Modeling of Change in Multi-Agent Organizations

Mark Hoogendoorn

Cover illustrations:
© USDA Forest Service, Black Hills National Forest

SIKS Dissertation Series No. 2007-08

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

VRIJE UNIVERSITEIT

Modeling of Change in Multi-Agent Organizations

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op maandag 18 juni 2007 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Mark Hoogendoorn

geboren te Nieuwkoop

promotoren: prof.dr. J. Treur
prof.dr. C.M. Jonker

If you want to truly understand something, try to change it.
- Kurt Lewin (1890 – 1947)

Preface

When writing a PhD thesis, and especially writing this part of the thesis, it provides the author with a good opportunity to reflect on the last four years of his working life. During these last four years, many people have contributed to this work in one way or the other. Well, I guess this is the opportunity to thank them.

My whole PhD project started in September 2003 after Catholijn Jonker had asked me whether I would be interested in becoming a PhD student in the agent systems group at the Vrije Universiteit. I did not have to think about it for long. My master thesis project within the research group of Maria Gini at the University of Minnesota, made me so enthusiastic about doing research, that I accepted the offer of Catholijn, and came into contact with my other promotor, Jan Treur. I must say that over the years Catholijn and Jan together have really been the ideal combination of promotores for me, for which I am very grateful. Coming back from the United States and before starting my PhD project I used to tell people how many hours people put into their work over there, which I found unimaginable in the Netherlands. Well, it is imaginable now. I would really like to thank Catholijn and Jan for always making time available for me whenever I needed them even though they were usually very busy. Both of them gave me all the opportunity to develop myself in every aspect that plays a role in doing research as well as working in an academic institution. I really learned a lot from them in all of these aspects. Furthermore, I would like to thank Catholijn for giving me an opportunity to investigate organizational change processes in real life, by switching her job two times ;-) .

The research I have conducted within my PhD project was part of a number of large research projects. In such projects both research as well as application partners are involved. This gave me the highly appreciated opportunity to apply my research in a practical setting and see the implications of it in real life. The projects in which I have participated are CIM (for Cybernetic Incident Management), funded by the Dutch Ministry of Economical Affairs; DEAL (for Distributed Engine for Advanced Logistics), funded by the same Ministry, and finally a project funded by Force Vision, the software development company of the Royal Netherlands Navy. I am grateful to all people involved in these projects for the various contributions they have made.

Of course, the atmosphere at work is a very important aspect when you are doing research. In that sense I have been very lucky to be part of the agent systems group. I would like to thank all of the members next to Catholijn and Jan over the past four years: Alexei, Annerieke, Charlotte, Egon, Fiemke, Ghazanfar, Lai, Lourens, Martijn, Peter-Paul, Pinar, Radu, Savas, Tibor, Vera, Viara, and Zulfiqar. I would like to thank them for being such great colleagues and friends, both during as well as outside working time, such as going out for a group dinner, the sailing trips, the Russian movie night and so on. Traveling to conference locations is one of the advantages of the type of work, but it is even more enjoyable if you have such great colleagues to accompany you. We have done lots of nice things over the years, from doing a road trip in the United States together, to enjoying the hectic city of Hong Kong.

Furthermore, I am very grateful to all my co-authors (other than those from the agent systems group) for their contribution to the chapters in this thesis: Hans Abbink, Bas de Bruin, Roel van Dijk, Tamas Dobos, Maria Gini, Peet van Tooren, Jeroen Valk, and Marian Verhaegh. Thank you very much for the great cooperation. Of course I would also like to thank the reading committee members of the PhD thesis for the effort and time they spent on reading the thesis, and their valuable comments. The reading committee consisted of Jaap Boonstra, Kathleen Carley, Virginia Dignum, Maria Gini, and Frank van Harmelen.

Finally, I would not have been able to perform all this work without the support of my family and friends. First of all, I would like to thank my parents for always supporting me in my work and the choices I have made. After a busy day at work, it is always good to know that you have people to come home to that care about you. They have really been a big support for me. Furthermore, I am very grateful to my grandmother who also supported me a lot and was always fascinated by what I was doing at the university. I feel very sad that she passed away one and a half year ago. Being her only grandson, I really would have liked her to witness me getting my PhD. Finally, I would like to thank my friends, whom I am not going to list here, for all the fun we had, and giving me the relaxation I needed to think about something else than work!

Mark

Contents

Part I:
Introduction

Chapter 1

Introduction

Introduction

1 Organizations

An organization is a systematic arrangement of elements that together aim at achieving a certain goal. The occurrence of organization is not limited to processes in human society. In biological systems and nowadays also in (software and hardware) computing systems organization occurs. In this thesis organizations are analyzed and simulated in a formal manner, with an emphasis on organizational change.

Historically, human organizations are studied in the fields of economics and social sciences. The characteristics of human organizations vary to a great extent. Organizations such as constructed for incident management and military organizations are extremely hierarchically structured and have detailed descriptions of what a person is supposed to do within an organization. On the other hand, adhocracies hardly have any specification of what one is supposed to do and rely on mutual adjustment of the individuals in the organization [19].

In biology various organizational forms have been investigated as well, for example, the organization of intracellular processes [3], circulatory systems, and the organization of for instance social insects [14]. One of the current trends is the investigation of *self-organization* in human and biological systems [6]. An interesting example thereof is a honeybee colony, where individual bees adapt in such a way that the hive's organizational form adapts to all normally occurring changes of situation. For example, when an attack is observed, the organization changes by adaptation of the bees within the hive, thereby creating a new organization which is suitable for the new circumstances [21].

Finally, in computer science organization of large software and hardware systems is being addressed. Such systems are becoming increasingly complex, and the coordination of the ever increasing number of software components, is of key importance to allow the systems to function properly. The formal nature of software and hardware systems make them particularly suitable objects for studying organizations.

2 Computational and Mathematical Organization Theory

The field of computational and mathematical organization theory aims at development and testing of organizational theories from disciplines as presented above, by means of both computational and mathematical models. Among the disciplines involved in computational and mathematical organization theory is the field of *multi-agent systems*. A multi-agent system can be defined as a system containing several interacting agents that pursue a particular process or goal, for more information, see e.g. [9; 20]. Although often studied for other reasons, multi-agent

systems aid the progress of computational and mathematical organization theory since they allow the investigation of collective behavior based upon individual agent behavior. The collective behavior is described by means of an abstract, organizational, perspective. The advantage of this additional abstraction level is that more complex processes can be modeled. As a result, more real world phenomena can be described. Examples of approaches that enable modeling of multi-agent systems at such an abstract level are AGR [10], MOISE [16], and Opera [8]. The approaches describe organizations from both a structural and behavioral perspective. In the structural description, the elements that are part of the organization are described whereas the behavior of those particular elements is described in the latter perspective.

The development of approaches to model organizations corresponds to a trend in agent technology to specify multi-agent systems from a more abstract, organizational perspective (see e.g. [2;12]), before going into the implementation of the agents themselves. This approach aids developers of multi-agent systems to maintain a clear perspective of the system as a whole. The modeling approaches used in computational and mathematical organization theory can be used for this purpose, as well as for the analysis and simulation of organizations in other domains.

3 Organizational Change

The percentage of change processes in human business organizations that do not achieve the intended goal is 70% [1; 13]. As a result of this, the literature on organization theory shows a growing emphasis on organizational change. In 1986 Cohen [7] stated that "*computational and mathematical models are particularly suited for the study of organizational evolution and change"*.

The development of an organizational abstraction level in multi-agent systems, and various approaches to model the organizational level of such systems is an important step towards better techniques for the analysis of organizational change processes. However, in order for an organizational modeling approach to be useful in describing dynamic multi-agent systems, it needs to have the ability of expressing such change processes in an adequate manner. Currently, some organizational modeling approaches do address some important aspects of change [15]. However, a more detailed analysis of theories of change is needed to provide modelers with a variety of templates and tools to address organizational change.

4 Research Goals

The goal of the research presented in this thesis is to analyze and model organizational change processes. In addition, the goal of this thesis is to provide modelers of organizations with a number of templates and tools that address organizational change, in such a way that these can be reused by these modelers.

5 Modeling Approach

The multi-agent organization modeling approach that has been used throughout this thesis for analysis and modeling of change processes consists of a structural model as well as a behavioral model to describe multi-agent organizations. The structural model is based on the AGR approach [10], the behavioral model extends AGR with formal behavioral specifications conform [11]. Both aspects are discussed in this section.

An organization is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. Note that by avoiding an agent-oriented viewpoint, an organization is regarded as a structural relationship between a collection of roles, where the roles can be fulfilled by agents. Thus, the idea of an organization can be described solely on the basis of its structure, i.e., by the way groups and roles are arranged to form a whole, without being concerned about the way agents actually behave. The specific architecture of the agents themselves is purposely not addressed in the organizational model, thus allowing for all possible realizations of the idea of an organization by agents. The approach followed in this thesis, therefore, allows studying the difference between the idea of an organization (in terms of groups and roles) and the realization of an organization (in terms of the behavior of the agents involved).

5.1 Structural Model

For the structural description of multi-agent organizations, the AGR (for agent/group/role) model has been adopted [10]. The three primitive definitions are:

- The *agents*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application.

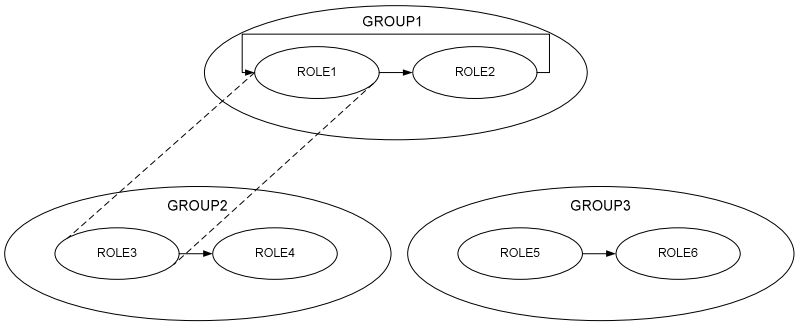

Fig. 1. Example organization modeled within AGR

- A *group* is defined as an atomic set of roles. Each agent plays a role in one or more groups. In its most basic form, the group is only a way to tag a set of roles. An agent can contribute to multiple groups at the same time. A major point of these groups is that they can freely overlap.

- A *role* is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group.

AGR distinguishes three aggregation levels: the organization as a whole, groups, and roles, as illustrated in Figure 1. The large ovals denote groups whereas the smaller ovals denote the roles within the organizations. Furthermore, the solid arrows denote intra-group interactions between roles within a given group, and the dashed lines represent inter-group interactions. Agents realizing the roles are not depicted. However, the specification of the aggregation levels can place additional constraints on the agents that are to realize the organization. For example, the dashed lines between role1 and role3 could indicate that those roles will have to be fulfilled by the same agent.

5.2 Behavioral Model

Describing the structure of an organization is not enough; the behavior has to be described as well. For example, the intra-group interactions in Figure 1, describe that Role5 can communicate to Role6, but it does not describe when this should occur, nor what content is to be communicated.

The specification of behavior follows the same aggregation levels as identified in AGR, namely the level of roles, groups, and the organization as a whole. The importance of such aggregation levels and the relation between these aggregations levels is emphasized by Lomi and Larsen [17]. In the introduction to their book they describe as a main challenge in the field:

- *"given a set of assumptions about (different forms of) individual behavior, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?"*
- *"given observable regularities in the behavior of a composite system, which rules and procedures - if adopted by the individual units- induce and sustain these regularities?"*

Both views and problems require means to express relationships between dynamics of different elements and different levels of aggregation within an organization. The different aggregation levels of the behavioral specification are shown in Figure 2 in the form of an AND tree.

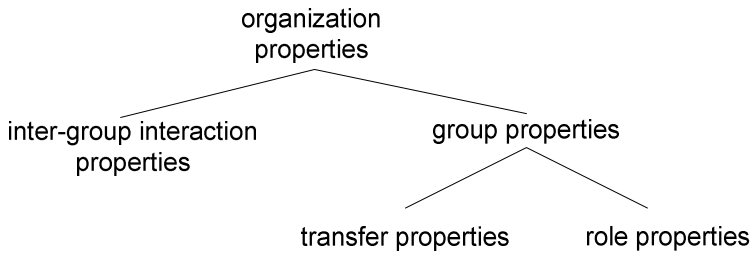

Fig. 2. AND tree of behavioral properties

As can be seen, on the highest level of the tree organizational properties are shown, which are properties the organization as a whole needs to achieve. At the level below the organizational level, group properties and inter-group interaction properties are specified, which together entail the organizational properties. The group properties are entailed by lowest level, namely role properties, and transfer properties, that specify interactions between roles within the same group.

The language TTL (for Temporal Trace Language), described in [4], has been adopted for the specification of behavior in organizational models.

5.2.1 Temporal Trace Language (TTL)

In TTL [4], ontologies for states are formalized as sets of symbols in sorted predicate logic. For any ontology Ont, the ground atoms form the set of *basic state properties* BSTATPROP(Ont). Basic state properties can be defined by nullary predicates (or proposition symbols) such as incident, or by using n-ary predicates (with n>0) like observes(amount_of_casualties, 7). The *state properties* based on a certain ontology Ont are formalized by the propositions (using conjunction, negation, disjunction, implication) made from the basic state properties and constitute the set STATPROP(Ont).

In order to express dynamics in TTL, important concepts are *states*, *time points*, and *traces*. A *state* S is an indication of which basic state properties are true and which are false, i.e., a mapping S: BSTATPROP(Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont). Moreover, a fixed *time frame* T is assumed which is linearly ordered. Then, a *trace* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by TRACES(Ont).

The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation, indicated by the infix predicate |=, comparable to the Holds-predicate in the Situation Calculus. Thus, state(γ, t) |= p denotes that state property p holds in trace γ at time t. Likewise, state(γ, t) $|\neq$ p denotes that state property p does not hold in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted predicate logic, using the usual logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and the quantifiers $\forall$, $\exists$ (e.g., over traces, time and state properties). The set DYNPROP(Ont, γ) is the subset of DYNPROP(Ont) consisting

of formulae with γ occurring in which is either a constant or a variable without being bound by a quantifier.

To model direct temporal dependencies between two state properties, not the expressive language TTL, but the simpler *leads to* format is used. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties (the leaves of the tree in Fig. 2). The format is defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold
for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [4]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can be depicted graphically in a causal graph like style.

6 Research Methodology

The modeling approach presented above has been repeatedly used to analyze and model particular organizational change cases according to the following general research methodology:

Identification and formalization of role and interaction properties. Identify and formalize the dynamic properties of the lowest level of aggregation in the organization, namely role properties, expressing the behavior of roles, and interaction properties, expressing nature and timing of the interaction between roles within the organization (also those within different groups).

Simulation using role and interaction properties. Simulate the organizational model based upon the identified role and interaction properties. To enable this, translate these properties in an executable format and input these properties into a simulation tool [5]. The result is a formal trace.

Formalization of an empirical trace. Obtain a log from the organization being studied, and formalize the occurrences observed in the trace.

Identification and formalization of group and organization properties. Identify and formalize the properties of the higher levels of aggregation within the organization, namely the group and organization properties.

Verification of properties against formal traces. Verify the group and organization properties identified against the formal traces obtained by simulation and/or by formalizing an empirical trace. In case of an empirical trace the properties for verification may also include the role and interaction properties. Such verification can be performed by a checker that uses the formal properties and the formal traces as input. The output of the checker states whether the properties are satisfied for all

formal traces, and in case a particular property is not satisfied, presents a counter example. Note that the results of this verification process only hold for the traces that have been checked, not the entire model. In case all possible traces have been generated, the results do hold for the entire model. An example of such a checker is the TTL checker [4].

Verification of organizational behavioral model. Besides verifying the properties against traces (simulated and/or empirical) of the organization, the inter-level relations within the behavioral model can also be verified. This can either be done by means of logical proof of the inter-level relations, or by means of model checking tools such as SMV [18].

The methodology presented above underlies all research presented in this thesis. In some chapters, however, not all elements of the general methodology are addressed.

7 Thesis Overview

This thesis is based on a collection of articles. The majority of the articles are reprints of refereed papers that have been published elsewhere, or are extensions thereof. Except for the layout of the papers, they have been left unchanged. As a consequence the overlap between the papers has not been removed, e.g., introductions to TTL and the organization modeling approach. Another consequence is that the chapters can be read independently. Note that, unless explicitly stated otherwise, all authors have a comparable share in the research presented in the articles and are therefore alphabetically ordered. The chapters in this thesis have been organized in seven parts.

7.1 Introduction

The introductory part positions the research described in the thesis. Furthermore, the goals of the research are stated. The general modeling approach that has been used throughout the thesis is briefly described, as well as the research methodology . Finally, the various parts of the thesis are introduced.

7.2 Organizational Change Preparation

The second part of this thesis addresses the preparation for organizational change. Such a preparation phase may cover aspects as monitoring and analyzing the current organization, and preparing a (re)design of the organization. This part presents a number of techniques that identify potential problems in the organization and tries to find organizational models that solve such problems. Chapter 2 presents a labeled graph approach to identify organizational elements that are overloaded when considering their specified capacity. Such signals could potentially result in a new organizational model that is able to handle the current load. In Chapter 3 a model of an agent which uses meta-reasoning capabilities to identify unpredicted occurrences

in an organization, and finds solutions that solve such occurrences, is presented. The approach is illustrated for the naval domain. A component-based model for organizational redesign is presented in Chapter 4. The model continuously monitors the environment in which the organization is participating, and the current requirements that have been posed upon the organization. In case it is observed that the requirements are no longer satisfied under the current environmental conditions, the component-based model generates a new organizational model. Finally, Chapter 5 presents an approach that adapts the capacity of an organizational model based upon max flow networks. Such a capacity adaptation could simply entail addition of capacity to particular roles or interaction elements, but copying of certain organizational elements is part of the presented method as well.

7.3 Organizational Change Process: Centralized Change

The implementation of organizational change is the actual process of moving from one organization form to another. Different types of organizational change can be distinguished, according to a centralized or a decentralized perspective. This part addresses organizational change addressed from a centralized perspective, in which a central decision is made upon the change to be performed. The first chapter, Chapter 6, provides a model and analysis of Lewin's classical unfreezing-movement-refreezing theory that is still used in social science nowadays. The analysis is illustrated by means of a case study drawing inspiration from the organization of the famous Dutch eleven cities tour. Chapter 7 presents a centralized organizational change process model for particular cases in the naval domain. The naval domain is characterized by central decision making; the commander is the one that decides upon organizational change. The model addresses the particularly important aspects in change processes of when particular roles are activated and deactivated. Finally, Chapter 8 presents an extensive case study of change in a number of Dutch municipalities. The study addresses both an analysis of a current organizational model, as well as the development of a potential organizational model. Furthermore, the centralized change process of moving from the current to a new organization is modeled and analyzed.

7.4 Organizational Change Process: Decentralized Change

Not all organizational change processes are orchestrated from a centralized perspective. This part addresses the decentralized perspective. In decentralized organizational change processes, no single entity coordinates the change, but various individual entities in the organization decide upon change for themselves. In order to analyze and model such processes, first of all inspiration has been taken from social insects, and more specific, honeybee colonies. Honeybee colonies are known to exhibit decentralized organizational change, and are also known to be very robust. A quantitative organizational model for honeybee colonies is presented in Chapter 9. Chapter 10 presents a more generic decentralized organizational change model which specifies adaptation on a more abstract level, covering both a quantitative and a

qualitative specialization of the model. For the qualitative model an extensive case study is presented in the field of incident management. The quantitative specialization is basically a generalized form of the organizational model presented in Chapter 9. Another form of decentralized change, is by negotiation between individual agents. For example, finding particular agents that perform certain tasks within the organization can be addressed by means of negotiation. Chapter 11 presents a negotiation model for the formation of virtual organizations. In the model the preferences and capabilities of individual agents are taken into account to form, by mutual agreement, an organization capable of performing the tasks presented to the organization over time. The results of the model are evaluated by means of analysis of the performance upon data obtained from a logistics company. Chapter 12 presents a model which aims to forming efficient organizations, ignoring the preferences of agents for the particular tasks. Again, a dataset in the field of logistics is used to evaluate how efficient the solutions found by the model are.

7.5 Organizational Change Process: Mixed Change

Besides models that strictly address centralized or decentralized change processes, also situations occur in which organizational change has both centralized and decentralized aspects. This part presents models that are able to both address centralized as well as decentralized (i.e. mixed) forms of organizational change. Chapter 13 presents an approach for evaluating coordination methods between various agents within an organization. The coordination approaches addressed vary from centralized methods, to completely decentralized coordination methods as well as completely pre-specified coordination methods, which exhaustively specify how to coordinate, versus methods specified in a more generic fashion. Furthermore, Chapter 14 presents a language for the specification of coordination between components in an organization. The language allows for the specification of the various types of coordination methods as mentioned above.

7.6 Organizational Change Process Evaluation

After change has been performed within an organization, the effectiveness of such a change can be evaluated. Such evaluation processes are the topic of this part. First of all, traces that have been obtained from an organization after a particular change has occurred can be analyzed. The analysis of such traces can be addressed by means of the verification of particular properties that should hold within such an organization (for instance particular role properties or performance properties for the organization as a whole). In Chapter 15 an approach is presented that enables automated verification of such properties against an empirical log. The approach is illustrated by means of a case study in the domain of incident management. Chapter 16 extends the approach presented in Chapter 15 in such a way that properties are also specified in a hierarchical form, and furthermore presents methods that allow for the analysis of human error. As a result, the approach can be used to analyze change within human organizations, thereby observing what types of errors are being made within the new

organization, allowing for correction of such errors. Finally, Chapter 17 presents an approach to formalize particular plans that exist for a change of organization, and evaluate such plans after the change has been performed. Such an evaluation consists of an automated evaluation of the plan for change compared to formalized traces corresponding to the current behavior of the organization.

7.7 Discussion

This final part presents conclusions, and discusses related work. The work presented in this thesis is compared to other approaches and to related literature in general. The part concludes with a perspective on future avenues for research in organizational modeling.

References

[1] Bashein, M.L., Marcus, M.L., and Riley, P., Business Process Reengineering: preconditions for success and failure, Information Systems Management 9, 1994, pp. 24-31.

[2] Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), Proc. of the 1st OOOP Workshop, 2005.

[3] Bosse, T., Jonker, C.M., and Treur, J., On the use of Organisation Modelling Techniques to Address Biological Organisation. *Multi-Agent and Grid Systems Journal*, vol. 3, 2007.

[4] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., Specification and Verification of Dynamics in Cognitive Agent Models, In: *Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06*. IEEE Computer Society Press, 2006.

[5] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), *Proc. of the Third German Conference on Multi-Agent System Technologies, MATES'05*. Lecture Notes in Artificial Intelligence, vol. 3550. Springer Verlag, 2005, pp. 165-178.

[6] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E., *Self-Organization in Biological Systems*, Princeton University Press, 2001.

[7] Cohen, M.D., Artificial Intelligence and the Dynamics of Performance in Organizational Designs, In: March, J.G. and Weissinger-Baylon, R. (eds.), Ambiguity and Command: Organizational Perspectives on Military Decision Making, Marshfield, MA, 1986.

[8] Dignum, V., A Model for Organizational Interaction: Based on Agents, Founded in Logic, PhD thesis, 2003.

[9] Ferber, J., *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Addison Wesley, 1999.

[10] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

[11] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems, in Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organizations. Proceedings of MAAMAW'01*, 2001.

[12] Giorgini, P., Müller, J., Odell, J. (eds.), *Agent-Oriented Software Engineering IV*, LNCS, vol. 2935, Springer-Verlag, Berlin, 2004.

[13] Hall, G., Rosenthal, T., and Wade, J. How to make reengineering really work, *Harvard Business Review*, 71(6), 1993, pp. 119-131.
[14] Hölldobler, B., and Wilson, E.O., The Ants, Harvard University Press, 1990.
[15] Hübner, J.F. and Sichman, J.S., Using the MOISE+ model for a cooperative framework of MAS reorganization, Boletim Técnico BT/PCS/0314, Escola Politécnica da USP, São Paulo, 2003.
[16] Hübner, J.F., Sichman, J.S., and Boissier, O., A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems. In: Proc. 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), Porto de Galinhas, Brasil, 2002. Extended abstract in: C. Castelfranchi and W.L. Johnson (eds.), Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02. ACM Press, 2002, pp. 501-502.
[17] Lomi, A., and Larsen, E.R.. Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park, 2001.
[18] McMillan, K., Symbolic Model Checking: An approach to the state explosion problem, Kluwer Academic Publishers, 1993.
[19] Mintzberg, H., Structure in Fives: Designing Effective Organizations, Prentice Hall, 1992.
[20] Weiss, G. (editor), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.
[21] Winston, M.L. and Punnet, E.N., 1982, Factors determining temporal division of labor in honeybees, *Canadian Journal of Zoology*, vol. 60, pp. 2947-2952.

Part II:
Organizational Change Preparation

Chapter 2

A Labeled Graph Approach to Analyze Organizational Performance

This chapter appeared as: Hoogendoorn, M., Treur, J., and Yolum , P., A Labeled Graph Approach to Analyze Organizational Performance. In: Nishida, T., Klusch, M., Sycara, K., Yokoo, M., Liu, J., Wah, B., Cheung, W., and Cheung, Y.-M. (eds.), *Proceeding of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006)*, IEEE Computer Society Press, 2006, pp. 482-489.

Furthermore, part of this chapter appeared as: Hoogendoorn, M., Treur, J., and Yolum, P., A Labeled Graph Approach to Support Analysis Organizational Performance. In: Fischer, K., Berre, A., Elms, K., and Muller, J.P. (eds.), *Proceedings of the Workshop on Agent-based Technologies and applications for enterprise interOPerability, (ATOP 2005)*, 2005, pp. 49-60.

A Labeled Graph Approach to Analyze Organizational Performance

Mark Hoogendoorn[1], Jan Treur[1], and Pınar Yolum[2]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, treur}@cs.vu.nl
[2]Bogazici University, Department of Computer Engineering,
TR-34342 Bebek, Istanbul, Turkey
pinar.yolum@boun.edu.tr

Abstract. Determining the performance of an organization is a must for both human and multi-agent organizations. The performance analysis enables organizations to uncover unexpected properties of organizations and allow them to reconsider their internal workings. To perform such an analysis, this paper represents organizations as labeled graphs that capture, not only the interactions of the entities, but also the characteristics of those interactions, such as their content, frequency, and so on through labels in the graph. Algebraic representation and manipulations of the labels enable analysis of a given organization. Hence, well-known phenomena, such as overloading of participants or asymmetric distribution of workload among participants can easily be detected. Finally, a case study is performed within the domain of incident management.

1 Introduction

Multi-agent organizations consist of agents that interact to carry out their tasks. Current models of multi-agent organizations usually represent organizations as consisting of roles that agents adopt. An organization model then specifies the structure and behavior of the organization in terms of the relations between the roles. An analysis of such an organization model could check if the model satisfies desired properties such as the possibility of completing a desired task given that all agents comply with the requirements of the organization. Whereas such an analysis is useful, it is not sufficient to analyze an executing organization. The main reason is that many design-time choices become concrete during execution. Agents choose who they want to interact with as well as how often they want to do so during run-time. For example, among two agents that enact a merchant role, one might be preferred over the other because the agent has better capabilities, more work capacity, and so on. These subtle interactions of agents at run-time can give rise to interesting situations that can only be detected during execution. That is, as a result of previous decision, one merchant agent will be more loaded than the second merchant will be. Further, the agents that participate in an organization might be designed and developed by independent

parties, which requires them to interoperate and execute intelligently at run-time. In other words, such facts about the workings of a multi-agent organization cannot be discovered from a static representation of an organization during design time, but can only be analyzed during the execution time.

Whereas there is a vast literature in the design of multi-agent organizations, there is little work on the analysis of executing multi-agent systems [9, 10]. For this reason, this paper provides a complementary treatment of multi-agent organizations, where in addition to existing design time dynamics of the organizations, a graph representation is used to analyze executing organizations. Executing multi-agent organizations are analyzed by logging the performance of the organization in traces. Graph representations are useful for analyzing organizations; for example for understanding the structure of an organization through theoretical concepts.

This paper presents a formal specification language based on a graph representation. The directed graph captures the relationships between participants in the organization and the labels give semantics to the relationships. Once the labeled graphs are constructed, they can be used to analyze the functioning of the organization at runtime, i.e. analyze traces of the execution of the multi-agent system. Organization designers or analyzers can study the graph to understand the shortcomings of the organizations and to restructure the organization as they see fit. This paper further shows that rules related to the organizations can be developed and automatically checked against the labeled graphs. As a concrete example, detection of overloaded agents is used.

The rest of this paper is organized as follows. Section 2 gives a representation of organizations as labeled graphs. Sections 3 discusses the usage of the graph for external analysis. Section 4 presents a case-study and Section 5 discusses the relevant literature.

2 Organizations as Labeled Graphs

A directed graph $G = (V, E)$ constitutes the basis of the description of an organization in this paper. V denotes the set of nodes, which represent agents that enact a role. E denotes the edges in the graph, which represent the interactions between agents. Graph-based representations are typically used to model processes in areas such as (distributed) workflow management, business process design, organization modeling and organizational performance measurement. Usually the graphs have no labels or simple labels; such as a number that denotes the strength of a link. However, in real organizations edges denote different types of relationships with different properties. To represent such relationships, this paper provides a more complex structure of the labels and formalizes the structure with an algebra.

The example organizations considered here contain agents that fulfill tasks, assign subtasks to other agents, and thus run a business together. There are two primitive concepts we consider: workloads and capacities. An edge e connecting u and v means in this particular application that *u* requires some work to be done by *v*; i.e., edge e denotes a request for *workload*. As in real life, *u* could request different tasks to be performed by *v*. A label on an edge specifies the task type and the strength of the task

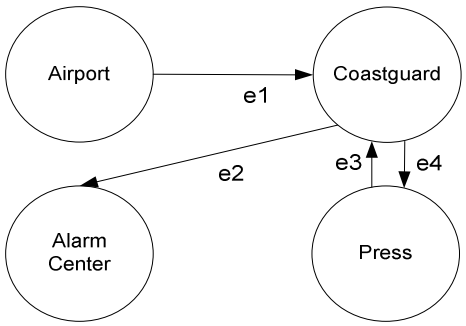

Fig. 1. An example organization graph

(i.e., how intensive the work is). The label also includes a list consisting of tasks the current task at hand originates from.

Example 1. Consider the organization in Figure 1. The figure gives a simplified representation of the disaster prevention organization in case of a plane crash in the Netherlands in the form of a labeled graph. Four agents enact the roles as shown in Figure 1: First of all, the airport role is present. This role takes care of the communication with airplanes and is the one that receives the mayday calls. After it has received a mayday call from a plane above the sea, it will contact the coastguard immediately to start a rescue task. The call causes the coastguard a lot of work, as they are in charge of the entire fleet of rescue ships. For possible precautions or backup from the land, the coastguard can contact the alarm center role which will arrange this type of help. The press is also represented as a role as they often request information regarding the number of casualties, information about the cause of the crash, and so on. The coastguard is responsible for fulfilling this task, which is called `Inform`.

Each agent in the organization has a certain capacity for each of the tasks that it can perform. Hence, the nodes of the graph are also labeled to denote the capacities of agents. First, a description of a formal language for the labels is given. Next, the capacities of the nodes will be discussed. Finally, the workload is defined.

2.1 Formal Specification Language

The formal language presented in this Section is based on many-sorted algebra. The sorts of the label specification language are shown and explained in Table 1. Based on these sorts, functions are defined to combine these sorts into labels. Statements of this language are equations as the examples accompanying the function definitions show. Throughout the text, when sorts and functions of the algebra are meant, they are denoted in `Courier` font.

First of all, a function is defined to construct a list containing pairs of subtasks. In general, the relation between tasks could be more general than the subtask relationship; for example, by incorporating information on the alternative tasks as

Table 1. Sorts used in the label algebra

Sort	Description
Value	Sort for real values.
Timepoint	Sort for moments.
TimeInterval	Sort for names of intervals that contain two time-points of sort Timepoint.
Node	Sort to identify a node.
Edge	Sort to identify an edge.
Task	Sort to identify tasks.
Load	Sort to identify loads.
LoadValue	Sort for a Load Value pair.
LoadValueList	Sort for a list of LoadValue pairs.
Label	Sort to identify a label.
LabeledLoad	Sort for a pair containing a LoadValueList and a Label.
TaskSubtaskList	Sort for a list of tasks with a subtask relationship between them.
TaskList	Sort for a list of tasks.
Capacity	Sort to identify a capacity.
CapacityValue	Sort for a pair containing the Capacity and a Value.
OverallCapacity	Sort to identify the overall capacity.
OverallCapacity Value	Sort for a pair containing the OverallCapacity and a Value.
EdgeActivation	Sort for specifying the Value of the amount of activations of an Edge during a certain TimeInterval

well. However, the focus here is on dividing a task into smaller pieces that will be performed by agents. Hence, only concentrating on the subtask relationship.

```
taskSubtaskPair: Task x TaskSubtaskList → TaskSubtaskList
```

Considering Example 1 one could express that the Rescue task has as a subtask LandOp which includes the operations that take place on land. Formally this can be expressed as follows:

```
tS=taskSubtaskPair(Rescue, taskSubtaskPair(LandOp, null))
```

Besides that, another function is specified which expresses a regular list of tasks without the subtask relationship between them.

```
taskList: Task x TaskList → TaskList
```

For example, a list containing the tasks that can be performed by the coastguard:

```
tL = taskList(Rescue, taskList(Inform, null))
```

For expressing the load three sorts are used: (i) the list which specifies the task from which this task originates, (ii) the node that carries the load, and (iii) the time interval for which this all holds. Intuitively, a load captures the intensity of the task a node has to do in a given time interval.

```
loadFor: TaskSubtaskList x Node x TimeInterval → Load
```

In the running example, the load for the coastguard can be expressed for TimeInterval I (for example 8 hours) and the Rescue task:

```
L = loadFor(tS, Coastguard, I)
```

A load is accompanied by a value expressing the amount of work caused by the load.

```
loadValuePair: Load x Value → LoadValue
```

For the Load defined above the value is set to 5:

```
LV = loadValuePair(L, 5)
```

Constructing a list from these `LoadValue` pairs can be done by means of a function. A communication from a role to another role can cause different kinds of load, therefore there is a need to express more than one load for each edge.

```
loadValuePairList: LoadValue x LoadValueList → LoadValueList
```

In the case of the example, only one `LoadValue` is present:

```
LVL = loadValuePairList(LV, null)
```

Now that the load caused by a connection in a graph can be fully specified it is combined with a label identifier.

```
loadLabel: LoadValueList x Label → LabeledLoad
```

The label specified above is now called `L1`:

```
LL = loadLabel(LVL, L1)
```

Now a label identifier is associated with an edge.

```
labeledEdge: Edge x Label → LabeledEdge
```

```
LE = labeledEdge(e1, L1)
```

Finally, at runtime an edge will be activated a certain number of times over a certain period, which can also be expressed in the algebra:

```
edgeActivation: Edge x TimeInterval  x Value → EdgeActivation
```

For example, the edge `E1` was activated 2 times during `TimeInterval I`:

```
EA = edgeActivation(e1, I, 2)
```

Capacities can also be expressed by means of the functions. Capacities belong to nodes, as they are the ones that need to carry the load. The next Section will go into more detail on expressing the capacities. The capacity of a node is the amount of task it can do in a certain time period. The amount of task is denoted by a `TaskList` and the time period is denoted by a `TimeInterval`.

```
capacityOf: TaskList x Node x TimeInterval → Capacity
```

A value can be added to the capacity, for example, during the time-interval for which the capacity is specified, one man-hour is available for rescuing.

```
capacityValue: Capacity x Value → CapacityValue
```

Besides a capacity for specific tasks, a node also has an overall capacity. This overall capacity exists independent of types of tasks it can do.

```
overallCapacity: Node x TimeInterval → OverallCapacity
```

A value can again be added to this kind of capacity. It can for example say that during the time-interval of a day a maximum of 8 man-hours are available for a specific node.

```
overallCapacityValue: OverallCapacity x Value →
OverallCapacityValue
```

Using the basic ontology of this algebra, its relations can be expressed, and logical relationships can be defined: The primitive terms used in the label algebra are defined by a many-sorted signature. The signature takes into account symbols for sorts, constants, functions and relations, including the equality relation. Among the relations, the equality relation has a special position: the identities (equations) between algebraic term expressions. Further relations can be defined by a relation symbol instantiated with term expressions. Logical relationships involve conditional statements involving relations, both the equality relation and other relations. For

simplicity these logical relationships are assumed to be in a clausal format. Examples of constants are names of values, examples of function symbols are +, x, examples of relation symbols are = and <. Examples of logical relationships are

if $t1 < t2$ then $f(t1) < f(t2)$
if $t1 < t2$ then $f(t1 + t2) = f(t2)$

If no other relations than the equality relation occur, the algebra is called functional.

2.2 Capacities

The capacity of a node should be represented flexibly so that realistic situations can be modeled. The following scenarios are seen frequently. For these scenarios, it is assumed that the unit of capacity is man-hours. The maximum man-hours available is fixed: in this case to eight man-hours.

1. **Fixed Capacities:** An agent has a fixed number of hours it can spend on each task as dictated by its role. The sum of these hours should not be more than the maximum amount available.
2. **Constant Task-Specific Capacities**: This time an agent is told how many hours it can spend on each individual task. For example, if the role enacted by this agent has two tasks, coordinating the rescue operations and informing the press, then a possible restriction could state that the agent playing the role can spend at most 5 hours on the rescue operations and 5 hours on informing the press. Of course, working on the rescue operations task for 5 hours still leaves 3 hours for the informing the press task. That is, the maximum number of hours is still constant.
3. **Group-Restricted Capacities:** This time the restriction is not on individual tasks but on sets of tasks. For example, a role can spend a maximum 5 hours on the rescue operations and informing the press and maximum of 4 hours on writing reports. The choice of distributing the 5 hours between the rescue operations task and the informing the press task is up to the agent that plays the role. However, the time spent on the rescue operations and informing the press together cannot exceed 5 hours.
4. **Flexible Capacities**: An agent can decide to work any number of hours on any of its tasks, as long as a certain maximum is not exceeded during the time-interval for which this capacity holds.

It is actually easy to see that both Scenarios 1 and 4 can be modeled in terms of Scenario 2. To model the first scenario, the only thing that needs to be ensured is that the total of the fixed capacities adds up to the maximum. This already defines the scenario in terms of constant task-specific capacities. For the fourth scenario, the individual restriction for each individual work has to be set to the maximum 8 hours. Additionally, Scenario 2 can be modeled a special case of Scenario 3 where each set consists of one task. Hence, accommodating Scenario 3 enables accommodating the remaining scenarios. For the sake of simplicity, disjoint sets of tasks are assumed for a specification of the capacity.

Example 2. To give an example, consider the node `Coastguard`, having capacity for tasks `Rescue` and `Inform`. The capacity of the `Coastguard` concerning the `Rescue` task in the `TimeInterval I` is 8. For the `Inform` task this maximum is

set to 2. Combined however, the overall capacity is set to 8, meaning that for the `Inform` and `Rescue` tasks together the time spent can not exceed 8. According to the formal notation as introduced in Section 2.1, the example can be formalized as shown below.

```
c1 = capacityOf(taskList(rescue, null), Coastguard, I)
cval1 = capacityValue(c1, 8)
c2 = capacityOf(taskList(inform, null), Coastguard, I)
cval2 = capacityValue(c2, 2)
co = overallCapacity(Coastguard, I)
coval = overallCapacityValue(cO, 8)
```

2.3 Workloads

A workload of a node is the amount of work it is required to do. Much work has been done to define the concept of workload more precisely, however there is still little consensus on a single definition. In [4] the 'human workload' is described as follows: "The intrinsic difficulty of the activities that an operator must perform establishes the target or nominal level of workload. The difficulty of a particular task may be influenced by any one or several of the following factors: (1) the goals and performance criteria set for a particular task; (2) the structure of the task; (3) the quality, format, and modality in which information is presented; (4) the cognitive processing required; (5) the characteristics of the response devices."

In operations management [8] research has been performed to define the time required to do a job in order to generate a unit of output, which is called work measurement. The initiator of this type of measurement was F.W. Taylor with his scientific management approach. It has however fallen into disfavor because if focuses on routine, repetitive tasks, but recently the labor-intensive service companies have resulted in a new popularity.

The workload of an agent in this paper is determined based on the tasks assigned to it now, how often these assignments take place, and how much of these tasks are delegated to other agents. In general, the agent would perform a percentage of the tasks on its own and assign the remaining tasks to other agents; i.e., create workloads for others. In principle, the newly created workload should be less than that of the initial workload of the agent. The workload of an agent is only determined during execution. Hence, it is not possible to know the workloads exactly during design time and distribute work accordingly.

3 Specification for labels with respect to loads

As has been mentioned before, labeled organization graphs can be used to analyze an organization. It can first be used to model the capacities and the workloads, and thereafter can be applied to analyze a trace representing the state of affairs within a multi-agent system during a certain period.

3.1 Calculations for values of loads

The workload of a node v during an interval I for a task t can be calculated in the following way: Let *workload(e,t)* be the workload for task t caused upon one activation of edge e. This number can be derived from the labeled algebra. First, look up the `taskSubtaskList` associated with this `Task t`: `taskSubtaskList(t, TSL)`. Thereafter get the label for edge e: `labeledEdge(e,L1)`. Now, look up the identifier of the `LoadValueList` via the `Label`: `loadLabel(LVL, L1)` and scan all entries of the `LoadValueList` for a `Load` in which the `TaskSubtaskList` starts with an element in `TSL` or starts with `t`, and holds for `TimeInterval I`. Finally, sum up the `Value` for each of these `Load` elements. Furthermore, for each of these edges, get the amount of activations, during `TimeInterval I`, then the workload can be calculated as shown in Definition 1:

Definition 1. Workload(v, t, I) =
$\sum_{e \in \text{ incomingEdges(v)}}$ a1 x workload(e, t) where edgeActivation(e, I, a1)
- $\sum_{e \in \text{ outgoingEdges(v)}}$ a2 x workload(e, t) where edgeActivation(e, I, a2)

Which entails summing up the workload caused by all incoming nodes, and subtracting from that the workloads distributed through the outgoing edges. The calculation of the overall workload of a node (for all tasks t) is simply summing up all separate workloads, as shown in Definition 2.

Definition 2. workload(v, I) = $\sum_{t \in \text{ tasks}}$ workload(v, t, I)

Example 3. Consider the organization as presented in Example 1 and 2. Imagine the following scenario (during an interval I): A Dakota airplane has crashed in the sea, the airport forwards this crash message to the coastguard (causing a load of 5), who in turn delegates the land operations to the alarm center (causing them a load of 1). Besides that, the press starts asking questions about the crash (causing a load of 1 each time), as they have observed the plane crashing in the sea. They request information 40 times, and the Coastguard replies the same number of times (causing the press a load of 0.8 each time). The workload calculation is as follows: workload(coastguard, rescue, I) = (1 * 5) – (1 * 1) = 4 man-hours during interval T for the rescue task workload(coastguard, inform, I) = (40 * 1) – (40 * 0.8)=8 man-hours during interval I for the inform task.

As the calculation for the workload has been explained, the workload of a node can be compared with the capacity of a node, this is referred to as the load of a node. Two different types of loads have been distinguished. First of all, the load for a specific task t can be calculated. To calculate this load, first remember that the capacities are defined for a list of tasks, let l be the list of which t is an element. As it is impossible to calculate loads for individual tasks, loads can only be calculated in terms of these lists of tasks, therefore the calculation of a load for a task t is done by means of the list l the task is part of. Let v be a node, t be a task and I be an interval and let *capacity(v, l, I)* be the capacity of the node v for task list l, during interval I. This can be derived from the labeled algebra as follows: Get the `capacity` for `TaskList L` in which task t is defined for node v during interval I: C = `capacityOf(L,v,I)`.

Thereafter, look up the `Value CV` of this capacity: `capacityValue(C,CV)`. Now the load is defined as shown in Definition 3.

Definition 3. load(v, t, I) =
$(\sum_{task \in l}$ workload(v, task, I))/ capacity(v, l, I) where $t \in l$

This defines that the load for a task is calculated by summing up all workload within the list l (so for every task within l) and dividing it by the capacity defined for that list.

The load can also be calculated for the node as a whole, this is simply done by taking the workload of the node, and dividing it by the overall capacity, *capacity(v, I)*, which can be found using the algebra: `CA = overallCapacity(v,I)` after which the `Value OCV` can be looked up: `overallCapacityValue(CA,OCV)`. The load is now calculated as shown in Definition 4.

Definition 4. load(v, I) = workload(v, I)/ capacity(v, I))

An example of an interesting type of information that can be derived from the load is the load distributions among the nodes in the graph. An organization with evenly balanced nodes is typically preferable over a very uneven distribution of loads.

Example 4. Picture the organization in case of an airplane crash in the North-Sea, the Netherlands again. Following the capacity example as given in Section 2.2 the coastguard has a capacity of 8 man-hours during I for the rescue task, and a capacity of 2 man-hours for the inform task, during that same period. Another capacity that is part of this organization is that of the press. The capacity of the press (which is not shown in a formalization) is defined as being 50 during the time-interval I in which the incident management occurs. The load of the coastguard and the press nodes can be calculated: The general load for the coastguard is: load(coastguard, I) = (12 / 8) = 1.5. More specifically, for the task rescue the load is 0.5 and for the inform task the load is 4.0. For the press, the workload is only caused by the information coming from the coastguard, which can not be distributed elsewhere. Therefore the workload of the press is 40 x 0.8 = 32. As they only have one task, the load of the press, load (press, I), is equal to 0.64. Based on this, it can be seen that the press has a relatively low load compared to the coastguard. By means of this information, a person that is analyzing an organization could suggest that the press should reduce the requests for information to the coastguard and try getting most of their information within the press organization, as they still have sufficient capacity.

3.2 Overloading

As the load for a node has been defined, the definition of a node being overloaded can be given. A certain role is overloaded in case one, or both of the following situations hold: (1) There exists a task t for which the load is greater than 1.0; (2) The load for the entire node, load(v, I) exceeds 1.0. A formal definition is presented below. Please note that due to the choice of representing the capacities by group restricted capacities it can occur that the loads for the individual group are not overloaded whereas the overall load is.

Definition 5: overloaded(v, I) =
$\exists$t:Task (load(v,t,I) > 1.0) $\vee$ (load(v,I) > 1.0)

Example 5. Following from example 4, it can be seen that the role of coastguard is heavily overloaded, for one of the tasks (inform) the load is 4.0, which means 4 times the capacity. The press however is not overloaded as it has a load value of 0.64.

4 Case-Study: Dakota Incident

This Section presents details regarding the implementation of the labeled graph approach into a software tool, and shows an empirical evaluation using a trace obtained from the domain of incident management.

4.1 Implementation

In order to be able to use the algebra and calculations for analyzing multi-agent organizations, a software tool has been created. First, the algebra presented in Section 2 has been implemented in PROLOG [1], including the calculations that are presented in Section 3. For a comparative study of translating an algebraic specification into a PROLOG program, see [2]. A specific interval can be specified over which the calculations of the organizational performance are done. Thereafter, in order to make the calculations of the workloads and loads for the nodes more insightful for e.g. domain experts to evaluate, a visualization tool has been created that graphically shows how much work is being transferred between different nodes within the graph, and represents the load for each of these nodes. Figure 2 shows a screen-shot of the visualization tool. The radius of a node is increased in case the load increases, so the bigger the node the heavier the load on that specific node. Further, communication

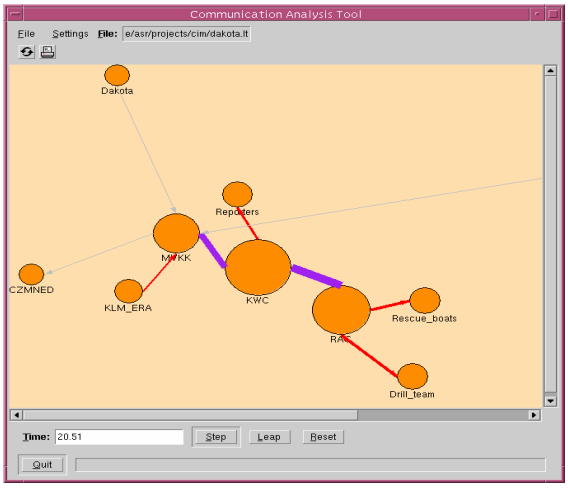

Fig. 2. Screenshot of the visualization tool

channels that are intensively used (i.e. edges that are activated many times during a particular time interval) are highlighted as well by turning red in case of a lot of activity (or in case of a huge amount of activity purple).

4.2 Empirical Evaluation

In order to evaluate the functioning of the implementation and the approach itself, a case study has been performed in the incident management domain. The case-study itself is based upon reports of a plane crash which occurred in the Netherlands in 1996. A trace of the events that occurred during the rescue of the passengers on board of the plane has been obtained from domain experts and logs that have been made of the communications that took place during the incident management in 1996. The examples used in Sections 2 and 3 include simplifications used for this case study. To enable an analysis, the organization, including the roles and the communications that took place, has been translated to a graph. Thereafter, a domain expert has labeled the graph with the values he thinks are appropriate values for workload caused by activation of a communication line (i.e. an edge). Furthermore, the expert has set capacities for the roles (i.e. nodes) within the incident management organization. According to the experts in the field (written down in incident management reports) the role of the coastguard (abbreviated in the figure to KWC) was heavily overloaded due to too many requests for information of the press, regional alarm center (RAC) and the military airport (MVKK). This indeed showed in the visualization, based on the capacities and workloads set in the graph. The coastguard has a large capacity for handling all the work, but is unable to handle all incoming requests. This shows that the analysis using the labeled graph approach is indeed in line with the manual expert evaluations.

5 Discussion

This paper has presented a formal language for specifying organizations. The specification is based on a graph formalism. The nodes of the graph represent agents and the edges between the nodes are labeled to denote why those edges exist. This allows us to represent the interactions between the agents in an expressive way. It has been shown that using this organization structure properties of executing organizations can be detected, such as the cases where the organization hosts overloaded agents, successfully.

Operations research is a closely related field to the research presented in this paper, see e.g. [5]. Many theories have been developed in that field of research to enable a proper functioning of the organization as a whole, creating a planning for these operations, etc. The research presented in this paper is meant to monitor the performance of these organizations, not to design these operations within the organization.

Another related field is workflow management, in which tools exist that measure and analyze the execution of processes so that continuous improvements can be made. The approach in workflow management can be used as a support tool to analyze the

execution, however workflow management systems constitute a huge system which is put into the organization to measure the performance, whereas the approach in this paper simply needs traces of the events and values for the capacities of nodes and workloads regarding tasks. This also enables the presented approach to be used for analyzing occurrences in the past and organizations in which introducing a workflow management system is not feasible.

There is a vast literature on designing multi-agent organizations. Zambonelli *et al.* develop a design methodology, GAIA [11]. GAIA identifies roles, organization rules, environment, and so on as necessary organizational abstractions. Using these constructs, GAIA methodology helps a system designer build its system in a systematic way. Padgham and Winikoff develop Prometheus, an agent-based software development methodology [7]. It consists of a system specification, architectural design, and detailed design phases. While these approaches are useful for designing multi-agent systems, they do not provide any mechanisms for analyzing executing organizations. That is, these methodologies only care for the design phase, but are not targeted for analyzing the multi-agent system during execution, which is the case for the methodology presented in this paper.

Handley and Levis create a model to evaluate the effect of organizational adaptation by means of colored Petri nets [4]. The Petri nets are used to represent external interaction of decision makers as well as internal algorithms the decision maker must perform, and are equipped with labels. In this model the workload of the decision makers is monitored and is used as a performance indicator. The concept of entropy is used to measure the total activity value (which is linked to the workload) of a decision maker. When an overload of a decision maker occurs, the execution time of the internal algorithm has a delay of one additional time point. Decision makers can also base decisions on who to forward an output to on the total activity of the decision maker that can be chosen. Their approach differs from the approach in this paper in the sense that they specify the entire process within the organization, and use the Petri nets to actually simulate an organization. Therefore, their aim is more towards the decision process and the evaluation thereof whereas the approach presented here is more intended as a separate method for evaluating the performance of an organization from an external viewpoint.

Fink *et al.* develop a visualization system to help monitor the performance of businesses [3]. The focus of their work is on presenting a tool that can incorporate different performance metrics from different sources. The aim of the approach presented here is to analyze workings of a business automatically. In this sense, the work of Fink *et al.* is complementary to the work in this paper. Once certain properties are detected by the approach in this paper, they could be feed into a visualization tool to ease the exposure.

The work presented in this paper is open for further improvements. Whereas this paper mainly deals with calculating the effect of an edge on its endpoints, it is also possible to calculate the effects of an edge on nodes that are not immediate endpoints. This can be regarded as calculating the cascading effects of interactions on third parties. Similarly, the representation can be made richer by adding capacities or workflows for groups of agents to model the smaller units in an organization. Ideas developed in this paper can also be used to help agents model others and reason about others' workloads to manage their interactions more efficiently. Such reasoning could

possibly even result in change of an organization in case the workload simply cannot be handled, see [6] for more extensive results on this. Furthermore, investigations on how well the approach scales up to large scale multi-agent systems will need to be performed in the future. One important possibility to note here is that of specifying such a system on multiple aggregation levels, whereby the analysis can take place at the highest level (e.g. the workload between departments) while at the lower level focus on parts of the organization (e.g. the workload within a department).

References

[1] Colmerauer, A., Kanoui, H., Pasero, R., and Roussel, P., Un Système de Communication Homme-Machine en Français. Groupe de Recherche en Intelligence Artificielle, Université d'Aix-Marseille, Lumini, 1971.

[2] Drosten, K., Translating algebraic specifications to Prolog programs: a comparative study. In: J. Grabowski, P. Lescanne, and W. Wechler, eds., *Algebraic and Logic Programming*, LNCS 343, pp 137–146, Springer, 1988.

[3] Fink G., Krishnamoorthy S., and Kanade A., Naval Crew Workload Monitoring and Visualization. In: First Annual Conf. on Systems Integration, NJ, 2003

[4] Handley, H., and Levis, A., A Model to Evaluate the Effect of Organizational Adaptation. *Computational & Mathematical Organization Theory* 7(1) pp 5–44, 2001.

[5] Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, McCraw-Hill, SF, 2002.

[6] Hoogendoorn, M., Adaptation of Organizational Models for Multi-Agent Systems based on Max Flow Networks, In: Proceedings of IJCAI 2007, Morgan Kaufmann Publishers, To Appear, 2007.

[7] Padgham, L. and Winikoff, M. Prometheus: A Methodology for Developing Intelligent Agents. In: F. Giunchiglia et al. (eds.) Proc. of the AOSE Workshop, LNCS 2585, pp. 174–185, 2003.

[8] Russell, R.S., and Taylor, B.W., Operations Management, Prentice Hall, New Jersey, 2003.

[9] Shehory O, Sturm A: Evaluation of modeling techniques for agent-based systems. Proceedings of the Intl. Conf. on Autonomous Agents, pa 624–631, ACM Press 2001.

[10] Sudeikat J., Braubach L, Pokahr A., and Lamersdorf W. Evaluation of Agent Oriented Software Methodologies Examination of the Gap between Modeling and Platform. Preproceedings of the AOSE Workshop, 2004.

[11] Zambonelli F., Jennings N.R., Wooldridge M. J., Developing Multiagent Systems: The Gaia Methodology, ACM Transactions on Software Engineering and Methodology, 12(3), September 2003.

Chapter 3

An Agent-Based Meta-Level Architecture for Strategic Reasoning in Naval Planning

Part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., Maanen, P.P. van, and Treur, J., An Agent-Based Meta-Level Architecture for Strategic Reasoning in Naval Planning. In: Kolp, M., Bresciani, P., Henderson-Sellers, B., and Winikoff, M. (eds.), *Agent-Oriented Information Systems III: Post-Proceedings of the 7th International Bi-Conference Workshop AOIS 2005*, LNAI 3529, Spinger Verlag, 2006, pp. 216-230. The original publication is available at www.springerlink.com.

Furthermore, a three page abstract of this paper appeared as: Hoogendoorn, M., Jonker, C.M., Maanen, P.P. van, and Treur, J., A Meta-Level Architecture for Strategic Reasoning in Naval Planning (extended abstract). In: Ali, M., and Esposito, F. (eds.), *Innovations in Applied Artificial Intelligence: Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, LNAI 3533, Springer Verlag, 2005, pp. 848-850. The original publication is available at www.springerlink.com.

An Agent-Based Meta-Level Architecture for Strategic Reasoning in Naval Planning

Mark Hoogendoorn[1], Catholijn M. Jonker[3], Peter-Paul van Maanen[1,2], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Dept. of Artificial Intelligence,
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
{mhoogen, treur}@cs.vu.nl
[2] TNO Human Factors, Dept. of Information Processing,
P.O. Box 23, 3769 ZG Soesterberg, The Netherlands
peter-paul.vanmaanen@tno.nl
[3] Radboud University Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
c.jonker@nici.ru.nl

Abstract. The management of naval organizations aims at the maximization of mission success by means of monitoring, planning, and strategic reasoning. This paper presents an agent-based meta-level architecture for strategic reasoning in naval planning. The architecture is instantiated with decision knowledge acquired from naval domain experts and is formed into an executable agent-based model which is used to perform a number of simulation runs. To evaluate the simulation results, relevant properties for the planning decision are identified and formalized. These important properties are validated for the simulation traces.

Keywords: Meta-reasoning, planning, intelligent agent systems.

1 Introduction

The management of naval organizations aims at the maximization of mission success by means of monitoring, planning, and strategic reasoning. In this domain, strategic reasoning more in particular helps in determining in resource-bounded situations if a go or no go should be given to, or to shift attention to, a certain evaluation of possible plans after an incident. An incident is an unexpected event, which results in an unmeant chain of events if left alone. Strategic reasoning in a planning context can occur both in plan generation strategies (cf. [15]) and plan selection strategies.

The above context gives rise to two important questions. Firstly, what possible plans are first to be considered? And secondly, what criteria are important for selecting a certain plan for execution? In resource-bounded situations first generated plans should have a high probability to result in a mission success, and the criteria to determine this should be as sound as possible.

In this paper a generic agent-based meta-level architecture (cf. [10]) is presented for planning, extended with a strategic reasoning level. Besides the introduction of an agent-based meta-level architecture, expert knowledge is used in this paper to formally specify executable properties for each of the components of the agent architecture. In contrast to other approaches, this can be done on a conceptual level. These properties can be used for simulation and facilitate formal validation by means of verification of the simulation results.

The agent architecture and its components are described in Section 2. Section 3 presents the method used to formalize the architecture. Section 4 presents each of the individual components on a more detailed level and instantiates them with knowledge from the naval domain. Section 5 describes a case study and discusses simulation results. In Section 6 a number of properties of the model's behavior are identified and formalized. A formal tool TTL Checker is used to check the validity of these properties in the simulated traces. Section 7 is a discussion.

2 An Agent-Based Meta-level Architecture for Naval Planning

The agent-based architecture has been specified using the DESIRE framework [2]. For a comparison of DESIRE with other agent-based modeling techniques, such as GAIA, ADEPT, and MetateM, see [13, 11]. The top-level of the system is shown in Figure 1 and consists of the ExternalWorld and the Agent. The ExternalWorld generates observations which are forwarded to the Agent, and executes the actions that have been determined by the Agent. The composition of the Agent is based on the generic

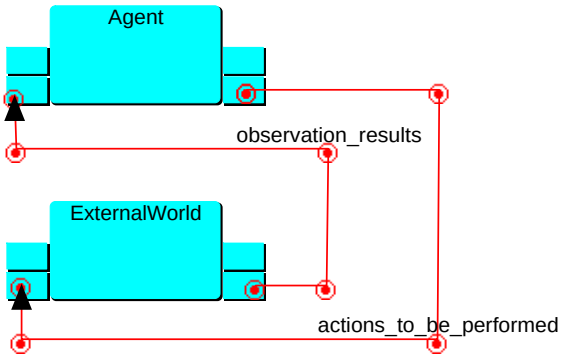

Fig. 1. Top-level architecture

agent model described in [3] of which two components are used: WorldInteractionManagement and OwnProcessControl, as shown in Figure 2. WorldInteractionManagement takes care of monitoring the observations that are received from the ExternalWorld. In case these observations are consistent with the current plan, the actions which are specified in the plan are executed by means of forwarding them to the top-level. Otherwise, evaluation information is generated and forwarded to the OwnProcessControl component. Once OwnProcessControl receives such an evaluation it determines whether the current plan needs to be changed, and in case it does, forwards this new plan to WorldInteractionManagement.

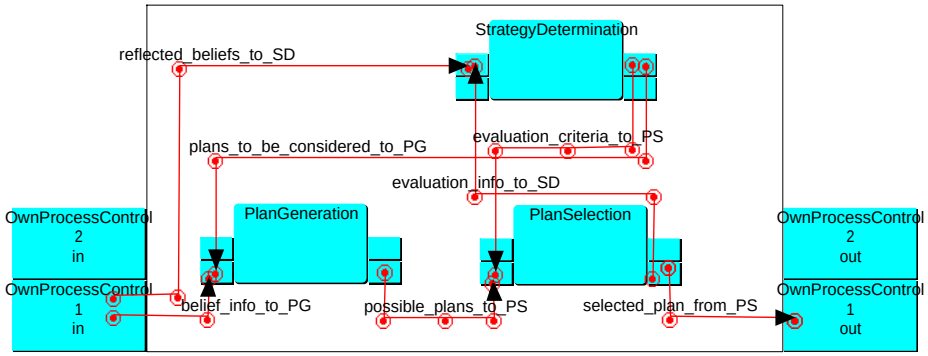

Fig. 2. Agent architecture

WorldInteractionManagement can be decomposed into two components, namely Monitoring and PlanExecution which take care of the tasks as previously presented (i.e. monitoring the observations and executing the plan). For the sake of brevity the Figure regarding these components has been omitted.

OwnProcessControl can also be decomposed, which is shown in Figure 3. Three components are present within OwnProcessControl: StrategyDetermination, PlanGeneration, and PlanSelection. The PlanGeneration component determines which plans are suitable, given the evaluation information received in the form of beliefs from WorldInteractionManagement, and the conditional rules given by StrategyDetermination. The candidate plans are forwarded to PlanSelection where the most appropriate plan is selected. In case no plan can be selected in PlanSelection this information is forwarded to the StrategyDetermination component. StrategyDetermination reasons on a meta-level (the input is located on a higher level as well as the output as shown in Figure 3), getting input by translating beliefs into reflected beliefs and by means of receiving the status of the plan selection process from PlanSelection. The component has the

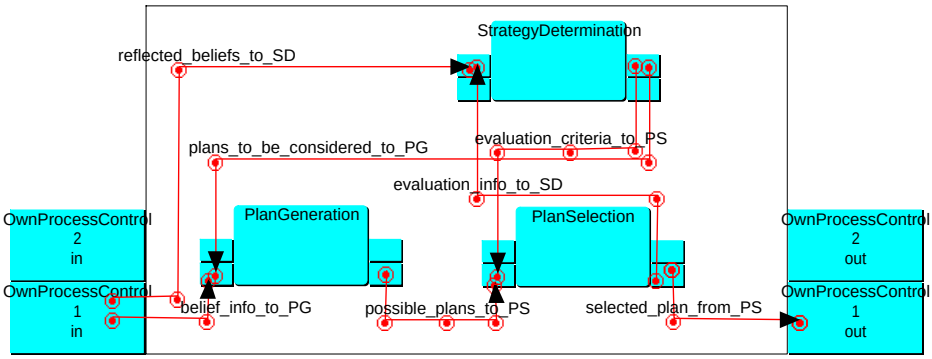

Fig. 3. Components within OwnProcessControl

possibility to generate more conditional rules and pass them to PlanGeneration, or can change the evaluation criteria in PlanSelection by forwarding these criteria.

The model has some similarities with the model presented in [7]. A major difference is that an additional meta-level is present in the architecture presented here for the StrategyDetermination component. The advantage of having such an additional level is that the reasoning process will be more efficient, as the initial number of options are limited but are required to be the most straightforward ones.

3 Formalization Method

In this section the method used for the formalization of the model presented in section 2 is explained in more detail. To formally specify dynamic properties that are essential in naval strategic planning processes and therefore essential for the components within the agent, an expressive language is needed. To this end the Temporal Trace Language (TTL) is used as a tool; cf. [8]. In this section of the paper both an informal and formal representation of the properties are given.

A state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of a component c within the agent (e.g., PlanGeneration, or PlanSelection) at time point t is denoted by state(γ, t, input(c)).
Analogously state(γ, t, output(c)) and state(γ, t, internal(c)) denote the output state, internal state and external world state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(c)) |= p denotes that state property p holds in trace γ at time t in the output state of agent-component c. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. In trace descriptions, notations such as state(γ, t, output(c))|= p are shortened to output(c)|p.

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [9]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can easily be depicted graphically.

4 Component Specification for Naval Planning

This Section introduces each of the components within the strategic planning process in more detail. The components presented in this section are only those part of OwnProcessControl within the agent as they are most relevant for the planning process. A partial specification of executable properties in formal format is also presented for each of these components. The properties introduced in this Section are generic for naval (re)planning and can easily be instantiated with mission specific knowledge. All of these properties are the result of interviews with officers of the Royal Netherlands Navy.

4.1 Plan Generation

The rules for generation of a plan can be stated very generally as the knowledge about plans. Conditions for those plans are stored in the StrategyDetermination component, which is treated later. Basically, in this domain the component contains one rule:

if belief(S:SITUATION, pos)
and conditionally_allowed(S:SITUATION, P:PLAN)
then candidate_plan(P:PLAN)

Stating that in case Monitoring evaluated the current situation as being situation S and the PlanGeneration has received an input that situation S allows for plan P then it is a candidate plan. This information is passed to the PlanSelection component.

4.2 Plan Selection

Plan selection is the next step in the process and for this domain there are three important criteria that determine whether a plan is appropriate or not: (1) Mission success; (2) safety, and (3) fleet morale criterion. In this scenario it is assumed that a weighed sum can be calculated and used in order to make a decision between candidate plans. The exact weight of each criterion is determined by the StrategyDetermination component. The value for the criteria can be derived from observations in the world and for example a weighed sum can be taken over time. To obtain the observations, for each candidate plan the consequence events of the plan are determined and formed into an observation. Thereafter the consequences of these observations for the criteria can be determined. In the examples shown below the

bridge between changes of the criteria after an observation and the overall value of the criteria are not shown in a formal form for the sake of brevity.

Mission Success. An important criterion is of course the mission success. Within this criterion the objective of the mission plays a central role. In case a certain decision needs to be made, the influence this decision has for the mission success needs to be determined. The criterion involves taking into account several factors. First of all, the probability that the deadline is reachable. Besides that, the probability that the mission succeeds with a specific fleet configuration. The value of the mission success probability is a real number between 0 and 1. A naval domain expert has labeled certain events with an impact value on mission success. This can entail a positive effect or a negative effect. The mission starts with an initial value for success, taking into consideration the assignment and the enemy. In case the situation changes this can lead to a change of the success value. An example of an observation with a negative influence is shown below.

```
if      current_success_value(S:REAL)
 and  belief(ship_left_behind, pos)
then  new_succes_value(S:REAL * 0.8)
```

Safety. Safety is an important criterion as well. When a ship loses propulsion the probability of survival decreases dramatically if left alone. Basically, the probability of survival depends on three factors: (1) the speed with which the task group is sailing; (2) the configuration of own ships, which includes the amount and type of ships, and their relative positions; (3) the threat caused by the enemy, the kind of ships the enemy has, the probability of them attacking the task group, etc.

The safety value influences the evaluation value of possible plans. The duration of a certain safety value determines its weight in the average risk value, so a weighed sum based on time duration is taken. The value during a certain period in time is again derived by means of an initial safety value and events in the external world causing the safety value to increase or decrease. An example rule:

```
if       current_safety_value(S:REAL)
 and   belief(speed_change_from_to(full, slow), pos)
then   new_safety_value(0.5 * S:REAL)
```

Fleet morale. The morale of the men on board of the ships is also important as criterion. Morale is important in the considerations as troops with a good morale are much more likely to win compared to those who do not have a good morale. Troop morale is represented by a real number with a value between 0 and 1 and is determined by events in the world observed by the men. Basically, the men start with a certain morale value and observations of events in the world can cause the level to go up or down, similar to the mission success criterion. One of the negative experiences for morale is the observation of being left behind without protection or seeing others solely left behind:

```
if      current_morale_value(M:REAL)
 and  belief(ship_left_behind, pos)
then  new_morale_value(M:REAL * 0.2)
```

An observation increasing the morale is that of sinking an enemy ship:

```
if      current_morale_value(M:REAL)
and   belief(enemy_ship_eliminated, pos)
and   min(1, M:REAL * 1.6, MIN:REAL)
then  new_morale_value(MIN:REAL)
```

4.3 Strategy Determination

The StrategyDetermination component within the model has two functions: First of all, it determines the conditional plans that are to be used given the current state. Secondly, it provides a strategy for the selection of these plans.

In general, naval plans are generated according to a preferred plan library or in exceptional cases outside of this preferred plan library. The StrategyDetermination component within the model determines which plans are to be used and thereafter forwards these plans to the PlanGeneration component. The StrategyDetermination component determines one of three modes of operation on which conditional rules are to be used in this situation:

1. **Limited action demand.** This mode is used as an initial setting and is a subset of the preferred plan library. It includes the more common actions within the preferred plan library;
2. **Full preferred plan library.** Generate all conditional rules that are allowed according to the preferred plan library. This mode is taken when the limited action mode did not provide a satisfactory solution;
3. **Exceptional action demand**. This strategy is used in exceptional cases, and only in case the two other modes did not result in an appropriate candidate plan.

Next to determining which plans should be evaluated, the StrategyDetermination component also determines *how* these plans should be evaluated. In Section 4.3 it was stated that the plan selection depends on mission success, safety, and fleet morale. All three factors determine the overall evaluation of a plan to a certain degree. Plans can be evaluated by means of an evaluation formula, which is described by a weighted sum. Differences in weights determine differences in plan evaluation strategy. The plan evaluation formula is as follows (in short):

```
evaluation_value(P:PLAN) =
   (α x mission_success_value(P:PLAN)) + (β x safety_value(P:PLAN)) + (γ x fleet_morale_value(P:PLAN))
```

where all values and degrees are in the interval [0,1], and $\alpha + \beta + \gamma = 1$. The degrees depend on the type of mission and the current state of the process. For instance, if a mission is supposed to be executed safely at all cost or the situation shows that already many ships have been lost, the degree β should be relatively high.

In case of equally important criteria the following rule holds:

```
if      problem_type(mission_success_important)
 and   problem_type(safety_important)
 and   problem_type(fleet_morale_important)
 and   candidate_plan(P:PLAN)
 and   mission_success_value(P:PLAN, R1:REAL)
 and   safety_value(P:PLAN, R2:REAL)
```

```
 and   fleet_morale_value(P:PLAN, R3:REAL)
then   evaluation_value(no_propulsion(ship), 0.33 * R1:REAL + 0.33 * R2:REAL + 0.33 *R3:REAL)
```

In case two criteria are most important the following rule holds:

```
if      problem_type(mission_success_important)
 and   problem_type(safety_important)
 and   not problem_type(fleet_morale_important)
 and   candidate_plan(P:PLAN)
 and   mission_success_value(P:PLAN, R1:REAL)
 and   safety_value(P:PLAN, R2:REAL)
 and   fleet_morale_value(P:PLAN, R3:REAL)
then   evaluation_value(no_propulsion(ship), 0.45 * R1:REAL + 0.45 * R2:REAL + 0.1 *R3:REAL)
```

This holds for each of the problem type combinations where two criteria are important: A weight of 0.45 in case the criterion is important for the problem type and 0.1 otherwise. Finally, only one criterion can be important:

```
if       problem_type(mission_success_important)
 and    not problem_type(safety_important)
 and   not problem_type(fleet_morale_important)
 and   candidate_plan(P:PLAN)
 and   mission_success_value(P:PLAN, R1:REAL)
 and   safety_value(P:PLAN, R2:REAL)
 and   fleet_morale_value(P:PLAN, R3:REAL)
then   evaluation_value(no_propulsion(ship), 0.6 * R1:REAL + 0.2 * R2:REAL + 0.2 *R3:REAL)
```

The plan generation modes and plan selection degrees presented above can be specified by formal rules which have been omitted for the sake of brevity.

5 Case-studies

This Section presents several case studies which have been formalized using the agent-based model presented in Section 2 and 4. These case studies are again based upon interviews with expert navy officers of the Royal Netherlands Navy. The formalization of this process follows the methodology presented in Section 3. Three case studies are presented: total steam failure, submarine threat, and frigate loss.

5.1 Total Steam Failure

The first scenario used as a case study is called total steam failure. First, the scenario is described, after which the simulation results are presented.

5.1.1 Scenario Description

The scenario used as an example is the first phase within a *total steam failure* scenario. A fleet consisting of 6 frigates (denoted by F1 – F6) and 6 helicopters

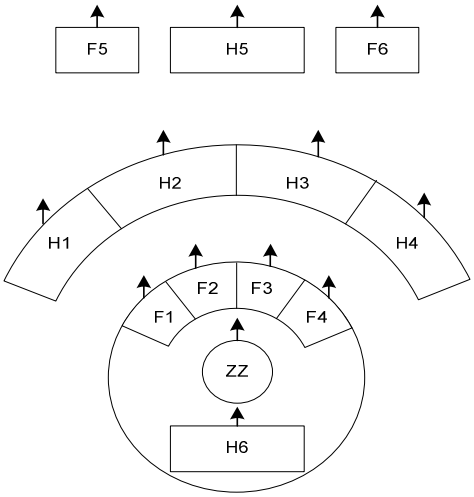

Fig. 4. Scenario for meta-reasoning

(denoted by H1 – H6) are protecting a specific area called Zulu Zulu (denoted by ZZ). For optimal protection of valuable assets that need to be transported to a certain location, and need to arrive before a certain deadline, the ships carrying these assets are located in ZZ. These ships should always maintain their position in ZZ to guarantee optimal protection. The formation at time T0 is shown in Figure 4. On that same time-point the following incident occurs: An amphibious transport ship, that is part of ZZ, loses its propulsion and cannot start the engines within a few minutes. When a mission is assigned to a commander of the task group (CTG), he receives a preferred plan library from the higher echelon. This library gives an exhaustive list of situations and plans that are allowed to be executed within that situation. Therefore the CTG has to make a decision: What to do with the ship and the rest of the fleet. In the situation occurring in the example scenario the preferred plan library consists of four plans:

1. **Continue sailing.** Leave the ship behind. The safety of the main fleet will therefore be maximal, however the risk for the ship is high. The morale of all the men within the fleet will drop.
2. **Stop the entire fleet.** Stopping the fleet ensures that the ship is not left behind and lost, however the risks for the other ships increase rapidly as an attack is more likely to be successful when not moving.
3. **Return home without the ship.** Rescue the majority of the men from the ship, return home, but leave a minimal crew on the ship that will still be able to fix the ship. The ship will remain in danger until it is repaired and the mission is surely not going to succeed. The morale of the men will drop to a minimal level. This option is purely hypothetical according to the experts.
4. **Form a screen around the ship.** This option means that part of the screen of the main fleet is allocated to form a screen around the ship. Therefore the ship is protected and the risks for the rest of the fleet stay acceptable.

Option 4 involves a lot more organizational change compared to the other options and is therefore considered after the first three options. The CTG decides to form a screen around the ship

5.1.2 Simulation Results

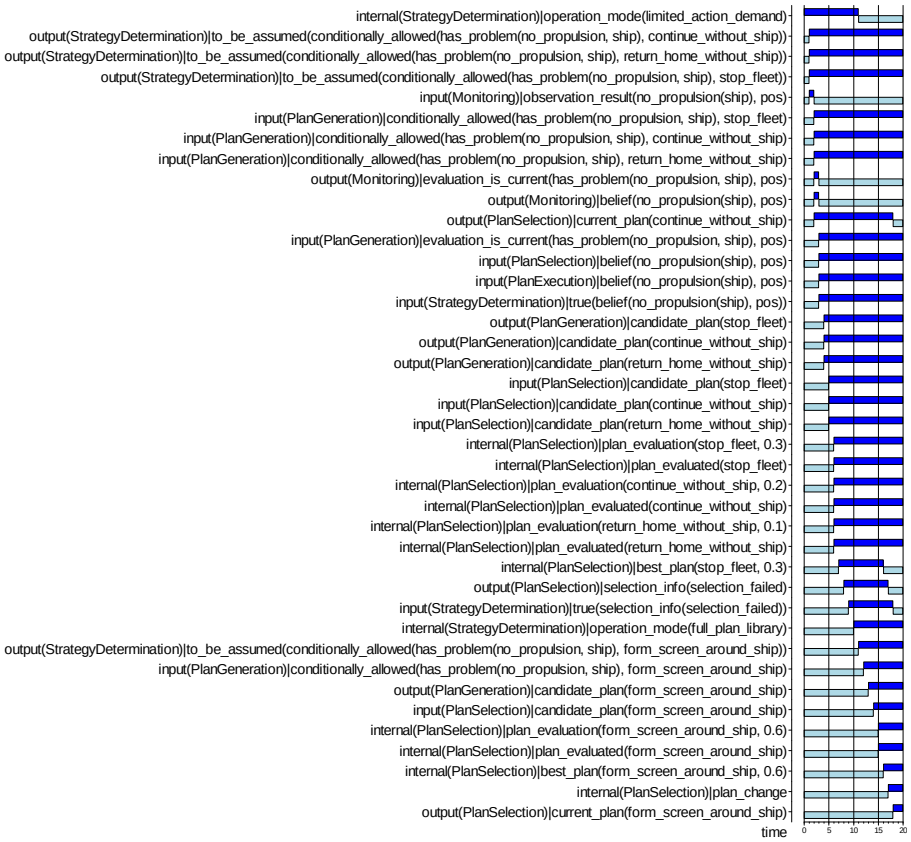

Fig. 5. Trace of the total steam failure simulation

The most interesting results of the simulation using the architecture and properties described in Section 2 and 4, and instantiated with the case-study specific knowledge from Section 5.1 are shown in Figure 5. The trace, a temporal description of chains of events, describes the decision making process of the agent which plays the role of Commander Task Group (CTG). The atoms on the left side denote the information between and within the components of the agent. To keep the Figure clear only the atoms of the components on the lowest level of the agent architecture are shown. The right side of the figure shows when these atoms are true. In case of a black box the atom is true during that period, in the other cases the atom is false (closed world

assumption). The atoms used are according to the model presented in Section 2. For example, internal(PlanGeneration) denotes that the atom is internal within the PlanGeneration component. More specifically, the trace shows that at time-point 1 the Monitoring component receives an input that the ship has no propulsion

```
input(Monitoring)|observation_result(no_propulsion(ship), pos)
```

The current plan is to continue without the ship, as the fleet continues to sail without any further instructions:

```
output(PlanSelection)|current_plan(continue_without_ship)
```

As the StrategyDetermination component always outputs the options currently available for all sorts of situations (in this case only a problem with the propulsion of a ship) it continuously outputs the conditionally allowed information in the limited action mode, for example:

```
output(StrategyDetermination)|to_be_assumed(
            conditionally_allowed(has_problem(no_propulsion, ship),continue_without_ship))
```

The information becomes an input through downward reflection, a translation from a meta-level to a lower meta-level:

```
input(PlanGeneration)|conditionally_allowed(
                              has_problem(no_propulsion, ship), continue_without_ship)
```

The Monitoring component forwards the information about the observation to the components on the same level as beliefs. The StrategyDetermination component also receives this information but instead of a belief it arrives as a reflected belief through upward reflection which is a translation of information at a meta-level to a higher meta-level:

```
input(StrategyDetermination)| true(belief(no_propulsion(ship), pos))
```

Besides deriving the beliefs on the observations the Monitoring component also evaluates the situation and passes this as evaluation info to the PlanGenerator.

```
input(PlanGenerator)|evaluation(has_problem(no_propulsion, ship), pos)
```

This information acts as a basis for the PlanGenerator to generate candidate plans, which are sent to the PlanSelection, for example.

```
input(PlanSelection)|candidate_plan(continue_without_ship)
```

Internally the PlanSelection component determines the evaluation value of the different plans, compares them and derives the best plan out of the candidate plans:

```
internal(PlanSelection)|best_plan(stop_fleet, 0.3)
```

This value is below the threshold evaluation value and therefore the PlanSelection component informs the StrategyDetermination component that no plan has been selected:

```
output(PlanSelection)|selection_info(selection_failed)
```

Thereafter the StrategyDetermination component switches to the full preferred plan library and informs PlanGeneration of the new options. PlanGeneration again generates all possible plans and forwards them to PlanSelection. PlanSelection now finds a plan that is evaluated above the threshold and makes that the new current plan.

```
output(PlanSelection)|current_plan(form_screen_around_ship)
```

This plan is forwarded to the PlanExecution and Monitoring components (not shown in the trace) and is executed and monitored.

5.2 Submarine Threat

The second scenario is called submarine threat, and deals with a hostile submarine being detected within the fleet. First, a description of the scenario is given and thereafter simulation results are presented.

5.2.1 Scenario Description

The initial fleet formation and mission for this scenario is identical to the one explained in Section 5.1.1. Another event however occurs that needs to be dealt with. Frigate F1 suddenly detects sonar contact with a high probability that it concerns a hostile submarine. The position of this submarine is such that the assets in Zulu Zulu are within torpedo range of the submarine. The plan library for the CTG in this particular situation is as follows:

1. **Eliminate and turn.** This option consists of two actions: First of all, F1 will fire a torpedo in the direction of the detected submarine. Thereafter, several frigates are sent to eliminate the submarine whereas the remainder of the fleet turns away from the submarine, positioning several frigates between the submarine and Zulu Zulu. This option results in risk for the frigates chasing the submarine whereas the remainder of the fleet remains relatively safe. Morale of the men will go up, and mission success is not so much endangered.
2. **Full attack.** This plan entails a full attack on the submarine with all available resources. Disadvantage is however that Zulu Zulu is no longer protected, and another enemy ship could possibly attack Zulu Zulu. The risk for mission success is therefore high, and morale of the men on board of the ships part of Zulu Zulu will drop, since they are being left behind without protection.
3. **Full throttle.** Accelerate to maximum speed, in order to try and outrun the submarine, zig zag to avoid the submarine getting a lock on one of the ships within Zulu Zulu. Morale of the troops will go down since they know there is a submarine somewhere trying to attack, and mission success will be much lower

as well since the submarine might have the ability to successfully fire torpedo's at Zulu Zulu. Safety is also low.

Option 3 is considered only after the first two have been considered as trying to escape from a submarine is highly dangerous and therefore seriously threatens mission success. Preferred plan is therefore to try and eliminate the submarine. The CTG decides to choose the eliminate and turn plan.

5.2.2 Simulation Results

Fig. 6. Trace of the submarine threat simulation

Figure 6 shows the results of a simulation of the submarine threat scenario. Initially, again the operation mode is set to limited action demand, which results in two plans being outputted by the StrategyDetermination component:

```
output(StrategyDetermination)|to_be_assumed(conditionally_allowed(has_problem(
                                    submarine_detected), ship), eliminated_and_turn)
output(StrategyDetermination)|to_be_assumed(conditionally_allowed(has_problem(
                                    submarine_detected), ship), full_attack)
```

Suddenly, an event occurs which is precisely the event for which these conditional plans are meant, namely that a submarine has been detected by a ship:

```
output(Monitoring)|belief(detected(submarine), pos)
```

As a result the current plan selected to handle the situation is again to continue with the current plan, which is to continue sailing. The PlanGeneration component generates

the currently available plans for handling the event, which it has received from the StrategyDetermination component:

```
output(PlanGeneration)|cadidate_plan(eliminated_and_turn)
output(PlanGeneration)|cadidate_plan(full_attack)
```

This output is received by the PlanSelection component, which starts to evaluate the two available plans. After evaluation, the plan to eliminate and turn is found to be best and is evaluated above the threshold value. As a result, it is selected as the new current plan:

```
output(PlanSelection)|current_plan(elminate_and_turn)
```

As can be seen in the simulation, only two out of three available plans have been evaluated before selecting a new plan. Since the plans being evaluated first are the ones typically best suitable in the situation, this saves a lot of precious evaluation most of the time.

5.3 Frigate Loss

Final scenario which has been investigated is that of a frigate being hit by a submarine torpedo.

5.3.1 Scenario Description

Again, the initial fleet configuration and mission are identical to the description presented in Section 5.1.1. Again, a submarine is detected, for which the CTG decides to send in H3 to eliminate the submarine. The submarine however fires a torpedo which strikes F3 causing it to sink. There are now several options how to continue:

1. **Eliminate and save.** Eliminate the submarine first by reinforcing the current attack units. Thereafter, save the drowning crew of frigate F3. This option maximizes the morale of the troops as they see their colleagues being saved, mission success is however slightly endangered as picking up the drowning crew will result in frigates lying still, which makes them more vulnerable for enemy attacks.
2. **Save crew.** Immediately use all resources to save the crew on board of the sunken ship. In this scenario this is devastating for mission success as the submarine can easily attack the ships within Zulu Zulu. Furthermore, the submarine could even attack the resources that are being used to save the crew of the sunken ship. The safety for the crew of the sunken ship is relatively high whereas the safety for the other ships is low.
3. **Surrender.** Hoist the white flag and surrender to avoid further casualties. Morale will be very low, mission success probability is down to zero, and safety is highly unknown as the crew and assets are now in the hands of the enemy.

Again, options 1 and 2 are first considered before the last option is taken into consideration since surrender is the last option a fleet commander wants to think of.

5.3.2 Simulation Results

Figure 7 shows the simulation results of the Frigate loss scenario. In this particular trace, the α, β, and γ value passed to the PlanSelection component by StrategyDetermination are shown as well. Again, initially the operation mode is set to limited action demand and the accompanying conditional rules for this scenario are passed as well, namely the following:

```
input(PlanGeneration)|conditionally_allowed(has_problem(submarine_attack_hit, ship),
                                                              eleminate_and_save)
input(PlanGeneration)|conditionally_allowed(has_problem(submarine_attack_hit, ship),
                                                                    save_crew)
```

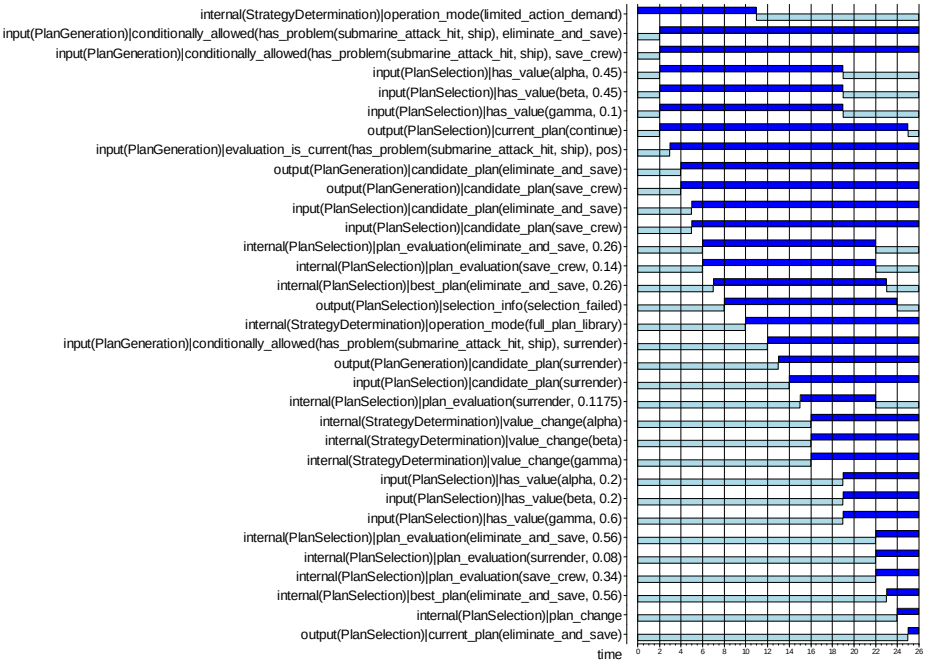

Fig. 7. Trace of the frigate loss scenario

The initial α, β, and γ values passed are respectively 0.45, 0.45, and 0.1:

```
input(PlanSelection)|has_value(alpha, 0.45)
input(PlanSelection)|has_value(beta, 0.45)
input(PlanSelection)|has_value(gamma, 0.1)
```

Denoting that in this case mission success and safety are considered to be more important aspects for plan evaluation than morale. Suddenly the problem of a frigate being hit by an enemy submarine is observed, which is forwarded to the PlanGeneration component:

input(PlanGeneration)|evaluation_is_current(has_problem(submarine_attack_hit), ship), pos)

Based on the detected problem, the two plans that are currently conditionally allowed are generated, and forwarded to PlanSelection:

input(PlanSelection)|candidate_plan(eliminate_and_save)
input(PlanSelection)|candidate_plan(save)

Based on the previously mentioned α, β, and γ values, the component evaluates the candidate plans, and concludes that eliminate and save is the best plan, with an evaluation value of 0.26:

internal(PlanSelection)|best_plan(eliminate_and_save, 0.26)

Since the threshold for plan selection is set to a higher value, namely 0.35, the component outputs that selection has failed for this set. As a result the StrategyDetermination component switches to full plan library mode:

internal(StrategyDetermination)|operation_mode(full_plan_library)

The plans that have been added to the library and which are appropriate for the current situation are again forwarded to PlanSelection which evaluates the new additional plan (surrender) to the even lower value of 0.1175:

internal(PlanSelection)|best_plan(eliminate_and_save, 0.26)

Again, selection has failed, however there are no additional plans available in the exceptional action demand mode. Therefore, the StrategyDetermination component decides to adapt the weights of the parameters, and gives more weight to moral (γ):

input(PlanSelection)|has_value(alpha, 0.2)
input(PlanSelection)|has_value(beta, 0.2)
input(PlanSelection)|has_value(gamma, 0.6)

As a result, the best plan is now eliminate and save which now evaluates above the threshold. Finally, the plan is set to be the current plan.

6 Validation by Verification

After a formalized trace has been obtained, either by formalization of an empirical trace or by means of simulation (such as done in the previous section), in this section it is validated whether the traces comply to certain desired properties for this trace. Below the verification of these properties against the traces are shown. The properties are independent from the specific scenario and should hold for every scenario for which the agent-based meta-level architecture presented in Section 2 and 4 is applied. The properties are formalized using Temporal Trace Language as described in Section 3.

P1: Upward reflection. This property states that information generated at the level of the Monitoring and PlanSelection components should always be reflected upwards to the level of the StrategyDetermination component. In semi-formal notation:

At any point in time t,
if Monitoring outputs a belief about the world at time t
then at a later point in time t2 StrategyDetermination receives this information through upward reflection
At any point in time t,
if PlanSelection outputs selection info at time t
then at a later point in time t2 StrategyDetermination receives this information though upward reflection.

In formal form the property is as follows:

$\forall t$ [[$\forall$O:OBS, S:SIGN [state(γ, t, output(Monitoring)) |= belief(O, S)
$\Rightarrow$ $\exists t2 \geq t$ state(γ, t2, input(StrategyDetermination)) |= true(belief(O,S))]]
& [$\forall$SI:SEL_INFO [state(γ, t, output(PlanSelection)) |= selection_info(SI)
$\Rightarrow$ $\exists t2 \geq t$ state(γ, t2, input(StrategyDetermination)) |= true(selection_info(SI))]]]

This property has been automatically checked and thus shown to be satisfied within the traces.

P2: Downward reflection. Property P2 verifies that all information generated by the StrategyDetermination component for a lower meta-level is made available at that level through downward reflection. In formal form:

$\forall$t, S:SITUATION, P:PLAN [state(γ, t, output(StrategyDetermination))
|= to_be_assumed(conditionally_allowed(S, P))
$\Rightarrow$ $\exists t2 \geq t$ state(γ, t2, input(PlanGeneration)) |= conditionally_allowed(S, P)]

This property is also satisfied for the given traces.

P3: Extreme measures. This property states that measures that are not part of the preferred plan library (extreme measures) are only taken in case some other options failed. In formal form:

$\forall t, t2 > t$, S:SITUATION, P1:PLAN, P2:PLAN
[[state(γ, t, output(Monitoring)) |= evaluation(exception(S), pos) & state(γ, t, output(PlanSelection)) |=
current_plan(P1) & state(γ, t2, output(PlanSelection)) |= current_plan(P2) & P1 $\neq$ P2
& $\neg$state(γ, t2, internal(StrategyDetermination)) |= to_be_assumed(preferred_plan(S, P2)]
$\Rightarrow$ $\exists$t' [t' $\geq$ t & t' $\leq$ t2 & state(γ, t', output(PlanSelection)) |= selection_info(selection_failed)]]

The property is satisfied for the given traces.

P4: Plans are changed only if an exception was encountered. Property P4 formally describes that a plan is only changed in case there has been an exception that triggered this change. Formal:

$\forall t, t2 \geq t$, P:PLAN [[state(γ, t, output(PlanSelection)) |= current_plan(P) &
$\neg$state(γ, t2, output(PlanSelection)) |= current_plan(P)] $\Rightarrow$ $\exists$t', S:SITUATION [t' $\geq$ t & t' $\leq$ t2 &
state(γ, t', output(Monitoring)) |= evaluation(exception(S), pos)]]

This property is again satisfied for the given traces.

7 Discussion

This paper presents an agent-based architecture for strategic planning (cf. [15]) for naval domains. The architecture was designed as a meta-level architecture (cf. [10]) with three levels. The interaction between the levels in this paper is modeled by reflection principles (e.g., [1]). The dynamics of the architecture is based on a multi-level trace approach as an extension of what is described in [6]. The architecture has been instantiated with naval strategic planning knowledge. The resulting executable model has been used to perform a number of simulation runs. To evaluate the simulation results desired properties for the planning decision process have been identified, formalized, and then validated for the simulation traces.

A meta-level architecture for strategic reasoning in another area, namely that of design processes is described in [4]. This architecture has been used as a source of inspiration for the current architecture for strategic planning. In other architectures, such as in PRS [5], meta-level knowledge is also part of the system, however this knowledge is not explicitly part of the architecture (it is part of the Knowledge Areas) as is the case in the architecture presented in this paper.

Agent models of military decision making have been investigated before. In [14] for example an agent based model is presented that mimics the decision process of an experienced military decision maker. Potential decisions are evaluated by checking if they are good for the current goals. A case study of decisions to be made at an amphibian landing mission is used. The outcome of the evaluations of the decisions that can be made in the case-study are compared to the decisions made by real military commanders. The approach presented is different from the approach taken in this paper as a more formal approach is taken here to evaluate the model created. Also the focus in this paper is more on the model of the decision maker itself and not on the correctness of the decisions, which is the case in [14]. The main advantage of the approach taken is that the system is specified and can be simulated on a conceptual level contrary to other approaches. Furthermore for knowledge intensive domains, such as the naval domain, there is the problem of scalability. We acknowledge this and suggest further research for different domains and variants. It is possible for instance to add or change the described criteria or apply particular planning algorithms. Finally, this paper addressed resource-bounded situations. In [12] an overview is presented of models for human behavior that can be used for simulations. Similar to research done in other agent-based systems using the DESIRE framework [2], future research in simulation and the validation of relevant properties for the resulting simulation traces is expected to give key insight for the implementation of future complex resource-bounded agent-based planning support systems used by commanders on naval platforms.

Acknowledgments

CAMS-Force Vision, a software development company associated with the Royal Netherlands Navy, funded this research and provided domain knowledge. The authors especially want to thank Jaap de Boer (CAMS-Force Vision) for his expert knowledge.

References

[1] Bowen, K. and Kowalski, R., Amalgamating language and meta-language in logic programming. In: K. Clark, S. Tarnlund (eds.), Logic programming. Academic Press, 1982.

[2] Brazier, F.M.T., Dunin Keplicz, B., Jennings, N., and Treur, J., DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems*, vol. 6, 1997, pp. 67-94.

[3] Brazier, F.M.T., Jonker, C.M., and Treur, J., Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal*, vol. 14, 2000, pp. 491-538.

[4] Brazier, F.M.T., Langen, P.H.G. van, and Treur, J., Strategic Knowledge in Design: a Compositional Approach. Knowledge-based Systems, vol. 11, 1998 (Special Issue on Strategic Knowledge and Concept Formation, K. Hori, ed.), pp. 405-416.

[5] Georgeff, M. P., and Ingrand, F. F., Decision-making in an embedded reasoning system. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pages 972-978, Detroit, MI, 1989.

[6] Hoek, W. van der, Meyer, J.-J.Ch., and Treur, J., Formal Semantics of Meta-Level Architectures: Temporal Epistemic Reflection. International Journal of Intelligent Systems, vol. 18, 2003, pp. 1293-1318.

[7] Jonker, C.M., and Treur, J., A Compositional Process Control Model and its Application to Biochemical Processes. Applied Artificial Intelligence Journal, vol. 16, 2002, pp. 51-71.

[8] Jonker, C.M., and Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.

[9] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. Cognitive Systems Research Journal, vol. 4, 2003, pp. 191-210.

[10] Maes, P, Nardi, D. (eds), Meta-level architectures and reflection, Elsevier Science Publishers, 1988.

[11] Mulder, M, Treur, J., and Fisher, M., Agent Modelling in MetateM and DESIRE. In: M.P. Singh, A.S. Rao, M.J. Wooldridge (eds.), *Intelligent Agents IV, Proc. Fourth International Workshop on Agent Theories, Architectures and Languages, ATAL'97.* Lecture Notes in AI, vol. 1365, Springer Verlag, 1998, pp. 193-207.

[12] Pew, R.W. and Mavor, A.S.. Modeling Human and Organizational Behavior, National Academy Press, Washington, D.C. 1999.

[13] Shehory, O., and Sturm, A., Evaluation of modeling techniques for agent-based systems, In: Proceedings of the fifth international conference on Autonomous agents, Montreal, Canada, May 2001, pp. 624-631.

[14] Sokolowski, J., Enhanced Military Decision Modeling Using a MultiAgent System Approach, In Proceedings of the Twelfth Conference on Behavior Representation in Modeling and Simulation, Scottsdale, AZ., May 12-15, 2003, pp. 179-186.

[15] Wilkins, D.E., Domain-independent planning Representation and plan generation. Artificial Intelligence 22 (1984), pp. 269-301.

Chapter 4

Redesign of Organizations as a Basis for Organizational Change

Part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., and Treur, J., Redesign of Organizations as a Basis for Organizational Change. In: Boella, G., Boissier, O., Matson, E., and Vazquez-Salceda, J. (eds.), *Proceedings of the Workshop on Coordination, Organization, Institutions, and Norms in Agent Systems (COIN @ ECAI 2006)*, 2006, pp. 38-46. (Also to appear in Springer LNAI post-proceedings).

Furthermore, part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., and Treur, J., Simulating Organizational Change Triggered by a Changing Environment. In: Borutzky, W., Orsoni, A., Zobel, R. (eds.) *Proceedings of the 20th European Conference on Modelling and Simulation (ECMS 2006)*, 2006, pp. 532-539.

An extended abstract of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., and Treur, J., Redesign of Organizations as a Basis for Organizational Change (Extended Abstract). In: *Poster Abstracts of the Second International Conference on Design Computing and Cognition (DCC '06)*, 2006, pp. 7-8.

Redesign of Organizations as a Basis for Organizational Change

Mark Hoogendoorn[1], Catholijn M. Jonker[2], and Jan Treur[1]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
email: {mhoogen, treur}@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands,
email: C.Jonker@nici.ru.nl

Abstract. Artificial Intelligence has contributed (formal) design models and software support tools to application areas such as architecture, engineering and software design This paper explores the effectiveness of applying design models to the area of organization (re)design. To that purpose a component-based model for (re)design of organizations is presented as a specialization of an existing generic design model. Using recently developed formalizations within Organization Theory organization models are described as design object descriptions, and organization goals as design requirements. A formal design process description is presented that models the redesign process for an organization that adapts to changes in the environment. The formally specified and implemented approach to organization redesign thus obtained has been tested for a well-known historical case study from the Organization Theory literature.

1 Introduction

Organizations are created to smoothen processes in all aspects of society, even in the artificial societies of software agents. From a design perspective organizations have goals to be achieved or maintained that serve as requirements for their functioning. The behavior of the elements or parts of the organization and their interaction together should result in overall organization behavior that fulfills the goals of the organization. Environmental circumstances impose constraints on the organization with respect to the way its goals can be fulfilled. As the environment changes over time, so do these constraints. To adapt to such changes in constraints, the organization might have to change itself. From a design perspective the changing constraints can be interpreted as changing requirements to a re-organization problem.

Within the area of AI and Design, in the last decade formally specified generic models for (re)design processes have been developed; e.g., [2, 4]. Application of a generic redesign model to the area of organizations requires specialized knowledge on: (1) organization goals; (2) how to derive refined requirements from such goals given a variable environment; (3) the current design object description, and (4) what

components for a design object satisfy which requirements. A redesign process results in a new design object description as a modification of the existing one and a specification of changed (new) design requirements.

The redesign process as formally modeled in [4] involves generation and modification steps for the specification of the requirement set and for the design object description. A formal model of a redesign process thus requires formalizations of design objects, design requirements, and of the dynamics of redesign processes. This paper proposes such formalizations for the area of organizational (re)design, in the context of a component based model for (re)design of organizations. Formalized organization models [5,10,11,14,18] serve as design object descriptions. Formalizations of organizational behavior are used for design requirements specifications [10,11,14,18]. Finally, for design process dynamics a formalization is used as put forward in [2]. The resulting formal approach contributes to the domain of organization redesign in that it facilitates formal modeling, simulation and verification of the redesign process. The approach is supported by tools to model and analyze such redesign processes.

Section 2 gives the components based model for the design and redesign process and describes the types of domain specific knowledge needed in such a process. Section 3 addresses the formalization of design object descriptions by means of an organization model format in which different components and aggregation levels can be distinguished. In Section 4 the relation between goals, a changing environment and requirements is described, including example cases described in Organization Theory. Section 5 presents the method of refinement of such requirements and shows a specific example. Thereafter, Section 6 presents examples of design object that are known to satisfy certain design requirements, and Section 7 presents generic properties which enable an evaluation of the successfulness of the whole (re)design process. Section 8 presents simulation results of the model whereas Section 9 verification of these simulation results is addressed. Finally Section 10 is a discussion.

2 A Component-Based model for (re)design of organizations

This Section presents a component-based generic model for design of organizations based on requirements manipulation and design object description manipulation. The component-based model presented draws inspiration from [4] and was specified within the DESIRE [3] framework. The model for design is composed of three components, see Figure 1:

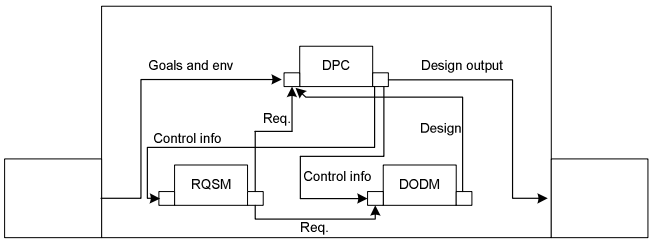

Fig. 1. Top level of the design model

- RQSM, which stands for Requirement Qualification Set Manipulation. Such requirements are for example acquired by elicitation in cooperation with managers within a company. Within RQSM the appropriate requirements are determined in relation to the goals set for the organization and the current environmental conditions. After having selected a set of requirements, these are refined to more specific ones.
- DODM, for Design Object Description Manipulation, creates a design object description based on the (specific) requirements received from RQSM. In order to determine such a design object description, a number of alternative solutions known to satisfy the requirements are generated and according to certain strategic knowledge one of those is selected.
- Design Process Coordination (DPC) is the coordinating component for the design process. The component determines the global design strategy (e.g., [4]) and can evaluate whether the design process is proceeding according to plan.

Information exchange possibilities are represented by the links between input and output of the components and the input and output of the model. Input and output are represented by the small boxes left and right of components.

The next sections describe the three components in more detail. The model as described here, is a generic design model for organizational design without application- or domain-specific knowledge. In later sections such knowledge is specified for a case study.

2.1 RQSM

The component RQSM is composed from two sub-components, namely Requirements Sets Generation and Requirements Set Selection, see Figure 2.

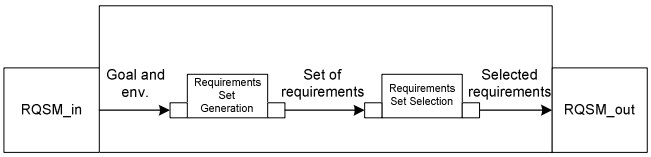

Fig. 2. Components within RQSM

The component Requirements Sets Generation receives as an input the current environmental conditions and the organizational goals. The sub-component contains knowledge on what requirements entail fulfillment of organizational goals given the environmental conditions. Such knowledge can be depicted in the form of AND/OR trees as shown in Figure 3.

If for example E1 is observed, requirement R1 is an example of a requirement that, when fulfilled, guarantees to satisfy goal G under environmental conditions E1. If the environment changes to situation E2, the requirement has to change as well; the example tree shows how R1 can be changed to requirement R2 that guarantees G under the new environmental conditions E2. After a requirement is determined, it can

be refined in order to obtain requirements on a more specific level. Making such a requirement more specific can result in several options being generated. For example, it might be possible to establish a certain market share by having the best quality products but also by having the lowest priced products. After having refined each of the requirements, all possible sets of refined requirements are forwarded to the component Requirements Set Selection.

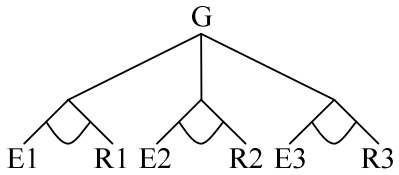

Fig. 3. Example AND/OR tree relating environmental conditions and requirements to a goal

After the component Requirements Set Selection has received the alternative sets of requirements its task is to select one of those alternatives, and to forward it to the component DODM which will in turn find a suitable organization design for such a requirement set. Different selection methods exist, e.g., explicit ranking, on the basis of strategic knowledge. Such strategic knowledge can for example be based on the source of requirements: requirements that originate from users can for example be preferred over those derived by default rules which are in turn preferred over requirements derived from previous requirements (see [12]).

2.2 DODM

DODM receives a set of refined requirements from RQSM, which is handled by two sub-components, Design Object Description Generation and Design Object Description Selection. The design object descriptions are descriptions of designs of the organization, including both structural aspects as behavioral aspects.

Design Object Description Generation receives the requirements and delivers descriptions of possible alternative design objects (i.e., organization design descriptions), such that the (specific) requirements as received from RQSM are satisfied. To establish satisfaction, knowledge is needed that specifies what part of a design object contributes to fulfillment of a specific requirement. If, for example, the requirement is to produce products of the highest quality, then a satisfactory design is an organization having a department dedicated to checking quality and repairing of production errors. Again, there can be many possibilities available that satisfy the requirements. All alternatives found are forwarded to the component Design Object Description Selection.

The component Design Object Description Selection can use several criteria to choose the optimal design, such as operational costs effectiveness, and production time effectiveness. In order to make such a selection, the component has (strategic) knowledge concerning these aspects. It might for example know the typical price for hiring an agent for a particular role Eventually, the component outputs a new design for the organization.

2.3 DPC

The component DPC is the component which determines the global design strategy and oversees whether the design process proceeds according to plan. Two different tasks are distinguished. DPC checks whether a design object description determined by DODM satisfies the refined requirements. It might for example be the case that the combination of two suitable design object parts causes a conflict. In case the refined requirements are not satisfied control information is passed to DODM stating that an alternative should be found (e.g., taking a different branch of an OR tree). In case these refined requirements are satisfied whereas the high-level requirements are not, the requirements refining process has failed, therefore control information is given to RQSM to refine the requirements in another way (again by for example taking another OR branch).

3 Organization Models as Design Objects

An organizational structure defines different elements in an organization and relations between them. The dynamics of these different elements can be characterized by sets of dynamic properties. An organizational structure has the aim to keep the overall dynamics of the organization manageable; therefore the structural relations between the different elements within the organizational structure have to impose relationships or dependencies between their dynamics; cf. [18]. In the introduction to their book Lomi and Larsen [20] emphasize the importance of such relationships:

- 'given a set of assumptions about (different forms of) individual behavior, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?'
- 'given observable regularities in the behavior of a composite system, which rules and procedures - if adopted by the individual units- induce and sustain these regularities?'

Both views and problems require means to express relationships between dynamics of different elements and different levels of aggregation within an organization. In [20] two levels are mentioned: the level of the organization as a whole versus the level of the units. Also in the development of MOISE [11,12,14] an emphasis is put on relating dynamics to structure. Within MOISE dynamics is described at the level of units by the goals, actions, plans and resources allocated to roles to obtain the organization's task as a whole. Specification of the task as a whole may involve achieving a final (goal) state, or an ongoing process (maintenance goals) and an associated plan specification.

The approach in this paper is illustrated for the AGR [9] organization modeling approach. Figure 4 shows an example

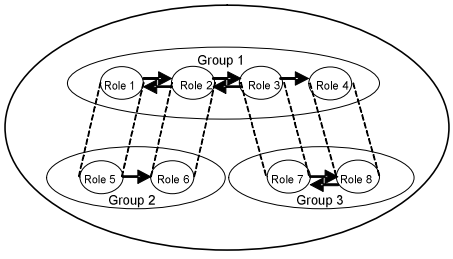

Fig. 4. An AGR Organization Structure

organization modeled using AGR. Within AGR organization models three aggregation levels are distinguished: (1) the organization as a whole; the highest aggregation level, denoted by the big oval, (2) the level of a group denoted by the middle size ovals, and (3) the level of a role within a group denoted by the smallest ovals. Solid arrows denote transfer between roles within a group; dashed lines denote inter-group interactions. This format is adopted to formalize organization models as design object descriptions. In addition, behavioral properties of elements of an organization are part of a design object description. TTL [17] is used to express such behavioral properties.

In TTL state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t $(t \in T)$ in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. A special software environment has been developed for TTL, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against a set of (simulated or empirical) traces.

4 RQSM: Changing Requirements upon Environmental Change

Organizational requirements change due to changing environmental circumstances. The circumstances are input to RQSM. The general pattern is follows. A certain organizational goal G (e.g. sufficient demand) is no longer reached, due to an environmental change, say from E1 to E2. In the old situation requirement R1 was sufficient to guarantee G under environmental condition E1: E1 & R1 $\Rightarrow$ G. Here R1 is a requirement expressing a relation which states that under the condition E1 the organization is able to achieve G. The change from E1 to E2 makes that requirement R1, which is still fulfilled but has become insufficient, is to be replaced by a new, stronger requirement R2 which expresses that under environment E2 goal G can be

achieved; therefore: E2 & R2 $\Rightarrow$ G. Thus, the organization is triggered to change to fulfill R2 and as a consequence fulfill goal G again.

Jaffee [16] distinguishes several of these external triggers for organizational change. This paper presents a classification (see Figure 5) of those triggers based on the flow of information for an organization. The input type of external trigger includes the triggers the organization notices on its input, for example changes in the resources or suppliers. *Enabling / constraining factors* are external triggers such as government rules and technology that concern processes within the organization. Finally, *output* can influence the input of an organization and can therefore affect the triggers received by an organization. Output information itself is however not considered a trigger for organizational change.

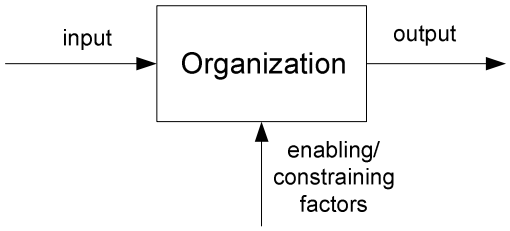

Fig. 5. Flow of information in an organization

4.1 Input Changes

The input of an organization can originate from a variety of different sources. Each of these sources can cause a change of requirements, and possibly trigger an organization to change.

A first source is formed by the *suppliers* who can increase their price of a product P, which is used by the organization for the production, at time t from M_1 to M_2. A formal form of this environmental condition is specified in E1 using the Temporal Trace Language (TTL) as explained in Section 3.

E1(P, M, t): Supplier Price
∃R:REAL state(γ, t) |= environmental_condition(price(P, R), pos) & R ≤ M

Before the environmental change, E1(P1, M_1, t) specifies the relevant property of the environment. After the change of supplier price however, this property no longer holds whereas E1(P1, M_2, t) does hold. The overall goal to be maintained within the organization is to keep the demand of product P above a threshold D. A formal specification of the goal is presented in OP1.

OP1(P, D, t): Sufficient demand
∃I:INTEGER
state(γ, t) |= environmental_condition(customer_demand(P, I), pos) & I ≥ D

The requirement imposed for the organization is to maintain the goal of keeping demand for product P2 above D, in the new situation given the environmental condition of the price M for product P1 which is needed for the production of P2. This requirement is specified below in property R.

R(P1, P2, M, D): Maintain demand
∀t :TIME
[state(γ, t) |= needed_for_production_of(P1, P2) & E1(P1, M, t)] $\Rightarrow$ OP1(P2, D, t)

Before the change in the environment, requirement R1 which is R(P1, P2, M_1, D) was sufficient to ensure the goal being reached. After the change however, this requirement is still satisfied but might be insufficient to ensure the goal. This is due to the fact that the environmental condition E1 in the antecedent of E1 & R1 $\Rightarrow$ G does not hold, and hence, cannot be used to entail G (although the requirement R1 is fulfilled all the time). The requirement is therefore withdrawn and replaced by the requirement R2 which is R(P1, P2, M_2, D). This R2, however, is not necessarily satisfied and may require an organizational change to enable fulfillment.

Secondly, an input trigger can be formed by *resources* that run out, becoming a lot more expensive. Therefore, the requirement for an organization triggered in such a way is to reduce the usage of the particular resource. This can for example be accomplished by focusing on a completely different, more viable product, or producing the same goods using different resources.

Another source is formed by the *customers* whose demands decreases for the good being produced. The organization can change direction (and thus change the organization) or keep producing the same good but decrease the output (and therefore also change the organization).

Finally, *competitors* might change their production methods causing a more efficient production process for products within the same product group as P, lowering their price from C_1 to C_2.

4.2 Changes in Enabling / Constraining Factors

Besides triggers on the input of an organization, another type of trigger exists: the enabling and constraining factors. First of all, the enabling factors within the organization include *technology*. In case the technology available to produce a product P changes from T1 to T2, the profit margin should remain at least at the same level D for a company.

OP'(P, D, t): Sufficient Profit Margin
$\exists$R:REAL state(γ, t) |= belief(profit_margin(P, R), pos) & R $\geq$ D

E'(P, T, t): New Technology
$\exists$R:REAL state(γ, t) |= environmental_condition(technology_available_for(T, P), pos)

R'(P, T, D): Maintain Profit
$\forall$t :TIME E3(P, T, t) $\Rightarrow$ OP1(P, D, t)

All properties have been specified similar to those presented in the previous subsection. Before the environmental change of available technology E'(P, T1, t) was the case whereas E'(P, T2, t) is the new environment. Secondly, constraining forces include *government regulations and labor aspects*. Government regulations for workers might affect human resource practices and composition of the workforce. Concerning labor aspects, the union might demand a reduction from 40 to 36 hours a week, which naturally causes organizational change. All these aspects should however not decrease overall profitability of the organization.

5 RQSM: Refining Requirements Based on Interlevel Relations

To fulfill requirements at the level of the organization as a whole as discussed in Section 4, parts of the organization need to behave adequately (see also the central challenges put forward by Lomi and Larsen [20] as discussed in Section 2). Based on this idea, in this paper dynamics of an organization are characterized by sets of dynamic properties for the respective elements and aggregation levels of the organization. An important issue is how organizational structure (the design object description determined in DODM) relates to (mathematically defined) relationships between these sets of dynamic properties for the different elements and aggregation levels within an organization (cf. [18]). Preferably such relations between sets of dynamic properties would be of a logical nature; this would allow the use of logical methods to analyze, verify and validate organization behavior in relation to organization structure. Indeed, following [18], in the approach presented below, logical relationships between sets of dynamic properties of elements in an organization turn out an adequate manner to (mathematically) express such dynamic cross-element or cross-level relationships.

A general pattern for the dynamics in the organization as a whole in relation to the dynamics in groups is as follows:

dynamic properties for the groups &
dynamic properties for inter-group interaction
dynamic properties for the organization

Moreover, dynamic properties of groups can be related to dynamic properties of roles as follows:

dynamic properties for roles &
dynamic properties for transfer between roles
dynamic properties for a group

The idea is that these are properties dynamically relating a number of roles within one group.

A generic overview of the logical relationships between dynamic properties at different aggregation levels is depicted as an AND-tree in Figure 6. It is possible that each level shown in the tree (for example organization properties) again consists of multiple levels. The logical relationships put forward above can be formalized further as shown in [18].

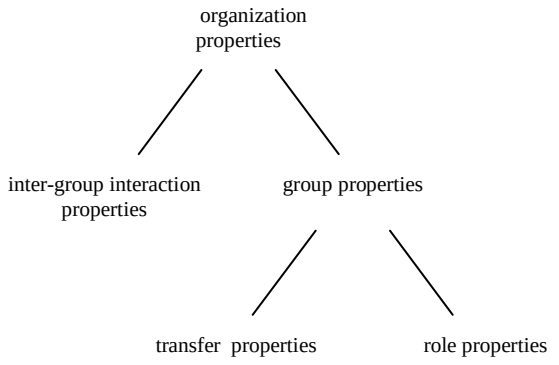

Fig. 6. Overview of relations between dynamic properties

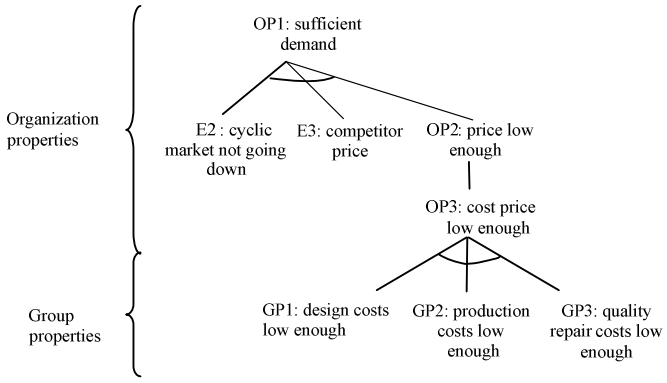

Fig. 7. Hierarchy of Organizational and Group properties

Figure 7 shows an example of a hierarchy of dynamic properties for an organization producing certain products, the properties follow field observations at the Ford Motor Company in 1980 described in [25]. The overall organizational goal is to maintain sufficient demand for the goods being produced, as was also the case in OP1 in Section 4. The organization has separate departments for design, production and quality control, which are modeled as groups in the organization. The highest levels represent organizational properties or goals at the aggregation level of the organization as a whole, whereas the lowest level shown here represents properties at the aggregation level of the groups. Note that the fact that these are group properties already restricts the design of the object in DODM, which makes the process less complex.

A definition for each of the properties in Figure 5 is presented below. Notice that this hierarchy could easily be extended by other aspects (e.g., of quality of the products as a reason for the demand decreasing or not).

Property OP1 is described in Section 4. One of the environmental conditions is that the cyclic market is not going down for a product P at time t in case the demand for the product group as a whole (i.e., all goods produced by different companies in this particular category) is not going down.

E2(P, t): Cyclic market not going down
∀G:PRODUCT_GROUP, I1,I2:INTEGER
[state(γ, t) |= belongs_to_product_group(P, G) &
state(γ, (t-1)) |= environmental_condition(customer_demand(G, I1), pos) &
state(γ, t) |= environmental_condition(customer_demand(G, I2), pos)]
⇒ I2 ≥ I1

Furthermore, an environmental condition E3 poses a requirement on the price of competitors in the form of the average price of products within the product group to which product P belongs. These prices should not be higher than V:

E3(P, V, t): Competitor Price
∀G:PRODUCT_GROUP, V1:REAL
[state(γ, t) |= belongs_to_product_group(P, G) &
state(γ, t) |= environmental_condition(average_price(G,V1), pos) & V1 ≥ V]

To achieve goal OP1 given environmental conditions E2 and E3, the price of the products being produced by the organization should be low enough, which in turn is

the requirement posed on the organization. Prices are considered low enough for a product P at time t in case the price for the product is equal or below the average price level within the product group (i.e. prices are ≤ V as set above).

OP2(P, V, t): Price low enough
∀G:PRODUCT_GROUP, V1:REAL [state(γ, t) |= price(P, V1)] ⇒ V1 ≤ V

Whether the price is low enough depends on the cost price for the particular product P at time t, which purely depends on the costs for the different groups within the organization, as expressed in the group properties (GP's).

OP3(P, V, t): Cost price low enough
∀V1,V2,V3:REAL
[state(γ, t) |= design_cost(P, V1) &
state(γ, t) |= production_cost(P, V2) &
state(γ, t) |= quality_repair_cost(P, V3)]
⇒ V1+V2+V3 ≤ V

Finally, the individual group properties can be specified such that the costs of each group are below a certain value. that the division of such costs over groups is a refinement choice. An example decision could be the to allow only a small percentage of the costs for quality repair and to divide the brunt of the costs equally over production and design. Each group should meet their individual requirements. First of all, design costs should be low enough:

GP1(P, V1, t): Design costs low enough
∀Q:REAL [state(γ, t) |= design_cost(P, Q)] ⇒ Q ≤ V1

Also, the production costs for product P should be low enough:

GP2(P, V2, t): Production costs low enough
∀Q:REAL [state(γ, t) |= production_cost(P, Q)] ⇒ Q ≤ V2

Finally, quality repair costs should be low enough for product P:

GP3(P, V3, t): Quality repair costs low enough
∀Q:REAL [state(γ, t) |= quality_repair_cost(P, Q)] ⇒ Q ≤ V3

After having generated all options in RQSM, selection knowledge is used to select one of the available options. In this paper, such selection knowledge is not further addressed. The output of RQSM is, however, of the form selected_basic_refinement_set(RS) where RS is a name for a requirements set. The elements within this set are defined as follows: in_selected_basic_refinement_set(R, RS) where R is a requirement, as the ones shown above, and RS is the selected basic refinement set.

6 DODM: Constructing Design Objects

As stated in Section 2, DODM contains a library of templates for (parts of) design objects which are known to satisfy certain requirements (of the form as specified in the last paragraph of the previous section). For the case study, the DODM library contains two templates. One of those is a template in which a mass production system is used to produce goods. Such a system produces goods at reasonable production costs but at high quality repair costs. The template for mass production includes a

group of production workers (e.g. a production worker for attaching a wheel to a car). The mass production template also contains a quality repair department of considerable size with quality repair worker roles.

The second template in the library is a lean production organization. Lean production has no quality repair costs, since there is no separate quality repair department. The production costs are at the same level as the production costs for mass production organizations. In the lean production method (see e.g. [25]), multi-task production workers are present which perform several tasks, and also handle errors in case they are observed. As a result of such immediate error detection and correction, a quality repair department is not present within a lean production model.

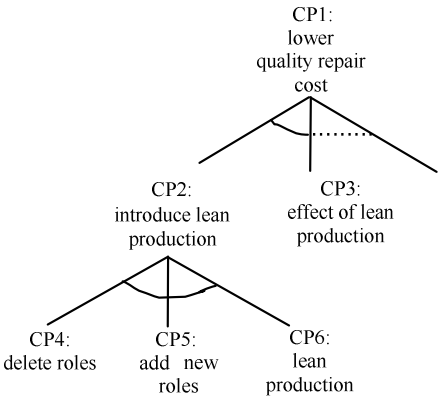

Fig. 8. Redesign options specified in the form of an AND/OR tree

Figure 8 shows an example AND/OR tree for DODM (focusing at lean production as a solution) in which options for changes in a design object not satisfying the requirement that design costs are low enough. The specific changes in the design object are presented below. First of all, the highest level property states that design costs will at least at the required level within a duration d:

CP1(P, D, t):Lower Quality Repair Costs
∀V1,V2:REAL
[state(γ, t) |= selected_basic_requirement_in(GP3(P, V1, t), RS) &
state(γ, t) |= DOD_includes(D, quality_repair_cost(P, V2)) & V1 < V2]
⇒ ∃t2:TIME > t, V3:REAL
[t2 < t+d & state(γ, t2) |=DOD_includes(D,quality_repair_cost(P, V3)) & V3 ≤ < V1]

On a lower level, property CP2(P, D, t) specifies the introduction of lean production into an organization. This reduces the quality repair costs to 0 as shown by CP3(P, D, t). Although more options are possible for reducing quality repair costs, shown by the dots in Figure 8, these are not addressed in this paper.

CP2(P, D, t): Introduce Lean Production
∀V1,V2:REAL
[state(γ, t) |= selected_basic_requirement_in(GP3(P, V1, t), RS) &
state(γ, t) |= DOD_includes(D, design_cost(P, R2)) & V1 < V2]
⇒ ∃t2:TIME > t
[t2 < t + d & state(γ, t2) |= DOD_includes(D, lean_production_method(P))]

CP3(P, D, t): Effect of Lean Production

[state(γ, t) |= DOD_includes(D, lean_production_method(P))
⇒ state(γ, t) |= DOD_includes(D, quality_repair_cost(P, 0))]

Introducing a lean production system entails that within the production process the specialized roles for mass-production and quality repair department are deleted.

CP4(P, D, t): Delete Roles

∀R1,R2:REAL
[state(γ, t) |= DOD_includes(D, lean_production_method(P))
⇒ ∃t2:TIME > t
[t2 < t + d &
state(γ,t2)|=¬DOD_includes(D,exists_role(spec_production_worker)) &
state(γ, t2)|=¬DOD_includes(D,exists_group(quality_repair_group))]]

Moreover, roles are created that perform multiple tasks, and teams are created such that the roles combined in the team have all the abilities to make a car.

CP5(P, D, t): Add New Roles

∀R1,R2:REAL
[state(γ, t) |= DOD_includes(D, lean_production_method(P))
⇒ ∃t2:TIME > t, ∀A:AGENT, R:ROLE
[t2 < t + d &
state(γ, t2) |= DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t2) |= DOD_includes(D, previously_allocated_to(A, R, quality_repair)) &
state(γ, t2) |= DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]]

Agents that were allocated to the deleted roles in the production process are allocated to the newly formed roles. Agents formerly allocated to a role in quality repair are fired. Once the system is organized in this fashion, quality repair in a separate department becomes obsolete, and quality repair costs are down to 0 as the production workers are now performing the task. CP6 expresses that the measures as described in CP4 and CP5 results in a lean production method for the product P:

CP6(P, D, t): Lean Production

∀A:AGENT, R:ROLE
[state(γ, t) |= ¬ DOD_includes(D, exists_role(spec_production_worker)) &
state(γ, t) |= ¬ DOD_includes(D, exists_group(quality_repair_group)) &
state(γ, t) |= DOD_includes(D, exists_role(multi_task_production_worker)) &
state(γ, t) |= DOD_includes(D, previously_allocated_to(A, R, quality_repair))
state(γ, t) |= DOD_includes(D, allocated_to(A, multi_task_production_worker, production_group))]
⇒
∃t2:TIME < t + d
state(γ, t2) |= DOD_includes(D,lean_production_method(P))

After such options for (re)design of the object have been generated based on the requirements, selection knowledge is used to select one of the options that have been generated. This knowledge is not addressed in this paper. Eventually, DODM outputs a design object description of the form selected_DOD_output(D) where D is the design object description. Furthermore to identify properties of the DOD or its parts, output of the form in_selected_DOD_output(P,D) is generated where P is a property of (a part of) the DOD and D is the selected DOD. This is based on the internal information represented in the form of DOD_includes(D, P).

7 (Re)design Process Evaluation

This section addresses the evaluation of the whole design process. The overall design process is successful when both RQSM and DODM show the proper behavior.

RQSM shows the proper behavior in case it generates requirements, and these requirements indeed result in the goal set for the organization being met. Such properties are formulated in a formal form below.

RQSM_generate
If RQSM receives new environmental conditions on its input, then RQSM eventually generates a set of requirements

∀t:TIME, γ:TRACE, E:ENV_COND
 state(γ, t, input(RQSM)) |= environment_property(E) &
 ¬∃t':TIME < t [state(γ, t', input(RQSM)) |= environment_property(E)]
⇒ ∃t2:TIME > t, G:GOAL, RS:REQUIREMENT_SET
 [state(γ, t2, output(RQSM)) |= main_requirement(G) &
 state(γ, t2, output(RQSM)) |= selected_basic_refinement_set(RS)]

RQSM_successful
If RQSM generates requirements, then the combination of these requirements entail the goal set for the organization.

∀t:TIME, γ:TRACE, RS :REQUIREMENT_SET, G :GOAL
 [state(γ, t, output(RQSM)) |= main_requirement(G) &
 state(γ, t, output(RQSM)) |= selected_basic_refinement_set(RS)]
⇒ entails_goal(RS, G)

DODM shows the proper behavior in case it first of all generates a design object description in case a new requirement set is received. Besides simply generating such a design object description, the object also needs to satisfy the requirements received on its input.

DODM_generate
If DODM receives a new requirements set on its input, then DODM eventually generates a design object description as output.

∀t:TIME, γ:TRACE, RS :REQUIREMENTS_SET
[state(γ, t, input(DODM)) |= selected_basic_refinement_set(RS) &
¬∃t':TIME < t
 state(γ, t', input(DODM)) |= selected_basic_refinement_set(RS)]
⇒ ∃t2:TIME, D:DESIGN_OBJECT_DESCRIPTION
 state(γ, t2, output(DODM)) |= selected_DOD_output(D)]

DODM_successful
If DODM generates a design object description as output, then the design object description satisfies the requirements set on the input of DODM.

∀t:TIME, γ:TRACE, R :REQUIREMENT_SET,
 D:DESIGN_OBJECT_DESCRIPTION
 [state(γ, t, input(DODM)) |= selected_basic_refinement_set(R) &
 state(γ, t, output(DODM)) |= selected_DOD_output(D)]
⇒ fulfills_requirements(D, R)

8 Simulation Results

In order to show the functioning of the model, and to validate whether the model indeed behaves correctly, simulation runs have been performed. The results for one of these simulation runs are presented in this Section. The simulation has been performed using a subset of the Temporal Trace Language (TTL) called *leads to*. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties (the leaves of the tree in Figure 6). The format is defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold
for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [1]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

The setup of the simulation is as follows: A historic case taken from [25] is used as an input for the model. The case concerns the Ford Motor Company who has been one of the leading car manufacturers since the introduction of mass production in 1913. In 1980 the Ford Motor Company suffered a major crisis. The company began to loose vast amounts of money a vast amount of car demand. The model presented in this paper is used to reorganize the Ford organization such that demand is restored again.

8.1 Simulation Results: High-Level Overview

First of all, results are presented in this section that abstract from the details of the organization and the internal functioning of the model. This is to show that on this high level the model indeed shows the expected results. In the following sections, more details will be shown regarding the internal functioning of the model.

The results of the simulation in the form of a trace are shown in Figure 9. In the figure, the left side shows the relevant atoms, the right part represents a time-line indicating when an atom is true (dark box) or false (lighter box). It can be observed in the figure that initially the market conditions are equal for the four car manufacturers included in the simulation. First of all, the average costs for design, production, and quality repair are the same:

environmental_condition(design_cost(average, 1000), pos)
design_cost(ford, 1000)
environmental_condition(production_cost(average, 15000), pos)
design_cost(ford, 15000)
environmental_condition(quality_repair_cost(average, 3500), pos)
quality_repair_cost(ford, 3500)

As a result, the price for these cars is the same, also resulting in the same demand for cars from the four manufacturers (it is assumed here that there is no preference of

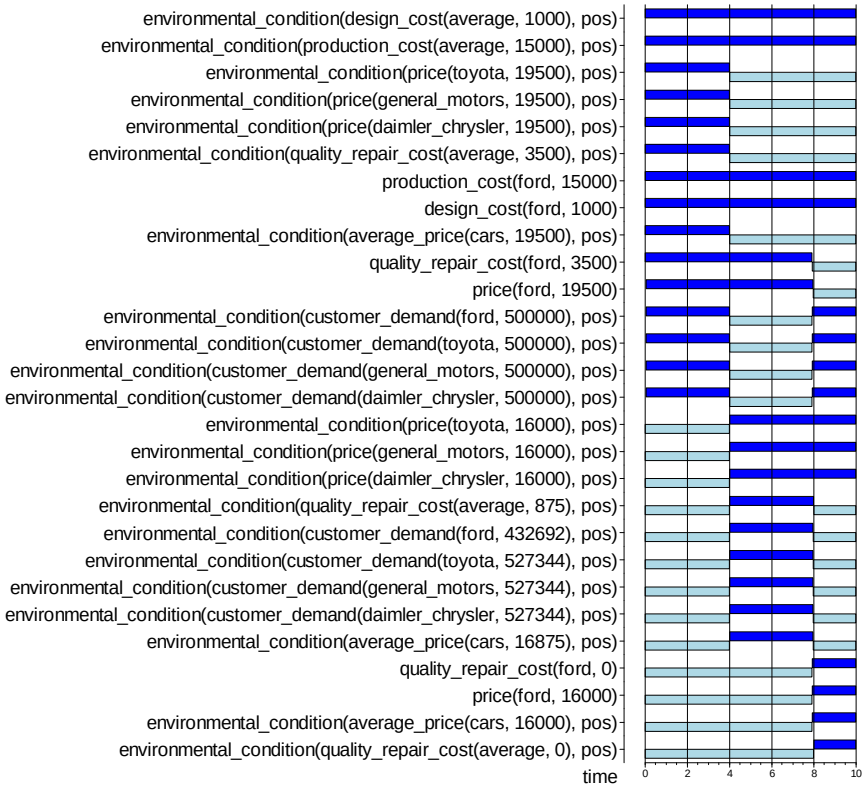

Fig. 9. High-level simulation results using the redesign model

customers for particular brands, if the price is the same, each manufacturer gets an equal share of the total demand).

environmental_condition(customer_demand(ford, 500000), pos)
environmental_condition(customer_demand(general_motors, 500000), pos)
environmental_condition(customer_demand(toyota, 500000), pos)
environmental_condition(customer_demand(daimler_chrysler, 500000), pos)

Suddenly however, at time point 4 the other three manufacturers lower their price whereas Ford does not:

environmental_condition(price(general_motors, 16000), pos)
environmental_condition(price(toyota, 16000), pos)
environmental_condition(price(daimler_chrysler, 16000), pos)

This lowering of the price is performed due to a drop in the cost for quality repair cost of the other companies:

environmental_condition(quality_repair_cost(average, 875), pos)

As a result, demand for Ford cars drops whereas the other manufacturers see an increase in demand:

environmental_condition(customer_demand(ford, 432692), pos)
environmental_condition(customer_demand(general_motors, 527344), pos)
environmental_condition(customer_demand(toyota, 527344), pos)
environmental_condition(customer_demand(daimler_chrysler, 527344), pos)

Now the model introduced in this paper comes into play. The results obtained after application of this model are shown in the figure as well, using the new organization structure brings the quality repair cost of Ford down to 0 as well:

quality_repair_cost(ford, 0)

As a result, demand is restored again to the old value of 500,000 cars. These results indeed correspond to the results described in the historic case.

8.2 RQSM Simulation Results

In order to achieve the result of restoring demand for Ford cars, RSQM and DODM are used to redesign the organization of Ford. In this section, RQSM is addressed. Figure 10 shows the atoms related to the RQSM component. As input, RQSM receives the environmental conditions as shown in the trace of the previous section. The goal of the organization is set to keep demand above or at least equal to 500,000 cars (a quarter of the constant total demand for cars of 2,000,000), which is initially satisfied:

internal(RQSM)|property(OP1(ford, 500000), pos)

The environmental conditions under which the initial Ford organization is obtaining its goal are the following:

internal(RQSM)|property(E2(ford), pos)
internal(RQSM)|property(E3(ford, 19500), pos)

Which means that first of all, the cyclic market is not going down, and secondly, that the competitor prices are not below 19,500. Given these environmental conditions, OP2(ford, 19500) is indeed a sufficient requirement posed upon the organization to guarantee satisfaction of the overall goal. From time point 4 and on however, the environmental condition E3(ford, 19500) no longer holds due to competitors lowering their price. Another condition does however hold:

internal(RQSM)|property(E3(ford, 16000), pos)

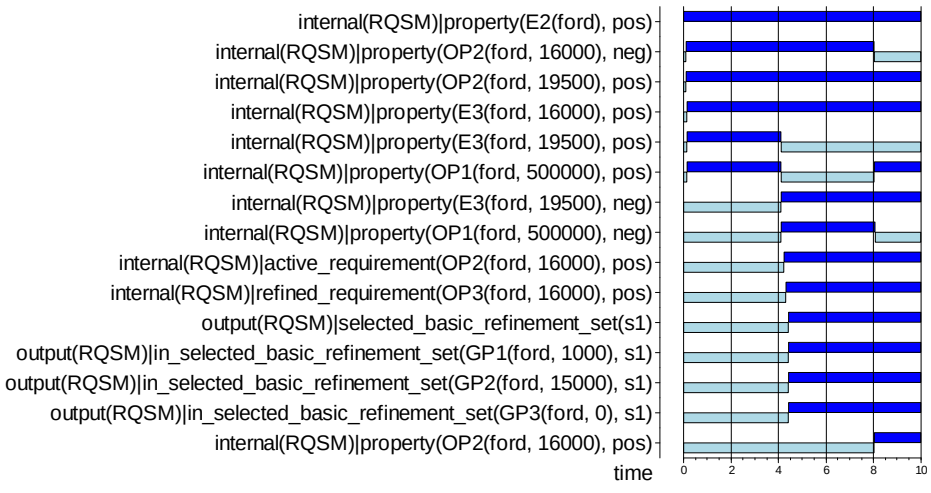

Fig. 10. RQSM reasoning process

Given this new environmental condition, property Op2(ford, 19500) is no longer sufficient to obtain the goal:

internal(RQSM)|property(OP1(ford, 500000), neg)

A new requirement is determined by RQSM that will satisfy the goal under these new environmental conditions:

internal(RQSM)|active_requirement(OP2(ford, 16000), pos)

This requirement is thereafter refined until the level of basic requirements of which a set is sent to the output:

output(RQSM)| selected_basic_requirement(s1)
output(RQSM)|in_selected_basic_refinement_set(GP1(ford, 1000), s1)
output(RQSM)|in_selected_basic_refinement_set(GP2(ford, 15000), s1)
output(RQSM)|in_selected_basic_refinement_set(GP3(ford, 0), s1)

In this case the selected basic refinement includes bringing down the cost of the quality repair cost to 0 whereas the requirements for the rest of the costs (i.e. production and design) remain the same.

8.3 DODM Simulation Results

Figure 11 shows the simulation results for the DODM component. After RQSM has refined and outputted these requirements, DODM receives these on its input. Furthermore, DODM has knowledge about the current organization used by the Ford organization:

DOD_includes(ford_design, exists_group(design_group), pos)
DOD_includes(ford_design, exists_group(production_group), pos)
DOD_includes(ford_design, exists_group(quality_repair_group), pos)
DOD_includes(ford_design,role_belongs_to_group(spec_prod_worker, production_group), pos)

The Ford organization consists of three groups, namely a design group, a production group, and a quality repair group. Furthermore, the production group consists of specialized production workers. In other words, Ford is using a mass production type of company. After having received the basic refinement set, DODM starts to search

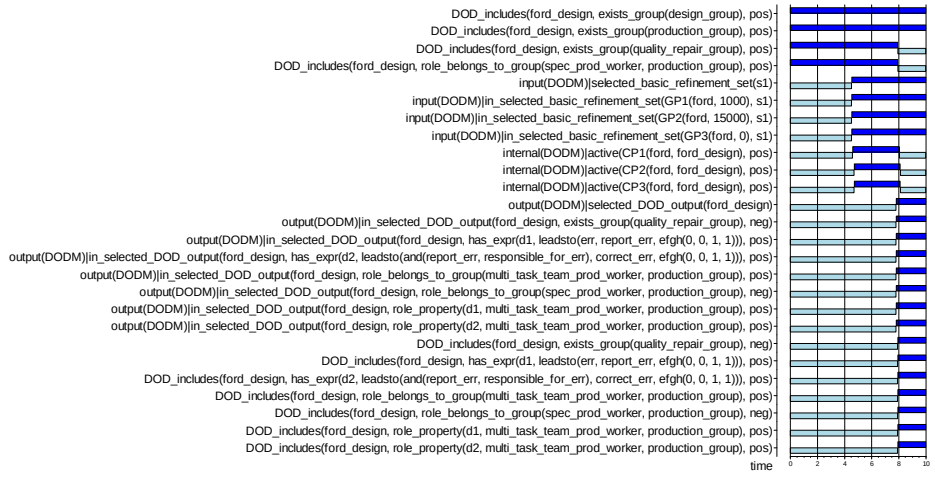

Fig. 11. DODM reasoning process

for an appropriate organization that indeed meets the requirements that have been set. In this case, it first determines that the quality repair cost should go down:

```
internal(DODM)|active(CP1(ford, ford_design), pos)
```

This is further refined to the point of the introduction of lean production within the organization, which is one of the solutions to bring down the quality repair cost to 0:

```
internal(DODM)|active(CP2(ford, ford_design), pos)
internal(DODM)|active(CP3(ford, ford_design), pos)
```

As a result of this choice to introduce lean production, many changes in the current Ford design are sent to the output of DODM. First of all, the quality repair group is deleted:

```
output(DODM)|in_selected_DOD_output(ford_design, exists_group(quality_repair_group),
                                    neg)
```

Furthermore, the specialized production worker role within the production group is deleted as well:

```
output(DODM)|in_selected_DOD_output(ford_design,
                role_belongs_to_group(spec_prod_worker, production_group), neg)
```

As a replacement for the specialized production workers, multi-task production workers are inserted into the organization.

```
output(DODM)|in_selected_DOD_output(ford_design,
                role_belongs_to_group(multi_task_prod_worker, production_group), pos)
```

Of course the behavior of this role is completely different from the behavior of the classical specialized production worker role. Since the approach which is used throughout the paper also allows for the specification of behavior of the roles, this behavior is also present on the output of DODM.

```
output(DODM)|in_selected_DOD_output(ford_design,
                role_property(d1, multi_task_prod_worker, production_group), pos)
output(DODM)|in_selected_DOD_output(ford_design,
                role_property(d2, multi_task_prod_worker, production_group), pos)
```

The actual behavior expected of an agent allocated to such a role is communicated in the form of a *leads to* property as introduced in the beginning of this section.

```
output(DODM)|in_selected_DOD_output(ford_design,
                has_expr(d1, leadsto(err, report_err, efgh(0,0,1,1)), pos)
```

This first property states that if an error is observed this error should be reported immediately. The second role property is specified as follows:

```
output(DODM)|in_selected_DOD_output(ford_design,
                has_expr(d2, leadsto(and(report_err, reposible_for_err),
                                     correct_err, efgh(0,0,1,1)), pos)
```

Stating that if an error is reported, and the worker is responsible for this error, he should correct the error immediately. Both properties are typical for the lean production system. Note that communicating such properties requires properties about properties, i.e. a meta-language in this case called meta-TTL. Finally, after all this has been sent to the output, the actual DOD is updated which eventually results in a restored demand again, as already shown in the high-level trace presented in section 8.1.

9 Verification

To see whether the properties as expressed in Section 7 hold for the simulation trace, first of all, the RQSM_generate and DODM_generate properties have been checked against

the trace shown in Figure 7 using a software tool called the TTL Checker [17]. Both properties were shown to hold for the trace.

In order to see whether the refinement process within RQSM is properly performed, the tree used for the simulation as presented before in Section 5 has been formally proven by means of the SMV model checker [22]. The translation of the properties expressed in Section 5 to the input language of SMV is not trivial. In order to improve the efficiency of the checking process, the numbers as introduced in the case study above have been divided by 1000. In order to verify whether the property hierarchy is indeed correct, the property hierarchy is indeed correct, four rules have been specified in the SVM input language. The first rule concerns the calculation of the average price of cars on the market, which is simply calculated by adding the average design cost, production cost, and quality repair cost:

```
next(average_car_price) := average_design_cost + average_production_cost +
average_quality_repair_cost;
```

Furthermore, the calculation of the price of Ford is also specified in the same fashion as the calculation of the average price for cars with one intermediate step, namely the cost price. In this case the two are considered to be equivalent.

```
next(ford_cost_price) := ford_design_cost + ford_production_cost +
                         ford_quality_repair_cost;

next(ford_price) := ford_cost_price;
```

Final element is the calculation of the demand for Ford cars, which is directly coupled to the cost price. Notice that the calculation presented here are identical to the ones used in the simulations.

```
next(ford_demand) := (2000 * 4 * average_car_price) / ford_price;
```

Now finally, two checks are performed after having inputted the initial facts based on the scenario as used in the simulation and the transition rules as specified above. These checks are specified in CTL. The first one states that if the costs at the lowest level of the Ford organization are all equal all lower to the average costs over all companies, demand for Ford cars will be at least equal to a quarter of the total demand (constant at 2000):

```
AG (((ford_design_cost <= average_design_cost) &
    (ford_production_cost <= average_production_cost) &
    (ford_quality_repair_cost <= average_quality_repair_cost))
  -> AX(ford_demand >= 500))
```

A second version is a stronger requirement. It states that if the sum of the costs of all different groups is lower or equal to the average, demand will be at least a quarter of the total demand:

```
AG (((ford_design_cost + ford_production_cost + ford_quality_repair_cost) <=
    (average_design_cost + average_production_cost + average_quality_repair_cost))
  -> AX(ford_demand >= 50))
```

Indeed, both properties are satisfied given the initial conditions and the rules specified. Besides checking whether the lowest level properties satisfy the highest level property, each of the interlevel relationships have also been checked in a similar manner, and were all shown to hold. Furthermore, to prove the successfulness of DODM, the property hierarchy shown in Figure 6 has also been proven by the SMV model checker which shows that introducing lean production in a design object indeed results in canceling the quality repair costs, which satisfied the property DODM_successful. Two input rules have been specified, first of all, the definition of lean production, and secondly the effect of lean production (i.e. 0 quality repair cost):

```
next(production_method) := case
      !q_r_group & multi_task_team_prod_worker & multi_task_team_prod_worker_beh &
                        !spec_prod_worker: lean;
            1 : mass;
                              esac;

next(ford_design_cost):= case
      production_method = lean: 0;
      1: 4;
                  esac;
```

The following property has been shown to hold:

```
AG ((!q_r_group & multi_task_team_prod_worker & multi_task_team_prod_worker_beh &
                        !spec_prod_worker)
-> AF (ford_design_cost = 0))
```

In other words, for all time points, in case CP4-5-6 are indeed accomplished this reduces quality repair cost to 0, which clearly satisfies property CP1. Again, the intermediate relationships have been checked as well, and all were proven to hold. As a result, the DODM_successful property is satisfied as well as the RQSM_successful property in case the components indeed generate the output based on these property hierarchies.

10 Discussion

Organizations aim to meet their organizational goals. Monitoring whether events occur that endanger fulfillment of these goals enables organizations to consciously adapt and survive. Adaptation is essential once an organizational goal becomes unreachable. This paper views such a change as a (re)design process. A component-based formal generic model for design developed within the area of AI and Design is specialized into a model for organization (re)design.

Formalizations developed within the area of Organization Theory and AI (or computational organization theory), have proved suitable for the description of organization models as design object descriptions, and organization goals as design requirements. Furthermore, different types of specialized knowledge have been identified: (1) about main organization goals and their relation for given environmental conditions to organization requirements, (2) about refinement of

organization requirements, (3) about design object descriptions, and (4) which components for a design object description satisfy which requirements. The generic design model was instantiated with such types of knowledge to constitute a specialized component-based model for (re)design of organizations. Example properties have been taken from a well known example in Organization Theory describing the introduction of lean production within an organization [25].

This paper focuses on external triggers for organizational change. Triggers are related to specific goals that play the role of design requirements which the organizational change should comply to. These requirements tend to be high-level goals and lack the detail needed for specifying how an organization should change. Therefore, design requirement refinement is introduced in the form of hierarchies of requirements. Such hierarchies relate objectives of the organization (e.g., high demand for cars) to organizational change properties at different organizational levels (e.g., change in some departments). Thus, they relate triggers at the level of the organization to properties at the level of parts (groups) within the organization. For example, the cause of why a certain type of car is not selling according to the goals that have been set is related to the costs of quality repair. Requirements hierarchies help to localize where to change the organization. High-level goals for an organization as well as goals for organizational redesign have been related to low-level executable properties. Formal verification has been performed and the results show satisfaction of the non-leaf properties in the property tree.

When comparing the approach to previous work in the redesign of organizations the main strength is the formal description of the whole redesign process in terms of a generic redesign model for organizations. In the field of management for example, an overview of which can be found in [7], only informal descriptions are given about redesign processes. In Systems Theory, see e.g. [23], goal oriented behavior is addressed. The gap observed between the actual state of the system and the desired state causes redesign, which corresponds with the approach taken in this paper. Formalizations by means of property hierarchies are, however, not present, therefore formal verification as done in this paper cannot be performed.

In [13] a general diagnosis engine is presented which drives adaptation processes within multi-agent organizations using the TAEMS modeling language as the primary representation of organizational information. In the design of the diagnostic engine three distinct layers are identified: symptoms, diagnosis, and reactions which in the approach presented in this paper roughly correspond to Section 4, 5, and 6 respectively. The implementation of these elements differs in both approaches. The goals and requirements in this paper are explicitly connected to each other. Once an organizational goal is observed not to be fulfilled, such a dissatisfaction is related directly to a goal for change. In the approach presented in [13] lacks such an explicit relation between goals and error diagnosis. Furthermore, this paper also introduces an approach to diagnose whether the whole reorganization process was successful, which is not the case in [13]. [6] explores dynamic reorganization of agent societies and focuses on changes to the structure of an organization, this paper presents an approach that enables such a dynamic reorganization.

In [15] an approach is introduced which aims to archive adaptive real-time performance through reorganizations of the society. As a domain of application, production systems are used throughout that paper. Whereas that paper focuses on

adaptive agents, this paper concentrates on adaptation of an organizational model that abstracts from agents and specifies elements on the level of roles the agents can fulfill.

The work presented in this paper can also be compared with the work on institutions as a way to describe multi-agent organizations. In [8] an institution is said to structure interactions and enforce individual and social behavior by obliging everybody to act according to norms. In that same paper, a formalization language is introduced for such an institution. The approach to use dynamic expression as a restriction of the behavior of agents allocated to that role used in this paper is also expressive enough to describe such norms. For example, in [21] an example of a norms is said to be the following: "Students are prohibited from sitting the exam if they have not completed the assignment" such can easily be formulated in terms of a dynamic property for the student role. The approach presented in this paper could therefore also be applied to institutions and normative organizations.

Finally, in the field of coalition formation (see e.g. [19, 24]), the main purpose of forming a coalition is to perform a task that cannot be performed by a single agent. That work can be combined with our approach by addressing the problem of the allocation of agents to roles, after the change of the organizational model by our approach.

References

[1] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., *LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn*. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), Proceedings of the Third German Conference on Multi-Agent System Technologies, MATES'05. Lecture Notes in AI, vol. 3550. Springer Verlag, 2005, pp. 165-178.

[2] Bosse, T., Jonker, C.M., and Treur, J., Analysis of Design Process Dynamics. In: R. Lopez de Mantaras, L. Saitta (eds.), Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04 , 2004, pp. 293-297.

[3] Brazier, F.M.T., Jonker, C.M., and Treur, J., Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering, vol. 41, 2002, pp. 1-28.

[4] Brazier, F.M.T., Langen, P.H.G. van, and Treur, J., Strategic knowledge in design: a compositional approach, Knowledge-Based Systems 11:405-415, 1998.

[5] Ciancarini, P., Wooldridge, M. (eds.), Agent-Oriented Software Engineering, Lecture Notes in Computer Science, vol. 1957, Springer-Verlag, Berlin, 2001.

[6] Dignum, V., Sonenberg, L., Dignum, F., 2004, Dynamic Reorganization of Agent Societies, In: Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004.

[7] Douglas, C., Organization redesign: the current state and projected trends, Management Decision 37(8), 1999.

[8] Esteva, M., Padget, J., and Sierra, C., Formalizing a language for institutions and norms, In: Intelligent Agents VIII, Lecture Notes in Artificial Intelligence volume 2333, 2002, pp. 348-366.

[9] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

[10] Hannoun, M., Sichman, J.S., Boissier, O., and Sayettat, C., Dependence Relations between Roles in a Multi-Agent System: Towards the Detection of Inconsistencies in Organization. In: J.S. Sichman, R. Conte, and N. Gilbert (eds.), Multi-Agent Systems and Agent-Based Simulation (Proc. of the 1st. Int. Workshop on Multi-Agent Based Simulation, MABS'98), Lecture Notes in Artificial Intelligence, vol. 1534, Springer-Verlag, 1998, pp. 169-182.
[11] Hannoun, M., Boissier, O., Sichman, J.S., and Sayettat, C., MOISE: An organizational model for multi-agent systems. In: M. C. Monard and J. S. Sichman (eds.), Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol. 1952, Springer-Verlag, Berlin, 2000, pp. 152-161.
[12] Haroud, D., Boulanger, S., Gelle, E., and Smith, I.F.C., Strategies for conflict management in preliminary engineering design, In: Proceeding of the AID 1994 Workshop Conflict Management in Design, 1994.
[13] Horling, B., Benyo, B, and Lesser, V., Using Self-Diagnosis to Adapt Organizational Structures, In: Muller, J.P., Ander, E., Sen, S., and Frasson, C., Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, 2001, pp. 529-536.
[14] Hubner, J.F., Sichman, J.S., and Boissier, O., A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems. In: Proc. 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), Porto de Galinhas, Brasil, 2002. Extended abstract in: C. Castelfranchi and W.L. Johnson (eds.), Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02. ACM Press, 2002, pp. 501-502.
[15] Ishida, T., Yokoo, M., and Gasser, L., An Organizational Approach to Adaptive Production System, In: Proceedings of the 8th National Conference on Artificial Intelligence, Boston, USA, 1990, pp. 52-58.
[16] Jaffee, D., Organization Theory: Tension and Change, McGraw-Hill Publishers, New York, 2001.
[17] Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. Int. J. of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
[18] Jonker, C.M., and Treur, J., Relating Structure and Dynamics in an Organisation Model. In: J.S. Sichman, F. Bousquet, and P. Davidson (eds.), Multi-Agent-Based Simulation II, Proc. of the Third Int. Workshop on Multi-Agent Based Simulation, MABS'02. Lecture Notes in AI, vol. 2581, Springer Verlag, 2003, pp. 50-69.
[19] Klusch, M. Gerber, A., Dynamic Coalition Formation among Rational Agents, IEEE Intelligent Systems 17(3), 2002, pp. 42-47.
[20] Lomi, A., and Larsen, E.R.. Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park, 2001.
[21] McCallum, M., Vasconcelos, W.W., and Norman, T.J., Verification and Analysis of Organisational Change. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. 1st OOOP Workshop*, 2005, pp. 91-106.
[22] McMillan, K., Symbolic Model Checking: An approach to the state explosion problem, Kluwer Academic Publishers, 1993.
[23] Rapoport, A., General System Theory, Abacus Press, 1986.
[24] Shehory, O., and Kraus, S., Task allocation via coalition formation among autonomous agents, In: proceedings of IJCAI 1995, 1995, pp. 655-661.
[25] Womack, J.P., Jones, D.T., and Roos, D., The Machine That Changed The World: The Story of Lean Production, HarperCollins Publishers, New York, 1991.

Chapter 5

Adaptation of Organizational Models for Multi-Agent Systems based on Max Flow Networks

This chapter appeared as: Hoogendoorn, M., Adaptation of Organizational Models for Multi-Agent Systems based on Max Flow Networks. In: Veloso, M.M. (ed.), *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, AAAI Press, 2007, pp. 1321-1326.

Adaptation of Organizational Models for Multi-Agent Systems based on Max Flow Networks

Mark Hoogendoorn

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
mhoogen@cs.vu.nl

Abstract. Organizational models within multi-agent systems literature are of a static nature. Depending upon circumstances adaptation of the organizational model can be essential to ensure a continuous successful function of the system. This paper presents an approach based on max flow networks to dynamically adapt organizational models to environmental fluctuation. First, a formal mapping between a well-known organizational modeling framework and max flow networks is presented. Having such a mapping maintains the insightful structure of an organizational model whereas specifying efficient adaptation algorithms based on max flow networks can be done as well. Thereafter two adaptation mechanisms based on max flow networks are introduced each being appropriate for different environmental characteristics.

1 Introduction

With the need for more complex software, arose the need for a higher abstraction level than the concept agent. As a result, organization modeling is becoming a practiced stage within multi-agent system design (see e.g. [2] and [1]). The organizational model poses various constraints on agents populating the organization. Frameworks have been introduced for representing such an organizational model e.g. AGR (Agent/Group/Role)[4], GAIA [10] and MOISE [6].

A common problem encountered with the current organizational modeling frameworks is their static nature. The frameworks do not support the organizational model itself to be dynamic in that it changes based on e.g. the environment. Especially when a multi-agent system participates in a dynamic and unpredictable environment, an organizational model might become obsolete, making dynamic adaptation of the model essential. Imagine a design for a negotiation system on the Internet, in which buyer and seller agents are present. The organizational model specifies the number of buyers and seller that should be present, based on a certain expected input for the system. Suddenly, an increase in usage requires much more buyer and seller agents. Such an event requires adaptation of the current organizational model to the new usage level, otherwise the system would no longer function correctly due to overload.

The aim of this paper is to introduce a method for capacity management of organizations by dynamically adapting an organizational model based on the

environmental fluctuation. For this purpose, the AGR framework is adopted. AGR has been chosen because the framework is closely related to graph representations. Furthermore, extensions of the framework with capacities for each of the elements within the organizational model are introduced. The method itself is based on graph theory, and more specifically, on max flow networks. Specifying methods for adaptation in max flow networks has the advantage of efficient algorithms being available to perform calculations on the network. The method can be incorporated into an agent maintaining such an organizational model, attributing the agent with the capabilities to properly adapt the organizational model. This paper however only deals with evaluating the effectiveness of the method itself.

The paper is organized as follows: Section 2 introduces max flow networks and the terminology associated with it. Thereafter, Section 3 discusses the existing modeling framework for organizations and extends it with capacity elements. Section 4 presents a mapping between the extended organizational modeling framework and max flow networks. Static analysis methods for analyzing the current functioning of the organizational model are presented in Section 5 whereas Section 6 expresses adaptation rules that can be used when the analysis shows an improper functioning. Section 7 presents simulation results of these adaptation mechanisms, and finally, Section 8 is a discussion.

2 Max Flow Networks

This Section provides a brief introduction to max flow networks within graph theory. Max flow theory (see e.g. [5]) is a very well known part of graph theory, appreciated because of its practical applicability. A max flow network is defined as follows.

Let $G=(V,E)$ be a directed graph with a set of nodes V and a set of edges E. Within V two special nodes are distinguished, namely the source $s \in V$ and the sink $t \in V$. The source has an indegree of 0 and the sink an outdegree of 0. Furthermore, let $c: E \rightarrow \mathbb{R}^+$ be a capacity function for the edges. A network is then defined as $N = (V,E,s,t,c)$. Now let $f: E \rightarrow \mathbb{R}_0^+$ denote the *flow value* under the following conditions:

$f(x,y) \leq c(x,y)$ for all $(x,y) \in E$

$f_{in}(v) = f_{out}(v)$ for all $v \in V-\{s,t\}$

Where $f_{in}(v)$ and $f_{out}(v)$, respectively the inflow and the outflow of a node v, are defined as follows:

$f_{in}(v) = \sum_{x \in V} f(x,v)$ with $(x,v) \in E$

$f_{out}(v) = \sum_{y \in V} f(v,y)$ with $(v,y) \in E$

For the source s and the sink t the following thus holds:

$f_{in}(s) = f_{out}(t) = 0$

The flow value throughout the network is now defined as follows:

$|f| := f_{out}(s)$ where $f_{out}(s) = f_{in}(t)$

The *max flow* is the maximum among all flows, and the *max flow problem* is to find such a flow. Several algorithms have been published which can find such a flow, in 1956 Ford and Fulkerson [5] were the first to publish such an algorithm where finding a minimal cut for the graph was proven to be equal to the max flow. Later efficiency improvements have been proposed, see e.g. [3].

To enable a formal mapping between the organizational modeling framework and max flow network, node capacities should be expressible. Specifying capacities for nodes can be incorporated into the classical max flow network as follows: let $c_{node}:V \rightarrow \mathbb{R}^+$ denote the capacity of such a node. Now split up the node v with capacity $c_{node}(v)$ into two nodes: v_1 and v_2 where node v_1 inherits all incoming nodes of v whereas v_2 inherits all outgoing edges. Finally, draw an edge (v_1,v_2) with the following capacity value:

$$c(v_1,v_2) = c_{node}(v)$$

3 Multi-Agent Organizational Framework and Extensions

In this Section, the AGR approach is introduced. AGR is used because the representation of the organizational modeling framework is closest to graph theory. As the purpose of this paper is to investigate adaptations in the capacity of an organizational model, AGR is extended with elements specifying such capacity.

3.1 Agent/Group/Role approach

As a basis for representing a multi-agent organization the AGR approach introduced by Ferber and Gutknecht [4] is used. In the approach, as the name already suggests, three main elements are used: (1) the **agent** which is only specified as an active communicating entity which plays roles within groups; (2) the **group** defined as atomic sets of agent behavior, and (3) the **role** which is an abstract representation of an agent function, service or identification within a group. More formally on an abstracter level, Ferber and Gutknecht define a *group structure* as a tuple $S=<R,G,L>$. In the definition, R is a set of role identifiers whereas G is an interaction graph specifying the valid interactions between two roles (later referred to as role links): $G: R \times R \rightarrow L$, where L is the interaction language. The *organizational structure* is defined as the set of group structures expressing the design of a multi-agent organization scheme. It is expressed as $O=<S,Rep>$, where S is a set of group structures. *Rep* is a representative graph specifying interactions between role of different groups (later referred to as group links): $Rep:S \times R \times S \times R$, e.g. $\text{Rep}(S_a, r_1, S_b, r_2)$ where $r_1 \in S_a$ and $r_2 \in S_b$, and $S_a, S_b \in S$. A constraint is that a single agent is playing both role r_1 and role r_2.

3.2 Agent/Group/Role Extensions

In addition to the AGR approach, it is assumed that in the specification of roles a certain capacity is present. This capacity places a requirement on what an agent to be allocated to the role within the organization should be able to handle computationally per time unit (universal for the whole organization). This capacity is denoted by *RC:* $R \rightarrow \mathbb{R}^+$. In addition, a capacity can also be set for role links: *CC:* $R \times R \rightarrow \mathbb{R}^+$ and group links: *SC:* $R \times R \rightarrow \mathbb{R}^+$.

One crucial aspect is however still missing, namely the interaction with the environment. AGR is mainly based on interaction between roles, whereas the emphasis of this paper is to adapt to environmental fluctuations. Therefore, it is assumed that the environment causes a certain pressure upon the organization. Such pressure is expressed as the amount of processing needed by the organization to deal with the pressure, it can be seen as the demand of the environment upon the system. In the organizational model this is represented by adding an entity called the environment e_{in} and having links from the environment to the roles receiving that stress directly: (e_{in}, r_i) which has a particular value at a certain time point: E_{in}: $R \rightarrow \mathbb{R}^+$. At different points in time the amount of pressure can differ, requiring different processing capabilities. Furthermore, besides receiving pressure from the environment, most roles are assumed to perform actions in the environment (e_{out}) as well, affecting the environment: (r_i, e_{out}) which again has a value: E_{out}: $R \rightarrow \mathbb{R}^+$. Assumed is that a correctly functioning multi-agent organization affects the environment to the exact same amount as the environment affects the organization.

4 Mapping the Organizational Framework to Max Flow Networks

A mapping between the extended AGR model and the max flow networks as introduced in Section 2 is presented. The translation algorithm of the extended AGR model to a max flow network can be described as follows:

- For each role $r_i \in O$ create a node v_i
- For each role link $G(r_i, r_j)$ create an edge (v_i, v_j)
- For each role link with capacity $CC(r_i, r_j)$ set the capacity of the edge (v_i, v_j) to that value: $c(v_i, v_j)=CC(r_i, r_j)$
- For each group link (r_i, r_j) where $r_i \in S_a$, $r_j \in S_b$ and $S_a \neq S_b$ create an edge (v_i, v_j)
- For each group link with capacity $SC(r_i, r_j)$ set the capacity of the edge (v_i, v_j) to that value: $c(v_i, v_j)=SC(r_i, r_j)$
- For each role with capacity $RC(r_i)$ set the capacity of the node v_i to that value: $c_{node}(v_i)=RC(r_i)$, reduce the graph to a classical max flow graph using the method presented in Section 2.
- Add a node *s* to represent the environment e_{in} and add a node *t* to represent the environment e_{out}

- For each (e_{in}, r_i) with capacity $E_{in}(r_i)$ create an edge (s, v_i). Set the capacity $c(s, v_i)=E_{in}(r_i)$
- For each (r_i, e_{in}) with capacity $E_{out}(r_i)$ create an edge (v_i, t). Set the capacity $c(v_i, t)=E_{out}(r_i)$

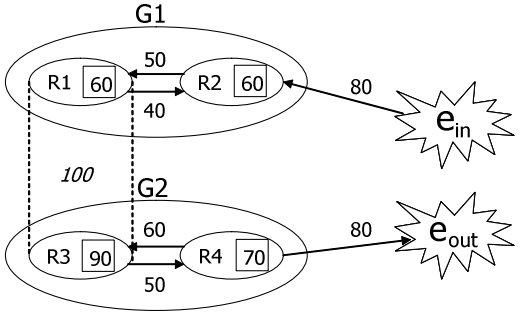

Fig. 1. Example organization represented in AGR

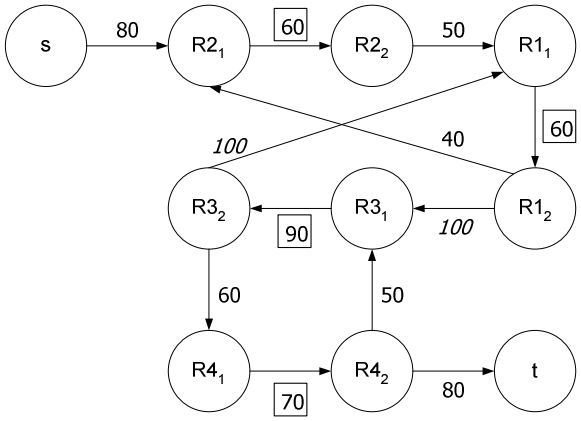

Fig. 2. Max Flow equivalent of the example organization

Figure 1 shows an example AGR organization. In the figure, the big ovals denote groups, whereas the smaller ovals denote the roles. Capacities of roles are depicted as a box with a number specifying the capacity. Furthermore, interactions between roles within a group or between a role and the environment are depicted by arrows, including a label specifying the capacity. Finally, capacities for interactions between groups are specified by dashed lines, including a capacity number depicted in italics. Figure 2 shows the accompanying max flow network using the previously presented translation algorithm (including the translation of node capacity to a classical max flow network) with a *max flow* of 50.

5 Analyzing an Organizational Model using the Max Flow Equivalent

Now that an equivalent max flow network can be derived from the extended form of an AGR organizational model, this Section shows in what way the max flow network can aid in the analysis of the organizational model. Such an analysis can be performed in two ways: (1) checking whether the current organizational model can meet the *expected* environmental conditions (i.e. an analysis beforehand), and (2) checking at runtime whether the organizational model can meet the *actual* environmental conditions.

5.1 Analysis of the Organizational Model based on Expected Values

Creating an organizational model is done having certain requirements in mind. In this paper organizational requirements in the form of organizational capacities are considered. Requirements on capacities are in the form of pressure from the environment (i.e. the organization should be able to handle *x* requests of a certain type). For checking whether such a requirement can theoretically be met by a multi-agent organization, the max flow equivalent can be used, enabling the usage of tools and algorithms from graph theory. To this end an organizational model, including capacities for the various organization elements as introduced in the previous section, is translated into a max flow problem. Remember the notation for the flow: $|f|:=f_{out}(s)$ where $f_{out}(s) = f_{in}(t)$. Now let the maximum flow be noted as follows:

$$|f|_{max}(N):=f_{out,\,max}(s) \text{ where } f_{out,\,max}(s) = f_{in,\,max}(t)$$

The requirements posed for the organizational model can be translated into requirements on the flow. Requirements regarding the amount of pressure can be translated to a max flow requirement of the form

$$f_{out,\,max}(s) \geq r$$

where *r* is the requirement. Using the max flow problem algorithms from e.g. [5];[3] it can be determined whether the organization can theoretically fulfill the requirements. Note that non-fulfillment of the requirements *guarantees* that the organization will never be able to meet the requirements when complying to the design specification.

5.2 Analysis of the Organizational Model based on Observed Values

Besides performing an analysis at design time, an analysis at runtime can also be performed. It can be the case that the environment in which the multi-agent system is participating is highly dynamic and hard to predict, causing an unknown amount of pressure for the organization. Therefore, the requirement *r* posed for the system is dynamic. The requirement can however be observed: observing the amount of pressure received by the multi-agent system. In case this exceeds the maximum flow in the network equivalent, the organizational model is incorrect. Note that it is still

possible that the multi-agent system is functioning correctly, since the agents allocated to the roles within the organizational model might have a higher capacity than required. The model however should always be updated to make sure that the system continues to function correctly. It could for example be that an agent can handle the pressure for a while, but after a certain duration suffers a burn out. Methods for updating such a model are presented in the next section.

6 Adaptation of an Organizational Model using the Max Flow equivalent

As shown in the previous Section, when participating in a highly dynamic environment the organizational model sometimes needs to be changed in order to handle the environmental fluctuations appropriately. This Section proposes two methods to perform such a change, each working under specific circumstances. These methods only concern extending the capacity as the aim of this paper is to adapt the organizational model in such a way that the environment can be handled, which does not include decreasing the capacity.

6.1 Adapting the Bottlenecks

The first method for improving the network equivalent of the organizational model involves finding the path which requires a minimum additional capacity, and adding capacity to the bottleneck within the path. In other words, pinpointing the bottleneck within the organization and improving it. Let $P = s \rightarrow ... \rightarrow t$ denote a path from the source *s* to the sink *t* in the network. In the explanation of the method, capacities of edges are assumed to be natural numbers, however this can easily be extended to rational numbers. Given that the environment has imposed requirement *r*, and the parameter η which represents the safety margin to be taken:

- calculate the current max flow of the network $|f|_{max}(N)$
- if $|f|_{max}(N) < (\eta \times r)$:
  1. $n = 1$
  2. try finding a path *P* of the form $P=s \rightarrow^{+1} ..... x_i \rightarrow^{0} y_i \rightarrow^{+1} t$ where *n* edges $\{(x_1,y_1),...,(x_n,y_n)\}$ do not have sufficient capacity to add a flow of 1 to the path. In case such a path cannot be found: $n=n+1$
  3. set the capacity for all the edges (x_i,y_i) in the set $\{(x_1,y_1),...,(x_n,y_n)\}$ to $c(x_i,y_i) = c(x_i,y_i)+1$
  4. if the new max flow $|f|_{max}(N) < (\eta \times r)$ then continue at point 1, else the algorithm ends

The specification of the algorithm draws inspiration from the algorithms proposed for finding the max flow through a network. These algorithms work with finding paths from source to sink that can be increased with a flow of 1. Finding a path which can almost be increased and extending the capacity therefore results in an immediate increase in the max flow of the network.

6.2 Adding Organizational Elements

The capacity required for allocation of an agent to a certain role is of course limited; agents with very high capacities might be too expensive. Therefore, an algorithm for adding roles and the accompanying role and group links to an organization is presented here. The algorithm as presented above is therefore adapted to cope with addition of organizational elements as well. The extension states the following: in case the max flow of the network can no longer be increased as a further increase of the max flow would necessarily require exceeding of the maximum capacity set (i.e. the current capacity $c(x_i,y_i)$ exceeds the maximum capacity $c_{max}(x_i,y_i)$:

1. For each two nodes v_{i1}, v_{i2} and set of edges of the form (v_x, v_{i1}) where $x \in V$ and $(v_x, v_{i1}) \in E$, (v_{i1}, v_{i2}), and (v_{i2}, v_y) where $y \in V$ and $(v_{i2}, v_y) \in E$ representing a role r_i and its role and group links:

   If at least one of the elements has reached the maximum capacity: calculate the increase of the max flow in case the nodes and edges were to be doubled (i.e. $N_{new}=N \cup \{v_{i1,2}, v_{i2,2}, (v_x, v_{i1,2}), (v_{i1,2}, v_{i2,2}), (v_{i2,2}, v_y)\}$): $|f|_{max}(N_{new}) - |f|_{max}(N)$

   - If the highest increase exceeds 0, in other words the network can be improved by copying a single role, copy the two nodes and edges having the highest value.
   - Otherwise, copy the two nodes v_{i1}, v_{i2} and accompanying edges (v_x, v_{i1}) where $v_x \in V$ and $(v_x, v_{i1}) \in E$, (v_{i1}, v_{i2}), and (v_{i2}, v_y) where $v_y \in V$ and $(v_{i2}, v_y) \in E$ which maximize the following:
     $min\ (|\{(v_x, v_{i1}) \mid (v_x, v_{i1}) \in E\}|, |\{(v_{i2}, v_y) \mid v(_{i2}, v_y) \in E\}|)$
     Where $\{(v_x, v_{i1}) \mid (v_x, v_{i1}) \in E\}$ is the set of incoming edges of v_{i1} and $\{(v_{i2}, v_y) \mid v(_{i2}, v_y) \in E\}$ the set of outgoing edges of v_{i2}. In other words, take the node which has a maximal connection with other nodes. Thereafter, return to 1.

The intuition behind the algorithm is to find the nodes and edges representing the role of which a copy would improve the max flow most. If no increase is possible, take the nodes and edges representing the role which is most connected within the organization to maximize the chances that an addition of a role after the copy will result in an increase of the max flow.

7 Evaluation of the Different Methods for Improving an Organizational Model

In order to characterize the methods presented in the previous Section, this section presents an evaluation method, and compares the results of the different methods using different settings. First, a cost model is presented expressing the cost function used for evaluation. Thereafter, the different methods are evaluated based on the cost model and several environmental settings.

7.1 Cost Model

Each element within the organizational model has a certain cost attached to it. The cost for role and group links is expressed as $cost_{link}$: $R \times R \rightarrow \mathbb{R}^{+}$ and defined as follows:

$cost_{link}(x,y) = e^{(CC(x,y)/\alpha)}$ where $(x,y) \in (G \cup Rep)$

In other words, cost for links increase exponentially (a commonly used type of cost function), where the parameter α can be varied. The same holds for the cost of a node $cost_{role}: R \rightarrow \mathbb{R}^{+}$ which is thus defined as follows:

$cost_{role}(r) = e^{(RC(r)0.5/\alpha)}$ where $r \in R$

The factor 0.5 is arbitrarily set in the cost function because typically interaction capacity costs are lower than agent capacity costs. In order to punish an organizational model not being able to meet the environmental pressure, a penalty is introduced of the form $p: N \times r \rightarrow \mathbb{R}^{+}$, where r is the environmental requirement. The penalty function is defined as follows:

$p(N, r) = \beta \times (r - |f|_{max}(N))$

The parameter β specifies the penalty for each requirement unit not fulfilled. The network N represents the multi-agent organization. Finally, the overall cost for the organizational model is defined as follows:

$cost_{total} = p(N, r) + \sum_{r \in R} cost_{role}(r) + \sum_{(x,y) \in G \cup Rep} cost_{link}(x,y)$

7.2 Evaluation

An implementation in Java has been created of the two algorithms for reorganization, and the translation procedure of an organizational model to the accompanying network. In order to evaluate the performance of the two algorithms the example organizational model shown in Figure 1 is used. As a benchmark, no adaptation of the organizational model is used. Using the implementation, simulation runs are performed with an environmental pressure causing a requirement r based on a normal distribution $f(x;\mu,\sigma)$. One step within such a simulation entails: (1) generating the environmental requirement r based on the normal distribution; (2) calculating the current max flow of the network: $|f|_{max}(N)$; (3) calculating the cost $cost_{total}$, and (4) updating the network for the next step, using one of the improvement methods. Each step is performed 100 times, and each simulation is performed 10 times, generating from a different seed each time. After the steps have been performed, the average of the $cost_{total}$ per step is calculated. In order to evaluate the different methods, two settings for the cost model have been used: relatively high penalty cost compared to agent/communication cost (e.g. a critical domain such as incident management), and relatively low penalty cost compared to agent/communication cost (a non-critical domain). Furthermore, the environment setting μ is by default set to the initial max flow (in this case 50), whereas the fluctuation σ has been set to different values: $\sigma=\{0,5,20\}$.

7.2.1 Relatively High Penalty Cost

The results of an organization in which penalty costs are relatively high compared to labor cost are presented here. This reflects in the cost model in which α is set to 80, meaning an initial cost of 34 for the whole organization, whereas β is set to 200, a penalty cost significantly higher than the initial cost of the whole organization.

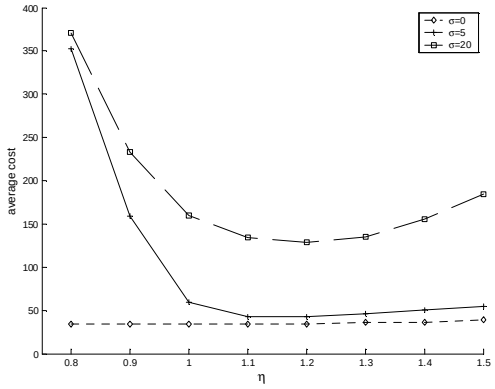

Fig. 3. Bottleneck algorithm performance for high penalty cost

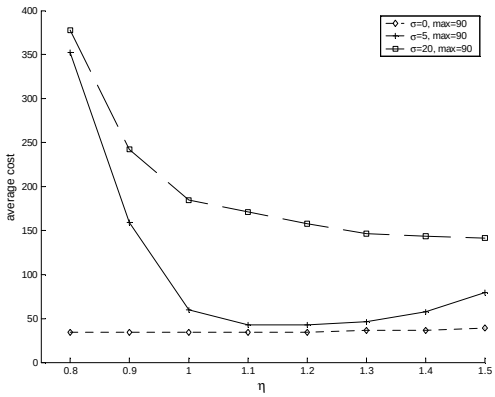

Fig. 4. Role addition algorithm performance for high penalty cost with max capacity 90

First, the bottleneck algorithm is used. Figure 3 shows the results for different setting given environmental fluctuation σ and varying η value (the algorithm parameter specifying how much to update the capacity). As can be seen, with no environmental fluctuation, the costs stay stable for the lower η values whereas they slightly increase for the higher settings, as the capacities are even increased when the current max flow is identical to the requirement r. For small environmental fluctuation (σ=5) an η value of 1.2 gives the best results, which is also the case for large environmental fluctuation. In both cases, having a higher η value results in the

capacity being too high (i.e. costs too high) whereas a lower η value increases the amount of penalties.

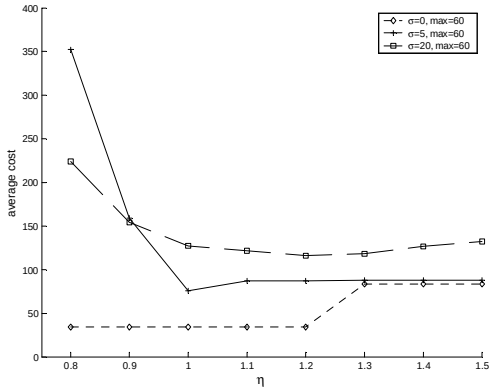

Fig. 5. Role addition algorithm performance for high penalty cost with max capacity 60

Table 1. Average costs of the organizational model resulting from the different algorithms, penalty is set to either 5 or 200

β	**σ**	**No adaptation max flow μ**	**No adaptation max flow 2μ**	**Optimal Bottleneck**	**Role addition max=90**	**Role addition max=60**
200	0	34	199	34	34	34
200	5	370	199	42	42	75.6
200	20	1518	210	129	141	116
5	0	34	199	34(η≤1.2)	34(η≤1.2)	34(η≤1.2)
5	5	42	199	35(η=1.0)	35(η=1.0)	37(η=0.9)
5	20	71	210	35(η=0.8)	37(η=0.7)	69(η=0.5)

Figure 4 and 5 show the results of the bottleneck algorithm extended with role addition. In Figure 5 the max capacity has been set to 90 whereas in Figure 6 60 is used. In the case of no environmental fluctuation, Figure 5 shows a similar shape as the bottleneck algorithm, whereas Figure 6 shows a large increase above a value of η=1.2. This is the result of roles being copied due to the low setting of the maximum capacity. Each role in the network has already reached the maximum capacity, resulting in a need for a copy, which causes a severe overcapacity. In case of little environmental fluctuation, the setting 90 for the maximum capacity is better since no large increase for capacity is required, whereas with high environmental fluctuation 60 is best since larger capacity increases are required.

Finally, the top part of Table 1 (the rows where β=200) shows the comparison between the different methods, given optimal parameter settings. As can be seen, no adaptation is always worse compared to the other algorithms, even when an overcapacity is initially present. Furthermore, in case of small environmental fluctuation adding capacity to the current roles is best (despite the exponential cost

function) whereas for larger fluctuation, adding roles immediately is the best option due to the exponential cost function.

7.2.2 Relatively Low Penalty Cost

Table 1 also shows the result when the penalty is set to a relatively low value (the rows where β=5). Still the algorithms are better than no adaptation, however the value for η needs to be set to a lower value as having too much capacity is relatively expensive compared to the penalty for not meeting the requirements. Therefore, the role addition algorithm with a maximum capacity 60 performs worse than the one with 90.

8 Discussion

This paper has presented an approach to adapt an organizational model to fluctuations within the environment. Such an approach can be incorporated into an agent responsible for maintaining the organizational model. In a highly dynamic and unpredictable environment, such an adaptation mechanism might be a necessity to guarantee successfulness of a multi-agent system. The approach used in this paper is to translate an organizational model to a max flow network and specify two algorithms for adaptation. Specifying analysis constructs and reorganization algorithms for the graph representation has as an advantage that knowledge and algorithms from the well established graph domain can be reused (such as calculation of the max flow [3]). The algorithm used for adaptation of bottlenecks is indeed a known algorithm within graph theory for addition of capacity. The addition of organizational elements however requires knowledge about the meaning of the elements within the max flow network (e.g. what the roles are), algorithms from graph theory can therefore not be re-used. The algorithms as proposed in this paper have been evaluated by simulation runs, and were shown to be more effective compared to no adaptation, especially in critical domains in which the penalty function is relatively high. Limits of the approach for instance include the case where the environmental pressure is at first above the capacity of the organizational model and thereafter steadily decreases. In such cases the organizational model using the adaptation mechanism will adapt to the initial high pressure, and suffer from an overcapacity at the later time points. The total sum of the penalties received in the beginning by the non-adaptive organizational model might be lower than the total cost of the overcapacity in the adaptive case. Very rare outliers with a very high environmental pressure can have the same effect.

In the field of adaptive agents and multi-agent systems (see e.g.[9];[8]) learning from the environment is an important topic. Adapting organizational models based on the environmental conditions is, as argued before in this paper, one of the necessities for the new organizational paradigm. Especially with continuously changing circumstances and agents leaving and arriving a well specified and up-to-date organizational model is required to guarantee proper functioning of the organization.

When comparing the approach with other organizational modeling approaches in multi-agent systems, those approaches often include much more concepts than

capacity of a role. An extension of GAIA [10] for example adds the notion of organizational rules. Such rules express relationships and constraints between roles, protocols, and roles and protocols. These relationships and constraints can be incorporated in the approach presented in this paper as well. When copying a role one can simply copy those relationships involving the role being copied and adapt them to specify the relationships and constraints of the copy of the role. MOISE [6] defines missions for roles, which can include concepts such as goals, plans, actions, and resources. Furthermore, authority links between roles can also be specified. Again, as already stated for GAIA, these concepts can be reused when copying a role. Several of these organizational models have been extended with organizational change notions, see for example [7]. These extensions are however typically very generic models without going into specific details on how to reorganize the organization whereas this paper does.

Finally, for future work an interesting continuation would be to look at the performance in other simulation settings, and possibly compare how well different types of organizational structures perform in changing circumstances.

Acknowledgements

The author would like to thank Catholijn Jonker, Jan Treur, and Evert Wattel for the fruitful discussions, and the anonymous reviewers for their useful comments.

References

[1] Boissier, O., Dignum, V., Matson, E., and Sichman, J. (editors), *Proceedings of the First OOOP Workshop*, Utrecht, 2005.

[2] Ciancarini, P. and Wooldridge, M. (editors),*Agent-Oriented Software Engineering*, LNCS, Spinger Verlag, 1957 edition, 2001.

[3] Edmunds, J. and Karp, R.M., Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the Association for Computing Machinery*, 19:248--264, 1972

[4] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organizations in multiagent systems, In *Proc. of ICMAS 1998*, pages 128--135. IEEE CS Press, 1998

[5] Ford, L.R. and Fulkerson, D.R., Maximum flow though a network, *Canadian Journal of Mathematics*, 8:399--404, 1956.

[6] Hannoun, M., Boissier, O., Sichman, J., and Sayettat, C., Moise: An organizational model for multi-agent systems, In: *IBERAMIA-SBIA '00: Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI*, pages 156—165, Springer-Verlag, 2000.

[7] Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modeling centralized organization of organizational change. Computational and Mathematical *Organization Theory*, In Press, 2006.

[8] Kudenko, D., Kazakov, D., and Alonso, E. (editors), *Adaptive Agents and Multi-Agent Systems II*, Springer Verlag, 2005

[9] Schaal, S., Ijspeert, A.J., Billard, A., Vijayakumar, S., Hallam, J., and Meyer, J.A. (editors), *From animals to animats 8*, MIT Press, 2004

[10] Zambonelli, F., Jennings, N., and Wooldridge, M., Organizational rules as an abstraction for the analysis and design of multi-agent systems. *Journal of Software and Knowledge Engineering*, 11:303--328, 2001.

Part III:
Organizational Change Process:
Centralized Change Processes

Chapter 6

Modeling Centralized Organization of Organizational Change

This chapter appeared as: Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modeling Centralized Organization of Organizational Change, *Computational and Mathematical Organization Theory*, vol. 13, 2007, pp.147-184. The original publication is available at www.springerlink.com.

Furthermore, part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modelling the Organisation of Organisational Change. In: Giorgini, P., and Winikoff, M., (eds.), *Proceedings of the Sixth International Workshop on Agent-Oriented Information Systems, AOIS'04* , 2004, pp. 29-46.

Modeling Centralized Organization of Organizational Change

Mark Hoogendoorn[1], Catholijn M. Jonker[2], Martijn C. Schut[1], and Jan Treur[1]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {mhoogen, schut, treur}@cs.vu.nl
URL: http://www.cs.vu.nl/~{mhoogen, schut, treur}
[2]Radboud University Nijmegen, Nijmegen Institute for Cognition and Information
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
Email: C.Jonker@nici.ru.nl
URL: http://www.nici.ru.nl/~catholj

Abstract. Organizations change with the dynamics of the world. To enable organizations to change, certain structures and capabilities are needed. As all processes, a change process has an organization of its own. In this paper it is shown how within a formal organization modeling approach also organizational change processes can be modeled. A generic organization model (covering both organization structure and behavior) for organizational change is presented and formally evaluated for a case study. This model takes into account different phases in a change process considered in Organization Theory literature, such as unfreezing, movement and refreezing. Moreover, at the level of individuals, the internal beliefs and their changes are incorporated in the model. In addition, an internal mental model for (reflective) reasoning about expected role behavior is included in the organization model.

Keywords: organizational change, formal organizational modeling, organizational simulation, multi-agent organizations, organization verification

1 Introduction

Within the literature on Organization Theory changing organizations play a dominant role [30; 21; 22]. As change processes involve many factors ranging from making the employees aware of changes to come and taking away resistance to change to the design of efficient organizational structures. Changes can concern rather simple processes of slight changes in one or more role descriptions. They may affect only a part of the organization or practically the whole organization. Roles or big parts of the organization may be deleted, new ones created. The realization of the organization probably changes, e.g., agents fulfilling other roles than before, agents leaving the organization, agents joining the organization [16]. A change may be initiated by the environment or by the organization itself. The organization of a change process may involve agents from outside the organization (e.g., consultation) or from inside. In this

paper, the process of (business) organizational change is analyzed in more detail. Methods used in this analysis are those of formalization, simulation and verification. To organize change processes, a generic organization model for organizational change is introduced and formalized. This organization model incorporates both multi-agent co-operation aspects and individual cognitive aspects in the form of the internal mental states (e.g., beliefs) of those involved in the change.

A specific area in which organizational change is inherent, is in the organization that is needed to cope with a big upcoming event. Such an event can be a planned event in the area of sports or concerts, for example, but also an incident that can grow out to a disaster. The latter area is the focus of the project CIM (for Cybernetic Incident Management); cf. [1; 18; 19]. A common characteristic for incidents and big planned events is that the organizational structures start almost at zero, i.e., no activity, and hence no organization, but (have to) grow out to a scale and form of organization that is able to address large and complex processes by multiple parties and multiple agents. To test ideas on organizational change modeling and to get more insight in cases with these characteristics, the organization of a big sports event has been chosen: the famous Dutch 11 cities ice skating tour (10.000s of people all performing 200 km of ice skating on one day, going from city to city). In this case study the usefulness of the developed organization model for organizational change is evaluated.

To model the organizational change process, the theory presented in [27; 20] has been used as inspiration, and has been evaluated on its usefulness in an operational (modeling) sense. The three phases unfreezing, moving, refreezing distinguished have been incorporated in the generic model for organizational change developed. The case study shows that this theory indeed can be integrated in an organizational change modeling approach in a useful manner.

This paper is organized as follows: Section 2 gives an overview of organizational change literature, and introduces the stages that can be identified in an organizational change process. Section 3 introduces the approach which has been used to model the stages in organizational change in a formal way. The model itself is specified in Section 4 both from a structural as well as from a behavioral perspective. Section 5 presents a language used to specify an organizational change and Section 6 presents results of a case study which has been performed to show how the approach can be applied. In Section 7 formal verification is performed upon the simulation results to show that the simulated organization indeed satisfies the desired properties. Finally, Section 8 draws conclusions based on the results presented in this paper.

2 Organizational Change Literature

Organizational change is a well studied topic in recent literature on sociology, psychology, and economics. Change within organizations has become part of everyday life, some organizations are even continuously undergoing change. Changing an organization is not a simple process, often difficulties are encountered within such a change process. Research has shown that over 70 percent of the change programs in organizations do not achieve the intended goal [17; 5]. Boonstra [6]

criticizes typical explanations given for these failures in that they pay insufficient attention to the complexity of the change process itself. Three types of organizational change are distinguished within the introduction of his book: First, *planned organizational change*, which addresses questions with respect to problems that require change in technical and instrumental aspects in which the problems and solutions are known. Secondly, *organizational development* which is said to be suitable when "the changes to be made are far-reaching, the problems not entirely unambiguous but still recognizable, and there is some idea as to the direction in which the solutions must be sought". Cummings and Worley [9] define organizational development as "a system-wide process of applying behavioral science knowledge to the planned change and development of strategies, design components, and processes that enable organizations to be effective". The final type of organizational change distinguished is *transformational change*, in which the change processes include "renewal processes involving actors from various organizations". In Ackerman [2] transformational change is said to be the emergence of a totally new state of being out of the remains of the old state.

Both in planned change and organizational development an approach is taken in which a move is performed from one stable state to another. The change processes involve the phases in which an organization is unfrozen, changed, and refrozen. These phases within the organizational change process originate from the ideas of Kurt Lewin [27]. He states that there are two opposing forces at work when changing an organization: forces that resist the change, and forces that drive towards the newly desired organization. Figure 1 presents the phases and forces within organizational change in a graphical manner (from [30]). The unfreezing phase begins at the moment that change becomes necessary and consists of the process of changing

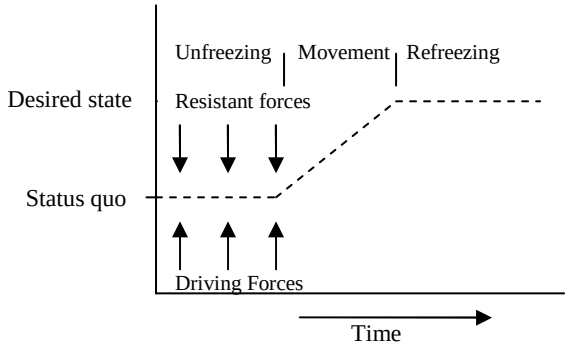

Fig. 1. Movement of an organization from a status quo to a desired state [30]

the resisting and driving forces in such a way that change becomes possible (i.e., the driving forces outweigh the resisting forces). Both Schein [31] and Hosking [20] stress the importance of communication within this unfreezing phase to enable a successful change. According to Cummings [10] organizational development has discovered a long list of causes for resistance to change, such as structural inertia, work habits, fear of the unknown, powerful interests, and members' security needs.

Forces that drive an organization to change can be found in Jaffee [22] and for example include change on the supply side, customer behavior, available technology (see e.g. [29]), etc. The actual change of the organization is contained in the movement phase in which the organization is moved from the current state to the desired stated. The refreezing phase involves freezing the newly formed organization so that there is no possibility to return to the former status quo or to continue changing in another unwanted direction. The whole re-organization process is completed when all phases have been completed. The unfreezing can be performed by increasing the driving forces and/or by decreasing the resisting forces. In their book, Cummings and Worley [9] state that Lewin's model remains closely identified with the fields of planned change and organizational development.

Since the model of Lewin is a highly generic model, effort in organizational development research has gone into making it more concrete. Lippitt *et al*. [28]for example arrange Lewin's model in seven steps: within the unfreezing phase they identify scouting, entry and diagnosis. The movement phase is split up into planning and action, and finally, stabilization and evaluation, and termination are placed within the refreezing phase.

Particularly of interest for this paper are further refinements regarding the actors within organizational change. Kotter [26] has defined characteristics for change managers to prevent organizations from falling into pitfalls due to bad change management. These include having industrial and organizational knowledge, relations in the firm and industry, and reputation and track record. Power is an important aspect related to actors in organizational change processes as well, since the resisting and driving forces of the actors need to be changed to enable an organizational change. This particular research branch is called power dynamics. Research started in 1946 when Kurt Lewin introduced T-groups in a laboratory training setting and was mainly based on group-based approaches where people learn about group dynamics, leadership and interpersonal relationships. Bradshaw and Boonstra (2004) identify several different notions of power. Firstly, *manifest-personal power* which takes the viewpoint that a person can have power over other people and can make them do something they would not do otherwise. Research concerning this form of power research is said to have started with the work of Dahl [11], Emerson [12] and Wrong [33]. In *manifest-structural power*, power is no longer viewed from the personal perspective, but from a group perspective. Bacharach and Lawler [4] is named as a reference for this notion of power. Negotiations are said to be an important part of the models regarding manifest structural power. *Latent-Cultural Power* sees organizing as "a process of the creation and reproduction of shared meanings that are largely latent or unconscious", they also refer to Alvesson [3] for more details about the notion of latent-cultural power. Finally, *latent-personal power* which is said to be relatively new in organization theory. This type of power is said to differ from latent-cultural power is several different ways. First of all, power is said to be scattered throughout the organization, even individuals at the bottom of the organization can deploy their power. Secondly, power relations are assumed to become part of the psyche of the individual.

As the theory of Lewin is still considered being the underlying theory for organizational change research and considered valid, this paper tries to model the theory in a generic sense as a first step towards modeling and understanding complex

organizational change processes. Further extensions might focus on the idea sketched above such as on more complex power relationships, the role of different characteristics for change managers, and the different ways to enable unfreezing an organization.

3 Modeling Approach for Organizations

Before being able to model the organizational change processes identified by Lewin, a methodology is required which enables modeling organizations in general. This Section presents such a methodology which allows modeling of organizations from two perspectives. First, the structural perspective, merely specifying the structural blueprint of an organization, and secondly, the behavioral perspective which specifies the behavior of an organization and the actors within such an organization.

3.1 The structural description of an organization

For the structural description of actual multi-agent organizations, the AGR (for agent/group/role) modeling approach has been adopted [14]. In that approach, an organization is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. But, by avoiding an agent-oriented viewpoint, an organization is regarded as a structural relationship between a collection of agents. Thus, an organization can be described solely on the basis of its structure, i.e. by the way groups and roles are arranged to form a whole, without being concerned with the way agents actually behave, and multi-agent systems will be analyzed from the outside, as a set of interaction modes. The specific architecture of agents is purposely not addressed in the organizational model. The three primitive definitions are:

- The *agents*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application.

- *Groups* are defined as atomic sets of agent aggregation. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. An agent can be a member of *n* groups at the same time. A major point of these groups is that they can freely overlap.

- A *role* is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group.

Figure 2 presents an example of an organization modeled in AGR. The large ovals denote groups whereas the smaller ovals denote the roles within the organizations.

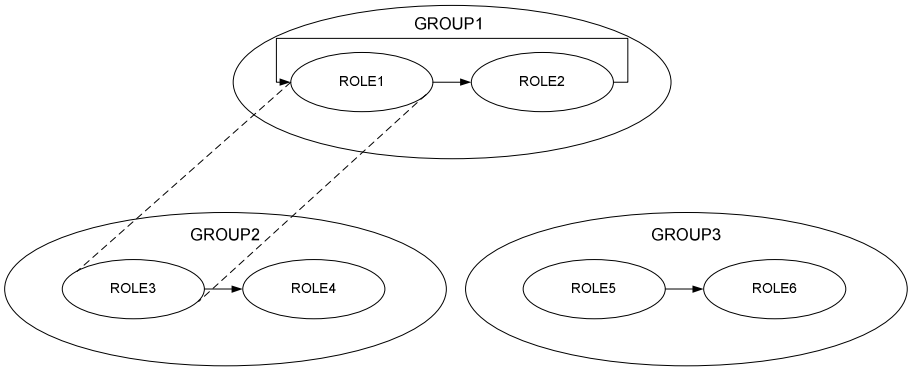

Fig. 2. Example organization modeled within AGR

Furthermore, the solid arrows denote interactions between roles, and the dashed lines represent inter-group interactions. Agents realizing the roles are not depicted.
To enable simulation and reasoning about such an organizational model, the Structural Language SL is used, based on the set of *sorts* (a class or type of objects) that is shown in Table 1. These sorts enable talking about structural elements in the organization model. Additionally, Table 2 shows a set of *predicates* within SL that define relations between the introduced sorts.

Table 1. Sorts in *SL*

Sort	**Description**
ROLE	Sort for a role within an organization.
AGENT	Sort for an agent that can be allocated to a certain role.
GROUP	Sort for a group within an organization.
TRANSFER	Sort for a connection between two roles within one group.
GROUP_INTERACTION	Sort for a connection between two roles in a different group.

Table 2. Predicates defined in *SL* to describe the structure of an organization

Predicate	**Description**
exists_role: ROLE	A role exists within an organization.
allocated_to: AGENT x ROLE x GROUP	An agent is allocated to a role within a group.
exists_group: GROUP	A group exists within the organization.
role_belongs_to_group: ROLE x GROUP	A role belongs to a group.
intra_group_connection: ROLE x ROLE x GROUP x TRANSFER	A role is connected to another role (directed) within a certain group by means of a transfer connection. The source and destination roles are allowed to be equivalent.
inter_group_connection: ROLE x GROUP x ROLE x GROUP x GROUP_INTERACTION	A role within a group is connected to a role within another group by means of a group interaction connection.

3.2 The behavioral description of an organization

In this section a method to express dynamics within an organizational model is addressed. To formally specify dynamic properties at the different aggregation levels that are essential in an organization, an expressive language is needed. To this end the Temporal Trace Language is used as a tool; cf. [23]. For the properties occurring in the paper informal, semi-formal or formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly described as follows; for more formal details, see Appendix A.

A state ontology Ont is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is defined as an indication of which state properties expressed in ontology Ont hold in the state and which do not hold. The set of all states is modeled by the sort STATE. A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is an indication of which state occurs at which time point, for example if a discrete time frame based on natural numbers is taken, a trace is a sequence of states γ_t ($t \in T$). The set of all traces over state ontology Ont is modeled by the sort TRACE. Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. A *dynamic property* over state ontology Ont is a temporal statement that can be formulated with respect to traces based on the state ontology. Such temporal statements can express, for example, a temporal relationship between the fact that in a given trace a certain state property holds at a certain time point and another state property holds at some other time point. For more formal details, see Appendix A.

The Temporal Trace language can be used to specify behavioral properties at different aggregation levels, according to the organizational structure. Within the AGR approach the aggregation levels are the level of the roles, the level of the groups and the level of the organization as a whole (see Figure 3). The lower level properties can often be modeled in simpler formats than the higher level properties. In particular, it is often possible to model the properties at the leaves of the tree in the form of directly executable properties, i.e., by direct temporal dependencies between state properties in two successive states. To model direct temporal dependencies between two state properties, not the expressive language TTL, but the simpler *leads to* format is used. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties (the leaves of the tree in Fig. 3). The format is defined as follows. Let α and β be conjunctions of elementary state properties, and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold
for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [23]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that simulation results can be depicted graphically.

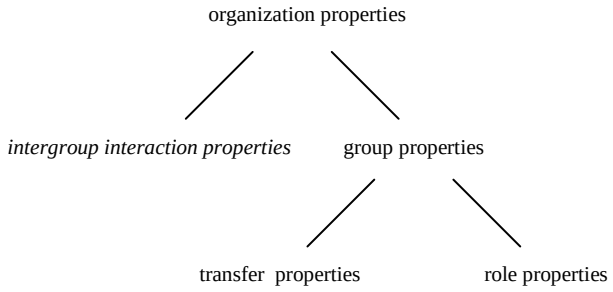

Fig. 3. Overview of interlevel relations between dynamic properties

Table 3 shows the predicates within the Behavioral Language BL which allows the specification of the behavioral part of the organization at different aggregation levels, using the TTL language as described above. The sort DYNPROP expresses an identifier of a dynamic property whereas DYNPROPEXP expresses the dynamic property itself in terms of TTL.

Table 3. Predicates defined in BL to define the dynamics within an organization

Predicate	**Description**
role_property: DYNPROP x ROLE x GROUP	A role within a group has a role property.
transfer_property: DYNPROP x ROLE x ROLE x GROUP	Within a group, a transfer property with an identifier holds between two roles.
group_property: DYNPROP x GROUP	A group has a certain group property.
group_interaction_property: DYNPROP x ROLE x GROUP x ROLE x GROUP	An interaction property with an identifier holds between two roles in different groups.
organization_property: DYNPROP	A certain or property holds for the organization.
has_expression: DYNPROP x DYNPROPEXP	A specific dynamic property has an expression.

Based on the sort DYNPROPEXP it is possible to put more constraints on particular types of properties. The constraints for the different properties are defined below. The formal representations of these properties can be found in Appendix B.

Role dynamic properties

Role properties involve only one role, namely the role for which the property holds. Therefore, a role property should only contain elements that are part of the ontology of that role. The group is also part of the definition of the ontology since roles in different groups can have the same name and might have a different ontology. Role properties can be divided into different types which in turn can be defined more restricted than the general definition. An example of such a refinement is an executable role dynamic property.

Transfer dynamic properties
Transfer properties relate the output of a role to the input of a destination role, therefore the restriction on this dynamic property is that it should be expressed in terms of the output ontology of the source role combined with the input ontology of the destination role.

Group dynamic properties
Group dynamic properties are dynamic properties expressed in terms of the state ontologies of (some of) the roles within the group. The most common type of group property relates an output state of a role within the group to an input state of another role within that group.

Intergroup interaction dynamic properties
Group interaction properties involve the input of a role within one group which is related to the output of a role within another group.

Organization dynamic properties
For the organization dynamic properties the same holds as for group properties: states of multiple roles (this time in different groups) can be involved; there is no further specific definition for this type of property.

4 Organizing Organizational Change

The term organizing organizational change makes it explicit that organizational change is a behavior process of that organization. Therefore, when formalizing organization dynamics, also the process of change must be formally specified as one of the possible ways of behavior of the organization. As all organizational behavior is described in terms of the behavior properties of the roles in that organization, also the whole process of organizational change is attributed to a set of roles in that organization. This section presents an organization model of organizational change that is based on the three stages of change introduced by Lewin.

4.1 Structure and Informal Behavior of the Change Organization

Modeling the forces indicated in Lewin's model entails attributing these forces to roles. Given an existing organization model that does not model organizational change, there are two basic choices that can be made: assigning these forces to roles already in the model, or extending the model with additional organizational elements. The first can be a part of the second approach by first extending the existing model with additional organizational elements, and then applying the first approach. Although the first approach can be a part of the second, when modeling an organization in which the realizing agents cannot reason about the change or even about the role that they are playing (e.g., when modeling an ant hill), only the first

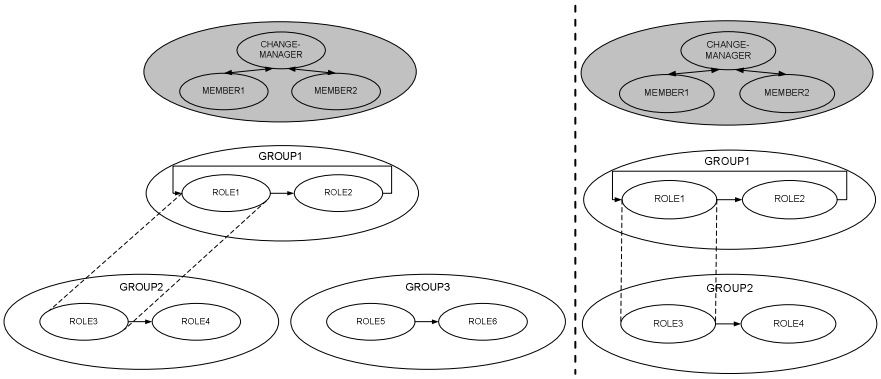

Fig. 4. (a) Organization before the change (b) Organization after the organizational change

approach can be followed and the roles must be modeled as adaptive roles to ensure the possibility of change. In this article, the realizing agents can reason about roles and organizations. The second approach is chosen to most explicitly show the organizational change process. In both cases the behavioral specification of the organization elements needs extension, resulting in an organization model that incorporates organizing organizational change.

Consider, as an example, the organization as presented in Section 3.1, Figure 2 which is also shown at the bottom part of Figure 4a. An organizational change might for example concern the removal of Group3, which in turn could imply that one of the agents realizing the organization will be fired. It might further entail a re-allocation of agents over roles in groups. The organization in its state before change resists change (resisting forces outweigh the driving forces). To formally model this phenomenon, the resisting and driving forces must be attributed to roles. Attributing them to the existing roles is counterintuitive, because different roles have been identified to specify different behaviors. The resisting and driving behaviors are of a different category. The way chosen in this article, is to recognize that all agents part of the realization of the organization have one thing in common: they are all members of the organization. Some members of the organization might be in favor of change, some against, and this might change over time. This is modeled by adding the role Member to the organization model, and attributing driving and resisting forces to that role. Given that the organization changes from one stable situation to a new stable situation, there is a need to model the focus existing in the organizational change. For this reason the role of Change Manager is added to the organizational model. The Change Manager is attributed with driving forces. This role can be realized by an agent from an external company, i.e. a consultant type of role, or by an agent from within the organization. In Figure 4(a), the new roles are grouped together in an organizational element called the Change Group, the members are represented by Member One, Member Two, etc.

The Change Group is depicted in grey in Figure 4(a) to indicate that in stable situations this group is inactive. The Change Manager can be of several different types, for example there can be a global Change Manager, that is allowed to change

the entire organization. It is however also possible to have a local Change Manager that is only allowed to change a certain part within an organization and therefore can only communicate with a sub-group of the members within the Change Group. Because the Change Manager can be a representative of the company itself or of an external company there is no predefined shared allocation between this role and another. Every realizing agent of the organization is (next to the role it was already allocated to) also allocated to one instance of the Member role of the Change Group. The Change Group has a meta-view on the organization, and can, therefore, be seen as a meta-group. The start of an unfreezing phase (meaning a change is due) is characterized by a sudden activity of the Change Manager within the Change Group. The Change Manager might, for example, inform (all or some of) the instances of the Member role of the impending organizational change and the reasons for this change. Aside from the resulting reduction of resisting forces that this information might bring about, this interaction can also be used to model the preparation for the movement phase.

At the end of a well-performed unfreezing stage, maybe all Member role instances, but at least every Member role instance whose realizing agent is somehow involved in the change, now has beliefs about which role its realizing agent may have to play in the new organization. These beliefs include the expected role behavior. The end of the unfreezing phase may be characterized by the presence of these beliefs in the respective member role instances or communication of this presence to the Change Manager. Note that this does not say anything about all activities required to accomplish these shared beliefs.

The start of the movement phase, after a well-performed unfreezing phase, is characterized by the Change Manager informing all Members of when the actual change in organization is to take place. At the indicated moment, all Member roles are to consider in their beliefs the new organization form to be the current organization form. The movement phase is used to achieve (for example, by being informed) that all involved will get the appropriate beliefs on the new structure and their roles in this structure. As a result, the affected parts of the organization will start behaving according to the behavior specification of the new organization form. This process is modeled by means of the shared allocation of agents. Behavior that has become obsolete within the organization will disappear over time.

The start of the refreezing phase is characterized by regular functioning of the new organization form and a de-activation of the Change Group, see Figure 4(b). The refreezing phase is complete when the behavior of the organization shows the routines that correspond to the expected behavior of the new, now current, organization.

Next to the structural properties of the organization model of organizational change, also the behavioral properties of the roles involved should be described to get a complete model. The next sections describe the behavioral properties of the main roles; the Change Manager and the Member.

4.2 Dynamic Properties for the Behavior of the Change Organization

The Change Manager is active in all stages of the organizational change. The properties in this section are described in a domain independent manner, more

describing the global behavior than the actual behavior. Examples of more specific properties can be found in Section 6. First, properties regarding the unfreezing phase are presented, after which the behavior during the movement phase is described. Finally, the behavior during the refreezing phase is described.

4.2.1 Dynamic properties for the Unfreezing Phase

First of all, the following property states the global behavior during the unfreezing phase, namely that once there is an upcoming change, eventually enough key Members (fraction e) within the Change Group will be unfrozen which takes the form of a communication of acceptance of the new organization model.

GP1(ChangeGroup): Unfreezing Organization
if at time t the Change Manager within the Change Group has access to a plan which specifies a condition C for a decision to reorganize based on a new organizational model OM
and condition C is met at time t,
and the Change Manager uses fraction e
then at a later point in time t2 , at least fraction e of the key Members within the Change Group will have informed the Change Manager upon their acceptance of the new organization model OM.

This property can be fulfilled by means of several lower level properties. First of all, the Change Manager informs all Member within the Change Group that are involved in the change based on the new organizational model.

RP1(ChangeManager): Communicate Change
if at time t the Change Manager within the Change Group has access to a plan which specifies a condition C for a decision to reorganize based on a new organizational model OM
and condition C is met at time t
and Member M1 is involved in the change to organizational model OM at time t
then at a later point in time t2 the Change Manager will inform Member M1 about the upcoming change to organizational model OM.

Furthermore, ideally once a Member is informed about such an upcoming change, the member will eventually communicate the acceptance of the new organizational model OM and thus will show to be unfrozen. Fraction e is given as a parameter for the property. Note that these properties describe a successful unfreezing phase where at least fraction e of the Members accepts the change.

GP2(ChangeGroup, e): Confirm Change Acceptance
for at least a fraction e of the key Members in the Change Group
if at time t a Member M1 is informed about the new organizational model OM
then at a later point in time t2 Member M1 will inform the Change Manager of its acceptance of the change to the new organizational model OM.

The property above is again specified in a general sense, as there might be a whole process involved in convincing the Member of the improvements that come with the new organizational model OM. Hence, there are two ways in which property GP2 can be fulfilled by the Members. First, the Member can immediately agree with the organizational model, and as a result be unfrozen at once.

RP2(Member): No Resistance to Change
if a Member M1 is informed about a new organizational model M at time t,
then at a later point in time t2 Member M1 will inform the Change Manager upon its acceptance of the new organizational model OM
and there does not exist a time t' between t and t2 at which Member M1 has expressed resistance

to the change to the organizational model OM.

Another option is that a Member expresses temporary resistance to the change.

GP3(ChangeGroup): Belief Change after Resistance
for all Members M1 in the Change Group
if Member M1 is informed about a new organizational model OM at time t,
then at a later point in time t2 Member M1 will inform the Change Manager of its resistance to the change to the new organizational model OM,
and at a time t3 later than t2 Member M1 will inform the Change Manager of its acceptance of the new organizational model OM.

Opposition to change can be split up into several lower level properties. First, the Member opposes the change.

RP3(Member): Oppose to Change
if a Member is informed about a change to a new organizational model OM at time t,
then at a later point in time t2 Member M1 will inform the Change Manager of its resistance to the change to the new organizational model OM.

In response the Change Manager puts forward a communication that hopefully will convince the Member that organizational model OM is an appropriate option for him. Note that these terms are kept abstract on purpose as there are many ways to convince such Members in organizational change literature, and depending on the particular case a choice can be made (see also Section 6).

RP4(ChangeManager): Convince Member
if Member M1 informs the Change Manager of its resistance to the change to the new organizational model OM at time t,
then at a later point in time t2 the Change Manager will put forward additional arguments to Member M1 for the change to the organizational model OM.

Once this information is received by the Member it is assumed that he will be unfrozen.

RP5(Member): Member Convinced
if Member M1 receives additional arguments for organizational model OM at time t,
and Member M1 is convinced by the additional arguments for organizational model OM at time t,
then at a later point in time t2 Member M1 will inform the Change Manager upon its acceptance of the organizational model OM

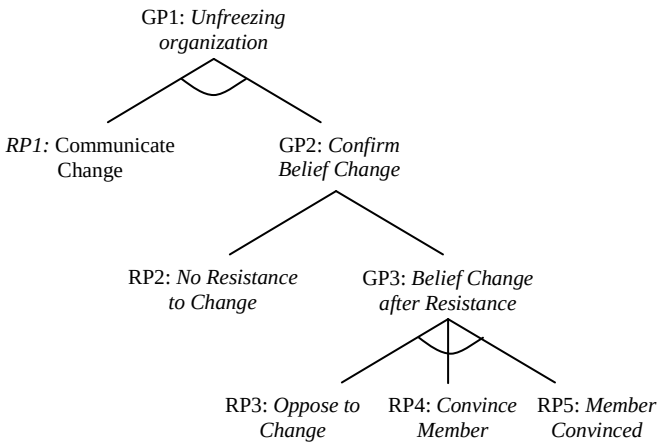

Fig. 5. Unfreezing property hierarchy specified by means of an AND/OR tree

There is also the possibility that a Member does not get convinced, which means that the Member again communicates resistance. The Change Manager can put forward more arguments in response (or use another method from organizational change theory). The possibility exists that not enough key Members of the organization communicate the acceptance of the organizational model OM, resulting in an organization which is not unfrozen. To show the relation between the different properties for a *successful* unfreezing phase, Figure 5 shows a property tree.

The tree depends upon the number of Members involved in the change of the organization, this tree covers only one Member.

4.2.2 Dynamic properties for the Movement Phase

The movement phase is rather straightforward after the unfreezing phase, in case a fraction e of the key Members have communicated their acceptance of the organizational change towards the organizational model OM, and the condition for the change to occur holds, the roles within the groups of the organization will show the behavior as specified in the organizational model OM. Property OP1 specifies this movement and is referred to as an organizational property as it also includes roles outside of the Change Group.

OP1: Successful move

if	at time t the Change Manager within the Change Group has access to a plan which specifies a condition C for a decision to reorganize based on a new organizational model OM
and	condition C is met at time t,
and	the Change Manager uses fraction e
and	at least fraction e of the key Members involved in the change have informed the Change Manager of their acceptance of the change to organizational model OM at time t,
and	organizational model OM specifies behavior B for a role R within group G at time t,
then	at a later point in time t2 role R within group G behaves according to the behavior specification B.

Satisfaction of this high level property can be accomplished by means of a group property for the Change Group and group interaction properties between the Change Group and the other groups within the organization. First, the group property states that all Members involved in the change will receive an announcement of the organizational model being activated, as expressed in GP4 below.

GP4(ChangeGroup, e): Change Activation

if	at time t the Change Manager within the Change Group has access to a plan which specifies a condition C for a decision to reorganize based on a new organizational model OM
and	condition C is met at time t,
and	at least fraction e of the key Members involved in the change have informed the Change Manager of their acceptance of the organizational model OM at time t,
then	at a later point in time t2 all Members involved in the change have received the announcement of organizational model OM being active.

This property is entailed by two lower level properties. First, the Change Manager announces the activation of the of the organizational model OM based on the conditions specified.

RP6(Change Manager): Announce Change

if	at time t the Change Manager within the Change Group has access to a plan which specifies a condition C for a decision to reorganize based on a new organizational model OM

and condition C is met at time t,
and at least fraction e of the key Members involved in the change have informed the Change Manager of their acceptance of the change to organizational model OM at time t,
then at a later point in time t2 the Change Manager announces the new organizational model OM being active.

And furthermore, this information is received by the Members via transfer property TP1.

TP1(Change Manager, Member): Transfer announcement
if at time t the ChangeManager announces organizational model OM being active
and at time t Member M1 is involved in the change to organizational model OM
then at a later point in time t2 Member M1 will receive this announcement on his input.

Finally, the group interaction properties state that after the announcement has been received by a Member role, the roles with which the Member that receives the announcement of activation will show the behavior as specified in the organizational model OM, expressed in GIP1.

GIP1(Member, ChangeGroup, R, G): New organization active
if at time t Member M1 is informed about a new organizational model OM being active,
and Member M1 has a shared allocation with a role R within group G at time t,
and role R has a behavior description B in organizational model OM at time t,
then at a later point in time t2 role R within group G behaves according to behavior B.

Figure 6 shows the property hierarchy for the movement phase.

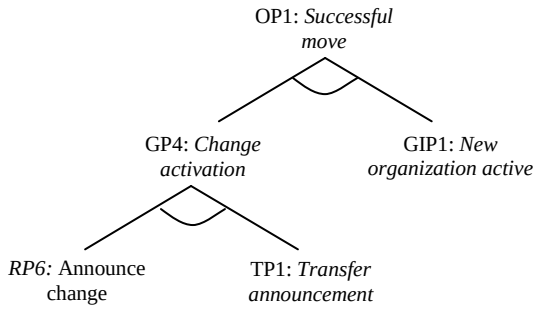

Fig. 6. Movement property hierarchy specified by means of an AND tree

4.2.3 Dynamic properties for the Refreezing Phase

The final step in the model of Lewin entails refreezing the organization. Within the model presented in this paper, this is expressed in the following way. There are two conditions to start the refreezing phase. First, the organizational model OM has been activated. Second, all roles are actually behaving according to the behavior specification. During the refreezing phase, key Members inform the Change Manager about what roles are showing the correct behavior. In case enough of these key Members (i.e. a fraction e1) communicate that a critical mass of roles (i.e. fraction e2) indeed show the correct behavior for a sufficient period of time p2, after a

conditioning phase of length p1, the refreezing phase is said to be ended successfully. This property is expressed as OP2.

OP2: Successful Refreezing
if before time t the Change Manager has informed the Members that a new organizational model OM is active,
and at time t all the roles within organization are just behaving according to behavior specification B within the organizational model OM,
and the Change Manager uses a conditioning period p1, a critical period of length p2, and fractions e1 and e2,
then there exists a time point t2 ($t2 > t + p2$) such that at t2 the Change Manager is informed by at least fraction e1 key elements that behavior B is efficiently performed by at least fraction e2 of the roles within the organization over the last period p2

The property can be accomplished by means of a group interaction property and a group property. First, the group property states that from the time point the behavior is first shown by a role R within group G, there exists a time point at which the role R has shown the correct behavior for the minimum duration p, set by the Change Manager. The fraction e2 and periods p1 and p2 are specified as parameters.

GP5(G, e2, p1, p2): Show Proper behavior
for at least a fraction e2 of the roles within group G,
if at time t a role R within group G just shows behavior B
then there exists a time point t2 ($t2 > t + p1 + p2$) such that at all time points between t2 and t2 – p2 role R within group G shows behavior B

Some roles will immediately satisfy this property within the group, as specified by property RP7. This means that the behavior shown is always according to the specified behavior.

RP7(R, p1, p2): Immediately Show Behavior
if at time t a role R within group G just shows behavior B
then for all time points t2 such that $t < t2 \leq t + p1 + p2$ role R within group G shows behavior B

Of course it is also possible that the role R within group G falls back into its old habits, not complying to the behavior specification within the new organizational model. After correction however, the role shows the correct behavior again in case of successful refreezing. Such temporarily falling back into old habits is specified in property GP6.

GP6(G, p1, p2): Show Behavior after Correction
for at least a fraction e2 of the roles within group G
if at time t a role R within group G just shows behavior B
and there exists a time point $t1 > t$ and $t1 < t + p1$ at which role R within group G does not show behavior B
then there exists a time point t2 ($t2 \leq t1 + p1 + p2$) such that at all time points between t2 and t2 – p2 role R within group G shows behavior B

Property GP6 is entailed by three lower level properties. First, RP8 expresses the improper behavior of the role R:

RP8(R, p1, p2): Improper behavior
if at time t a role R within group G just shows behavior B
then there exists a time point $t2 < t + p1$ at which role R within group G does not show behavior B

To correct this improper behavior, another role within the same group can correct role R by reminding the role of the proper behavior B:

RP9(R): Correct Improper Behavior
if at time t a role R within group G does not show the required behavior B
then at a later point in time another role R2 within group G will remind role R within group G of the proper behavior.

In a successful refreezing phase the correction indeed works, and role R returns to the correct behavior again (RP10). In case the role R is not properly refrozen such a correction might not work and therefore role R will continue to show the unwanted behavior.

RP10(R): Behave correct again
if at a time point t role R within group G is reminded by role R2 within group G of the proper behavior he should show
then for all later points in time t2 > t role R within group G shows behavior B as long as no new reorganization has been announced

Finally, GIP2 specifies that after having shown the correct behavior for a period longer than length p, the Member within the Change Group communicates this to the Change Manager.

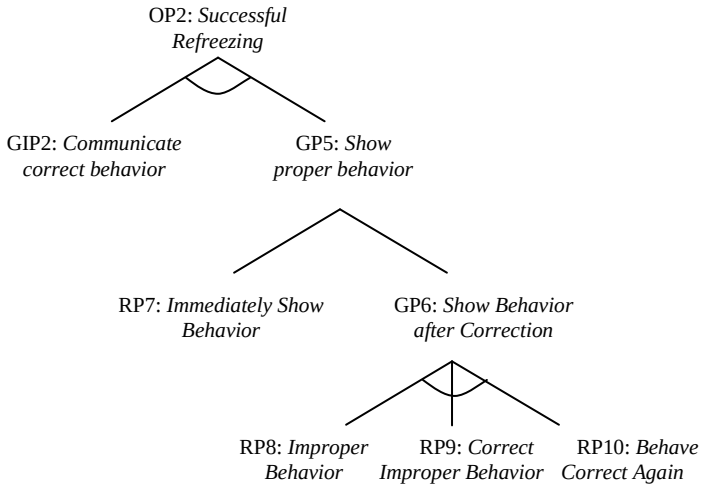

Fig. 7. Refreezing property hierarchy specified by means of an AND/OR tree

GIP2(R, G, Member, ChangeGroup, p1, p2): Communicate correct behavior
if between time point t and t2 (where t2 > t + p2) role R within group G shows the behavior according to B
and role R within group G has a shared allocation with Member M1
then at time t2 +1 Member M1 informs the Change Manager within the Change Group that behavior B is efficiently performed by role R within group G over the last period p2.

The property hierarchy for the refreezing phase is shown in Figure 7.

5 Change Language

Since communication between the Change Manager and the Members within the Change Group also concerns changes to the current organization (i.e., a new

organizational model), this section describes functions for describing such changes to be made. The sorts that have been used for this language are shown in table 4, and are basically the sorts that make it possible to use the structural and behavioral languages SL and BL introduced in Section 3. Moreover, the sort ACTION models actions that can be performed. If a conjunction of elements of ORG_ELEMENT is deleted or added, then all conjuncts are removed from or added to the model.

Table 4. Sorts used for the functions to describe organizational change.

Sort	Description
ORG_BEHAVIOR_ELEMENT	Defined by the behavioral language *BL*.
ORG_STRUCTURE_ELEMENT	Defined by the structural language *SL*.
ORG_ELEMENT	Union of the sorts ORG_BEHAVIOR_ELEMENT or and ORG_STRUCTURE _ELEMENT
ORG_PART	Conjunctions of elements from ORG_ELEMENT.
ACTION	Sort for actions.

The functions and predicates that can be used to describe organizational change are shown in table 5. The modify function is basically a combination of the delete and add function, but because it is most likely that change includes modification of certain elements it is more intuitive to include it as a function. The add function possibly takes a conjunction of ORG_ELEMENT as an input (denoted as ORG_PART), this however is impossible for the delete because this would not result in a unique system configuration. The performance of the actions is done internally within the role, resulting in a communication that the structure is in place.

Table 5. Functions and predicates used to describe organizational change.

Function or Predicate	Description
add: ORG_PART → ACTION	Add takes an ORG_PART and creates the action to add that part.
delete: ORG_ELEMENT → ACTION	Delete takes an ORG_ELEMENT and creates the action to delete it.
modify: ORG_PART x ORG_PART → ACTION	The first element models the current organization, the second specifies the modifications that need to be done. An action is constructed by means of this.
to_be_performed: ACTION	Predicate that a certain action is to be performed. This can be add, delete or modify.

An organization model for organizational change as described informally in Sections 2 and 3, involves a number of issues:

- changing internal (belief) states of all those involved in the changing organization
- changing organization structure
- taking up new roles by agents

- internal state properties of the agents involved incorporate beliefs on organization structure as well as beliefs on dynamic properties characterizing role behavior
- internal state properties (beliefs) play a role as part of the dynamic properties characterizing role behavior

A language to express dynamic properties of a changing organization has to be a rich language able to express all these aspects in combination. Such a language is defined in Appendix C as an extension of TTL [23] called meta-TTL. Note that in this language not only dynamic properties are defined on top of state properties, but also state properties (in particular beliefs) are defined on top of dynamic properties. This makes it possible to express a dynamic property built using a belief state property which itself refers to a dynamic property, and so on. So on the top level this is a dynamic property built on state properties (the beliefs), which themselves refer to state properties concerning the organization structure and to a dynamic (leads to) property again. An example property is the following, describing that a role performs the behavior it believes that is expected from the role:

```
If at time t
  a role believes that
      this role has as part of its behavior description that
          upon input v the output action w is done,
  and
    v occurs as input,
then
    at a next point in time this role will provide output w.
```

More formal details can be found in Appendix C.

6 Simulation of the Case Study: the Eleven Cities Tour example

This Section presents a case study to illustrate the usage of the organizational change model as presented in the previous Sections. First, the organization under investigation is explained and thereafter simulation results are presented as well as domain specific properties that have been used to enable a simulation.

6.1 Case study description

The organization model of organizational change has been applied to the organization that is responsible for the famous Frisian skating tour called the Eleven Cities Tour. The association is called “De Friesche Elf Steden” in Dutch.

Although the association has fixed parts in the organization, it also has an annual dynamics in its structure. The association has a board consisting of 3 members namely the Chairperson, the Treasurer, and the Secretary. The Board has two responsibilities: running the association smoothly at all times and organizing the tour. Most of the year only the board is active, but there is also a permanent group which

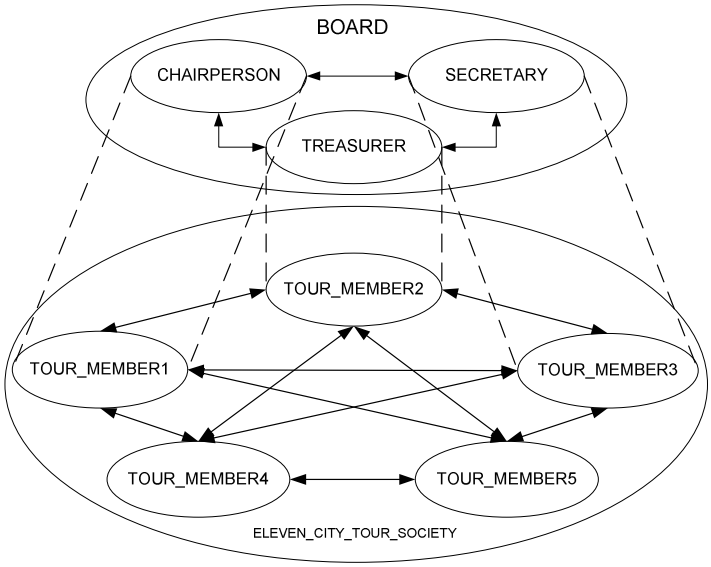

Fig. 8. Off-season eleven cities tour organization

contains all members of the eleven cities tour society, which includes the people within the Board as well. This off-season organization is shown in Figure 8. Once a year, at the beginning of winter, the organization changes its structure by formation of Region groups and the election of Region Heads for the coming winter season to enable monitoring of the ice conditions. This change process takes place within the Eleven Cities Tour Society group where the Member with a shared allocation to the Chairperson in the board is in charge of the change process. In the real organization, 21 Region groups are formed, for the case study however only the groups for the cities of Woudsend and Sneek are assumed to be created. The Region groups consist of more roles than the Region Head role (Monitor roles), however these roles have been left out of the case study for the sake of clarity. The election of the Region Heads is always a difficult part of the organizational change, as many people resist to the election of certain people because they think these people are not suitable for the job, or because they prefer another candidate, but in this case study we only consider suitability. Once the Regions have been formed and the Region Heads have been appointed, they start their work of monitoring the ice condition along the route. After certain conditions are met, such as a certain period of frost, another change occurs within the organization: A group called Region Representatives is formed which consists of representatives of all Region groups and representatives of the Board. This group discusses the conditions along the entire trajectory of the tour. If the conditions are good, this group organizes the Tour. The organization after formation of the Regions and Region Representatives group is shown in Figure 9. Note that the shared allocations between the members of the Board and the representatives of these in the Region Representatives group have been omitted to keep the Figure clear. To ensure

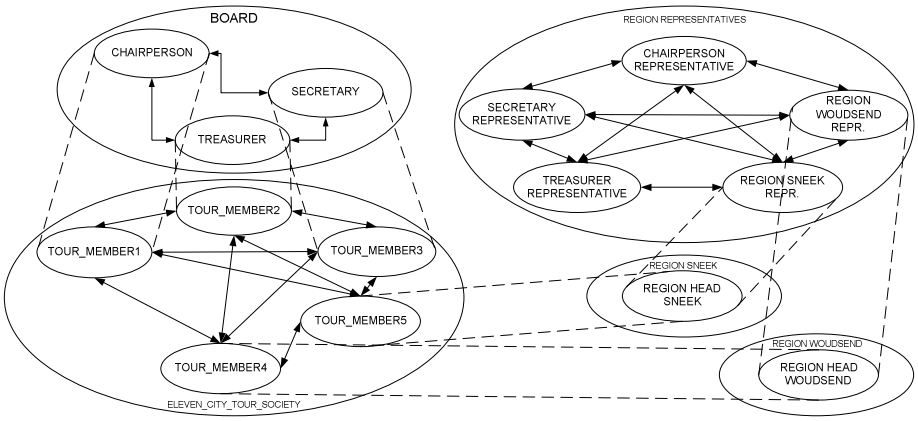

Fig. 9. Eleven cities tour organization after formation of the Regions and Region Representatives groups.

that indeed all roles within the Region Representatives group show the desired behavior, the Chairperson Representative monitors whether the representatives of each of the Regions are indeed behaving according to the specification and do not fall back into their prior behavior.

Finally, at the end of the winter, the Chair of the Meeting of Region Heads thanks all participants and deactivates all roles in that group as well as all Region Head role instances. At this point in time the agents are de-allocated from their roles, and the roles immediately cease to exist. The involved agents only remain allocated to the continuous roles / role instances in the Board and Eleven City Tour Society group.

6.2 Simulation Results

Based on the generic properties as specified in Section 4, a domain specific simulation model for the eleven cities tour has been created. All of the properties that underline the basis of this model have been specified in the *leadsto* format as introduced in Section 3. Since this format is executable, simulations can be performed using the *leadsto* software tool [7]. This Section presents a selection of the simulation results, and gives example of the domain specific properties that have been used for the simulation. Furthermore, several events are put into the model to see how well the organization changes in case this is required. The results have been ordered based on the different phases in organizational change distinguished by Lewin.

6.2.1 Initial organization

The initial organizational setup for the simulation is shown in Figure 10. On the left hand side of the Figure statements are shown about the organization whereas the right

Fig. 10. Initial setup of the organization for the simulation.

hand side presents a timeline where a black box indicates that the statement is true at that particular time point. The Eleven Cities Tour Society group is called Change Group within the initial organization since this group's only function is organizational change, hence it is considered a Change Group. Note that the Figure only presents part of the initial organization: only a selection of the intra and inter group interactions, and only the beliefs of the Change Manager are shown. The Figure for example shows the presence of the role Chairperson:

internal(GlobalChangeManager|ChangeGroup)|belief(exists_role(Chairperson))

Furthermore, the existence of the group Board is shown:

internal(GlobalChangeManager|ChangeGroup)|belief(exists_group(Board))

The role Chairperson is specified to be part of the Board group:

internal(GlobalChangeManager|ChangeGroup)|belief(role_belongs_to_group(Chairperson, Board))

Intra group interaction is part of the Board group as well, the Secretary can for example communicate with the Chairperson:

internal(GlobalChangeManager|ChangeGroup)|belief(intra_group_connection(Secretary, Chairperson, Board, t1))

And finally, inter group connections are part of the beliefs of the Change Manager. The inter group connection shown is the one between the Chairperson in the Board and the Change Manager within the Change Group:

internal(GlobalChangeManager|ChangeGroup)|belief(inter_group_connection(Chairperson, Board, GlobalChangeManager, ChangeGroup, gi1))

This inter group connection is based on shared allocation, which means that the agent playing the role of Chairperson within the Board also plays the role of Change Manager within the Change Group. Within the Figure, the Change Group consists of five Member roles and one Change Manager. The additional Members in the Change Group are played by agents that are not yet part of the organization, but can be used by the Change Manager for the fulfillment of new roles to be played.

6.2.2 Unfreezing phase for region formation

After the initial setup of the organization, an event is put into the simulation which

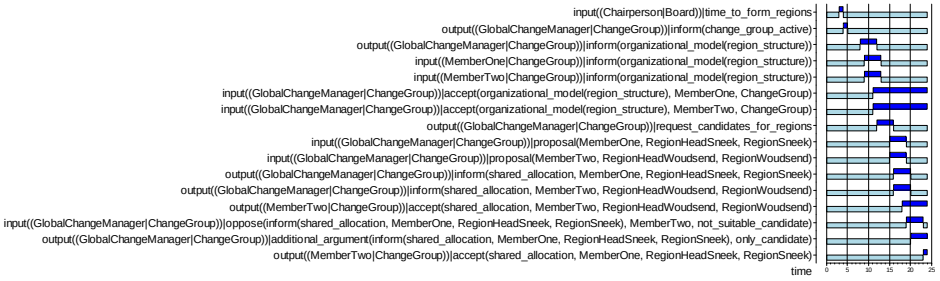

Fig. 11. First unfreezing phase during simulation

requires an organizational change, namely the onset of winter meaning that it is time to form the regions within the organization. The first phase within this change process is unfreezing. The occurrences during this phase are shown in Figure 11. The event requiring change is the Chairperson within the board observing that it is time to form the regions:

input(Chairperson|Board)|time_to_form_regions

An inter-group interaction property in the form of a leadsto rule now fires which specifies that if the Chairperson within the Board observes it is time to form the regions, the Change Manager activates the Change Group and announces the organizational model for the region structure:

GIP_specific(Chairperson, Board, ChangeManager, ChangeGroup): Form Regions when winter

if	at time t the Chairperson within the Board observes that it is time to form the Region groups
then	at time t + 1 the Global Change Manager within the Change Group informs the Members within the Change Group that the group is now active
and	at time t + 1 the Global Change Manager within the Change Group announces the new organizational model regarding the Regions.

The results of this rule show in the trace by the following elements:

output(GlobalChangeManager|ChangeGroup)|inform(change_group_active)
output(GlobalChangeManager|ChangeGroup)|inform(organizational_model(region_structure))

Only a reference i.e. the statement region_structure, to the whole specification of this organizational structure is presented in the Figure for the sake of clarity. None of the Members oppose the change as all are skating fanatics that long for a tour and all are convinced that winter has started. For them the onset of winter naturally means the formation of regions, so all communicate the acceptance of the organizational model, for example Member One:

input(GlobalChangeManager|ChangeGroup)|accept(organizational_model(region_structure), MemberOne, ChangeGroup)

The unfreezing for this particular organizational structure is therefore accomplished, following RP2 as described in Section 4. Another element of the change is to allocate the appropriate agents to the specific roles within the new organizational model: appoint the Region Heads. For this a more complicated unfreezing phase is performed.

First, the Change Manager within the Change Group requests candidates for the newly formed roles:

output(GlobalChangeManager|ChangeGroup)|request_candidates_for_regions

The Members within the ChangeGroup receive the request and propose candidates for the positions, based upon their availability during the winter:

input(GlobalChangeManager|ChangeGroup)|proposal(MemberOne, RegionHeadSneek, RegionSneek)

input(GlobalChangeManager|ChangeGroup)|proposal(MemberTwo, RegionHeadWoudsend, RegionWoudsend)

After receiving the proposals, the Change Manager decides upon an optimal allocation. Since there are two roles that need to be fulfilled and there is one proposal per role, these allocations are chosen and communicated:

output(GlobalChangeManager|ChangeGroup)|inform(shared_allocation, MemberOne, RegionHeadSneek, RegionSneek)

output(GlobalChangeManager|ChangeGroup)|inform(shared_allocation, MemberTwo, RegionHeadWoudsend, RegionWoudsend)

Member Two is however not convinced about the suitability of Member One for the role of Region Head Sneek and opposes to the organizational model following a domain specific instantiation of RP3 in Section 4:

RP3_specific(MemberTwo): Oppose to Change

if	at time t Member Two is informed about a change to an organizational model M in which a Member M1 has a shared allocation to a Role R1
and	Member Two observes M1 is unsuitable for the Role R1 at time t
then	at time t + 1 Member Two opposes to the change to the organizational model stating that M1 is not suitable for R1

The result of this rule is shown in the trace by the following statement:

input(GlobalChangeManager|ChangeGroup)|oppose(inform(shared_allocation, MemberOne, RegionHeadSneek, RegionSneek), MemberTwo, not_suitable_candidate)

As a result a domain specific instantiation of RP4 fires which is specified below.

RP4_specific(ChangeManager): Convince Member

if	at time t a Member M1 communicates opposing to the change to organizational model M to the Change Manager because candidate M2 is considered not suitable for the allocation to role R1
and	the Change Manager observed M2 is the only candidate for the role R1 at time t
then	at time t + 1 the ChangeManager communicates that M2 is the only candidate for role R1.

In the trace, the communication can be seen in the following format:

output(GlobalChangeManager|ChangeGroup)|additional_argument(inform(shared_allocation, MemberOne, RegionHeadSneek, RegionSneek), MemberTwo, only_candidate)

Finally, a rule RP5 is specified for this domain as well, as shown below. Since the successful organization of an eleven cities tour is most important for the Members and all roles being allocated is essential for such a successful organization, they seize to oppose to an allocation in case they are informed about the existence of only one candidate.

RP5_specific(Member Two): Member Convinced

if	at time t Member Two opposes to the change to the organizational model M regarding the allocation of Member M1 to Role R1
and	at time t2 later than t Member Two receives the argument that Member M1 is the only candidate for role R1
then	at time t2 + 1 Member Two will inform the Change Manager upon its acceptance of the change to the organizational model M

In the trace the Member indeed outputs the belief upon the shared allocation:

output(MemberTwo|ChangeGroup)|accept(shared_allocation, MemberOne,
RegionHeadSneek, RegionSneek)

Since all Members have now communicated their acceptance of the new organizational model, the unfreezing phase is performed successfully.

6.2.3 Movement and Refreezing of the region formation

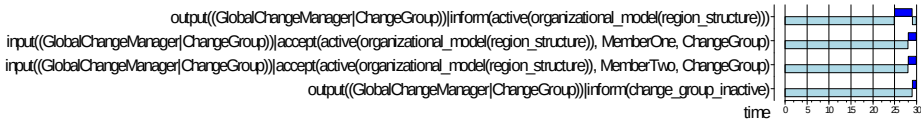

Fig. 12. First movement and Refreezing

The movement and refreezing phase for the case study are much shorter than the unfreezing phase, as the new organizational model is already accepted by all Members of the organization. The two phases are shown in Figure 12. Trigger for the ChangeManager to start the movement phase is when an acceptance on all parts of the organizational model M has been communicated to the Change Manager, as specified before in RP6. The movement phase starts with the communication of the region structure being active:

output(GlobalChangeManager|ChangeGroup)|
inform(active(organizational_model(region_structure)))

The phase ends after all participants of the change have confirmed that the organizational model will be active, which they instantly do as they are already unfrozen:

input(GlobalChangeManager|ChangeGroup)|
accept(active(organizational_model(region_structure)), MemberOne, ChangeGroup)
input(GlobalChangeManager|ChangeGroup)|
accept(active(organizational_model(region_structure)), MemberTwo, ChangeGroup)

Finally, the refreezing phase ends after the duration set by the Change Manager. In this particular refreezing phase, all roles immediately behave correctly after the change (according to RP7 in Section 4.2.3) which is not shown in the trace for the sake of brevity. Eventually, the Change Group is deactivated:

output(GlobalChangeManager|ChangeGroup)|inform(change_group_inactive)

6.2.4 Unfreezing phase for regions representatives group

The second unfreezing phase which is required to form the Region Representatives group is shown very briefly in Figure 13. As a start of the unfreezing phase the following events are put into the simulation:

input(Chairperson|Board)|one_week_frost_period_just_passed
input(Chairperson|Board)|before_that_week_no_frost
input(Chairperson|Board)|no_tour_held_this_winter

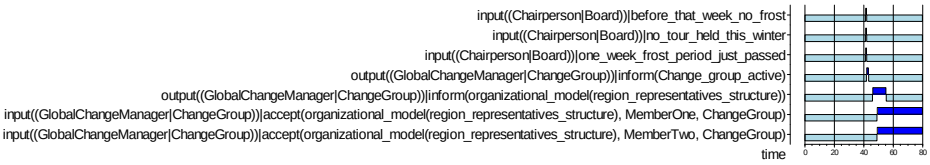

Fig. 13. Second unfreezing phase

As a result, the following inter group interaction property fires:

GIP_specific(Chairperson, Board, ChangeManager, ChangeGroup): Form Region Representatives group after frost period

if	at time t the Chairperson within the Board observes a period of one week of frost
and	at time t – (1 week) the Chairperson within the observed that there was no frost
and	at no time point this year the Chairperson within the Board observed that a tour has been held
then	at time t + (1 day) the Change Manager within the Change Group informs the Members within the Change Group that the group is now active
and	at time t + (1 day) the Change Manager within the Change Group announces the new organizational model regarding the Regions Representatives group.

The resulting communication of the Change Manager is shown in the trace: A communication of the Change Group being active again, and communication of the new organizational model:

```
output(GlobalChangeManager|ChangeGroup)|inform(change_group_active)
output(GlobalChangeManager|ChangeGroup)|
 inform(organizational_model(region_coordination_structure))
```

All Members accept the new structure, as they are very eager just thinking about a possible eleven cities tour, the event of the year, and are therefore immediately unfrozen, communicating their acceptance to the Change Manager. Therefore, the unfreezing is performed using RP2. Resistance can however easily be incorporated using properties such as presented in Section 6.2.2. The unfreezing process has now ended successfully.

6.2.5 Movement and Refreezing of the region representatives group

After unfreezing the organization, the Change Manager communicates that the new organization with the new Region Representatives structure is now active, which is shown in the partial trace in Figure 14:

```
output(GlobalChangeManager|ChangeGroup)|inform(active(
organizational_model(region_coordination_structure))
```

As a result, the organizational model becomes active in the actual organization, not only in the internal beliefs of the Members of the Change Group. Within the simulation there is a mapping between the name of general organizational structures (e.g. region coordination structure) and the actual changes on a lower level. For the Region Head Woudsend for example, the internal belief that a new role Region Representative Woudsend exists is added:

```
internal((RegionHeadWoudsend|RegionWoudsend))|
 belief(exists_role(RegionRepresentativeWoudsend))
```

Fig. 14. Second Movement and refreezing

Furthermore, the group Region Representatives is added to the internal beliefs:

```
internal((RegionHeadWoudsend|RegionWoudsend))|
 belief(exists_group(RegionRepresentatives))
```

The role RegionRepresentativeWoudsend belongs to the group RegionRepresentatives:

```
internal((RegionHeadWoudsend|RegionWoudsend))|belief(role_belongs_to_group(
 RegionRepresentativeWoudsend, RegionRepresentatives))
```

A belief on an inter-group connection is added between the Region Head Woudsend within the Region Woudsend and the Region Representative Woudsend within the Region Representatives:

```
internal((RegionHeadWoudsend|RegionWoudsend))|
 belief(inter_group_connection(RegionHeadWoudsend, RegionWoudsend,
 RegionRepresentativeWoudsend, RegionRepresentatives, gi24))
```

Besides the structure itself, the new roles also require new behavior. When first starting to perform a new role, the new behavior associated with the role is far from automated, and requires internal beliefs on the desired behavior. Such elements are shown in the trace of Figure 9 as well. First of all, there is an internal belief about the existence of a group interaction property gip1:

```
internal((RegionHeadWoudsend|RegionWoudsend))|belief(group_interaction_property(gip1,
 RegionHeadWoudsend, RegionWoudsend, RegionRepresentativeWoudsend,
 RegionRepresentatives))
```

The specification of the behavior required by such a property is done using a TTL expression, more particular in leadsto format:

```
internal((RegionHeadWoudsend|RegionWoudsend))|belief(has_expression(gip1,
 leads_to(
   (input((RegionHeadWoudsend|RegionWoudsend))|report(RegionWoudsend,good)),
    output((RegionRepresentativeWoudsend|RegionRepresentatives))|
        report(RegionWoudsend, good),
    efgh(0,0,1,1))))
```

This specifies that if the Region Head Woudsend receives a report that the ice is good, then this will be communicated by the Region Representative Woudsend in the Region Representatives group as well, with an efgh value of (0,0,1,1). Around time point 65 the antecedent of this rule becomes true, however, the consequent is not true after 1 time point within the Region Representatives group. As a result, the Chairperson Representative within the Region Representatives group reminds the role of the desired behavior gip1 (according to RP9 in Section 4.2.3):

```
output((CharipersonRepresentative|RegionRepresentatives))|remind(gip1)
```

After having received this reminder, the Region Representative Woudsend does behave according to gip1 and outputs the consequent:

```
output((RegionRepresentativeWoudsend|RegionRepresentatives))|report(RegionWoudsend,
 good)
```

This refreezing therefore takes the form of GP6 (Section 4.2.3) and the properties below it in the property tree. In exceptional years, all Region Representatives report that the ice is good, and the Chairperson within the board announces the date the tour will take place:

```
output((Chairperson|Board))|let_the_tour_be_held_on_date
```

7 Verification of the Case Study Simulation

As verification of the organization process of the Eleven Cities Tour is concerned, a distinction is made between two types of verification. Firstly, guarantees are given that concern the tour itself (so-called content properties). For example, it the circumstances permit so (if the ice is thick enough over the whole trajectory) then a tour should be organized as soon as possible. Secondly, guarantees on the organization of organizational change for setting up the tour are verified (so-called organizational change properties). This Section presents both verification types.

Logical relationships between properties, as depicted in the tree of Section 4, can be very useful in analyzing the dynamic properties of an organization. For example, if for a given trace of the system some global property OP is not satisfied, then by a refutation process it can be concluded that either one of the group properties, or one of the group interaction properties in the tree does not hold. If, after checking these properties, it turns out that a group property does not hold, then either one of the role properties or the intra group interaction properties is not satisfied. By this refutation analysis it follows that if OP does not hold for a given trace, then, via the intermediate properties, the cause of this malfunctioning can be found in the set of leaves of the tree of Section 4.

In order to determine which one of the properties encountered in this refutation process actually is refuted, some mechanism is needed to check if a certain property holds for a given trace. To this end, the simulation software described in Section 6 automatically produces log files containing the traces. In addition, software has been developed that is able to read in these log files together with a set of dynamic properties (in TTL format), and to perform the checking process. Traces are thus analyzed with an automated logic-based checker. This checker takes as input a property of interest about the trace and logically validates whether the property holds in the given trace. If the property holds in the trace, the checker outputs success otherwise it outputs failure. But the software determines not only whether the properties hold for the trace or not, but in case of failure, it also pinpoints which parts of the trace violate the properties. The results of different checks that have been performed are described below.

7.1 Content Properties

The overall goal of the Eleven Cities Tour organization is to arrange for a tour to be organized when possible, i.e., when the ice along the tour is thick enough to ensure a safe passage. This following property expresses this goal: the tour has to be organized whenever possible, ensuring a safe passage over the ice for all skaters.

OP3: Organize tour in case of good conditions
if the ice conditions in all regions are good
then it is announced that the tour will be held

This property has been checked against the simulation trace that was presented in the previous Section and is indeed satisfied within that trace. Other content properties to

consider in this context are, for example, the organization daily decides on the possibility and date (if appropriate) of a tour: ‘it giet oan’ (in Frisian language a go decision) decisions, and in wintertime, the organization daily monitors the weather. However, only OP3 is addressed in this paper.

7.2 Organizational Change Properties

The properties as presented in the previous Section depend on some organizational structure to ensure the fulfillment of each property and all of them combined. For this purpose, the aim of this paper is exactly this: a way to specify and model such an organization itself has been presented, as well as the actual process of setting up the organization. As such, this organization can support the organizational properties as presented above.

For the purpose of verifying the organizational change in the Eleven Cities Tour simulation, automatic checking of the high-level properties presented in Section 4.2 has been performed on the generated trace. The results are shown in Table 6. In the simulation trace, there are 2 change moments: the formation of the regions structure and the formation of the region representatives group. For the last changes, there is no resistance to the organization change, while there is for the first one. The automated checker has verified that all properties specifying a successful phase are indeed satisfied, hence, both changes have passed a successful unfreezing, movement, and refreezing phase. There is however a difference in how this success was accomplished. In the first change, property GP3 was satisfied, specifying that there was resistance to the change which was taken away. In the second change however, the change went without resistance; property GP3 was not satisfied in that change. In the refreezing phase of the first change, property GP6 was not satisfied as no improper behavior was encountered. In the second change however, improper behavior did show, after which the behavior was corrected, satisfying property GP6. The following setting were used for checking. For the unfreezing phase e was set to 1.0. Regarding the refreezing phase both e1 and e2 have been set to 1.0, for p1 a value of 10 was used, and finally, p2 was set to 20.

Table 6. Checked Properties (Yes = satisfied, No = not satisfied)

Stage	**Property**	**Change 1**	**Change 2**
Unfreezing	GP1	Yes	Yes
	GP3	Yes	No
Moving	OP1	Yes	Yes
Refreezing	OP2	Yes	Yes
	GP6	No	Yes

8 Conclusions

Organizations often have to survive in a dynamic world. To enable organizations in practice to adapt to the dynamics of the world, certain facilities, structures and capabilities are needed that support organizational change. This paper shows how the organization of organizational change processes can be modeled within a formal organization modeling approach. A generic organization model for organizational change was presented and formally verified for a case study concerning the organization of a major event in the Netherlands: the eleven cities tour. The formal verification sets it apart from existing work on organization modeling, e.g., [15; 32]. Previous work of the authors on organizational change [25] considered change as an instantaneous event instead of a process of change as is done in this paper. Additionally, previous work did not include the distinction between formal languages for expressing the change process. The change model in this paper takes into account different phases in a change process (unfreezing, movement and refreezing) considered in [27], which is still considered valid in current organizational change literature, see e.g. [30; 29]. In [29] a distinction is made between anticipated change (for which the model of Lewin is said to be suitable), emergent change and opportunity-based change. In this paper only anticipated change is being modeled and therefore the other two types of change are not addressed. In change processes the internal (mental) states of those involved in the organization are important. Therefore, also internal states of individuals have to be part of a model for organizational change. In particular, beliefs and their changes have been incorporated in the model. In addition, an internal model for (reflective) reasoning about expected role behavior was included. Hence, a model was created that combines organization aspects and cognitive aspects.

Acknowledgements

This research has been performed as part of two projects: CIM, for Cybernetic Incident Management, and DEAL, for Distributed Engine for Advanced Logistics. Both projects are funded by the Dutch Ministry of Economic Affairs. Furthermore, the authors would like to thank the anonymous reviewers for their useful comments that helped to improve the paper.

References

[1] Abbink, H., Dijk, R. van, Dobos, T., Hoogendoorn, M., Jonker, C.M., Konur, S., Maanen, P.P. van, Popova, V., Sharpanskykh, A., Tooren, P. van, Treur, J., Valk, J., Xu, L., Yolum, P., (2004). Automated Support for Adaptive Incident Management. In: Walle, B. van de, and Carle, B. (eds.), *Proc. of the First International Workshop on Information Systems for Crisis Response and Management, ISCRAM'04*, pp. 69-74..

[2] Ackerman, L.S., Development, transition, or transformation: the question of change in organization, OD Practitioner, 18(4) , 1986, pp. 1-9.

[3] Alvesson, M., Cultural Perspectives on Organizations, Cambridge University Press, New York, 1993.
[4] Bacharach, S.B. and Lawler, E.J., Power and politics in organizations, Jossey-Bass, San Francisco, 1980.
[5] Bashein, M.L., Marcus, M.L., and Riley, P., Business Process Reengineering: preconditions for success and failure, Informations Systems Management 9, 1994, pp. 24-31.
[6] Boonstra, J.J. (editor), Dynamics of Organizational Change and Learning, Wiley, 2004.
[7] Bosse, T., Jonker, C.M., Mey, L. van der, and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., et al. (eds.), Proc. of the Third German Conference on Multi-Agent System Technologies, MATES'05. Lecture Notes in Artificial Intelligence, vol. 3550. Springer Verlag, 2005, pp. 165-178.
[8] Bradshaw, P., and Boonstra, J.J., Power Dynamics in Organizational Change, In: Boonstra, J.J. (editor), Dynamics of Organizational Change and Learning, Wiley, 2004, pp. 279-299.
[9] Cummings, T.G., and Worley, C.G., Organization Development and Change, South Western College Publishing, 2001.
[10] Cummings, T.G., Organization Development and Change, In: Boonstra, J.J. (editor), Dynamics of Organizational Change and Learning, Wiley, 2004, pp. 25-42.
[11] Dahl, R.A., The concept of power, Behavioral Science, 2, 1975, pp. 201-215.
[12] Emerson, R.M., Power dependence relations, American Sociological Review, 27, 1962, pp. 31-41.
[13] Ferber, J. and Gutknecht, O. (1998), "A meta-model for the analysis and design of organizations in multi-agent systems," Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.
[14] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J. (2001), "Organization Models and Behavioural Requirements Specification for Multi-Agent Systems," in Y. Demazeau, F. Garijo (Eds.), Multi-Agent System Organizations. Proceedings of MAAMAW'01.
[15] Fox, M.S., and Gruninger, M. (1998), "Enterprise Modelling,". AI Magazine, 19(3), AAAI Press, pp. 109-121.
[16] Glaser, N., and Morignot, P. (1997), "The Reorganization of Societies of Autonomous Agents," in Boman, M. Velde, W. van de (eds.), Multi-Agent Rationality, 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Lecture Notes in Computer Science, vol. 1237, Springer, pp. 98-111.
[17] Hall, G., Rosenthal, T., and Wade, J. How to make reengineering really work, *Harvard Business Review*, 71(6), 1993, pp. 119-131.
[18] Hoogendoorn, M., Jonker, C. M., Konur, S., Maanen, P.P. van, Popova, V., Sharpanskykh, A., Treur, J., Xu, L., Yolum, P., (2004). Formal Analysis of Empirical Traces in Incident Management, In: Macintosh, A., Ellis, R., and Allen, T. (eds.), *Applications and Innovations in Intelligent Systems XII, Proceedings of AI-2004, the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer Verlag, 2004, pp. 237-250.
[19] Hoogendoorn, M., Jonker, C.M., Popova, V., Sharpaskykh, A., Xu, L., Formal Modelling and Comparing of Disaster Plans. (2005). In: Carle, B., and Walle, B. van de, (eds.), *Proceedings of the Second International Conference on Information Systems for Crisis Response and Management ISCRAM '05*, pp. 97-107.
[20] Hosking, D.M., Social construction as process: some new possibilities for research and development, *Concepts and Transformation*, 4(2), 1999, pp. 117-132.
[21] Huczynski, A., and Buchanan, D. (2001), *Organizational Behaviour*, Prentice Hall.
[22] Jaffee, D. (2001), *Organization Theory – Tension and Change*, McGraw-Hill Companies.

[23] Jonker, C.M., and Treur, J. (2002), "Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness," International Journal of Cooperative Information Systems, 11, pp. 51-92.
[24] Jonker, C.M., and Treur, J. (2003), "Relating Structure and Dynamics in an Organization Model," In J.S. Sichman, F. Bousquet, and P. Davidson (eds.), Multi-Agent-Based Simulation II, Proceedings of the Third International Workshop on Multi-Agent Based Simulation, MABS'02. Lecture Notes in AI, vol. 2581, Springer Verlag, pp. 50-69.
[25] Jonker, C.M., Schut, M.C., and Treur, J. (2003), "Modelling the Dynamics of Organizational Change," In Klusch, M., Omicini, A., and Ossowski, S., and Laamanen, H., (eds.), Cooperative Information Agents VII, Proceedings of the Seventh International Workshop on Cooperative Information Agents, CIA 2003, Lecture Notes in AI, vol. 2782, Springer Verlag, pp. 336-344.
[26] Kotter, J.P., Leading Change, Harvard Business School Press, Boston, 1999.
[27] Lewin, K. (1951), *Field Theory in Social Science*, Harper & Row, New York.
[28] Lippit, R., Watson, J., Westley, B., The Dynamics of Planned Change, Harcourt, New York, 1958.
[29] Orlikowski, W. and Hofman, D. (1997), "An Improvisational Model of Change Management: The Case of Groupware Technologies," Sloan Management Review vol. 38 (2), pp 11-22.
[30] Robbins, S.P. (1998), *Organizational Behaviour*, Prentice Hall, New Jersey.
[31] Schein, E.H., On dialogue, culture, and organizational learning, *Organizational Dynamics* 22(2), 1993, pp. 40-51.
[32] Steen, M.W.A., Lankhorst, M.M. and Wetering, R.G. van de (2002), "Modelling Networked Enterprises," Proceedings of the 6th International Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society, pp. 109-119.
[33] Wrong, D.H., Some problems in defining social power, *American Journal of Sociology*, 73, 1968, pp. 673-681.

Appendix A The Temporal Trace Language TTL: more formal details

A state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of some role r within a group g at time point t is denoted by

state(γ, t, input(r|g))

analogously

state(γ, t, output(r|g))
state(γ, t, internal(r|g))

denote the output state and internal state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r|g)) |= p denotes that state property p holds in trace γ at time t in the output state of role r within group g. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts TIME or T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. In trace descriptions, notations such as state(γ, t, output(r|g))|= p are shortened to output(r|g)|p.

Appendix B Constraints on the Language Elements

Role dynamic properties

```
if      has_expression(p:DYNPROP, d:DYNPROPEXP)
  and   role_property(p, r:ROLE, g:GROUP)
then    element_of(d, DYNPROPEXP(r|g, ONT(r|g)))
```

The group is also part of the definition of the ontology since roles in different groups can have the same name and might have a different ontology.
Role properties can be divided into different types which in turn can be defined more restricted than the general definition. An example of such a refinement is an executable role dynamic property. This special type is defined as follows:

```
if      has_expression(p:DYNPROP, d:DYNPROPEXP)
  and   role_property(p, r:ROLE, g:GROUP)
then    element_of(d, DYNPROPEXP((r|g), role_input_ontologies(r|g) ∪
          role_ouput_ontologies(r|g)))
```

Transfer dynamic properties

```
if      has_expression(p:DYNPROP, d:DYNPROPEXP)
  and   transfer_property(p, r1:ROLE, r2:ROLE, g:GROUP)
then    element_of(d, DYNPROPEXP({r1|g, r2|g}, role_output_ontologies(r1|g) ∪
                                    role_input_ontologies(r2|g)))
```

Group dynamic properties

```
if      has_expression(p:DYNPROP, d:DYNPROPEXP)
  and   group_property(p, g:GROUP)
then    element_of(d, DYNPROPEXP(g, ONT(g)))
```

Intergroup interaction dynamic properties

```
if      has_expression(p:DYNPROP, d:DYNPROPEXP)
  and   group_interaction_property(p, r1:ROLE, g1:GROUP, r2:ROLE, g2:GROUP)
then    element_of(d, DYNPROPEXP({r1|g1, r2|g2}, role_input_ontologies(r1|g1)
                                    ∪ role_output_ontologies(r2|g2)))
```

Appendix C Changing Organizations Formalized in meta-TTL

This is the formal part from Section 5.

C.1 Sorts and Subsorts in meta-TTL

Table C.1. Sorts in meta-TTL

Sort	Description
TRACE	for traces
STATE	for states within a trace.
T	time frame.
STATOMS	expressions for state atoms.
CONSTATOMS	expressions for conjunctions of state atoms.
STATPROPEXP	expressions for state properties.

The sorts that are included in meta-TTL are shown in Table C.1. The subsort relation STATOMS $\subseteq$ CONSTATOMS holds.
The function

and: CONSTATOMS x CONSTATOMS $\rightarrow$ CONSTATOMS

is used to build conjunctions of state atoms; it is also written as $\wedge$ in infix notation
Furthermore, the relation <: T x T for time ordering is used , and the function

state: TRACE x T x PART $\rightarrow$ STATE

that indicates the state of part of the considered system within a trace at some point in time.
For the changing organization it is needed to use names and expressions for dynamic properties within other formulae. Therefore two sorts

DYNPROP names for dynamic properties
DYNPROPEXP expressions for dynamic properties

have been introduced in the Appendix A.
Moreover,

holds: STATE x STATPROPEXP $\rightarrow$ DYNPROPEXP

indicates the dynamic property that a state property expression is true in a state; this predicate holds is often written as |= in infix notation.

C.2 Example formalization in change language

By means of an example the use of the functions combined with the language is shown below.

to_be_performed(delete(exists_role(RoleTwo))) ∧
to_be_performed(delete(role_belongs_to_goup(RoleTwo, Group1))) ∧
to_be_performed(delete(intra_group_connection(RoleOne, RoleTwo, Group1, t1))) ∧
to_be_performed(delete(intra_group_connection(RoleTwo, RoleOne, Group1, t1))) ∧
to_be_performed(delete(transfer_property(tp1, RoleOne, RoleTwo, Group1))) ∧
to_be_performed(delete(has_expression(tp1, {expression1}))) ∧
to_be_performed(delete(transfer_property(tp2, RoleTwo, RoleOne, Group1))) ∧
to_be_performed(delete(has_expression(tp2, {expression2})))

SL

BL

The example models the deletion of Role One from Group1. Both specification languages have been used to model this change as is shown by the braces at the side.

C.3 Building properties for the changing organization

In a change process it is needed that the roles have beliefs about the organization structure. Therefore all organization structure representations described in Section 4 are included ; some examples are shown in Table C.2,

Table C.2. Examples of included organization structure representations

exists_role : ROLE → STATPROPEXP
role_belongs_to_group: ROLE x GROUP → STATPROPEXP
role_property: DYNPROP x ROLE x GROUP → STATPROPEXP
has_expression: DYNPROP x DYNPROPEXP → STATPROPEXP
allocated_to: AGENT x ROLE x GROUP → STATPROPEXP

Moreover, to express beliefs, the following language construct is used :

belief: STATPROPEXP → STATPROPEXP

An example of its use is: belief(exists_role(s) ∧ role_belongs_to_group(s, g))

Furthermore it is needed that the roles have beliefs about the behavioral properties that are expected from a certain role. Therefore first a representation

leads_to: CONSTATOMS x CONSTATOMS → DYNPROPEXP

is introduced for a simple type of such properties. A more general type of dynamic property is built using:

& : DYNPROPEXP x DYNPROPEXP → DYNPROPEXP

and similarly for other logical connectives such as not, ⇒, ∀, ∃.

Thus within the sort DYNPROPEXP two types of expressions are built:

- temporal statements based on atoms of the form state(γ, t, P) |= p for state properties p
- leads to statements of the form leads_to(V, W) with V and W conjunctions of atoms

Although the latter type of expressions can be mapped to (are definable in terms of) the former type of expressions, for simplicity they are kept separate.

An example of an expression that can be built using the constructs above is the following

∃t state(γ, t, internal(r)) |= belief(exists_role(s) ∧ role_belongs_to_group(s, g)) ∧
belief(role_property(d1, s, g)) ∧
belief(has_expression(d1, leads_to(a∧b, c)))

This expression states that

there will be a time that
within role r there is the belief that
the organization structure includes role s in group g, and
this role has dynamic property d1 which
is expressed by leads_to(a∧b, c).

Another example property is the following, describing that a role performs the behavior it believes that is expected from the role:

If at time t
a role believes that
this role has as part of its behavior description that
upon input v the output action w is done,
and
v occurs as input,
then
at a next point in time this role will provide output w.

Here the nesting is visible in the informal structured text representation using tabs. The formalization of this property also shows a nesting as indicated.

[state(γ, t, internal(RegHead)) |= belief(role_property(d, RegHead, RegGroup1)) ∧
belief(has_expression(d, leads_to(v, w))) & state(γ, t, input(RegHead)) |= v]
⇒ ∃t' ≥ t state(γ, t', output(RegHead)) |= w

Chapter 7

Modeling Organizational Change for Naval Missions

Part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Simulation, Visualization, and Validation of Adaptive Multi-Agent Organizations in Naval Applications. In: *Proceedings of the Military Modeling and Simulation Symposium. Part of the Spring Simulation Multiconference (SpringSim'06)*, 2006.

Modeling Organizational Change for Naval Missions

Mark Hoogendoorn[1], Catholijn M. Jonker[2], Martijn C. Schut[1], and Jan Treur[1]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, schut, treur}@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. The naval domain is characterized by a dynamic environment. This requires constant adaptation of the organization, choosing between a wide variety of options. The consequences of the different options are difficult to foresee and hence, it is hard to judge which option is best. This paper presents automated support for the simulation, visualization, and validation of such adaptive multi-agent organizations. Generic simulation properties are specified using a formal modeling approach. Furthermore, results of a realistic case study are presented, and validated by means of properties obtained from naval experts. Finally, a tool is introduced that enables an insightful visualization of the simulation results.

1 Introduction

The process of setting up a simulation study involves steps of problem formulation, data collection, model definition, experimental design, running the simulation, output data analysis and reporting of results [9]. Throughout this process, intermediate validation steps assure that the simulation model corresponds with the actual system under investigation. The work described in this paper relates to two steps in particular, i.e., model definition and output data analysis, and describe these in more detail.

Model definition concerns setting up a conceptual model of the actual system with respect to project objectives, performance measures, data availability, computer constraints, etcetera. Many tools exist nowadays to support modelers with this activity. For ones specific interest, one may choose from a variety of simulation languages and software packages. These tools provide natural frameworks for model construction. As such, they are based on formal system descriptions and include concepts like entities, states, events, time, variables, etcetera.

Agent-based modeling techniques are often used to model and simulate (natural or artificial) agent systems that have to deal with dynamic and uncertain environments. Therefore, an important challenge for the area of agent-based modeling is the notion of adaptivity. Adaptation can take place within a single agent (e.g., an individual learning process), or at the level of the multi-agent organization (e.g, change of roles

of agents within the organization). In order to create (multi-)agent-based simulations with adaptive abilities, adaptation mechanisms have to be incorporated in agent-based simulation models.

Adaptation mechanisms can involve not only quantitative numerical aspects but also qualitative, logical aspects (for example, a role switch between agents within an organization). If formalization is used for an adaptation mechanism, this is often based on mathematical models using differential equations. In contrast, agent-based simulation models traditionally make use of qualitative, logical languages. Most of these languages are appropriate for expressing qualitative relations, but less suitable to work with more complex numerical structures as, for example, in differential equations. Therefore, integrating such mathematical models within the design of (multi-) agent-based simulation models is difficult. To achieve this integration, it is needed to bridge the gap between quantitative approaches and the type of languages typically used in agent-based simulation.

The model definition includes *validation* of the simulation model: "the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" [9]. Validation is essential for assuring that the simulation model corresponds with the actual system. Various validation techniques exist, of which one is mentioned in particular. By letting the simulation program generate a run or *trace*, i.e., the series of states over time of the simulated system (e.g., state variables, statistical counters), it is possible to compare the states with hand calculations to check the validity of the program.

Analysis of output data is in practice still rather undervalued as the simulation process is concerned. Much time goes into model development and programming, rather than addressing the generated output results appropriately. A commonly made "error" is that a single run is made of some arbitrary length, supposedly to provide insight into the workings of the actual system. Instead, suitable statistical techniques must be used to design the simulation experiments and analyze the results.

Since the output processes of simulations are almost all nonstationary and autocorrelated [9], classical techniques may not always be applicable. Validation of a model is usually not formally supported. Often validation is done informally, by hand (or eye), based on comparison of a simulation trace with an empirical trace. In addition, sometimes specific (e.g., statistical) techniques are used to support certain aspects of validation; e.g., termination conditions, mean and average estimations (for analysis of single systems), and measuring response differences, ranking, selection (for analysis of multiple systems). However, formal analysis and validation of global dynamic properties describing the system *behavior* has not received much attention in the simulation modeling literature. Usually in the domain that is modeled, *global properties* that should hold for the behavior of a simulation model can be identified. As the languages used to specify a simulation model are directed to *local properties* (the steps between successive states), such global properties cannot be formalized in these languages. To obtain more support, also for validation of a simulation model, it is needed to integrate the modeling of such global properties in a formal manner as well, so that their specification and automated checking on simulation traces also can be supported by the modeling environment.

In accordance with the findings mentioned above, this paper introduces an approach for simulation and analysis of adaptive (multi-)agent systems and underlying mechanisms that is integrative in two ways:

1. It combines in one modeling framework both *qualitative, logical* and *quantitative, numerical* aspects.
2. It allows to model dynamics at *different aggregation levels*, from a more local level (e.g., behaviors of roles within the organization) to a global level (behavior of the multi-agent organization as a whole); moreover, *interlevel relations* can be specified that express relationships between dynamic properties at different levels

Modeling dynamics at a local level often concerns expressing temporal relationships between pairs of successive states, such as described, for example, by basic steps within an adaptation mechanism. Local level specifications are the basis for the computation steps for a simulation model. From the more global perspective, more complex relationships over time can be used to model dynamics for adaptive multi-agent organizations: for example, how the system's behavior is changing during a history of events to which it adapts.

Based on the generic approach for simulation as presented above, this paper presents a simulation model for the naval domain. The model mainly concentrates on adaptation of such naval organizations using replanning.

The main objective of the research described in this paper is to investigate the suitability of a system involving planning, simulation, visualisation, and validation with respect to automated planning support in naval missions. The longer term aim of this research is to contribute to the development of a tool that allows for personnel to plan with a confidence and speed that would not be otherwise possible.

The remainder of this paper is structured as follows. Section 2 gives some details about the naval domain addressed and how adaptive organization forms play a role. In Section 3, the modeling methodology that has been used is presented. Section 4 presents a number of simulations that have been conducted based on local executable properties, and describes a case study that has been investigated. Section 5 presents the plan visualisation tool. Section 6 describes validation results in the form of non-local properties for the case study. Finally, Section 7 concludes and describes future work.

2 Dynamic Aspects in Naval Missions

Within the dynamic naval environment actions of possibly opposing parties, but also possible interference of non-military bystanders might induce a need for change in the organization to ensure the safety of the mission. Which response to choose in a given situation depends on a variety of factors. Elements such as enemy resources and innocent bystanders have to be taken into consideration and it is hard to predict the consequences of a plan that has been chosen. This paper presents an automated support system for the simulation, visualization, and validation of such processes. Two requirements must be met concerning such support: 1) the support must agree with the current way of working, and 2) guarantees must be given over the resulted planning with respect to given conditions including intended outcome and required

resources. The work presented here researches an approach for implementing automated support that meets these two requirements.

As the current way of working is concerned, the naval domain knows a large volume of well thought out plans that are scheduled for and during a mission (the so-called 'doctrine'). Everyone involved in a mission is familiar with these plans. The performed planning during a mission consists mostly of switching between and carrying out those plans. On the one hand, such planning during a mission may be a matter of executing the plans that were decided upon for the mission; on the other hand, unexpected events may happen that ask for necessary replanning during a mission. Concerning the latter, these situations require appropriate and speedy response. It is essential that in these situations, current circumstances are taken into account, a suitable plan is selected from the doctrine, the situation is dealt with and the mission will continue as originally planned.

Adaptation in the form of replanning in the naval domain frequently involves organizational change: it actually affects the organizational structure. For example, in response to an unexpected event, a ship that was originally only an escort of a high-value unit, may have to change its role to an attack unit. Such replanning situations are not rare: organizational changes are frequent and substantial.

Another important aspect of naval planning involves spatial information. Feasibility of a plan is partly determined by the nature of the available resources (helicopters, frigates, transporters) and the relative location of those resources. Combining the specific capabilities of the resources with spatial information and timing aspects plays a key role in the planning. Therefore plan visualisation that includes spatial information is necessary for successful implementation of automated planning support in naval applications.

In naval missions, it is crucial to consider the planning within the broader context of mission goals, available resources, intended outcomes, etcetera. In this respect, performed planning before and during a mission must be checked against such kinds of conditions. For example, when an agent is reallocated to another role (e.g., because of prevailing circumstances), it must inform others at the time that it is able to fulfill its role. It is important to recognise that this reallocation does not happen instantaneously (e.g., because a ship may have to sail towards some location to fulfill its new role), and therefore the communication is essential for others to know when the agent can receive orders in its new role.

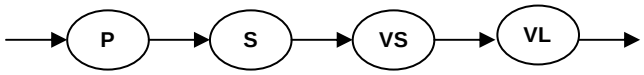

Fig. 1. Global overview of the simulation model.

This paper presents a simulation model that includes: a planner (P) for organizational change; a simulator (S) for those plans that reflects the meta-knowledge (see for example [5]) of the roles involved regarding organizational change; a visualisation tool (VS) for the spatial effects of plan execution that is dedicated to the naval domain; and a validation tool (VL) for the validation of the resulting planning.

The essential virtue of the model is that it recognises the importance of spatial information in naval planning (by means of the visualisation) and it offers an inventive way to check whether given conditions hold while planning (by means of the validation). The model may be used offline for analysis purposes and/or mission planning, as well as during execution of a mission as an automated planning support tool.

3 Modeling Methodology

To facilitate formal modeling of a multi-agent organization and its dynamics, this section introduces an organizational modeling approach and, in addition, a modeling language that enables specifying the dynamics within an organization (see also [3]). The organizational modeling approach is described in Section 3.1, and the formal language for expressing dynamics is addressed in Section 3.2.

3.1 AGR Organization Modeling Approach

For the description of actual multi-agent organizations, the AGR (for agent/group/role) model has been adopted [2]. In that approach, an organization is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. But, by avoiding an agent-oriented viewpoint, an organization is regarded as a structural relationship between a collection of agents. Thus, an organization can be described solely on the basis of its structure, i.e. by the way groups and roles are arranged to form a whole, without being concerned with the way agents actually behave, and multi-agent systems will be analyzed from the outside, as a set of interaction modes. The specific architecture of agents is purposely not addressed in the organizational model. The three primitive definitions are:

- The *agents*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application. In this paper, the agents are however assumed to be reflective agents, allowing them to reason about the role they are playing.
- *Groups* are defined as atomic sets of agent aggregation. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. An agent can be a member of n groups at the same time. A major point of these groups is that they can freely overlap.
- A *role* is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group. Roles can also have beliefs due to the assumed reflective capabilities of the agents; they can reason about whether they should have a particular belief given a certain role. These beliefs can be seen as an additional requirement on the agents playing that role.

3.2 Modeling Organizational Behavior

In this section a method to express dynamics within an organizational model is addressed. To formally specify dynamic properties at the different aggregation levels that are essential in an organization, an expressive language is needed. To this end the Temporal Trace Language is used as a tool; cf. [7]. For the properties occurring in the paper informal, semi-formal or formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly defined as follows.

A state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of some role *r* within a group *g* at time point t is denoted by

state(γ, t, input(r|g))

analogously

state(γ, t, output(r|g))
state(γ, t, internal(r|g))

denote the output state and internal state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r|g)) |= p denotes that state property p holds in trace γ at time t in the output state of role r within group g. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts TIME or T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. In trace descriptions, notations such as

state(γ, t, output(r|g))|= p

are shortened to

output(r|g)|p.

The Temporal Trace language can be used to specify behavioral properties at different aggregation levels, according to the organizational structure. Within the AGR approach the aggregation levels are the level of the roles, the level of the groups and the level of the organization as a whole (see Figure 2). The lower level properties can often be modeled in simpler formats than the higher level properties. In particular, it is often possible to model the properties at the leaves of the tree in the form of directly

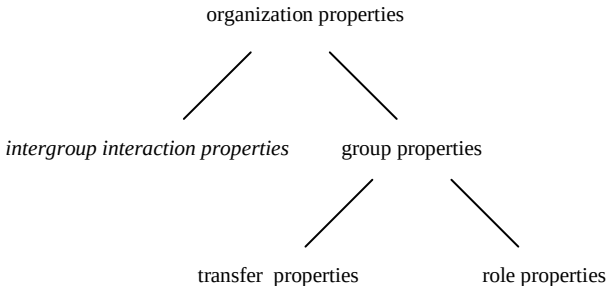

Fig. 2. Overview of interlevel relations between dynamic properties

executable properties, i.e., by direct temporal dependencies between state properties in two successive states. To model direct temporal dependencies between two state properties, not the expressive language TTL, but the simpler *leads to* format is used. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties (the leaves of the tree in Fig. 2). The format is defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold
for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [8]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

4 Local Properties and Simulations

This Section presents the simulator component within the system. First of all, the executable (leads to) properties which specify the simulation model for the simulator are presented in Section 4.1. After that, Section 4.2 addresses the case study that has been investigated, followed by the results of the simulations of the case study.

4.1 Simulation Model Specification

This Section describes generic local properties that constitute the basis for the simulation model. Each of these generic properties can be formed into more scenario specific properties whenever necessary. The generic properties in the framework work are based on goals, plans, beliefs and events.

It has to be mentioned that beliefs in this respect are used for storing information about the environment as well as information about oneself. As shown in the scenario below, many plans involve *organizational change*. This means that the actual organizational structure adapts to occurring events. Thus, in addition to knowing about the environment by observation, it is assumed that the agent (reflectively) knows about its role in the organization and can change to another role if necessary. The formalization is explained in the remainder of this section. Firstly, it is assumed that a goal has been given.

```
internal(r:ROLE|gr:GROUP)|belief(g:GOAL, pos)
```

denotes that role *r* within group *gr* holds the belief that *g* is a goal. Based on this goal, a plan is selected to achieve it:

```
internal(r:ROLE:gr:GROUP)|belief(current_plan(p:PLAN), pos)
```

says that plan *p* is selected as to achieve goal *g*. This plan will generate actions as long as no disturbing events occur. If such an event occurs and *r* is informed, this is denoted by

```
input(r:ROLE:gr:GROUP)|communication_from_to(
 r1:ROLE|gr1:GROUP, r:ROLE|gr:GROUP ,inform, e:EVENT)
```

stating that *r1* within group *gr1* informs *r* within group *gr* about event *e*. This event causes another goal to become active.

```
internal(r:ROLE|gr:GROUP)|belief(g1:GOAL, pos)
```

says that *g1* is now a goal and a subsequent plan is selected:

```
internal(r:ROLE|gr:GROUP)|belief(current_plan(p1:PLAN), pos)
```

This plan may involve organizational change. If this is the case (as it is in the scenarios below), a modeling approach is adopted as developed elsewhere [6]. This involves the existence of a ChangeManager who directs the organizational change. This approach is explained in more detail below. If the plan has been fully executed, this is denoted by

```
internal(r:ROLE|gr:GROUP)|belief(plan_executed(p:PLAN),pos)
```

where the parameter might be left out if it is assumed that only one plan can be executed at a time. This causes role *r* to reflect on other still existing goals and resuming the plans to achieve these goals. If there are no existing goals, a new goal may be generated or given.

Execution of a certain plan that has been selected often consists of organizational change. Therefore, generic simulation rules for these organization structure changes are needed to enable a generic simulation model. The properties shown below are based on the approach presented in [6] which is partially based on the AGR organization modeling approach as presented in Section 3.1. In that approach, organizational change can be performed in a meta-group called ChangeGroup, in which Member roles are present that represent agents within the organization. Each agent in the organization is represented by exactly one Member role within the ChangeGroup. The Member roles have beliefs about the organization and these beliefs are transferred to the roles the agent is currently playing. To initiate the change process as described above, triggers are needed. These are specified in the current plan, and are domain specific. Given this specific information for the particular plan, generic simulation rules fire to simulate the process of informing the members involved and changing their current beliefs on the organization. Some example executable local properties are presented below.

RP(ChangeManager):Communicate Activity
[output(ChangeManager|ChangeGroup)|communication_from_to(ChangeManager|ChangeGroup, all_involved, inform, active(C:CHANGE_GROUP)) &
internal(ChangeManager|ChangeGroup)|belief(involved_in_group(R:ROLE, C:CHANGE_GROUP), pos)]

$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
input(R:ROLE|ChangeGroup)|communication_from_to(ChangeManager|ChangeGroup, R:ROLE|ChangeGroup, inform, active(C:CHANGE_GROUP)

RP(Member): Believe Change Activity
input(R:ROLE|ChangeGroup)|communication_from_to(ChangeManager|ChangeGroup, R:ROLE|ChangeGroup, inform, active(C:CHANGE_GROUP)

$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
[internal(R:ROLE|ChangeGroup)|belief(active(C:CHANGE_GROUP, pos) &
output(R:ROLE|ChangeGroup)|communication_from_to(R:ROLE|ChangeGroup, ChangeManager|ChangeGroup, inform belief(active(C:CHANGE_GROUP), pos))]

Properties such as the examples above cause the ChangeGroup to be activated, knowledge about a new structure to be communicated, and finally belief emerging at the roles that need to have this information. After all of this has been performed, the ChangeGroup is deactivated and the new structure is in place (part of the internals of the roles).

Roles are attributed with reflective knowledge in the approach presented in this paper. This means that roles have beliefs on the expected behavior concerning the role. For example, a role has the internal belief that when the role receives an input *x* he eventually has to output *y*, formally:

internal(Role|Group)|belief(leadsto(input(Role|Group)|*x*, output(Role|Group)|y, efgh(0,0,1,1)),pos)

4.2 Simulation Results

This section contains results of simulations using the model presented in Section 3 and the generic properties presented in Section 4.1 which have been formalized in terms of the formal languages presented in Section 3. First of all, two case studies are introduced, thereafter some example formal properties which specify the behavior in the situations that occur in the case study are shown. Finally, the simulation traces for the case studies are shown.

4.2.1 Case studies

This section presents two case studies that has been obtained from experts of the Royal Netherlands Navy. The scenarios contain events that are typical within the naval domain.

Total Steam Failure

The first scenario that has been studied is called *total steam failure*. The initial configuration of the fleet is shown in Figure 3. In total there are six frigates, denoted by F1 - F6, each allocated to a certain area within which they reside. Besides the

frigates there are also helicopters (H1- H6) flying in a particular zone of the fleet. Finally, there are certain High Value Units (HVU) within the area called ZZ (for Zulu Zulu) that need to be protected. These might for example be ships containing troops, or amphibian landing vehicles. In total there are five ships within ZZ, which is called MainBody throughout this paper.

At a certain point in time the Officer in Tactical Command (OTC) receives an assignment to sail to Peterselie island and chooses a fleet configuration. On the way however, several unexpected events occur. First of all, one of the ships within the MainBody gets a total steam failure, meaning that it has lost all propulsion. On the basis of this event, the OTC has to decide what plan to apply. A few hours later, a nixie (a torpedo decoy) hit is observed at one of the members of the MainBody, which means that a torpedo was fired in the direction of that ship and implies re-planning as well. Finally, an hour after that, the ship that was suffering from a total steam failure gets back up to speed again.

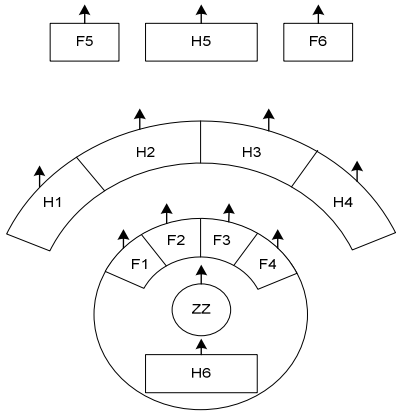

Fig. 3. Initial Fleet configuration

Submarine Threat

Another scenario which has been under investigation is that of a *submarine threat*. The initial fleet configuration is almost identical to the configuration shown in Figure 3, except that H6 is missing. The mission remains the same, which is to sail to Peterselie island. After a certain time-point however, frigate F1 detects sonar contact with a high probability that it is a submarine. The OTC now has to plan the actions to be performed to deal with such an event.

4.2.2 Case Specific Local Properties

This section presents some example properties that have been formalized to enable the simulation of the different case studies.

Total Steam Failure

First, two properties for the total steam failure case study is the following: In case a total steam failure is communicated to the OTC, then the new current plan is to form a screen around this ship. Formal:

RP(OTC): Handle total steam failure

input(OTC|Fleet)|communication_from_to(R:ROLE|MainBody1, OTC|Fleet, inform, total_steam_failure)
$\rightarrow_{0,0,1,1}$ internal(OTC|Fleet)|belief(current_plan(form_screen_around_ship(R:ROLE|MainBody1)), pos)

Furthermore, if the plan is indeed set to forming a screen around the ship, then the ship playing the role of FrontLeftProtector within the current screen will be allocated to the role of LeftProtector2 in the newly formed screen. Formally:

RP(OTC): Perform plan to form screen

∀A:AGENT, R:ROLE, G:GROUP
[internal(ChangeManager|ChangeGroup)|current_plan(
form_screen_around_ship(R:ROLE|MainBody1)), pos) &
internal(ChangeManager|ChangeGroup)|belief(allocated_to(A:AGENT,
FrontLeftProtector1, G:GROUP), pos)]
$\rightarrow_{0,0,1,1}$
[internal(ChangeManager|ChangeGroup)|belief(delete(allocated_to(
A:AGENT, FrontLeftProtector1, G:GROUP)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_group(Screen2)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_role(LeftProtector2)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(allocated_to(A:AGENT, LeftProtector2,
Screen2)), pos)]

Submarine Threat

Regarding the submarine threat case study, if a role informs the OTC that sonar contact with a submarine has been made, he forms a search and attack unit:

RP(OTC): Handle sonar contact

input(OTC|Fleet)|communication_from_to(R:ROLE|Screen1, OTC|Fleet,
inform, sonarcontact_sub)
$\rightarrow_{0,0,1,1}$ internal(OTC|Fleet)|belief(current_plan(eliminate_submarine_threat), pos)

The plan to eliminate such a submarine threat involves forming a search and attack unit. In case such a unit if formed, a new group is created called SAU. Furthermore, the role of commander within the SAU, the SAUC is performed by the agent previously allocated to LeftProtector1. Formally:

RP(OTC): Perform plan to form SAU

∀A:AGENT, R:ROLE, G:GROUP
[internal(ChangeManager|ChangeGroup)|current_plan(eliminate_submarine_threat), pos) &
internal(ChangeManager|ChangeGroup)|belief(allocated_to(A:AGENT,
LeftProtector1, G:GROUP), pos)]
$\rightarrow_{0,0,1,1}$
internal(ChangeManager|ChangeGroup)|belief(delete(allocated_to(
A:AGENT, FrontLeftProtector1, G:GROUP)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_group(SAU)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_role(SAUC)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(allocated_to(A:AGENT, SAUC, SAU)), pos)]

4.2.3 Simulation Trace

The results of the case studies that have been performed are presented here. First, the results of the *total steam failure* case study are presented after which the results of the *submarine threat* case study are addressed.

Total Steam Failure

The simulation results of the *total steam failure* case study are shown in Figure 4. The left side of the Figure shows a selection of the atoms that occur during the simulation. The right side shows a time-line where a black box indicates when an atom is true and a grey box when an atoms is false. This subset of the trace focuses on the OTC within the fleet, as he is the commander, he is the most interesting role to show. More

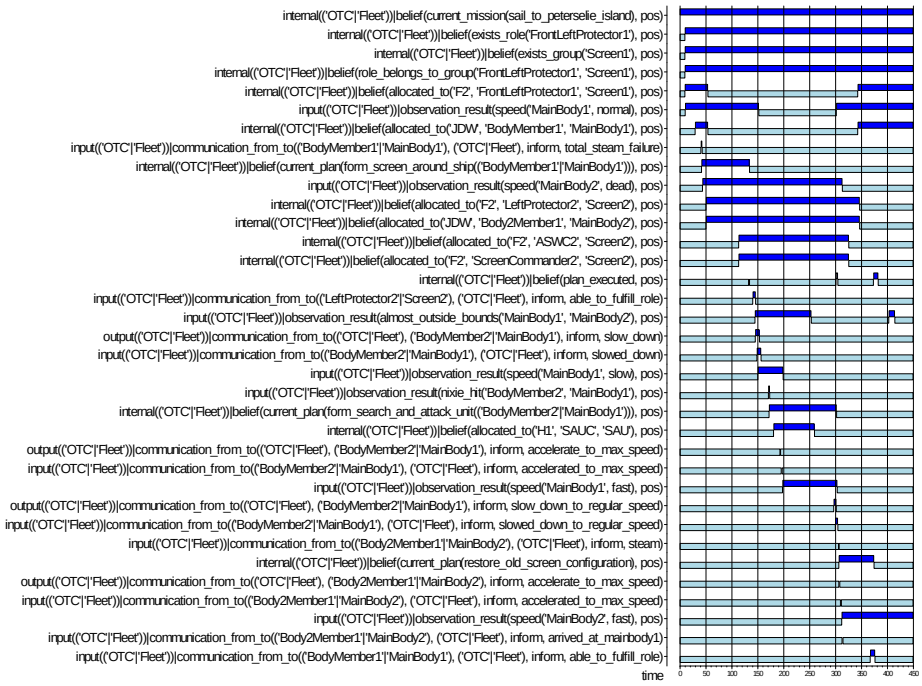

Fig. 4. Simulation result of the Total Steam Failure scenario

specifically, the trace shows that during all time points the current mission is to sail to Peterselie island:

internal(OTC|Fleet)|belief(current_mission(sail_to_peterselie_island), pos)

After the mission has been received, the initial organization is set-up according to the approach presented in Section 3.1. After the organization change process has ended the OTC has beliefs on the structure and allocations within the fleet, such as:

internal(OTC|Fleet)|belief(exists_role(FrontLeftProtector1), pos)
internal(OTC|Fleet)|belief(allocated_to(F1,LeftProtector1, Screen1), pos)

Suddenly, the OTC receives a communication from the role BodyMember1 within the MainBody1 group stating that the role has a total steam failure:

input(OTC|Fleet)|communication_from_to(BodyMember1|MainBody1, OTC|Fleet, inform, total_steam_failure)

Based on this communication, the OTC decides to form a screen around the ship, which means that the current fleet configuration as presented in the case-study changes drastically. As organizational change comes into play, the ChangeManager becomes active again, who forms a new group Screen2 (denoting the additional screen) and an additional main body (MainBody2). Several agents that were at first

allocated to the screen around MainBody1 are now re-allocated to roles in Screen2 around the newly formed MainBody2. To determine which agents to re-allocate, specific properties are present in the simulator that define a preference for which agent to take. Once the agents are in their new positions, they communicate this in their new role:

input(OTC|Fleet)|communication_from_to(LeftProtector2|Screen2, OTC|Fleet, inform, able_to_fulfill_role)

After these communications have been received, the OTC believes that the plan is executed successfully. A few time-points later however, the OTC observes that the distance between MainBody1 and MainBody2 is almost out of the bounds that have been set. As a response, the OTC commands the member of MainBody1 to slow down. Just after that command has been executed, an unexpected event occurs: A nixie hit is observed. This trigger causes the OTC to choose a new plan to be executed, because there is a severe danger of being attacked. The plan chosen is to form a search and attack unit, which will try to pinpoint the ship that fired the torpedo. Therefore, another organizational change is observed, creating the roles for the search and attack unit and re-allocating agents to these roles. In the trace this organization change involves a dynamic property being communicated, stating what the search and attack unit should perform:

internal((SAUC|SAU))|belief(leadsto(internal((SAUC|SAU))|belief(able_to_fulfill_role, pos), output((SAUC|SAU))|communication_from_to((SAUC|SAU), (OTC|Fleet), inform, started_plan_spencer), efgh(0, 0, 1, 1)), pos)

This states that once the role is fulfilled, the role will execute plan spencer and inform the OTC about this. Due to the reflective capabilities of the agent, they are able to reason about these dynamic properties and adopt them. After the OTC has observed that plan spencer is indeed being executed, he orders the remainder of MainBody1 to accelerate to maximum speed. After a while, the search and attack unit has fully executed plan spencer, resulting in the OTC deleting the group and re-allocating the agents to their old role. The final event that changes the organization is the communication from MainBody2 that it has steam again which is a trigger for a new plan, to restore the old fleet configuration. This is established by having MainBody2 and Screen2 accelerate to maximum speed and when it arrives at the MainBody1 allocated all the ships and helicopters to their old position again.

Submarine Threat

Figure 5 shows the trace regarding the simulation of the “submarine threat” case study. Briefly, the trace shows the following elements: First of all, OTC is informed by the LeftProtector1 within Screen1 about a sonar contact with a sub. At that same time-point the OTC derives a new plan:

internal(OTC|Fleet)|belief(current_plan(eliminate_submarine_threat, pos)

As a result, a search and attack unit (SAU) is formed again, and the submarine is located. After the location is known, the OTC orders the rest of the fleet to turn away. The command is confirmed by the ships within the MainBody1 and they eventually communicate to have turned away:

input(OTC|Fleet)|communication_from_to(BodyMember1|MainBody1, OTC|Fleet, inform, turned_away)

Following the observation that the ships must be outside of range for the torpedo’s, the ships are told to turn back to their old direction again. All confirm and execute the

order. The OTC commands the helicopters to replace the frigates that take part in the SAU because the helicopters are much faster and the distance between the SAU and the rest of the Fleet is increasing.

Fig. 5. Simulation result of Submarine Threat Scenario

output(OTC|Fleet)|communication_from_to(OTC|Fleet,LeftDetector|Screen1, inform, replace_sau)

Due to the open position in Screen1 that is left, helicopter F3 is allocated to two roles within the Screen. After a certain time, the OTC believes the submarine in no threat anymore and orders the roles within the SAU group to return to their mother ship:

output(OTC|Fleet)|communication_from_to(OTC|Fleet, SAUC|SAU, inform, return_to_mothership)

This denotes that at a later point in time, the helicopter is allocated to the role of FrontLeftProtector1 within Screen1, which is already allocated to frigate F2:

internal(OTC|Fleet)|belief(allocated_to(H1, FronLeftProtector, Screen1)

The commands to refuel and change the crew of the helicopter are therefore sent to the role to which F1 and H2 are allocated. After the refuel is done, the old fleet configuration is restored.

5 Visualization

For the simulator a visualization tool has also been developed. Figure 6 shows a screenshot of the tool. On the left side of the figure the fleet is shown in a visual manner as previously shown in Figure 3 whereas on the right side the trace (of which parts were explained already in Section 4.2), that acts as a basis for the visualization, is shown. A bar in the trace shows the accompanying time-point for which this visualization holds. For Navy domain experts such a visualization tool is easily

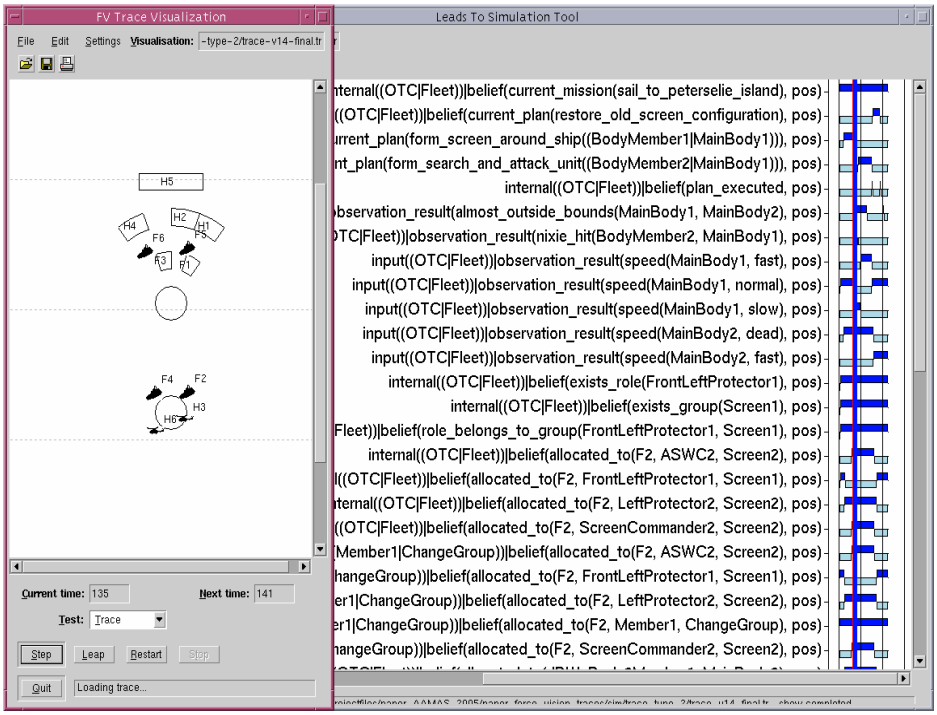

Fig. 6. Screenshot of the visualization tool

interpretable whereas a trace as shown on the right side of Figure 6 is hard to interpret especially due to the fact that one needs to be familiar with such kind of formalisms.

6 Non-Local Properties and Validation

When a formalized trace has been obtained either by a formalization of an empirical trace or by means of simulation it is useful to verify certain essential properties in the trace. Below the properties that have been checked against the traces presented in Section 4 are shown. The properties are independent from the specific scenario and should hold for every trace. The properties are formalized using the Temporal Trace Language as described in Section 3.

P1: Reflective Behavior
This property states that in case a role has a belief about an executable property that should be used when the role is being performed, the role should actually show this behavior. Formally:

∀γ:TRACES, t:TIME,
 [∃A:ANTECEDENT, C:CONSEQUENT, R:ROLE, G:GROUP
  state(γ, t, internal(R|G)) |= belief(leadsto(A, C, efgh(_,_,_,_)), pos)

$\Rightarrow \forall t2 \geq t$ [state(γ, t2) |= A $\Rightarrow \exists t3 \geq t2$ state(γ, t3) |= C]]

This property is indeed satisfied for the presented traces.

P2: Ship always allocated to a role
The fact that a ship should always be allocated to a role (after the initial fleet setup) is specified using this property. In formal form the property is formulated as follows:

```
∀γ:TRACES, t:TIME > 20, A:AGENT
  [∃R:ROLE, G:GROUP
      state(γ, t, internal(OTC|Fleet)) |= belief(allocated_to(A, R, G), pos)]
```

This property is also satisfied for the given traces.

P3: Communication that an agent is able to fulfill its role
This property expresses that when an agent is re-allocated to another role, it should always communicate when it is able to fulfill the role. There can be a time-delay between the re-allocation because the ship might have to sail to a particular place to execute the newly assigned role. Formally the property can be specified in the following way:

```
∀γ:TRACES, t:TIME > 20, A:AGENT, R:ROLE, G:GROUP
 [∃R2:ROLE state(γ, t, input(ChangeManager|ChangeGroup)) |=
           communication_from_to(R2|ChangeGroup, ChangeManager|ChangeGroup, inform,
           belief(add(allocated_to(A, R, G)), pos))
  ⇒ [∃t2:TIME ≥ t1 state(γ, t2, output(R|G)) |= communication_from_to(R|G, OTC|Fleet, inform,
able_to_fulfil_fole)]]
```

This property is satisfied as well for the given traces.

P4: Determine a plan to handle exceptions
When an exception occurs the OTC within the fleet always has a belief about a current plan that handles the exception:

```
∀γ:TRACES, t:TIME
  [∃E:EXCEPTION state(γ, t, input(OTC|Fleet)) |= E ⇒
   ∃t2:TIME ≥ t, P:PLAN [state(γ, t2, internal(OTC|Fleet)) |= belief(current_plan(P), pos)]]
```

This property is satisfied for the trace presented in Section 4.

7 Discussion

This paper introduces an integrative modeling approach for simulation and analysis of adaptive behavior of multi-agent organizations. The approach is integrative in two ways. First, it combines both *qualitative, logical* and *quantitative, numerical* aspects in one modeling framework. Second, it allows to model dynamics at *different aggregation levels* from local to more global levels.

The organizational processes during naval missions have been formalized by identifying executable local dynamic properties for the basic dynamics. On the basis of these local properties simulations have been made. Moreover, dynamic properties describing the behavior at a global level have been identified. These properties have been checked automatically on the simulation traces. To this end a system has been

introduced that consists of four components: (1) A planning component; (2) a simulation engine; (3) a visualization tool, and (4) a component which enables formal validation. The planning component has been equiped with typical plans for the naval domain from the so called 'doctrine'. The simulation engine has as a basis an organizational model which is specified by means of dynamics in the form of formal executable properties. Organizational change and change of plans are visualized in an understandable manner for naval experts by means of the visualization tool. Finally, the validation component enables formal validation of traces.

The approach taken in this paper has a number of advantages over other approaches. When comparing with planning achitictures such as [4] and [1], the approach presented in this paper provides validation functionalities for the simulation results, which is not the case in the other architectures. The models of these architectures can be formally proven to be correct, however for the complex naval domain it might be too diffult to prove such a thing. Furthemore the approach in this paper also has the ability to validate and visualize empirical traces who can for example be obtained from logbooks. These advantages could be used to monitor a current mission, and constantly check whether the properties that should hold for the mission are satisfied. In case a property is not satisfied, a warning could for example be given.

Other simulation engines have been developed specifically for the naval domain, such as for example presented in [10]. For a matter of validation of the model however, navy experts were asked what they considered to be the optimal solution. In the approach used in this paper, this process is automated due to the formal specification of properties provided to us by naval domain experts.

Acknowledgements

CAMS-Force Vision, the software development department associated with the Royal Netherlands Navy, has provided funding and domain knowledge to enable the scenarios and simulations presented in this paper. The authors especially want to thank Jaap de Boer (CAMS-ForceVision) for his expert knowledge.

References

[1] d'Inverno, M., Luck, M. Georgeff, M., Kinny, D. and Wooldridge, M., The dMARS Architechure: A Specification of the Distributed Multi-Agent Reasoning System. Journal of Autonomous Agents and Multi-Agent Systems, 9(1-2):5-53, 2004.

[2] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

[3] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Y. Demazeau, F. Garijo (eds.), Multi-Agent System Organisations. Proc. of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'01.

[4] Georgeff, M. P., and Ingrand, F. F., Decision-making in an embedded reasoning system. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pages 972-978, Detroit, MI, 1989.
[5] Goodwin, R., Meta-Level Control for Decision-Theoretic Planners. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.
[6] Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modelling the Organisation of Organisational Change. In: Proc. of the Sixth International Workshop on Agent-Oriented Information Systems, AOIS'04.
[7] Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
[8] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. Cognitive Systems Research Journal, vol. 4, 2003, pp. 191-210.
[9] Law A.M. and Kelton D.W., Simulation, Modeling and Analysis. McGraw Hill, 2000. Third edition.
[10] Sokolowski, J., Enhanced Military Decision Modeling Using a MultiAgent System Approach, In Proceedings of the Twelfth Conference on Behavior Representation in Modeling and Simulation, Scottsdale, AZ., May 12-15, 2003, pp. 179-186.

Chapter 8

A Formal Organizational Modeling Approach to Support Change Processes: A Case Study in Dutch Municipalities

This chapter appeared as: Bruin, B. de, and Hoogendoorn, M., A Formal Organizational Modeling Approach to Support Change Processes: A Case Study in Dutch Municipalities. In: Dignum, V., Dignum, F., Matson, E., and Edmonds, B. (eds.), *Proceedings of the Workshop on Agent Organizations: Models, and Simulation (AOMS @ IJCAI 2007)*, 2007, pp. 13-25.

A Formal Organizational Modeling Approach to Support Change Processes: A Case Study in Dutch Municipalities

Bas de Bruin and Mark Hoogendoorn

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{sbdbruin, mhoogen}@cs.vu.nl

Abstract. Change processes within organizations are cumbersome, over 70% of such processes does not achieve the intended goal. This paper presents an multi-agent organizational modeling approach which can support such change processes. The approach consists of three parts: (1) analysis of an existing organization by means of simulation and verification; (2) the possibility to simulate and analyze possible new organizations, and (3) the analysis of the process of moving from the one organization to another. The approach is based on formal modeling and simulation techniques, enabling an analysis using formal verification techniques. The approach has been evaluated by means of an extensive case study within several municipalities within the Netherlands.

1 Introduction

As the development of the field of multi-agent systems continues, the systems that are being developed within this field are becoming increasingly complex. Due to this increase in complexity, the need arises for an abstraction level higher than the concept agent to support the design and analysis of such systems. As a result, organization modeling is becoming a practiced stage within the development of multi-agent systems (see e.g. [5, 7, 11, 16]). Typically, approaches for multi-agent organizational modeling draw inspiration from the fields of social sciences and economics and apply such approaches to artificial or computational organizations. Some approaches however also claim that they can be used for modeling and analysis of human organizations as well (e.g. AGR extended with dynamic properties [6]).

In human organizations, change has nowadays become part of everyday life, some organizations are continuously undergoing such change. In the fields of social sciences, economics, and psychology recent research has focused on the analysis of such change process, see for example [2, 13]. Still, the effectiveness of change is low: research has shown that over 70% of all change processes does not achieve the intended goal [1, 9]. Three types of organizational change have been distinguished in the literature [2]. Firstly, *planned organizational change* is distinguished, in which the problems and solutions are known. Another type of change is *organizational development* in which the direction of change is approximately known and there exist

some ideas, but they are not entirely unambiguous. Finally, *transformational change* is the emergence of a new organization.

The goal of this paper is to present an approach which supports practitioners in decision making regarding organizational change based upon an existing multi-agent organization modeling approach, namely [6]. The system offers support in three phases of the change process. First of all, it allows for the investigation of the current way of functioning of an organization. Such an investigation takes place on the basis of formal simulation and verification techniques. In order to guide this formalization process, a three step formalization method is introduced. Second, a possible new organization can be simulated and analyzed to see whether improvements are indeed accomplished and whether the organization functions according to the expectations. Finally, the approach can be used for the change process itself, to investigate possible bottlenecks within the process (e.g. resistance of employees) and solutions for such bottlenecks. The approach is both suitable for planned organizational change and organizational development. For the former, it can be investigated whether the ideas that exist indeed show the expected result, whereas for the latter different alternatives can be weighed to determine the most promising change of organization. Since the approach is meant as a way to support practitioners and there is hardly any knowledge whether the multi-agent organizational modeling approach is suitable for human organizations, this paper presents the approach by means of an extensive case study which has taken place in several municipalities within The Netherlands.

The paper is organized as follows: Section 2 introduces the organizational modeling approach that has been adopted. In Section 3 it is shown how the approach can be used for modeling an existing organization whereas Section 4 shows this for a potential new organization after change has occurred. Analysis and modeling of the change process itself is addressed in Section 5, and finally, Section 6 is a discussion.

2 Organization Modeling Approach

This Section presents the organizational modeling approach which has been used throughout the paper; cf. [6]. Two parts can be distinguished within this approach, namely the structural description of an organization, and the behavioral description.

2.1 Structural Model of an Organization

For the structural description of actual multi-agent organizations, the AGR (for agent/group/role) model has been adopted [7]. Within AGR organization models three aggregation levels are distinguished: (1) the organization as a whole; the highest aggregation level; (2) the level of a group, and (3) the level of a role within a group. In addition, transfer between roles within a group can be specified as well as inter-group interactions.

2.2 Behavioral Model of an Organization

To enable formal specification of dynamic properties at the different aggregation levels that are essential in an organization, an expressive language is needed. To this end the Temporal Trace Language is used as a tool; cf. [12]. For the properties occurring in the paper informal, semi-formal or formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly described as follows;

A state ontology Ont is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is defined as an indication of which state properties expressed in ontology Ont hold in the state and which do not hold. The set of all states is modeled by the sort STATE. A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is an indication of which state occurs at which time point, for example if a discrete time frame based on natural numbers is taken, a trace is a sequence of states γ_t ($t \in T$). The set of all traces over state ontology Ont is modeled by the sort TRACE. Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers), or any other form, as long as it has a linear ordering. A *dynamic property* over state ontology Ont is a temporal statement that can be formulated with respect to traces based on the state ontology. Such temporal statements can express, for example, a temporal relationship between the fact that in a given trace a certain state property holds at a certain time point and another state property holds at some other time point.

The Temporal Trace language can be used to specify behavioral properties at different aggregation levels, according to the organizational structure. These aggregation levels follow those identified in the AGR approach. At the lowest level role properties describe the behavior of an individual role whereas transfer properties describe the dynamics of (intra-group) transfer between roles. For the roles within a given group, such role properties, together with the transfer properties, entail the group properties that characterize the behavior of the group as a whole. The group properties for the different groups, together with the inter-group relationship properties (for transfer between groups), entail the overall organization properties.

The lower level properties can often be modeled in simpler formats than the higher level properties. In particular, it is often possible to model the properties at the leaves of the tree in the form of directly executable properties, i.e., by direct temporal dependencies between state properties in two successive states. To model direct temporal dependencies between two state properties, not the expressive language TTL (which can express very complex dependencies which are hard if not impossible to execute), but the simpler *leads to* format is used. This is an executable format that can be used to obtain a specification of a simulation model in terms of local dynamic properties. The format is defined as follows. Let α and β be conjunctions of elementary state properties, and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,
then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [3]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that simulation results can be depicted graphically.

3 Analysis of an Existing Organization

A first step in the support for organizational change processes is the analysis of the existing organization. As a result of this analysis possible points for improvements or points of failure can be identified. Such an analysis can be performed in two ways: (1) logs of the functioning of an organization can be obtained; (2) simulation runs can be performed, using the current organization as a basis, resulting in a log as well. Thereafter, either one of these types of logs can be automatically analyzed by means of formal verification techniques. This Section first of all shows how an existing organization can be simulated and thereafter presents an analysis of the results.

3.1 Simulation of an Existing Organization

Getting to a model which enables simulation consists of two phases: (1) specifying the structural description of the organization; (2) creating a behavioral description of the organization. Both phases are explained by means of a case study which has taken place at several Dutch municipalities.

3.1.1 Structural Description of an Organization

There are several steps that have to be taken in order to simulate an existing organization The first step taken is to determine the structure of the current organization. Since AGR is used for the representation of such a structure, the roles, groups, and interaction between those elements for the organization are determined. Typically, organizations have their structure described in diagrams, so this process denotes the translation to AGR.

For the case of one of the municipalities, the organization structure is shown in Figure 1. Note that for the sake of brevity this Figure only represents a part of the total organization. In the Figure, the big ovals denote groups, whereas the small ovals denote roles. Furthermore, a solid arrow indicates an interaction between two roles within a group, and a dashed arrow indicates an interaction between two roles in different groups.

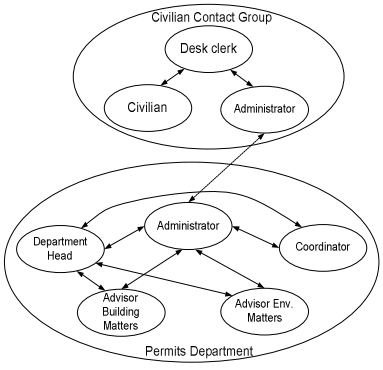

Fig. 1. AGR representation of part of a municipality organization

As can be seen in the Figure, two groups are present, namely the Civilian Contact Group, and the Permits Department. The former group is present to interact with civilians whereas the latter concerns the procedure of handling requests from civilians, in this case concerning

permits that have been requested. Within the Civilian Contact Group, three roles are present. First of all, the Civilian which has a particular request. Furthermore, the Desk Clerk role is the only role which interacts with the Civilian and processes its request, after which the request is forwarded to the Administrator, which is the third role. The Administrator has an inter group interaction with a role Administrator in the Permits Department. In the Permits Department it is the central role overseeing the whole process and passing information between the roles. In addition, four other roles are present. First of all, the Department Head, who needs to approve particular decisions. Furthermore, two Advisor roles are present, namely the Advisor for Building Matters, and the Advisor for Environmental Matters. Both are experts in their specific domain and on the permit procedures. Finally, the Coordinator makes sure all procedures are executed properly and keeps track of the time restrictions on the process.

3.1.2 Behavioral Description of an Organization

Given that the structure of an organization has been represented, the behavior of such structural elements needs to be specified next. Specifying such behavior is not a trivial matter, often procedures can be used as a basis for such a behavioral description. However, such a specification typically lacks sufficient detail to obtain a complete behavioral description. Therefore, interviews with experts are usually unavoidable. After sufficient information has been obtained, this informal information needs to be formalized to enable logical simulation such as described in Section 2.2. The formalization can be performed in a three step process.

First of all, informal behavior descriptions are translated into a semi-formal format. This is a necessary step as typically this information comes from various sources that are difficult to oversee all at once. For the case of the municipality for example, the following informal behavioral property has been acquired:

> "once a desk clerk has received a request for a permit from a civilian, the desk clerk forwards this request to the appropriate administrator"

When looking at the informal rule, it can be seen that such a rule can easily be translated into a semi formal format of the if-then form:

```
if      a desk clerk D has received a request R for a permit from civilian C
 and    desk clerk D believes that administrator A should handle request R
then    desk clerk D forwards the request R to administrator A
```

As can be seen, variables have now been introduced into the rules. Actually, a belief has been introduced into the rule as well which specifies a part of the knowledge the Desk Clerk must have in order to be able to perform its task properly.

Second step in the formalization process is to define an ontology suitable for this particular organization which is based upon the semi-formal rules that have been distinguished, and the terms that occur in such rules. For example from the rule as presented above the ontology presented in Table 1 and Table 2 can be extracted.

Table 1. Sort definition

Sort	**Explanation**
DESK_CLERK	A desk clerk role
CIVILIAN	A civilian role
ADMINISTRATOR	An administrator role
ROLE	DESK_CLERK ∪ CIVILIAN ∪ ADMINISTRATOR
REQUEST	A request from a civilian
BELIEF_ELEMENT	This specifies what can be believed by a role

Table 2. Predicate definition

Predicate	Explanation
communication_from_to: ROLE x ROLE x REQUEST	A communication can take place from one role to another concerning a request
belief: BELIEF_ELEMENT	A certain role believes something
should_handle_request: ROLE x REQUEST → BELIEF_ELEMENT	One example BELIEF_ELEMENT which concerns knowledge on what ROLE should handle a particular REQUEST

The final step in the process is to translate the semi-formal rules into formal ones using the ontology which has been created. Take for example the semi-formal rule which was specified previously. In formal format using TTL this rule can be expressed as follows:

∀t:TIME, C:CIVILIAN, D:DESK_CLERK, R:REQUEST, A:ADMINISTRATOR
[state(γ, t, input(D)) |= communication_from_to(C, D, R) &
state(γ, t, internal(D)) |= belief(should_handle_request(A, R))
⇒ ∃t' > t [state(γ, t', output(D)) |= communication_from_to(D, A, R)]]

Note that the state(γ, t, input(D)) |= communication_from_to(C, D, R) specifies that within trace γ at time point t on the input state of D a communication from C concerning request R is present. To enable simulation using such formal rules, the lowest level behavioral properties should be expressed in the executable LEADSTO format. A translation from the formalized rules into the LEADSTO format is often straightforward. For example, for the rule specified above the translation is specified as follows:

∀ C:CIVILIAN, D:DESK_CLERK, R:REQUEST, A:ADMINISTRATOR
[input(D)|communication_from_to(C, D, R) ∧ internal(D)|belief(should_handle_request(A, R))
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ [output(D)|communication_from_to(D, A, R)]]

which expresses that if the antecedent holds for a duration of 1 time point, then the consequent will hold for a duration of 1 time point as well with a delay of 0. Note that the parts such as state(γ, t, input(D)) |= as specified in the TTL formula are now abbreviated to input(D)| for the sake of clarity and simplicity.

3.1.3 Simulation of an Organizational Model

Based upon the LEADSTO properties obtained as a result of the process described in Sections 3.1.1 and 3.1.2 simulation runs can be performed using the LEADSTO simulation environment [3]. Output of such a simulator is a trace. A small portion of the trace resulting from the organizational model specified for the municipality is shown in Figure 2. This part precisely concerns the part in which the rule used as a running example in this Section applies, this trace shows one particular application of the rule. In the Figure, the left side shows the atoms that occur during the simulation whereas the right side shows a timeline where a dark box indicates an atom is true at that particular time point and a grey box indicates the atoms is false.

As can be seen in the Figure, the Desk Clerk receives a communication from a

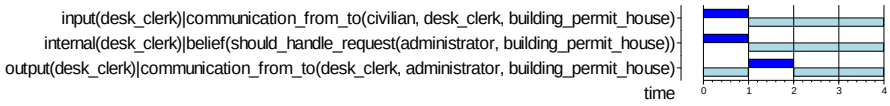

Fig. 2. Partial trace of the municipality organization

civilian who requests a permit to build a house:

input(desk_clerk)|communication_from_to(civilian, desk_clerk, building_permit_house)

Furthermore, in the simulation the desk_clerk has a belief that administrator is the one that should handle the request of the civilian:

internal(desk_clerk)|belief(should_handle_request(administrator, building_permit_house))

Note that such a fact is inserted into the simulation and can be varied, possibly resulting in different simulation results. Finally, as a result of the two previous atoms a rule fires which causes an output of the desk_clerk:

output(desk_clerk)|communication_from_to(desk_clerk, administrator, building_permit_house)

Of course the actual simulation of the organization is much more extensive and has a variety of possible outcomes due to the possibility of varying certain facts in the form of beliefs. To give an idea of the complexity of such a simulation model: The specification of one single process in the municipality comprises of over 60 role, transfer and group interaction properties and the resulting trace consists of 125 atoms.

3.2 Verification of Simulation Runs

Given that either an empirical formalized trace exists, or that a trace has been obtained using simulations (such as presented in Section 3.1), such a trace can be analyzed. In order to be able to analyze such a trace, a formal checker called the TTL Checker [12] is used to determine whether certain logical properties indeed hold for a given trace. Such logical properties are precisely those properties within the property hierarchy that are not in an executable format (i.e. the organization and group properties). The logical properties that are checked upon such a trace are again obtained from experts within the organization. In the case of the municipalities that have been investigated, the municipality needs to follow the law in order to have a successful organization. Several properties are specified within such a law.

P1: Communicate decision. In case the municipality has decided to either approve or reject a request submitted by a civilian, then the civilian will eventually be informed about this decision. In formal form this property is specified as follows:

$\forall$t,t2:TIME, R:REQUEST, C:CIVILIAN, D:DESK_CLERK, CO:COORDINATOR, S:SIGN
[[state(γ, t, output(C)) |= communication_from_to(C, D, R) &
state(γ, t2, internal(CO)) |= decision(R, S) & t2 $\geq$ t]
$\Rightarrow$ $\exists$t3:TIME > t2:TIME, R2:ROLE [state(γ, t3, input(C)) |= communication_from_to(R2, C, decision(R, S))]

The sort SIGN consists of both pos and neg indicating whether an acceptance or rejection was the result of the decision process. This property has been checked against a number of simulation traces that were based upon scenarios given by the organizational experts and were shown to hold for these traces.

P2: Check decision. If an Administrator passes a decision about a request to the Department Head then eventually the Department Head will pass back a decision. This property is also satisfied for all traces that have been generated.

P3: Timely entering of requests. If a request for a permit is communicated by a civilian, then within 5 time points such a request should be entered into an administrative software system. This property is satisfied for the traces belonging to one municipality, however is not satisfied in the other. This is due to a difference in insight in how to interpret the law in this matter.

4 Analysis of a Possible New Organization

This Section shows how, by means of simulations, a potential new organization can be evaluated in order to investigate how well the structure would function in practice.

4.1 Simulation of a Possible New Organization

In principle, the same techniques as presented in Section 3.1 can be used for simulation of such an organization. One difference is however that the future organization is often not so well described as the current organization. Hence, a new organization might need to be designed first before being able to perform this stage.

Within the case study of the Dutch municipalities, a possible merger between two municipalities has been a subject of discussions for a number of years due to the increasing complexity of tasks carried out by Dutch municipalities and the expected cost benefit. Interviews with key players within both municipalities have been conducted, resulting in a list of demands from each of the municipalities. Based upon these results, processes within the municipalities have been merged as well as the structure. The new organization is not simply a union of two separate structures such as shown in the Section concerning the analysis of the current organization. First of all, only one Desk Clerk is present in the combined new organization, whereas previously there existed two (one in every municipality). Furthermore, two Administrator roles are present within the Civilian Contact Group, taking the Administrator from both municipalities. This choice has been made because the role is a central player within the processes and must not become a bottleneck. In the Permits Department, two roles for Administrator (Administrator One and Administrator Two), a Department Head and a Coordinator role are present. For the Advisor roles, each of these roles are now present in twofold. This choice has been made due to the increasing complexity and the increasing number of the tasks that will be performed. Since two Advisors together know more than one (principle of synergy), such a combination is able to handle this higher complexity. A result of this new organizational structure is that fewer role instances are needed within the organization (i.e. cost benefit), whereas more expertise is present within the organization (i.e. the handling capability for more complex tasks is improved).

To see whether the new organization indeed behaves according to plan, the procedures currently in use are modified to match the organizational structure. These modifications are made using organizational experts. Thereafter, these modified processes are translated using the proposed method in Section 3. As a result, a formal specification of the new organization, including its behavior, is created. A combination of the scenarios used for the analysis of the individual municipalities is now used to analyze the newly designed organization.

4.2 Verification of a Possible New Organization

For the verification of the possible new organization, the same properties can be checked as expressed in Section 3, however, other properties can also be specified. In

this case study for example, it could be verified whether the new organization is able to handle the more complex tasks. Results of checking the properties specified in Section 3.2 show that all properties are satisfied by the proposed new organization.

5 Analysis of the Process of Organizational Change

Final element which can be used to support change within organizations is the simulation of the change process itself. Typically, such change processes are a painful process in which many people for example object to particular structural or behavioral changes within the organization, a new allocation, etc. Simulation can aid in finding the bottleneck within such a process.

5.1 Modeling of an Organizational Change Process

In order to model such a change process, the approach presented in [10] is adopted. The approach uses a well known theory in Organizational Change Theory, namely the *unfreezing – movement – refreezing* theory by Kurt Lewin [13]. He states that there are two opposing forces at work when changing an organization: forces that resist the change, and forces that drive towards the newly desired organization. The unfreezing phase begins at the moment that change becomes necessary and consists of the process of changing the resisting and driving forces in such a way that change becomes possible (i.e., the driving forces outweigh the resisting forces). The actual change of the organization is contained in the movement phase in which the organization is moved from the current state to the desired stated. The refreezing phase involves freezing the newly formed organization so that there is no possibility to return to the former status quo or to continue changing in another unwanted direction. The whole re-organization process is completed when all phases have been completed. The unfreezing can be performed by increasing the driving forces and/or by decreasing the resisting forces. In MOISE+ [11] such phases that occur in human organizational change are not represented in the model, making it less suitable for analysis of such change processes.

In order to model such a change process, [10] proposes two types of roles within an organization, namely the regular roles within the organization and roles placed within a so called Change Group. The roles within the Change Group include Member roles, and the Change Manager. The Member roles have the ability to reason about change within an organization and have a meta-view upon the organization. Furthermore, a Change Manager is present who directs the change. Properties for such Members and the Change Manager can be specified for each particular phase within the change process. As a result, possible options for resistance of Members to an organizational change can be investigated, the response of the Change Manager based upon that, and the effect of that resistance upon the change process can be determined. The advantage of the approach is that it handles organizational change in a fashion which abstracts from the specific agents allocated to the roles within the organization.

For the case study, a number of executable properties have been specified concerning the behavior of the Members and the Change Manager within the Change

Group for the Municipality when changing from the old to the new organization. These properties have been separately specified for each phase identified by Lewin.

5.1.1 Unfreezing Phase

First of all, properties have been specified for the unfreezing phase. For instance, the following property specifies the behavior of a Member who is currently playing the role of Department Head within one of the municipalities:

if	Member M has a shared allocation with the role Department Head within Municipality MU
and	Member M receives a communication about a new organization O including only one Department Head
and	Member M receives a communication that another Member M2 is appointed Department Head within the new organization O
then	Member M communicates to the Change Manager that he opposes to the new organization O giving the reason that he wants to be Department Head

In order to successfully unfreeze both municipalities, such resistance of a Member must be reduced. A property which might convince the Member is by stating that the other Member has more experience:

if	The Change Manager has sent a communication to Member M about a new organization O including only one Department Head
and	The Change Manager has sent a communication to Member M that another Member M2 is appointed Department Head within the new organization O
and	The Change Manager has received a communication from Member M in which he opposes to the new organization O giving the reason that he wants to be Department Head
then	The Change Manager informs Member M that he is less experienced than Member M2.

In case this indeed considered to be a valid and convincing argument by the Member, the Member communicates the acceptance of the new organization. Many more of such properties have been specified for the municipality change process. The unfreezing phase ends when all Members within the organization have acknowledged the new organization.

5.1.2 Movement Phase

The movement phase is a rather straightforward phase in which the new organization is simply put into place. New shared allocations are created to new roles, new behavioral description are put into place, new groups are formed, and the agents that no longer have a shared allocation leave the organization. The movement of an organization is completed after all Member roles have acknowledged the movement to the new organization, meaning that agents no longer playing a role in the organization acknowledge that they leave the organization, and do so immediately.

5.1.3 Refreezing Phase

In the refreezing phase, the organization is already functioning in its new form. Due to the fact that the behavior is not routine behavior yet, it could be the case that the wrong behavior is sometimes shown. In the case of the municipality, it might occur that the new Coordinator suddenly starts to approve decisions, which is not allowed. Therefore, properties that correct such behavior are specified as well (causing the role R2 to show the correct behavior again):

if	a role R within the Permit Department observes that a role R2 within the Permit Department is performing property P
and	role R2 is not allowed to perform property P
then	role R warns role R2 that he is not allowed to perform property P

5.2 Simulation and Analysis of an Organizational Change Process

In order to see whether the change process such as for example specified as presented in Section 5.1 indeed shows the desired behavior, again simulation runs can be performed. Such simulation could show behavior which was not predicted at first sight. After such a simulation has taken place, properties specifying the successfulness of such a change process can be identified to verify whether the change process indeed went according to plan. Properties that have been verified based on the simulation runs include the following: successful unfreezing (everybody has accepted the new organization); successful movement (all new roles and behaviors are known in the organization), and finally, successful refreezing (eventually everybody shows the correct behavior or is corrected otherwise). All of these properties indeed hold for the given traces.

6 Discussion

This paper presented an approach to aid practitioners in organizational change processes. Therefore, a three step support methodology was presented. The first step allows for the analysis of the current organization, enabling the pinpointing of bottlenecks. Such an analysis can be performed based upon empirical logs of the organization or logs resulting from simulations of the current organization. A second step consists of the simulation of a potential solution to the bottlenecks, i.e. simulation of a possible new organization. Again, such results can be analyzed to see whether the improvements are indeed met by such a new organization. Finally, in the last step the change process itself can be analyzed by means of simulations to help predict and overcome typical point of failure within such processes.

To enable the simulation and analysis of such organizational models, formal modeling and simulation techniques were used. In order to guide the specification of such models, a three step process has been identified to create a model from an informal organizational description. Using such formal techniques enables an automated evaluation of simulation results using formal verification techniques. Furthermore, model checking techniques can be used to formally prove that certain properties hold within a particular organization model. The latter approach is future work.

Using these techniques, an extensive case study was performed in several Dutch municipalities. Organizational experts were consulted to create simulation models, and specify properties for the analysis of the simulation results. It was shown that the approach indeed has the ability to handle the complexity of such a municipality organization. In an evaluation session, the organizational experts found the results very insightful and a good basis for discussing future cooperation.

Other organizational simulation approaches in the domain of artificial intelligence or computer science typically focus on the simulation of organizations using agents, see e.g. [15, 4]. Although such approaches are very useful, it prevents the analysis of the organizational model itself without looking at the details of the agents allocated to the roles within the organization. The approach presented in this paper does provide

an approach for such an analysis, fully abstracting from the agents, only looking at the expected behavior of the agents once allocated to their role. Other well known methods to perform social simulations use approaches such as differential equations and cellular automata (see e.g. [8]) which differ completely from the approach presented in this paper.

Acknowledgements
The authors would like to thank the Dutch municipalities of Aalsmeer and Uithoorn for participating in this research, and Jan Treur for the fruitful discussions.

References

[1] Bashein, M.L., Marcus, M.L., and Riley, P., Business Process Reengineering: preconditions for success and failure, Inf. Systems Management 9, 1994, pp. 24-31.
[2] Boonstra, J.J. (editior), Dynamics of Organizational Change and Learning, Wiley, 2004.
[3] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), Proc. of MATES'05. LNAI, vol. 3550. Springer Verlag, 2005, pp. 165-178.
[4] Conte, R., Esmonds, B., Moss, S., and Sawyer, R.K., Sociology and Social Theory in Agent Based Social Simulation: A Symposium, Computational and Mathematical Organization Theory, vol. 7, 2004, pp. 183-205.
[5] Dignum, V., Sonenberg, L., Dignum, F., Dynamic Reorganization of Agent Societies, In: Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies, 2004.
[6] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., "Organization Models and Behavioural Requirements Specification for Multi-Agent Systems," in Y. Demazeau, F. Garijo (Eds.), Multi-Agent System Organizations. Proceedings of MAAMAW'01, 2001.
[7] Ferber, J. and Gutknecht, O., "A meta-model for the analysis and design of organizations in multi-agent systems," Proc. of ICMAS '98, IEEE Comp. Soc. Press, 1998, pp. 128-135.
[8] Gilbert, N., and Troitzsch, K.G., Simulation for the Social Scientist, Open U. Press, 1999
[9] Hall, G., Rosenthal, T., and Wade, J. How to make reengineering really work, *Harvard Business Review*, 71(6), 1993, pp. 119-131.
[10] Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modeling Centralized Organization of Organizational Change, Computational and Mathematical Organization Theory, in press 2006.
[11] Hübner, J.F. and Sichman, J.S., Using the MOISE+ model for a cooperative framework of MAS reorganization, In: Proc. 17th Brazilian Symposium on Artificial Intelligence (SBIA'04), São Luís, Brasil, 2004. A. Bazzan and S. Labidi eds. Advances in Artificial Intelligence, vol. 3171 of LNAI series, Springer Verlag, 2004, pp. 506-515.
[12] Jonker, C.M., and Treur, J., "Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness," International Journal of Cooperative Information Systems, 11, 2002, pp. 51-92.
[13] Lewin, K., Field Theory in Social Science, Harper & Row, New York, 1951.
[14] Robbins, S.P., Organizational Behaviour, Prentice Hall, New Jersey, 1998.
[15] Terna, P., Simulation Tools for Social Scientists: Building Agent Based Models with SWARM, Journal of Artificial Societies and Social Simulation, vol. 1, 1998.
[16] Zambonelli, F., Jennings, N., Wooldridge, M., Organizational Rules as an Abstraction for the Analysis and Design of Multi-agent Systems, Journal of Software and Knowledge Engineering, Vol. 11, 2003, pp. 303-328.

Part IV:
Organizational Change Process:
Decentralized Change Processses

Chapter 9

Modeling Decentralized Organizational Change in Honeybee Societies

This chapter appeared as: Hoogendoorn, M., Schut, M.C., and Treur, J., Modeling Decentralized Organizational Change in Honeybee Societies. In: Minai, A., Braha, D., and Bar-Yam, Y., *Proceedings of the Sixth International Conference on Complex Systems*, NECSI, 2006.

Modeling Decentralized Organizational Change in Honeybee Societies

Mark Hoogendoorn, Martijn C. Schut, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, schut, treur}@cs.vu.nl

1 Introduction

The concept of organization has been studied in sciences such as social science and economics, but recently also in artificial intelligence [3; 4; 7]. With the desire to analyze and design more complex systems consisting of larger numbers of agents (e.g., in nature, society, or software), the need arises for a concept of higher abstraction than the concept agent. To this end, organizational modeling is becoming a practiced stage in the analysis and design of multi-agent systems. Hereby, the environment in which the multi-agent organization participates has to be taken into consideration. An environment can have a high degree of variability which might require organizations that change to adapt to the environment's dynamics, to ensure a continuous proper functioning of the organization. Hence, such change processes are a crucial function of the organization and should be part of the organizational model.

An organizational model incorporating organizational change can be specified in two ways: from a centralized perspective, in which there is a central authority that determines the changes to be performed within the organization, taking into account the current goals and environment, see e.g. [5]. A second possibility is to create a model for organizational change from a decentralized perspective, in which each agent decides for himself if and how to change its own role allocations. In the latter approach, it is much more difficult for the organization as a whole to change in a coherent way, still satisfying the goals set for the organization, as there is no overall view of the organizational change. The approach might however be the only possibility for an organization to perform change as a central authority for performing change could be non existing or infeasible due to the nature of the organization. In the domain of social insects, such as honeybees and wasps, organizations are known to adapt in a decentralized fashion to environmental changes. This paper presents a generic model for decentralized organization change appropriate for such phenomena as occur in Nature. Such a model can aid developers of multi-agent systems in creating and analyzing such an organization. The description of the model is done from a generic perspective, abstracting from the actual tasks being performed by the organization. The scope of the model is broader than simply being able to model social insects: the mechanisms incorporated in the model facilitating decentralized organizational change may work in other types of organizations as well. In [1] for example, a comparable approach is used for finding an optimal allocation of cars to paint booths.

To evaluate the generic model being proposed, as a case study the honeybee (Apis Mellifera) has been investigated. For this domain the generic model has been instantiated. The instantiated model has been validated against properties as acquired from biological experts. A number of different roles have been identified in the literature (see e.g., [8;11]). For the sake of brevity only five will be addressed here: (1) a *brood carer* takes care of feeding the larvae within the bee hive; (2) a *patroller* guards the hive by killing enemies entering the hive; (3) a *forager* harvests food to be stored in the hive; (4) an *undertaker* cleans the hive of corpses, and (5) a *resting worker* simply does nothing.

Switching between roles is triggered by changes in the environment observed by the bees. Such observations differ per bee. Each role has a specific trigger, for which a bee has a certain threshold that determines whether this is the role it should play. The bee always plays the role for which it is most triggered. For example, bees are triggered to start playing the *brood carer* role when they observe the larvae emitting a too high level of hunger pheromones. Once they are allocated to the role, they start getting food from the combs and feed the larvae that are emitting the pheromones. A trigger for the *patroller* role is the amount of enemies observed around the hive. *Foragers* that have returned from their hunt for food, communicate the location where they found the food by means of the honeybee dance (see [2]). For other bees currently not playing the *forager* role, such a dance is a trigger to start playing the *forager* role. The more corpses there are, the more bees are being triggered to switch from their current role to being *undertaker*. Bees perform the *resting worker* role in case they are not sufficiently triggered for any other role.

Section 2 presents the methodological approach used. The generic model for decentralized organizational change is described in Sections 3 (properties at organization level) and 4 (role properties). Results of a simulation of the generic organizational model instantiated with domain-specific knowledge of the bee colony are shown in Section 5, and finally Section 6 concludes the paper.

2 Modeling Organizational Dynamics

To enable modeling an organization, an expressive language is needed that has the ability to describe the dynamics of such an organization. For this purpose TTL (Temporal Trace Language) has been adopted cf. [6]. TTL allows for the formal specification of dynamic properties on multiple levels of aggregation. The bottom level addresses role properties, describing the required behavior for each of the roles within the organization. On the top level organization properties are defined, expressing the overall goals or requirements for the organization. An advantage of using TTL is that an executable subset has been defined called *leadsto* which is of the form $\alpha \rightarrow\!\!\!\!\rightarrow_{e,f,g,h} \beta$ that states that if α holds for duration g then β will holds for duration h with a delay between e and f. In case role properties are expressed in this executable format, the organizational model can be simulated by putting certain (e.g., environmental) events in the model (without including agents in the model), resulting in a trace of the organizational behavior. The top level organization properties can thereafter be checked against the trace by means of an automated tool called *TTL*

checker to see whether the organizational model indeed satisfies the goals or requirements set for it, given the events that have been put into the model. Using the results of these checks, statements can be made about the behavior of the organization, when the agents comply to the role properties that have been defined. More details and the semantics for TTL can be found in [9]. Examples and explanation of properties expressed in TTL are shown in Appendix A, which shows the formal form of all properties expressed in informal or semi-formal form below.

3 Organizational Properties

The model for decentralized organizational change presented here takes the form of a hierarchy of dynamic properties at two aggregation levels: that of the organization, and that of the roles within the organization. This section describes a number of such properties as well as the relationships between them.

The highest level requirement for the organization as a whole as inspired by the biological domain experts, is survival of the population given a fluctuating environment, in other words, population size needs to stay above a certain threshold M.

OP1(M) Surviving Population
For any time t, a time point t'≥t exists such that at t' the population size is at least M.

Such a high-level requirement is refined by means of a property hierarchy, depicted as a tree in Figure 1. At the highest level OP1 is depicted which can be refined into a number of properties (in Figure 1 *n* properties) each expressing that for a certain aspect the society is in good condition, characterized by a certain *value* for a variable (the *aspect variable*) that is to be maintained. The property template for an aspect X is as follows:

OP2(X, P1, P2) Organization Aspect Maintenance
For all time points t
If v is the value of aspect variable X at t, then v is between P1 and P2

Sometimes one of the two bounds is omitted, and it is only required that value v is at least P1 (resp., at most P2). For the example bee society the aspects considered are *wellfed brood*, *safety*, *food storage*, and *cleanness* (addressed, respectively, by Brood Care, Patroller, Forager, and Undertaker roles). For each of these aspects a variable was defined to indicate the state of the society for that aspect. For example, for wellfed brood, this variable concerns relative larvae hunger, indicated by the larvae pheromone rate.

In order to maintain the value of an aspect variable X, a certain effort is needed all the time. To specify this, a property that expresses the *effort* made by the organization on the aspect, is introduced. Notice that the notion of provided effort at a time point t can be taken in an *absolute* sense (for example, effort as the amount of feeding work per time unit), but it can also be useful to take it in a *relative* sense with respect to a certain overall amount, which itself can vary over time (for example, effort as the fraction of the amount of feeding work per time unit divided by the overall number of larvae). Below the latter, relative form will be taken. The general template property for aspect effort is as follows:

OP3(X, W1, W2) Sufficient Aspect Effort

For all time points t the effort for aspect X provided by the organization is at least W1 and at most W2.

For the bee colony, for instance, the brood care workers take care that the larvae are well-fed. The effort to maintain the hunger of larvae at a certain low level is feeding the larvae. Here provided effort for brood care is defined as the brood care work per time unit divided by the larvae population size. Brood care work is taken as the amount of the (average) brood care work for one individual brood carer times the number of brood carers.

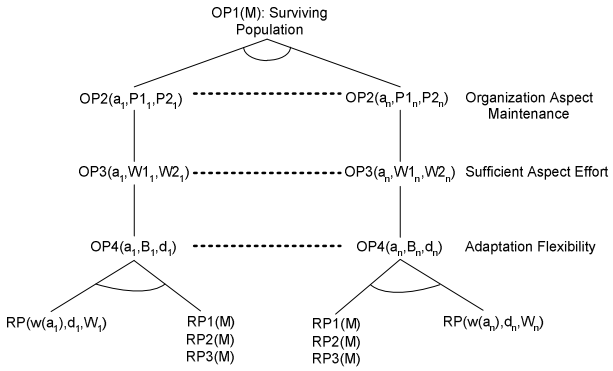

Fig. 1. Property hierarchy for decentralized organizational change

Whether the refined properties given above will always hold, depends on the flexibility of the organization. For example, in the bee colony case, if the number of larvae or enemies increases, also the number of brood care workers, respectively patrollers should increase. If the adaptation to the new situation takes too much time, the property Brood Care Effort will not hold for a certain time. In principle, such circumstances will damage the success of the organization. Therefore, an adaptation mechanism is needed that is sufficiently flexible to guarantee the properties such as Brood Care Effort. For this reason, the adaptation flexibility property is introduced, which expresses that when the effort for a certain organization aspect that is to be maintained is below a certain value, then within a certain time duration d it will increase to become at least this value. The smaller this parameter d is, the more flexible is the adaptation; for example, if d is very large, the organization is practically not adapting. The generic property is expressed as follows:

OP4(X, B, d) Adaptation Flexibility

At any point in time t, if at t the effort for aspect X provided by the organization is lower than B, then within time duration d the effort will become at least B.

An assumption underlying this property is that not all aspects in the initial situation are critical, otherwise the adaptation mechanism will not work. OP3 expressing that sufficient effort being provided directly depends on this adaptation mechanism as shown in Figure 1. OP4 depends on role properties at the lowest level of the hierarchy, which are addressed in the next Section.

4 Role Properties

Roles are the engines for an organization model: they are the elements in an organization model where the work that is done is specified. The properties described in Section 3 in an hierarchical manner have to be grounded in role behavior properties as the lowest level properties of the hierarchy. In other words, specifications of role properties are needed that entail the properties at the organizational level described in Section 3. In the behavioral model two types of roles are distinguished: Worker roles which provide the effort needed to maintain the different aspects throughout the organization, and Member roles which have the function to change Worker roles. Each Member role has exactly one shared allocation with a Worker role. The role behavior for the Worker roles within the organization is shown in Section 4.1, whereas Section 4.2 specifies the behavior for the Member roles.

4.1 Worker Role Behavior

Once a certain Worker role exists as an active role, it performs the corresponding work. What this work exactly is, depends on the application: it is not part of the generic organization model. The property directly relates to OP4 which specifies the overall effort provided, as shown in Figure 1. Note that Figure 1 only shows the generic form of the role property (depicted as $RP(w(a_i),d_i,W_i)$ where a_i is the specific aspect and $w(a_i)$ the Worker role belonging to that aspect) whereas in an instantiated model a role property is present for each instance of the Worker role providing the effort for the specific aspect. In a generic form this is specified by:

RP(R, d, W) Worker Contribution
For all t there is a t' with $t \leq t' \leq t + d$ such that at t' the Worker role R delivers a work contribution of at least W.

4.2 Member Role Behavior

By a Member role M decisions about taking up or switching between Worker roles are made. As input of this decision process, information is used about the well-being of the organization, in particular about the different aspects distinguished as to be maintained; these are input state properties indicating the value of an aspect variable X: has_value(X, v). Based on this input the Member role M generates an intermediate state property representing an indication of the aspect that is most urgent in the current situation. In the generic model the decision mechanism is indicated by a priority relation priority_relation(X_1, v_1, w_1, …, X_n, v_n, w_n, X) indicating that aspect X has priority in the context of values v_i, respectively norms w_i for aspects X_1, .., X_n. This priority relation can be specialized to a particular form, as shown below by an example specialization in the last paragraph of this section.

RP1(M) Aspect Urgency
At any t, if at t Member role M has norms w_1 to w_n for aspects X_1 to X_n
and receives values v_1 to v_n for X_1 to X_n at its input,

and has a priority relation that indicates X as the most urgent aspect for the combination of these norms and values,
then at some t' ≥t it will generate that X is the most urgent aspect.

Based on this, the appropriate role for the aspect indicated as most urgent is determined. If it is not the current role sharing an allocation with M, then another intermediate state property is generated expressing that the current Worker role sharing an allocation with M should be changed to the role supporting the most urgent aspect. In other words, the shared allocation of Member role M in the Change Group should change from one (the current) Worker role R1 in Worker Group WG1 to another one, Worker role R2 in Working Group WG2:

RP2(M) Role Change Determination
At any t, if at t Member role M generated that X is the most urgent aspect,
and Worker role R2 is responsible for this aspect,
and R1 is the current Worker role sharing an allocation with M, and R1 ≠ R2,
then at some t' ≥t it will generate that role R2 has to become the Worker role sharing an allocation with M, instead of R1.

Based on this intermediate state property the Member role M generates output indicating which role should become a shared allocation and which not anymore:

RP3(M) Role Reallocation
At any t, if at t Member role M generated that Worker role R2 has to become sharing an allocation with M, instead of Worker role R1,
then at some t' ≥t it will generate the output that role R1 will not share an allocation with M and R2 will share an allocation with M.

All three role properties for the Member roles are depicted in Figure 1. The adaptation step property OP4 for all organizational aspects dependent upon it, so each of the OP4 branches depends upon RP1, RP2, and RP3 which have therefore been depicted two times in the Figure.

The generic description for the Member role behavior can be specialized one step further by incorporating a specific decision mechanism. This gives a specific definition of the priority relation priority_relation(X_1, v_1, w_1, …, X_n, v_n, w_n, X) as has been done for the following decision mechanism based on norms used as thresholds (see e.g. [10]).

1. For each aspect X to be maintained a norm w(X) is present. For the Worker role R1 for X sharing an allocation with Member role M, each time unit the norm has a decay described by fraction r.
2. For each X, it is determined in how far the current value is unsatisfactory, expressed in a degree of urgency u(X) for that aspect.
3. For each aspect with urgency above the norm, i.e., with u(X) > w(X), the relative urgency is determined: u(X)/ w(X)
4. The most urgent aspect X is the one with highest relative urgency.

5 Simulation Results

This section discusses some of the results of simulations that have been performed based on the generic organizational model, in particular the role properties presented in Section 4 have been put in an executable format and have been instantiated with domain-specific information for bee colonies.

To validate the instantiated simulation model, the high-level dynamic properties from Section 3 were used (in accordance with biological experts). Proper functioning of such an organization in Nature is not self-evident, therefore two simulation runs are compared: one using the adaptation mechanism, and one without. Note that the results presented here are the results of a simulation of the instantiated organizational model, abstracting from allocated agents. Performing such high-level simulations of an executable organizational model enables the verification of properties against these simulation runs. Hence, it can be checked whether or not the model satisfies the properties or goals considered important. When such properties are indeed satisfied, by allocating agents to the roles that comply to the role properties, the multi-agent system delivers the desired results as well. In the two simulations, several parameters have been set to certain values, where the circumstances are kept identical for both simulations. See appendix B for the details on the settings used.

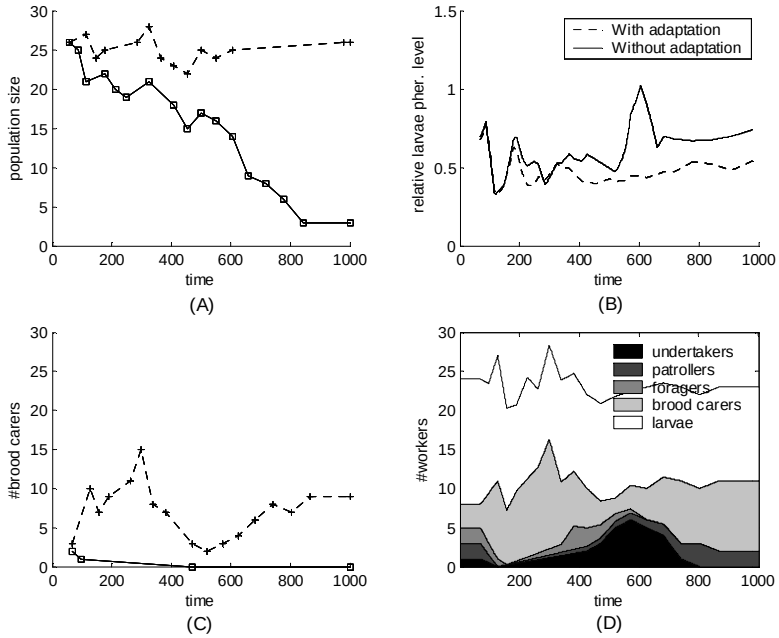

Fig. 2. Results of simulating the bee colony with and without adaptation. Note that (D) only shows the worker types for the adaptive case

Figure 2 shows results on the performance of the two settings of the organizational model. Figure 2a shows the overall population size over time. The population size of the simulation with adaptation remains relatively stable, whereas without adaptation it drops to a colony of size 3, which is equal to the amount of larvae living without being fed. Figures 2b and 2c show information regarding brood care: Firstly, the average pheromone level, the trigger to activate the allocation to brood carer. Furthermore, the number of active brood carers in the colony is shown. In the case with adaptation their number increases significantly in the beginning of the simulation, as the amount of pheromones observed is relatively high. Therefore, a lot

of the brood carer roles are allocated. For example, at time point 300, 15 out of a population of 28 are brood carers.

Despite the fact that the overall pheromone level is not decreasing rapidly, the amount of brood carer roles drops significantly after time point 300. This is due the fact that Member roles can only share an allocation with one Worker role at a time. When another role receives a higher urgency (e.g., there is a huge attack, demanding many patrollers) a switch of worker role takes place. Figure 2d shows the amount of worker roles of the different types (except the resting workers) within the bee colony for the setting with adaptation. The amount of brood carers decreases after time point 300 due to an increase in the amount of shared allocations to the undertaker and forager roles. This results in an increase in pheromone level again, causing a higher delta for brood care again, resulting in more brood carers, etc. The pheromone level finally stabilizes around 0.5 in the organizational model with adaptation. For the setting without adaptation, the brood carers simply cease to exist due to the fact that none of the larvae are growing up. The pheromone level stabilizes at a higher level.
The properties from Section 3 have been checked by the automated TTL checker. With the following parameter settings, the properties were validated and confirmed for the organizational model with adaptation and falsified for the one without adaptation: OP1(20), OP2(broodcare,0,0.9), OP3(broodcare,0.15,10000), OP4(broodcare, 0.3, 200).

6 Discussion

The generic organizational model for decentralized organizational change has been formally specified by means of a methodology which describes the behavior of an organization on multiple aggregation levels; cf. [6]. The model is inspired by mechanism observed in Nature. As a first evaluation, the model was validated for a honeybee colony case study. The scope of the model is not limited to being a model for social insects: in [1] the effectiveness of such approaches is shown for other domains as well. The model can therefore support organizational modelers and analysts working with multi-agent organizations in highly dynamic environments, without a central authority directing change, in general in designing and analyzing such an organization. The formal specification of the behavior of the organization is described by dynamic properties at different aggregation levels. Once the lowest level properties within the organization are specified in an executable form, the organizational model can be used for simulation abstracting from agents (to be) allocated. Such low level properties can be indicative for the behavior of the agent allocated to that particular role. The possibility also exists to specify the role properties at the lowest aggregation level in a more abstract manner, in a non-executable format. Hierarchical relations between the properties can be identified to show that fulfillment of properties at a lower level entails the fulfillment of the higher level properties. Simulations using agents can be performed and checked for fulfillment of these properties. Properties for the behavior of roles regarding decentralized organizational change have been specified on an executable level to be able to perform simulation, and higher-level properties have been identified as well.

The case study of the honeybee colony was used as a first evaluation of the model. Simulation of this instantiated model showed that given the external circumstances, it was effective, given overall properties put forward by biological experts. For a comparison of the work presented in this paper with related multi-agent organization research, see Appendix C.

References

[1] Bonebeau, E. and Theraulaz, G., 2000, Swarm Smarts, Scientific American, 282 (3): 72-79.

[2] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E., 2001, *Self-Organization in Biological Systems*, Princeton University Press, Princeton, USA.

[3] Furtado, V., Melo, A., Dignum, V., Dignum, F., Sonenberg, L., 2005, Exploring congruence between organizational structure and task performance: a simulation approach. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. of the 1st OOOP Workshop*.

[4] Giorgini, P., Müller, J., Odell, J. (eds.), 2004, *Agent-Oriented Software Engineering IV*, LNCS, vol. 2935, Springer-Verlag, Berlin.

[5] Hoogendoorn, M., Jonker, C.M., Schut, M., and Treur, J, 2004, Modelling the Organisation of Organisational Change. In: Giorgini, P., and Winikoff, M., (eds.), *Proceedings of the 6th International Workshop on Agent-Oriented Information Systems (AOIS'04)*, pp. 29-46.

[6] Jonker, C.M., Treur, J. 2002, Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. *Int. Journal of Cooperative Information Systems*,vol.11, pp.51-92.

[7] McCallum, M., Vasconcelos, W.W., and Norman, T.J., 2005, Verification and Analysis of Organisational Change. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. 1st OOOP Workshop*.

[8] Schultz, D.J., Barron, A.B., Robinson, G.E., 2002, A Role for Octopamine in Honey Bee Division of Labor, *Brain, Behavior and Evolution*, vol. 60, pp. 350-359.

[9] Sharpanskykh, A., Treur, J., 2005, Temporal Trace Language: Syntax and Semantics, Technical Report, Vrije Universiteit Amsterdam, Department of Artificial Intelligence, Amsterdam.

[10] Theraulaz, G., Bonabeau, E., and Deneubourg, J.L., 1998, Response thresholds reinforcement and division of labor in insect societies. Proceedings of the Royal Society of London Series B-Biological Sciences, 265: 327-332.

[11] Winston, M.L. and Punnet, E.N., 1982, Factors determining temporal division of labor in honeybees, *Canadian Journal of Zoology*, vol. 60, pp. 2947-2952.

Appendix A: Properties Formalized in TTL

This Appendix presents the formal form for each of the properties presented throughout the paper.

OP1(M) Surviving Population

$\forall t\ \exists t' \geq t, v$: state(γ, t') |= total_living_population_count(v) & $v \geq M$

Here state(γ, t') |= total_living_population_count(v) denotes that within the state state(γ, t') at time point t' in trace γ the state property total_living_population_count(v) holds, denoted by the (infix) predicate |= for the satisfaction relation.

OP2(X, P1, P2) Organization Aspect Maintenance

$\forall t, v$: state(γ, t) |= has_value(X, v) $\Rightarrow$ $P1 \leq v \leq P2$

OP3(X, W1, W2) Sufficient Aspect Effort

$\forall t, v$: state(γ, t) |= provided_effort(X, v) $\Rightarrow$ $W1 \leq v \leq W2$

OP4(X, B, d) Adaptation Flexibility

$\forall t, v1$ [[state(γ, t) |= provided_effort(X, v1) & $v1 < B$] $\Rightarrow$
$\exists t' \geq t, v2$: [$t' \leq t+d$ & state(γ, t') |= provided_effort(X, v2) & $v2 \geq B$]]

RP(R, d, W) Worker Contribution

$\forall t$ $\exists t' \geq t, v$: [$t' \leq t+d$ & state(γ, t') |= work_contribution(R, v) & $v \geq W$]]

Here work_contribution is part of the state ontology for the output of the role. For each of the specific roles it can be specified what the work contribution is in terms of the domain specific state ontology (e.g., the number of larvae to be fed for the brood carer role).

RP1(M) Aspect Urgency

$\forall t, v1, .., vn, w1, .., wn, X$
state(γ, t) |= has_value(X_1, v_1) & … & has_value(X_n, v_n) &
has_norm(X_1, w_1) & … & has_norm(X_n, w_n) &
priority_relation(X_1, v_1, w_1, …, X_n, v_n, w_n, X)
$\Rightarrow \exists t' \geq t$ state(γ, t') |= most_urgent_aspect(X)

RP2(M) Role Change Determination

$\forall t$, X, R1, R2 state(γ, t) |= most_urgent_aspect(X) &
role_responsible_for(R2, X) & role_reserved_for(R2, M) &
state(γ, t) |= has_shared_allocation(M, R1) & $R1 \neq R2$
$\Rightarrow \exists t' \geq t$ state(γ, t') |= shared_allocation_change(M, R1, R2)

RP3(M) Role Reallocation

$\forall t$, R1, R2
state(γ, t) |= shared_allocation_change(M, R1, R2)
$\Rightarrow \exists t' \geq t$ state(γ, t') |= not has_shared_allocation(M, R1) &
has_shared_allocation(M, R2)

Appendix B: Setting Used for the Simulation

This appendix addresses the settings that have been used for the simulation as presented in the paper.

External world. Initially, 15 larvae and 10 workers are present for which the initial type of the latter is randomly assigned. The natural mortality age is set to 500 time steps, whereas a larva is grown up after 250 time steps. Every 20 time steps, a new larva is added to the population. The initial food stock is set to 40 units of food. Once every 100 time points an attack of 40 enemies occurs, who stay there until a patroller defeats them. In case over 200 enemies are present in the hive, each individual in the organization is removed with a probability of 0.05 per time step. In case more than 20 dead bodies are present in the hive, individuals are removed with the same probability. Food used by larvae is 0.5 per feed, for workers 1 unit of food per time step.
Larvae. Larvae have an initial pheromone level of 0.5, increasing 0.006 per time step. In case pheromone emissions exceed 0.95, the larva dies. After being fed, the emission level is set to 0.1.
Foragers. Foragers each collect 1 food unit per 3 time steps.
Brood carers. Feed 1 larvae per 8 time steps, and only feed the larvae with a pheromone level above 0.55.
Undertaker. Carry 1 body per 12 time steps.
Patroller. Defeat 1 enemy per time step.
In the adaptation simulation, the Member thresholds are randomly generated, being somewhat above or below the average observed value of the various triggers.

Appendix C: Discussion of Related Multi-Agent Organization Research

Describing multi-agent organizations from a normative perspective is a popular viewpoint nowadays (see e.g. [2]). In such approaches, the behavior of agents is restricted via the norms specified for the role they are playing within the organization. In this paper, role properties can be used to formulate such restrictions. The following norm, for example (from [6]) "Students are prohibited from sitting the exam if they have not completed the assignment" can easily be formulated in terms of a dynamic property for the student role. In this way, the approach in this paper is suitable for modeling organizations in terms of norms. In case simulations using agents are performed, the role properties specifying such norms can be checked to see whether the behavior of the agents violates the norms or not.

A number of approaches have been introduced that enable an analysis of organizations in combination with organizational change. In [6] a framework is introduced which enables verification and analysis of organizational change. In the framework, changes in the organizational structure are allowed, however the process of organizational change itself is not addressed nor modeled, contrasting it from this paper. Their framework enables verification on the static organizational model to check whether the organizational model is workable. Furthermore, analysis is performed on simulations of possible outcomes of the organizational model, which is meant to see how the organization will act when populated by different societies of agents. In this paper, a simulation framework by means of executable role properties is used which abstracts from agents, and enables an analysis of the model for decentralized organizational change both from a static and dynamic viewpoint. It also allows the identification of hierarchical (interlevel) relations between dynamic properties at different aggregation levels. Verification is used to show that compliance of all agents to their role properties constitutes the satisfaction of the overall organizational properties that have been set. The relationship between the agents and the role specification was not addressed here as this is beyond the scope of this paper. [3] emphasize the necessity for multi-agent organizations to have the ability to reorganize, and state that additional requirements are needed for agents that have the ability to change. This paper makes such requirements more specific for decentralized organizational change in terms of role properties, and aggregates these to organizational properties, such as shown in Section 3. In [4] the performance of different organizational structures is compared by using agent simulations. The emphasis is again on finding an effective model to change towards, not on modeling the change process itself which this paper does address.

Organizational modeling approaches initially designed to model nonchanging organizations have been extended to include reorganization as well. The reorganization process in MOISE+ is addressed in [5], in which four phases in a reorganization are identified for controlled organizational change: (1) monitoring phase; (2) design phase; (3) selection phase, and (4) implementation. The reorganization is being controlled by an organization manager within a reorganization group. This makes the change centrally directed, based on certain organizational goals that have been set and demand a reorganization. In the decentralized organizational

change description in this paper however, the fulfillment of the organizational goals is reached by the trigger based behavior of the members roles within the change group, which is in fact the other way around. In GAIA [7] organizational modeling is addressed from a perspective in which firstly the requirements are identified after which the appropriate organization satisfying these particular requirements is selected. The methodology is however not intended to specify organizational models that include organizational change as being part of the model. The same can be said for Agent UML [1].

References

[1] Bauer, B., Muller, J.P., Odell, J., 2001, Agent UML: A formalism for specifying multiagent interaction, In: Ciancarini, P, and Wooldridge, M. (eds.), *Proc. AOSE'01*, Spinger-Verlag, pp. 91-103.

[2] Castelfranchi, C., 1998, Modelling Social Action for AI Agents. *Artificial Intelligence*, 103: pp.157 - 182.

[3] Dignum, V., Sonenberg, L., Dignum, F., 2004, Dynamic Reorganization of Agent Societies, In: *Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004*.

[4] Furtado, V., Melo, A., Dignum, V., Dignum, F., Sonenberg, L., 2005, Exploring congruence between organizational structure and task performance: a simulation approach. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. of the 1st OOOP Workshop*.

[5] Hübner, J.F. and Sichman, J.S., 2003, Using the MOISE+ model for a cooperative framework of MAS reorganization, Boletim Técnico BT/PCS/0314, Escola Politécnica da USP, São Paulo.

[6] McCallum, M., Vasconcelos, W.W., and Norman, T.J., 2005, Verification and Analysis of Organisational Change. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. 1st OOOP Workshop*.

[7] Zambonelli, F., Jennings, N., Wooldridge, M., 2001, Organizational Rules as an Abstraction for the Analysis and Design of Multi-agent Systems, *Journal of Software and Knowledge Engineering*, Vol. 11, pp. 303-328.

Chapter 10

An Adaptive Multi-Agent Organization Model Based on Dynamic Role Allocation

Part of this chapter appeared as: Hoogendoorn, M., and Treur, J., An Adaptive Multi-Agent Organization Model Based on Dynamic Role Allocation. In: Nishida, T., Klusch, M., Sycara, K., Yokoo, M., Liu, J., Wah, B., Cheung, W., and Cheung, Y.-M. (eds.), *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006)*, IEEE Computer Society Press, 2006, pp.474-481.

An Adaptive Multi-Agent Organization Model Based on Dynamic Role Allocation

Mark Hoogendoorn and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, treur}@cs.vu.nl

Abstract. Organizations involving multiple agents require adaptation mechanisms to guarantee robustness, especially in critical domains. This paper presents an organizational template to aid analysis and design of organizations with adaptation mechanisms based on dynamic role reallocation. The adaptive organization model can be used both for qualitative and quantitative domains, as is shown in two application cases made to evaluate the applicability of the model.

1 Introduction

Robustness of a multi-agent organization functioning in critical domains is essential. Unpredictability can both be in the internal functioning of the system itself (e.g., an incorrect functioning agent), or external to the system (e.g., a sudden increase in environmental pressure). To enable a multi-agent organization to be robust, capabilities are required that allow the organization to adapt in order to continue functioning adequately.

An approach could be to model a multi-agent system in which each of the agents have those specific capabilities, and show the effectiveness of the system as a whole in this domain. However, it is hard to generalize results obtained beyond the specific domain and the specific agents occurring in this domain. Recently, an abstraction level higher than the concept agent has become in use: the organizational level (see e.g. [3], [12]). On this level, templates can be specified to aid analysts in modeling appropriate multi-agent organization models. These templates, for example, include specification of roles, possibly in the form of required behavior. In a given application, agents can be allocated to such roles. The templates can be reused each time a new domain is analyzed for which the characteristics comply to the ones specified for the template. Once the correctness of the template is proven (given certain domain assumptions) for a desired property, each model which complies to the specified template will satisfy that property as well, making the approach reusable. Of course, for each new case in which the template is used, an instantiation with domain-specific knowledge is still required.

This paper presents such an organizational model or template for the analysis of multi-agent organizations with the ability to adapt to unpredictable circumstances,

maintaining the robustness of the system. The essential part of the organizational model is the specification of roles, since those can be seen as the engines of the organization. The approach taken distinguishes a number of aggregation levels, starting with the highest level dynamic property desired (i.e., robustness) and refining this property in a number of steps until the level of role behavior has been reached. Interlevel relations between dynamic properties at the different aggregation levels have been specified and verified using the model checker SMV [21]. The applicability of the model has been evaluated by using it to analyze two application case studies in different domains, one qualitative, and one quantitative.

The remainder of this paper is organized as follows. Section 2 introduces the modeling approach used to specify the organization model, which includes both a structural and a behavioral specification of the organization. The structural model within the template is specified in Section 3, whereas Section 4 presents the behavioral model, without taking into account adaptation. The adaptation model is presented in Section 5. Section 6 presents a qualitative application of the template in the domain of incident management and presents simulation results. In Section 7 a quantitative specialization of the model is specified which is applied in the domain of social insects in Section 8. Section 9 is a discussion.

2 Modeling Approach

This Section presents the modeling approach used to specify the adaptive multi-agent organization model. First, the framework used to model organization structure will be explained, and thereafter the method for describing the behavior of such an organization. For describing the structural part of the model for adaptive multi-agent organizations, the AGR (Agent-Group-Role) modeling approach introduced in [7] is used. Here three basic concepts are used to model a multi-agent organization: agent, group, and role. An agent is an active communicating entity which plays roles within groups. Groups are sets of roles; a role is an abstract representation of an agent function or service.

The approach presented in [8] which is partly based on AGR is used to specify behavioral or dynamic properties on multiple aggregation levels: following AGR, the functioning of the particular roles, the functioning of groups, and of the organization as a whole. In general, behavioral properties are expressed in terms of temporal relations over input states and output states of roles over time. Properties at the different levels can be structured by means of interlevel relations in a hierarchy. At the lowest level role properties describe the behavior of an individual role whereas transfer properties describe the dynamics of (intragroup) transfer between roles. For the roles within a given group, such role properties, together with the transfer properties, entail the group properties that characterize the behavior of the group as a whole. The group properties for the different groups, together with the inter-group relationship properties (for transfer between groups), entail the overall organization properties. Properties about the environment are treated the same way as groups and roles.

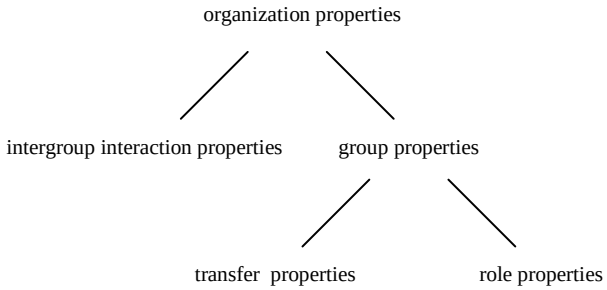

Fig. 1. Overview of interlevel relations between dynamic properties

Specification of dynamic properties is done in the *Temporal Trace Language* (TTL) [16]. This language can be classified as a reified predicate-logic-based temporal logic (see, e.g., [10], [11]), in contrast to, for example, modal-logic-based temporal logics as the ones discussed in, e.g., [9]. The language is briefly introduced here. For more details, see [4].

2.1 States and Traces

In TTL, ontologies for world states are formalized as sets of symbols in sorted predicate logic. For any STATE ontology Ont, the ground atoms form the set of *basic state properties* BSTATPROP(Ont). Basic state properties can be defined by nullary predicates (or proposition symbols) such as n-ary predicates (with n>0) like has_temperature(environment, 7). The *state properties* based on a certain ontology Ont are formalized by the propositions (using conjunction, negation, disjunction, implication) made from the basic state properties; they constitute the set STATPROP(Ont).

In order to express dynamics in TTL, in addition to state properties, important concepts are *states*, *time points*, and *traces*. A *state* S is an indication of which basic state properties are true and which are false, i.e., a mapping S: BSTATPROP(Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont). Moreover, a fixed *time frame* T is assumed which is linearly ordered. Then, a *trace* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by TRACES(Ont), i.e., TRACES(Ont) = STATES(Ont)^T.

2.2 Dynamic Properties

Patterns in world dynamics are described by dynamic properties. The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a

trace γ over state ontology Ont, a certain state of the world at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate |=, comparable to the Holds-predicate in event calculus [17] or situation calculus [23]. Thus, state(γ, t) |= p denotes that state property p holds in trace γ at time t. Here state properties are considered objects and denoted by term expressions in the TTL language. Likewise, state(γ, t) |$\neq$ p denotes that state property p does not hold in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted predicate logic, using the usual logical connectives such as $\neg$, &, $\vee$, $\Rightarrow$, and the quantifiers $\forall$, $\exists$ (e.g., over traces, time and state properties). For example, consider the following dynamic property for a pattern concerning belief creation based on observation:

if at any point in time t1 the agent A observes that it is wet outside,

then there exists a time point t2 after t1 such that at t2 in the trace the agent A believes that it is wet outside

This property can be expressed as a dynamic property in TTL form (with free variable for trace γ) as follows:

$\forall t{:}T\ [\ state(\gamma, t) \models observes(itswet) \ \Rightarrow\ \exists t' \geq t\ \ state(\gamma, t') \models belief(itswet)\ ]$

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \twoheadrightarrow_{e, f, g, h} \beta$, means:

if state property α holds for a certain time interval with duration g,

then after some delay (between e and f) state property β will hold for a certain time interval of length h.

3 Adaptive Organization Model: Organizational Structure

This Section presents the structural model within the adaptive organization model; see Figure 2. Here, small ovals denote roles, bigger ovals denote groups, solid arrows denote transfers between roles, and dashed lines denote inter-group interaction. The model can be composed from two parts, of which the structure of one (the lower layer in Figure 2) is dependent upon the specific domain of application, and the other structure (top layer, depicted in gray), consisting of the *Change Group*, by which adaptation of the organization takes place, is generic for any type of application. All the agents participating in the organization are assumed to have an *Adaptor* role in this group. This role has the ability to monitor the existing agent-role allocations, and role, group and organization properties (hence, the group has a meta-view on the organization). In case it is observed that a property is not satisfied, an Adaptor role makes decisions about change, thereby possibly requesting other Adaptor roles to perform a role for which errors have been observed. The exact specification of the Adaptor role is addressed in Section 5.

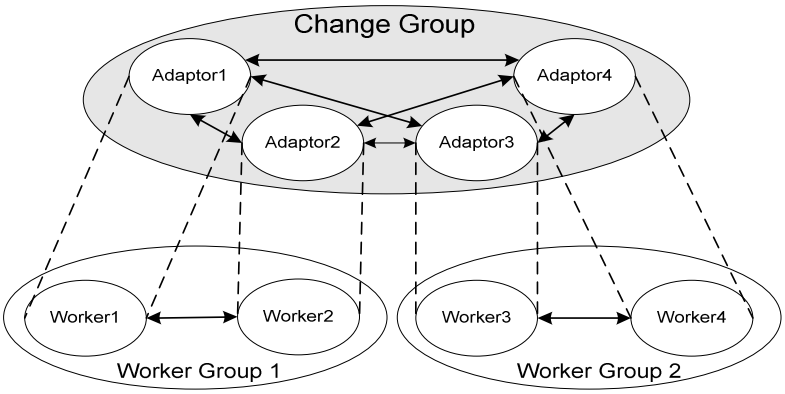

Fig. 2. Adaptive Organizational Model

In the lower part of Figure 2, *Worker Groups* are shown of which each one addresses a particular part of the tasks needed to be performed within an organization. In incident management there would, for example, be a fire fighting group, medical group, and a police group. Notice that names in small ovals such as Worker1, .., Worker4, used here denote the roles in these groups, not the agents allocated to these roles. The adaptativity in the model is in the flexibility of allocations of agents to these roles, not in the change of the roles themselves. Such an allocation change may involve, for example, that for an agent A that was allocated to role Worker1 within Worker Group 1, its allocation is changed to an allocation to role Worker4 in Working Group 2.

4 Adaptive Organization Model: Organizational Behavior

The behavioral model within the adaptive organization model takes the form of a hierarchy of dynamic properties at the different aggregation levels of the organization (see Figure 1 and 3). The relationships between the different levels within the hierarchy have been verified using the SMV model checker; cf. [21]. The highest organizational properties express what one wants a particular organization (as a whole) to establish, e.g., based on *performance indicators* of an organization. This section shows how such organizational properties are refined into more specific properties for particular aspects or parts of an organization (Section 4.1). These are further refined to the aggregation level of the particular groups within the organization (Section 4.2).

4.1 Organization-Level Properties

The highest level organization properties, expressing the well-being or satisfactory functioning of the organization, can take various forms; e.g., see the specific applications presented in Sections 6 and 8. In the adaptive organization model presented here the assumption is that robustness of the organization is a main organization property to be achieved. An organization is said to be robust in case all relevant aspects of the organization are well maintained, despite environmental or internal fluctuations. Therefore, to achieve the goals of the organization a number of aspects X1, …, Xn can be distinguished that have to be maintained; e.g., [1] p. 58, 83. Examples of such aspects in the context of incident management are fire fighting, health care, and traffic care. Thus, a main property for the whole organization is that the organization functions well for the combination of these aspects X1, …, Xn. The organization property OP expresses that at all points in time proper maintenance of the combination of aspects is satisfied:

OP = ∀t:TIME state(γ, t, O) |= satisfied(combination(X1, … Xn)).

Here

state(γ, t, O) |= satisfied(combination(X1, … Xn))

denotes that within the state state(γ, t, O) at time point t of the organization O in trace γ the state property satisfied(combination(X1, … Xn)) holds, with the infix predicate |= denoting the satisfaction relation between a state and a state property. Notice that such a state property can have different truth values at different points in time.

Other relevant organization properties (e.g., survival) are assumed to be entailed by this primary organization property OP. The organization property OP is refined using properties for different aspects of the organization: For any of the aspects X, the property OAP(X), expresses that at all points in time aspect X is maintained in a satisfactory manner:

OAP(X) = ∀t:TIME state(γ, t, O) |= satisfied(X).

These properties are assumed to relate to the overall organization property by

∀X OAP(X) ⇔ OPl

In other words, as long as all aspect properties are satisfied, the organization as a whole functions in a satisfactory manner. To this end it is assumed that:

satisfied(combination(X1, … Xn)) ↔ ∀X satisfied(X).

Then the bi-implication above can be rephrased as

∀X state(γ, t, O) |= satisfied(X) ⇔ state(γ, t, O) |= ∀X satisfied(X)

which is one of the axioms for the predicate |= within the logical language TTL.

4.2 Group-Level Properties

In order to achieve the robustness of the organization, depending on circumstances, the organization needs to spend a certain effort on each of the distinguished aspects. As circumstances may change, it is here that adaptive control is possible and needed; such control can be insufficient, leading to one or more not well-maintained aspects, or sufficient, leading to all aspects well-maintained. In the organizational structure

within the model, for each of the aspects a *Worker Group* is included to provide sufficient effort at each point in time as required to maintain this aspect, given the circumstances at that point in time.

GP1(X, G) : group provides required effort

For all time points t the effort provided by group G for aspect X is sufficient for the aspect.

$\forall$t:TIME, E:EFFORT

[state(γ, t, G) |= group_relates_to(G, X) $\wedge$
provides_group_effort_for(G, E, X)
$\Rightarrow$ state(γ, t, O) |= satisfies_required_effort_for(E, X)]

Here the antecedent denotes that within the state state(γ, t, G) at time point t of group G in trace γ the state property

group_relates_to(G, X) $\wedge$ provides_group_effort_for(G, E, X)

holds, expressing that group G relates to aspect X and provides effort E. Moreover,

state(γ, t, O) |= satisfies_required_effort_for(E, X)

expresses that at time t in trace γ the effort E is the effort required to satisfy aspect X. It is assumed as part of the organization model that when the effort provided by group G relating to X satisfies the required effort for aspect X, then X is considered satisfied:

group_relates_to(G, X) $\wedge$
provides_group_effort_for(G, E, X) $\wedge$
satisfies_required_effort_for(E, X) $\rightarrow$ satisfied(X)

Therefore, group effort property GP1(G, X) relates to the corresponding aspect property OAP(X) as follows:

GP1(X, G) $\Rightarrow$ OAP(X).

The current roles within the group G are the ones that actually provide the effort for X. Each role has a particular effort it can provide, based on the role specification. In order to provide the required effort, sufficient effort of specific roles within a group is needed that together deliver enough combined effort; this is expressed in GP2. Here the sorts ROLECOMBINATION and EFFORTCOMBINATION denote sorts for combinations of (a finite number of) roles and of efforts, respectively. The latter sort is a subsort of EFFORT.

GP2(X, G) : roles provide required effort

For all time points t the total effort E1,...,En provided by the roles R1,...,Rn within group G addressing aspect X provides a combined effort satisfying the effort required for X.

$\forall$t:TIME, RC:ROLECOMBINATION, EC:EFFORTCOMBINATION:

[state(γ, t, O) |= group_relates_to(G, X) $\wedge$
group_has_roles(G, RC) $\wedge$
provides_effort_combination(RC, EC)
$\Rightarrow$ state(γ, t, G) |= provides_group_effort_for(G, EC, X) &
state(γ, t, O) |= satisfies_required_effort_for(EC, X)]

This property relates to the previous one as follows:

GP2(X, G) $\Rightarrow$ GP1(X, G)

4.3 Role-Level Properties

One role property is present on the lowest level not devoted to adaptation: Each of the active worker roles performs a certain amount of work. This is expressed as follows:

RP1(R) Worker Contribution

For all t the Worker role R delivers an effort E.

∀t :TIME ∃E :EFFORT

state(γ, t, R) |= provides_role_effort(R, E)

5 Adaptive Organization Model: Organizational Adaptation

This Section presents the adaptation properties for the organization model. First, the adaptation properties expressing how the organization can achieve or maintain its goals under changing circumstances are introduced. Thereafter, the Adaptor role properties are presented which form the engines of the adaptation process.

5.1 Adaptation Properties

Within the organization the aspects distinguished are monitored all the time, in the sense that it is verified whether the provided effort is expected to stay sufficient for the required effort. To this end a signaling property is specified, based on desired effort. The property indicates those cases and time points that the effort observed for a certain aspect is close to becoming insufficient to satisfy the effort required for that aspect. The margin between the time point of signaling not satisfying the desired effort and the time point that the required effort is at risk of not being satisfied, is assumed large enough to have time to adapt. The adaptation mechanism within the organization has to guarantee that the effort will satisfy the desired effort again within a certain duration, without dissatisfying the required effort in the meantime; this to prevent property GP1 not being satisfied. This adaptation is expressed by the group adaptation property AP1.

AP1(X, G, d): Group adaptation for desired effort

For all time points t, in case the current effort E provided by group G for aspect X is not satisfying the desired effort, then at a later point in time t2 (where t2 > t and t2 < t+d) the organization has changed such that the effort provided satisfies the desired effort and in between will still satisfy the required effort.

∀t:TIME, G:GROUP, E1, E2:EFFORT:

[[state(γ, t, O) |= group_relates_to(G, X) &

state(γ, t, G) |= provides_group_effort_for(G, E1, X) &

state(γ, t, O) |= satisfies_required_effort_for(E1, X) &

state(γ, t, O |= not satisfies_desired_effort_for(E1, X)]
⇒ ∃ E2:EFFORT, t2 > t [t2 < t+d &
state(γ, t2, G) |= provides_group_effort_for(G, E2, X) &
state(γ, t2, O) |= satisfies_desired_effort_for(E2, X) &
∀t1 [t ≤ t1 ≤ t2 ⇒ state(γ, t1, O) |= satisfies_required_effort_for(E2, X)]]
]

This property relates to the previous properties as follows:

AP1(X, G, d) ⇒ GP1(X, G)

The group property for adaptation can be related to adaptation properties of individual roles taken as follows.

AP2(X, G, d) : Role adaptation for desired effort

For all time points t, in case the current effort combined from role efforts E1,…,En provided by the roles R1,….,Rn in G is not satisfying the desired effort, then at a later point in time t2 (where t2 > t and t2 < t+d) the organization has changed such that the effort combined from efforts provided by the roles within G satisfies the desired effort and in between will still satisfy the required effort.

∀t:TIME, RC1:ROLECOMBINATION, EC1:EFFORTCOMBINATION
[[state(γ, t, G) |= group_relates_to(G, X) ∧
group_has_roles(G, RC1) ∧
provides_effort_combination(RC1, EC1) &
state(γ, t, O) |= satisfies_required_effort_for(EC1, X) &
state(γ, t, O) |= not satisfies_desired_effort_for(EC1, X)]
⇒
∃ t2 > t, RC2:ROLECOMBINATION, EC2:EFFORTCOMBINATION
[t2 < t+d & state(γ, t2, O) |= group_relates_to(G, X) ∧
group_has_roles(G, RC2) ∧
provides_effort_combination(RC2, EC2) &
state(γ, t2, O) |= satisfies_desired_effort_for(EC2, X) &
[∀t'≤ t2 [t'> t ⇒
∃RC3:ROLECOMBINATION, EC3:EFFORTCOMBINATION
state(γ, t', O) |= group_relates_to(G, X) ∧
group_has_roles(G, RC3) ∧
provides_effort_combination(RC3, EC3) &
state(γ, t', O) |= satisfies_required_effort_for(EC3, X)]]

This property relates to the others as follows:

AP2(X, G, d) ⇒ AP1(X, G, d)

AP2(X, G, d) ⇒ GP2(X, G)

The next Section presents role properties which enable the adaptation.

5.2 Adaptor Role Properties

By an Adaptor role M, decisions about taking up or switching between Worker roles are made. As input information is used about the effort E currently being delivered by the different Worker groups G for a certain aspect X as expressed in

provides_group_effort_for(G, E, X).

In the model the decision mechanism is indicated by a relation expressing that an aspect has urgency:

has_urgency(X_1, E_1, …, X_n, E_n, X)

indicating that aspect X needs to be addressed in the context of efforts E_i, for aspects X_i. This relation can be specified as only deriving one aspect to be addressed (i.e., the most important aspect) or multiple aspects (e.g., all aspects currently not being addressed properly). The relation takes into account which effort E suffices for the required effort to be delivered for aspect X:

satisfies_required_effort_for(E, X)

and which effort E suffices for the desired effort for aspect X:

satisfies_desired_effort(E, X).

A simple form of an urgency relation that is taken by default is:

has_urgency(X_1, E_1, … X_n, E_n, X_i) $\leftrightarrow$ not satisfies_desired_effort(E_i, X_i).

This expresses that all aspects for which the desired effort is not satisfied are urgent. Based on the input on urgency, the Adaptor role M generates in an intermediate state an indication of the aspect that needs to be addressed in the current situation.

RP1(M) Aspect Urgency

At any t, if at t Adaptor role M observes the group efforts for each of the aspects,

and has a urgency relation that indicates X an urgent aspect at that time,

then at some t' ≥t it will generate that X needs to be addressed.

$\forall$t, X_1, .., X_n, E_1, .., E_n, X, M

state(γ, t, G_1) |= provides_group_effort_for(G_1, E_1, X_1) & … &

state(γ, t, G_n) |= provides_group_effort_for(G_n, E_n, X_n) &

state(γ, t, M) |= has_urgency(X_1, E_1, …, X_n, E_n, X)

$\Rightarrow \exists$t'≥t state(γ, t', M) |= to_be_addressed(X)

Based on this, the appropriate role(s) R within the Worker Group(s) WG for the aspect(s) is/are determined, and that a candidate is to be found for the role:

RP2(M) Role Change Determination

At any t, if at t Adaptor role M generated that X is an urgent aspect,

and role R in WG is responsible for this aspect,

then at some t' ≥t it will generate that a candidate for role R in WG has to be found.

$\forall$t, X, R, WG, M [state(γ, t, M) |= to_be_addressed(X) &

state(γ, t, M) |= role_responsible_for(R, WG, X)

$\Rightarrow \exists$t'≥t state(γ, t', M) |= to_be_found_candidate(M, ChangeGroup, R, WG)]

Finding the right Adaptor to be allocated to the role is the next step in the process. Assumed is that there exists shared knowledge of the capabilities of the Adaptors of the organization. An Adaptor may only have a partial view on this, and simply choose a local optimum. The decision mechanism states that the Adaptor will perform the role itself in case it has the capabilities or otherwise appoints another Adaptor which does have the capabilities and is preferred.

RP3(M) Candidate Selection: Own Selection

∀t:TIME, M,R:ROLE, WG:GROUP, C1,C2:CAPABILTIES
[state(γ, t, M) |= to_be_found_candidate(M, ChangeGroup, R, WG) &
state(γ, t, M) |= required_capabilities(R, WG, C1) &
state(γ, t, M) |= has_capabilities(M, ChangeGroup, C2) &
state(γ, t, M) |= capabilities_match(C1, C2)
⇒ ∃t2>t [state(γ, t2, M) |= shared_allocation(M, ChangeGroup, R, WG)]]

RP4(M) Candidate Selection: Request

∀t:TIME, M1,M2,R:ROLE, WG:GROUP, C1,C2,C3:CAPABILTIES
[state(γ, t, M1) |= to_be_found_candidate(M1, ChangeGroup, R, WG) &
state(γ, t, M1) |= required_capabilities(R, WG, C1) &
state(γ, t, M1) |= has_capabilities(M1, ChangeGroup, C2) &
state(γ, t, M1) |= not capabilities_match(C1, C2) &
state(γ, t, M1) |= has_capabilities(M2, ChangeGroup, C3) &
state(γ, t, M1) |= preferred_candidate(M2, ChangeGroup) &
state(γ, t, M1) |= capabilities_match(C1, C3)]
⇒ ∃t2>t [state(γ, t2, M1) |= request_shared_allocation(M2 ,ChangeGroup, R, WG)]

Furthermore, once a shared allocation is requested, the shared allocation will be in place:

RP5(M) Candidate Selection: Response

∀t:TIME, M, R:ROLE, WG:GROUP
[state(γ, t, M1) |= request_shared_allocation(M1,ChangeGroup, R, WG)]
⇒ ∃t2>t, M2 :ADAPTOR [state(γ, t2, M2) |= shared_allocation(M2, ChangeGroup, R, WG)]]

Finally, the following relationship is assumed to hold, given that roles R1..Rn are devoted to Group G addressing aspect X:

RP1(M) & RP2(M) & RP3(M) & RP4(M) & RP5(M) &

RP1(R1) & .. & RP1(Rn) ⇒ AP2(X,G,d)

This logical relationship is an assumption imposed on the domain of application. It is assumed that by adding more roles to the group involved, the effort for an aspect X can be strengthened so that the required effort is kept satisfied, and the desired effort will become satisfied again within duration d. In many qualitative and quantitative domains this assumption is fulfilled, for example, in the two domains addressed in Sections 6 and 8 in this paper. In quantitative cases it gets the form of the assumption that by adding role efforts for X, the total sum of efforts can be increased until a

certain value is reached, which has some relationship to the Archimedean principle in the real numbers:

$\forall a,b>0 \exists n \in \mathbb{N} \quad n*a > b$

In qualitative cases the assumption can be related to an assumption on the availability of the right capabilities within the organization, as is shown in Section 6.

6 The Interlevel Relations and Their Verification

In this section an overview of the interlevel relations between the dynamic properties at different levels of aggregation is given, as occurred at various places in Sections 4 and 5. They can be summarized and classified as shown in Table 1.

Table 1 Overview of the logical interlevel relations

From aspect properties to organization properties	$\forall X$ OAP(X) $\Rightarrow$ OPl
From global group properties to aspect properties	GP1(X, G) $\Rightarrow$ OAP(X)
From aggregated group properties to global group properties	GP2(X, G) $\Rightarrow$ GP1(X, G)
From group adaptivity properties to global group properties	AP1(X, G, d) $\Rightarrow$ GP1(X, G)
From role adaptivity properties to group adaptivity properties	AP2(X, G, d) $\Rightarrow$ AP1(X, G, d)
From role adaptivity properties to aggregated group properties	AP2(X, G, d) $\Rightarrow$ GP2(X, G)
From role properties to role adaptivity properties	RP1(M) & … & RP5(M) & RP1(R1) & .. .& RP1(Rn) $\Rightarrow$ AP2(X,G,d)

A graphical representation of the property hierarchy is shown in the AND/OR tree in Figure 3.

All relationships for the generic model as expressed within the tree have been verified using the SMV model checker under the assumptions as stated before. This has been done for every implication A $\Rightarrow$ B by rewriting property A in the transition system format that can be used by SMV as a description of a system. Moreover, B has been rewritten in CTL format, as required for the properties to be checked in SMV. As an example, this is shown for the relation between the role properties at the lowest level and the adaptivity property AP2. Here the antecedent A is a conjunction of a number of role properties for the adaptor role and for the worker roles. In SMV RP1 for the Member role has been specified as follows:

```
next(urgency) := case
          aspect = desired: 0;
          1: 1;
              esac;
```

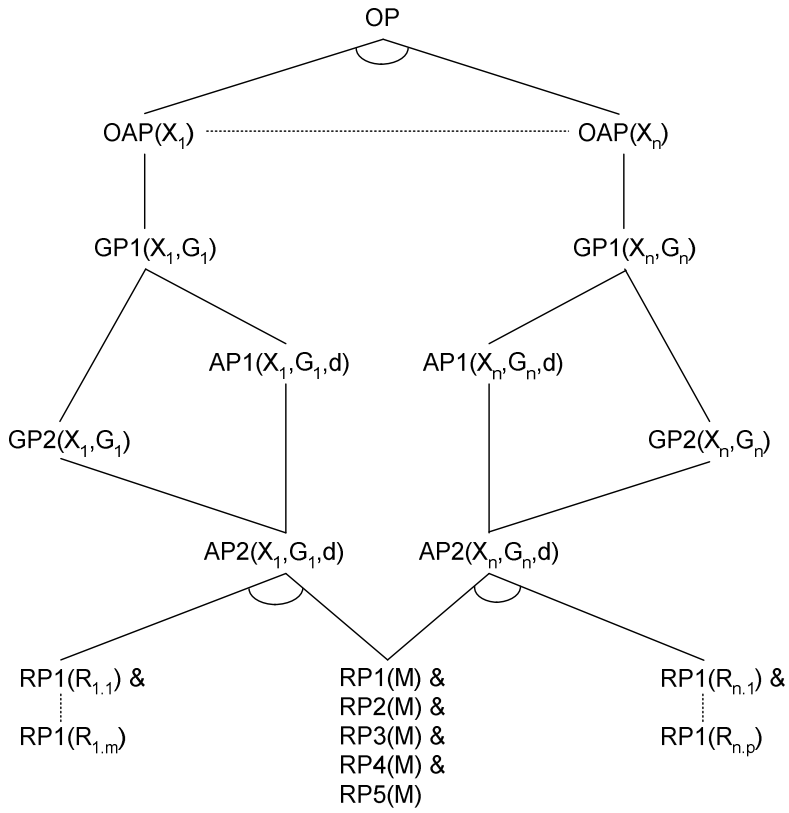

Fig. 3. Property hierarchy presented in graphical form as an AND/OR tree

This specifies the default urgency function, namely that in case the desired effort is not provided the aspect is considered to be urgent. This is expressed in the SMV transition system as follows: The next state for urgency is false (i.e., 0) in case the effort provided for the aspect is as desired, whereas urgency for the aspect is true in any other case. Furthermore, RP2 concerns the search of a candidate, which is true in the next state in case the aspect is considered urgent, and otherwise false.

```
next(search_candidate) := case
          urgency : 1;
          1:0;
                    esac;
```

The search of such a candidate is expressed in RP3 – RP5 and has been simplified for the sake of clarity. It is simply specified that a search for a candidate results in a candidate which has been found (which is also the case for the role properties specified).

```
next(found_candidate) := case
          search_candidate : 1;
          1:0;
                    esac;
```

Furthermore, the role property for the role which now has an agent allocated it is specified as follows:

```
next(effort_added) := case
          found_candidate : 1;
          1:0;
                    esac;
```

This specifies the effort added being by the newly found candidate. The added effort is set to true in case a candidate has been found, whereas it is false in case no candidate has been found. Final element of the transition system is to specify the effect of the added effort. Note that the addition of effort is not quantified, effort is simply added. This choice has been made for the sake of simplicity but has no consequences for the proof of the model. Adding effort has the following effect on the maintenance of the aspects within the organization:

```
next(aspect) := case
          aspect = failure & effort_added: required;
          aspect = required & effort_added: desired;
          aspect = desired & effort_added: desired;
          1 : aspect;
              esac;
```

This specifies that the next value for an aspect in case of effort being added is required in case the effort was previously failure; desired in case the previous aspect value was required, and desired remains desired upon additional effort. The consequent B is here property AP2. This is expressed in SMV's format based on CTL as follows.

```
AG ((!( aspect = desired) & (aspect = required))
     -> AF (aspect = desired) &
     A [(aspect = required|aspect=desired) U
  aspect=desired])
```

This expresses that when the considered aspect is required and not desired, then eventually the aspect will be desired again while in the meantime the value for the aspect will be either required or desired. This CTL property indeed holds for the specified model.

7 A Qualitative Application of the Organizational Model

This Section presents one of the two case studies undertaken to evaluate the applicability of the adaptive organization model presented in Sections 3, 4 and 5. This case study provides an analysis of the functioning of incident management organizations, in which adaptation of the organization by dynamic role reallocation is often observed. The analysis is based on a qualitative model for this domain made on the basis of extensive documentation of one of the disasters taking place in the Netherlands in the recent past [22]. First, domain specific variants of properties are introduced, after which simulation results are presented.

7.1 Domain Specific Properties

On the highest level of this qualitative incident management model, the property OP is defined as each aspect of the organization being satisfied. In the case of incident management, the aspects considered are fire fighting, health care, and traffic care:

OP(disaster)

For all time points t each aspect for incident management in the organization is satisfied.

$\forall$t:TIME [state(γ, t, O) |= satisfied(fire_fighting) $\wedge$ satisfied(health_care) $\wedge$ satisfied(traffic_care)]

On a lower level each individual aspect X is satisfied within the organization, for example, for the traffic care aspect of the organization:

OAP(traffic_care)

For all time points t aspect traffic care is satisfied in the organization.

$\forall$t:TIME [state(γ, t, O) |= satisfied(traffic_care)]

One group is responsible for the aspect traffic care: the police department. An instance of property GP1 requires a definition of satisfaction of the required effort. The effort of a group is defined as an abstract name; the required effort is always satisfied in case a route plan for ambulances is created which passes all wounded people within duration d from the start of the incident:

satisfies_required_effort_for(police_effort, traffic_care) $\leftrightarrow$

$\forall$t,t0:TIME [present_time(t) $\wedge$ memory(t0, incident_started) $\wedge$ t0+d < t] $\rightarrow$

$\exists$t2 $\exists$R:ROUTE_PLAN [t2 < t0+d $\wedge$

memory(t2, proposed_route_plan(R)) $\wedge$

$\forall$W :WOUNDED memory(t2, passes_wounded(R, W))]

Here it is assumed that there are memory states. The desired effort is defined by:

satisfies_desired_effort_for(police_effort, traffic_care) $\leftrightarrow$

$\forall$t,t0:TIME [present_time(t) $\wedge$ memory(t0, incident_started) $\rightarrow$ t0 + rd > t] $\vee$

$\exists t2\ \exists R{:}ROUTE\_PLAN\ [\ memory(t2, proposed\_route\_plan(R))\ \wedge$

$\forall W{:}WOUNDED\ memory(t2, passes\_wounded(R, W))\ ]$

Here $0 < r < 1$. In other words, the desired effort states that the correct route plan should be present before the required deadline already. The desired effort is always satisfied in the time interval from the start of the incident until rd after this start. It is not satisfied in the time interval starting rd after the start of the incident where no route plan was proposed yet. In view of property AP1 this means that after a correct route plan has not been generated by the police department within rd from the start of the incident, adaptation will be initiated at this time point, in order that the required effort will still be guaranteed before d after the start of the incident. As soon as indeed a route plan is proposed, the required effort remains satisfied and the desired effort becomes satisfied again.

Failure of the satisfaction of desired effort means that there is no role within the police department which has generated the correct route plan. By property AP2, this ultimately results in an adapted police department with roles which do perform the desired effort. To enable this change, the Adaptor within the Change Group uses the standard default definition of the urgency relation, in this case specifically for the police:

has_urgency(fire_fighting, fire_brigade_effort, health_care, health_effort, traffic_care, police_effort, traffic_care) $\leftrightarrow$ not satisfies_desired_effort_for(police_effort, traffic_care)

expressing that the traffic care aspect has urgency in case no route plan is generated within the desired duration.

7.2 Simulation Results

In order to show how a multi-agent organization functions using the organizational model as presented above, simulation runs have been performed based on observations at the Volendam bar fire as described in [22]. In order to be able to simulate these adaptation processes, the lowest level properties (i.e. role properties) as presented in the Sections above have been translated into the executable subset of

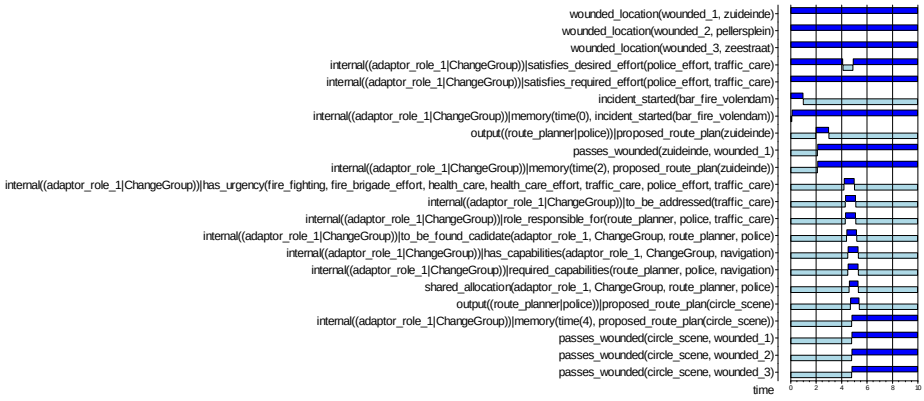

Fig. 4. Simulation results using the adaptive organization model

TTL called *leadsto* [5] which is used as an input for a simulation tool as described in [5]. Figure 4 shows the result of the simulation using this tool. In the Figure the left side shows the atoms that occur during the simulation run whereas the right side shows a timeline where a dark gray box indicates an atom being true whereas a light gray box indicates false.

As can be seen in the trace, at time point 0 the bar fire starts: incident_started(bar_fire_volendam). Three wounded people are present at the scene, at different locations, namely "zuideinde", "pellersplein", and "zeestraat":

wounded_location(wounded_1, zuideinde)
wounded_location(wounded_2, pellersplein)
wounded_location(wounded_3, zeestraat)

Note that in reality much more wounded are present. Based on these circumstances an Adaptor role observing the current state of affairs at the scene derives that both the desired and required effort concerning traffic care are being delivered by the police, since they have until time point 4 to come up with a correct plan:

internal(adaptor_role_1|ChangeGroup)|satisfies_desired_effort(police_effort, trafic_care)
internal(adaptor_role_1|ChangeGroup)|satisfies_required_effort(police_effort, trafic_care)

Underlying this derivation is the following *leadsto* property:

$\forall$I1,I2:INTEGER, RO:ROLE, IN:INCIDENT
[internal(RO|ChangeGroup)|memory(time(I), incident_started(IN)) $\wedge$
present_time(I2) $\wedge$ (I2 $\leq$ I+R*D)]

$\rightarrow_{0,0,0.1,0.1}$

internal(RO|ChangeGroup)|satisfies_desired_effort(police_effort, traffic_care)

The constants for R and D have been set to 0.5 and 6 respectively. Besides this property for desired effort a similar property exists for required effort. The only difference between these two properties is the multiplication with the factor R in the condition, which is not performed in the required effort property. In order for this rule to fire, the Adaptor roles have a memory state. For example, at time point 1 the Adaptor role stores the previously observed start of the incident:

internal(adaptor_role_1|ChangeGroup)|memory(time(0),
incident_started(bar_fire_volendam))

At time point 2 the route planner within the police group proposes a route plan which consists of merely one drive up route which is the location "zuideinde":

output(route_planner|police)|proposed_route_plan(zuideinde)

This plan however only passes the wounded person at the location "zuideinde" and not the other wounded:

passes_wounded(zuideinde, wounded_1)

Since the requirement is that the route plan should pass all wounded, the current proposed plan does not satisfy the requirements. However, due to the fact that the police has 4 time points before the desired effort needs to be provided, it takes until time point 4 before this failure is addressed (they could also have thought out a new,

correct, route plan before the fourth time point). At that time point, an Adaptor role is unable to derive that the desired effort for traffic care is satisfied. Such a desired effort can be derived by means of two rules, the first rule being the *leadsto* property presented above (i.e., there is still remaining time to come up with a good solution) whereas the second rule specifies that a correct traffic plan has been generated which passes all wounded:

∀I1,I2:INTEGER, RO:ROLE, IN:INCIDENT, RP:ROUTE_PLAN
[[internal(RO|ChangeGroup)|memory(time(I), incident_started(IN)) ∧
internal(RO|ChangeGroup)|memory(time(I2), proposed_route_plan(RP)) ∧
∀W:WOUNDED [passes_wounded(RP, W)]]
$\rightarrow_{0,0,0.1,0.1}$
[internal(RO|ChangeGroup)|satisfies_desired_effort(police_effort, traffic_care) ∧
internal(RO|ChangeGroup)|satisfies_required_effort(police_effort, traffic_care)]]

Since not all wounded are being passed, this rule cannot fire either. As a result, the Adaptor role derives that the police effort does not satisfy the desired effort regarding traffic care by means of a *closed world assumption* (i.e., everything which cannot be derived is assumed false). The required effort is however still satisfied because this can still be derived by means of the first variant presented in this Section. The fact that the desired effort is not longer satisfied causes an urgency for the traffic care task:

internal(adaptor_role_1|ChangeGroup)|has_urgency(fire_fighting,.., traffic_care)

As a result, the role immediately derives that traffic care needs to be addressed:

internal(adaptor_role_1|ChangeGroup)|to_be_addressed(traffic_care)

Since the route planner is the role responsible within the police department for this task, a candidate must be found to take over the role:

internal(adaptor_role_1|ChangeGroup)|to_be_found_candidate(adaptor_role_1,
ChangeGroup, route_planner, police)

The capabilities required for the role are navigation skills, a skill present at the particular Adaptor role, which therefore starts a shared allocation with the role itself (following the properties specified in Section 5.2) according the following *leadsto* property:

∀R1,R2:ROLE, G:GROUP, C:CAPABILITIES
[[internal(R1|ChangeGroup)|to_be_found_candidate(R1, ChangeGroup, R2, G) ∧
internal(R1|ChangeGroup)|required_capabilities(R2, G, C) ∧
internal(R1|ChangeGroup)|has_capacbilities(R1, ChangeGroup, C)
$\rightarrow_{0,0,0.1,0.1}$
internal(R1|ChangeGroup)|shared_allocation(R1, ChangeGroup, R2, G)]

As a result, the shared allocation occurs in the trace:

shared_allocation(adaptor_role_1, ChangeGroup, route_planner, police)

Resulting from this new shared allocation, the role outputs a new route plan which described a route that circles the scene and therefore passes all the wounded:

output(route_planner|police)|proposed_route_plan(circle_scene)

Thereafter, the desired effort is satisfied again. Note that during the entire adaptation process the required effort was always fulfilled since the requirement stated by the guidelines says that a route plan that passes all wounded should be present within 6 time points, which is the case within the simulation. Would there however not have been any adaptation, the required effort would not have been satisfied after time point 6.

8 Quantitative Specialization of the Adaptive Organization Model

For domains that can be quantified, the adaptive organization model can be specialized. As a starting point each aspect X can be quantified by some value V (real or integer number), indicated by

has_value(X, V).

For each aspect a lower bound V1 and upper bound V2 is specified, indicated by

lower_bound(X, V1) and upper_bound(X, V2)).

The aspect is satisfied whenever its value is between these values:

satisfied(X) $\leftrightarrow$
$\forall$V1,V2, V:VALUE [[has_value(X, V) $\wedge$ lower_bound(X, V1) $\wedge$ upper_bound(X, V2)] $\rightarrow$
V1 $\leq$ V $\wedge$ V $\leq$ V2]

Each of these aspects has a particular type of role attached to it, in which work is performed which contributes to that particular aspect. On the highest level, each aspect simply needs to be satisfied, expressed by the property OP in the following manner:

OPquantitative

For all time points t each aspect X has a value V which is below the upper bound V2 and above the lower bound V1.

$\forall$t:TIME state(γ, t) |= $\forall$X:ASPECT, V1,V2, V:VALUE
[has_value(X, V) $\wedge$ upper_bound (X, V2) $\wedge$ lower_bound (X, V1)] $\rightarrow$ V1 $\leq$ V $\wedge$ V $\leq$ V2]]

On the lower level of OAP(X), the same is expressed per aspect X. The effort required to maintain each of the aspects throughout the organization can change over time. A value to be maintained might for example express that a certain percentage of environmental pressure needs to be dealt with, which means more effort in case of more environmental pressure. The group properties which express the effort being

delivered by the groups addressing the aspects can again be reused from the model. However, the definitions for required effort and desired effort can be tailored towards the quantitative perspective. Here the assumption is made that V depends on E in a monotonic manner (when E is increasing, either V is increasing or decreasing). First of all, the required effort for each group is satisfied in case the current effort is between the minimum effort required (based either on the upper or lower bound of the aspect value) and the maximum effort (again from either the upper of lower bound of the aspect value).

satisfies_required_effort_for(E,X) $\leftrightarrow$
$\forall$V1,V2:VALUE, E1,E2:EFFORT
[[upper_bound(X, V1) $\wedge$ lower_bound(X, V2) $\wedge$
required_effort_for_value(E1, V1) $\wedge$ required_effort_for_value(E2, V2)$\wedge$
is_max_of(Emax, E1, E2) $\wedge$ is_min_of(Emin, E1, E2)]] $\rightarrow$ Emin $\leq$ E2 $\wedge$ E $\leq$ Emax]

Regarding the desired effort, the effort should be farther away from the bounds set. In other words, a parameter for a value ε with $0 < \varepsilon < 0.5$ is added, as follows:

satisfies_desired_effort_for(E,X) $\leftrightarrow$
$\forall$V1,V2:VALUE, E1,E2:EFFORT
[[upper_bound(X, V1) $\wedge$ lower_bound(X, V2) $\wedge$ required_effort_for_value(E1, V1) $\wedge$
required_effort_for_value(E2, V2) $\wedge$ is_max_of(Emax, E1, E2) $\wedge$ is_min_of(Emin, E1, E2)]
$\rightarrow$ Emin + ε (Emax – Emin) $\leq$ E $\wedge$ E $\leq$ Emax - ε (Emax – Emin)]

The decision properties for the Adaptor role again are reused from the generic properties as specified in Section 5, and also the default urgency relation:

has_urgency(X_1, E_1, …, X_n, E_n, X_i) $\leftrightarrow$ not satisfies_desired_effort_for(E,X)

In other words, an aspect is considered to be urgent in case the effort is outside the bounds of the desired effort.

9 A Quantitative Application of the Organizational Model

As a further evaluation of the applicability, the quantitative specialization of the organizational model has been applied in the domain of social insects and, more specifically, to analyze the functioning of honeybee colonies. In honeybee colonies several aspects need to be maintained in order for the population to be robust enough to be successful (in this case to survive), which include foraging, brood care, and undertaking. For the aspect brood care for example, the larvae need to have sufficient food, which requires more effort in case more larvae are present. If the larvae are insufficiently fed, the population size will eventually drop, endangering population survival. For each of these aspects, a specific Worker Group is present within the honeybee colony, i.e. all brood carer roles are part of the brood care group, etc. Furthermore, it is known from the biological domain that all honeybees within the organization have the capabilities to play each of the roles (see [6]).

Application of the quantitative specialization of the model is for the highest levels straightforward as these can simply be reused. On the lowest level however, the urgency of an aspect is defined by a threshold and trigger mechanism in each of the Adaptor roles, as described in biological literature; e.g., [6]. The mechanism informally works as follows. Each bee has a specific threshold for each of the aspects to be maintained in the organization. For the aspect the Adaptor is triggered most, relative to the threshold value it has, it will start a shared allocation with a role within a worker group devoted to that aspect. Following what is known in biological literature [6], for this case such a mechanism can be specified as follows:

has_urgency(X_1, E_1, …, X_n, E_n, X_i) ↔
$\forall V_1,\ldots, V_i,, \ldots, V_n$:VALUE, $T_1,\ldots, T_i,, \ldots, T_n$:VALUE,
[has_value(X_1, V_1) ∧ … ∧ has_value(X_n, V_n) ∧
has_threshold(X_1, T_1)∧.. ∧ has_ threshold (X_n, T_n) ∧
is_max_of(V_i/ T_i, V_1/ T_1, …, V_n/ T_n)]

As can be seen, the effort provided is not used for the decision process within the Adaptor roles, only the triggers (the value of the aspect) and the thresholds are used to determine which aspect is most important. Note that the thresholds are defined for each individual Adaptor within the ChangeGroup. For simulations, see [14].

10 Discussion

This paper presented a organizational model for the analysis and design of multi-agent organizations that are able to adapt to unpredictable events. The organization model was specified distinguishing a number of aggregation levels. At the highest level the goal for the organization as a whole is expressed and this is refined to lower aggregation levels until role properties are reached that have to be fulfilled by agents allocated to the role. The model has been formally specified and verified using the model checker SMV. Besides a generic template, also specific variants have been presented, addressing both quantitative and qualitative models. Applicability of the model was evaluated positively, using it to analyze two cases: social insects and incident management. For both cases simulations have been performed, based on translating the lowest level properties to an executable format.

Research as described in [2], [18], [19], [20] has some similarity to the approach presented in this paper: when only looking at the agents, they adapt their behavior based on an event. The difference is however that in this paper, the adaptation of the behavior of the agents over time is described using the roles they play. As a result, it abstracts from the specifics of the agent that describe this change behavior, but simply poses a requirement upon the adaptation behavior of the agent in the form of a role.

In the domain of organizational modeling for multi-agent systems several frameworks have been extended with capabilities to model organizational change as well. [13] for example introduces an approach where a Change Manager is present, deciding what to change within the organization, and following a model from a well known social scientist. Such a model is however concerned with centrally directed organizational change whereas this paper concentrates on adaptation brought about by

individuals within the organization detecting unsatisfactory occurrences in the organization. In an extension of MOISE, namely MOISE+ [15] a central director for change is present as well; decision rules as detailed as presented in this paper are not presented.

In order to incorporate new behavior which is not pre-specified, the approach presented in this paper can be enriched with adaptation of role properties or addition of roles. Such adaptations could for example include a new specification of role behavior. This is however future work and is not addressed in this paper.

References

[1] Ashby, W. R., Design for a Brain – The origin of adaptive behaviour, John Wiley and Sons, 1960.

[2] Barber, S.K., Goel, A., and Martin, C.E., Dynamic adaptive autonomy in multi-agent systems, *Journal of Experimental and Theoretical Artificial Intelligence*, 12:129-147, 2000.

[3] Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichmann, J., and Vбzquez-Salceda, J. (eds.), Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, Proc. of the First International Workshop From Organizations to Organization-Oriented Programming, OOOP'05. Lecture Notes in AI, vol. 3913. Springer Verlag, 2006

[4] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., Specification and Verification of Dynamics in Cognitive Agent Models. In: Proceedings of the Sixth International Conference on Intelligent Agent Technology, IAT'06. IEEE Computer Society Press, 2006, to appear.

[5] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., *LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn.* In: Proceedings of MATES'05. LNAI 3550. Springer Verlag, 2005, pp. 165-178.

[6] Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E., Self-Organization in Biological Systems, Princeton University Press, Princeton, USA, 2001.

[7] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organizations in multi-agent systems, *Proceedings of ICMAS'98*, IEEE Computer Society Press, pp. 128-135, 1998.

[8] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems, in Y. Demazeau, F. Garijo (Eds.), *Multi-Agent System Organizations. Proceedings of MAAMAW'01*, 2001

[9] Fisher, M. (2005). Temporal Development Methods for Agent-Based Systems, Journal of Autonomous Agents and Multi-Agent Systems, vol. 10, pp. 41-66.

[10] Galton, A. (2003). Temporal Logic. *Stanford Encyclopedia of Philosophy*, URL: http://plato.stanford.edu/entries/logic-temporal/#2.

[11] Galton, A. (2006). Operators vs Arguments: The Ins and Outs of Reification. *Synthese*, vol. 150, 2006, pp. 415-441.

[12] Giorgini, P., Müller, J., Odell, J. (eds.), *Agent-Oriented Software Engineering IV*, LNCS, vol. 2935, Springer-Verlag, Berlin, 2004.

[13] Hoogendoorn, M., Jonker, C.M., Schut, M.C., and Treur, J., Modeling Centralized Organisation of Organizational Change. *Jounal of Computational and Mathematical Organisation Theory*. In press, 2006.

[14] Hoogendoorn, M., Schut, M.C., and Treur, J., Modeling Decentralized Organization Change in Honeybee Colonies, Technical Report (see http://www.cs.vu.nl/~mhoogen/report/tr-asr-06-01.pdf), Vrije Universiteit Amsterdam, 2006.
[15] Hübner, J.F. and Sichman, J.S., Using the MOISE+ model for a cooperative framework of MAS reorganization, Boletim Técnico BT/PCS/0314, Escola Politécnica da USP, São Paulo, 2003.
[16] Jonker, C.M., and Treur, J., Relating Structure and Dynamics in an Organization Model, In J.S. Sichman, F. Bousquet, and P. Davidson (eds.), *Multi-Agent-Based Simulation II, Proc. of the 3rd Int. Workshop on Multi-Agent Based Simulation, MABS'02*. Lecture Notes in AI, vol. 2581, Springer Verlag, pp. 50-69, 2003.
[17] Kowalski, R., and Sergot, M. (1986). *A Logic-Based Calculus of Events*. New Generation Computing, 4:67--95, 1986.
[18] Krieger, M. J. B. and Billeter, J.-B. (2000) The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, vol. 30, pp. 65-84.
[19] Kube, C.R., and Zhang, H., (1994). Stagnation recovery behaviours for collective robotics. In *Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS'94)*, pp. 1883-1890.
[20] Maes, P., Modeling adaptive autonomous agents, *Artificial Life*, 1:1-37, 1994.
[21] McMillan, K., Symbolic Model Checking: An approach to the state explosion problem, Kluwer Academic Publishers, 1993.
[22] Ministry of the Interior, Investigation Bar Fire New Years Night 2001 (in Dutch), SDU, The Hague, 2001
[23] Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

Appendix A Example SMV Specifications

From role properties to adaptivity property AP1

```
MODULE main

VAR
  aspect:{desired, required, failure};
  urgency: boolean;
  search_candidate: boolean;
  found_candidate: boolean;
  effort_added: boolean;

ASSIGN
  init(aspect) := required;
  next(urgency) := case
          aspect = desired: 0;
          1: 1;
                  esac;
  next(search_candidate) := case
          urgency : 1;
          1:0;
                   esac;
  next(found_candidate) := case
          search_candidate : 1;
          1:0;
                   esac;
  next(effort_added) := case
          found_candidate : 1;
          1:0;
                   esac;
  next(aspect) := case
          aspect = failure & effort_added: required;
          aspect = required & effort_added: desired;
          aspect = desired & effort_added: desired;
          1 : aspect;
                  esac;

SPEC
   AG ((!( aspect = desired) & (aspect = required))
   -> AF (aspect = desired) &
   A [(aspect = required|aspect=desired) U aspect=desired])
&
   AG (aspect=desired | aspect = required)
```

From adaptivity property AP1 to group property GP1

```
MODULE main

VAR
  aspect:{desired, required, failure};
```

```
ASSIGN
  init(aspect) := required;
  next(aspect) := case
          aspect = desired: desired;
          aspect = required: desired;
          1: aspect;
                  esac;

SPEC
   AG (aspect=desired|aspect=required)
```

Chapter 11

Formation of Virtual Organizations through Negotiation

This chapter appeared as: Hoogendoorn, M, and Jonker, C.M., Formation of Virtual Organizations through Negotiation. In: Fisher, K., Timm, I.J., Andre, E., and Zhong, N. (eds.), *Multiagent System Technologies: Proceedings of the 4th German Conference on Multi-Agent Systems and Technology (MATES 2006)*, LNAI 4196, Springer Verlag, 2006, pp. 135-146. The original publication is available at www.springerlink.com.

Formation of Virtual Organizations through Negotiation

Mark Hoogendoorn[1] and Catholijn M. Jonker[2]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
mhoogen@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute of Cognition and Information
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. In this paper negotiation is presented as a solution to the formation of virtual organization in domains with many parties having (partially) unknown constraints and profiles and in which the environment is dynamic by nature. The solution presented is based on the MAGNET negotiation system, for which an extension is presented, that allows for last minute changes and failure management. An efficient algorithm is presented for supplier agents, incorporating preferences, and other constraints related to existing individual plans). Combining the algorithms for supplier agents, with a simple customer agent specification, and the ability to iterate the bidding, MAGNET is extended to deal with domains as described above. A case study in logistics using real data from a logistics company shows the validity of the approach.

1 Introduction

Virtual organizations have been defined as organizations where "complementary resources existing in a number of cooperating companies are left in place, but are integrated to support a particular product effort for as long as it is viable to do so" [7]. Nowadays, companies tend to outsource many non-core operations to upstream and downstream partner firms whose capabilities complement their own [9]. The relationship between such firms precisely complies to the definition of a virtual organization, making it an interesting type of organization to investigate given the current trends in organization theory.

Existence of a virtual organization can be long term or short term, where in the latter case the organization might only be formed to perform a few tasks. Especially for the cases where only a small number of tasks is involved in formation of a virtual organization, the overhead of the formation itself might be relatively large, possibly even causing more time than the task itself. One crucial aspect that for instance needs to be addressed in the formation process is what agents to allocate to what tasks. In order to cope with this problem, techniques from AI are being used to reduce the effort accompanying formation of a virtual organization.

This paper presents the application of one AI technique, namely automated negotiation between agents, to formation of virtual organizations. More in particular, the paper presents a system which enables automated allocation of agents to particular tasks that need to be performed within the virtual organization. The system tries to find a suitable allocation of tasks from two perspectives: (1) that of the agent looking for an agent to perform the task, and (2) that of the agent who can perform the task. Since both can have different, possibly partially conflicting interests, negotiation is most suitable to get to a solution for both parties. Besides the initial formation, the system also has facilities to cope with failure of agents to perform their allotted tasks and to redistribute tasks.

Section 2 presents MAGNET as the negotiation platform the supplier and consumer agents can use to find a solution to their needs. The techniques and extensions needed to be able to use the MAGNET for the dynamic formation of virtual organizations are presented in Section 3. Special attention is paid to obtain robustness with respect to failures in task performance and changes in the environment warranting the change of existing virtual organizations and the formation of new virtual organizations capable to cope with the situation at hand. The system was tested using real data from a logistic company. The test results are presented in Section 4. Section 5 discusses alternative approaches in literature and presents the conclusions.

2 The MAGNET System

This Section describes the negotiation system used as a basis for the development of the system supporting the formation of virtual organizations. The negotiation system used is the MAGNET (for Multi-AGent NEgotiation Testbed) system [4]. In [1] the MAGNET system is described as follows: the MAGNET architecture provides a framework for secure and reliable commerce among self-interested agents. What makes MAGNET particularly suitable is its ability to support negotiation of contracts for tasks that have temporal and precedence constraints [4]. MAGNET shifts much of the burden of market exploration, auction handling, and preliminary decision analysis from human decision-makers to a network of heterogeneous agents. Two types of agent are distinguished within such a network: The *supplier* agent and the *customer* agent. The main interactions between the two agent types are as follows:

- A customer agent issues a *Request for Quotes* (RFQ) which specifies tasks, their precedence relations, and a time line for the bidding process. For each task, a time window is specified giving the earliest time the task can start and the latest time the task can end.
- Supplier agents submit bids. A bid includes a set of tasks, a price, a portion of the price to be paid as a non-refundable deposit, and estimated duration and time window data that reflect supplier resource availability and constrain the customer's scheduling process.
- The customer agent decides which bids to accept. Each task needs to be mapped to exactly one bid (i.e. no free disposal [11]), and the constraints of all awarded bids must be satisfied in the final work schedule. In MAGNET the

customer can chose from a collection of winner-determination algorithms (A*, IDA*, simulated annealing, and integer programming).

The customer agent awards bids and specifies the work schedule.

3 Formation of a Virtual Organization

An overview of the activities accompanying the formation of a virtual organization supported by the system introduced in this paper is presented in this Section. Note that for evaluation and communication concerning the negotiation the MAGNET system can be used whereas more specific implementations for the *customer* and *supplier* agent are needed for specific domains such as the formation of virtual organizations.

3.1 High-Level System Overview

A high-level activity diagram of the system is shown in Figure 1. At the starting point the tasks to be fulfilled by the virtual organization come in, which are bundled in an RFQ and sent via the MAGNET system. The RFQ is sent to all *supplier* agents that might want to participate in the virtual organization. These *supplier* agents bid on the tasks they are able to perform and prefer and send a bid including these tasks back via the MAGNET system. After receiving all the bids, the MAGNET system evaluates these and selects the best set of bids possible. In case this set does not fully cover the tasks, an RFQ is sent again. For the bids that are in the set of optimal bids, an award is sent. The *supplier* agent that receives such a reward takes place in the virtual organization and starts executing the tasks, possibly reporting trouble requiring sending another RFQ for the task. Finally, after all tasks have been performed, the virtual organization is terminated.

3.2 Customer Agent

The *customer* part of the system mainly includes the formation of Request for Quotes (RFQs), the sending of awards for bids, and reassignment of tasks which are not properly performed. Tasks in the system include the following elements: intake time, early start time, late start time, deadline, and a task description, including details on the task and constraints. After an RFQ is sent, the *customer* eventually gets a set of bids to be awarded from the MAGNET system. In case there is no bid for a particular task, a new RFQ is sent concerning the particular task. After a task is assigned by means of awarding a bid, the *supplier* agent is placed in the virtual organization and starts to perform the task, which can result in an error report. In case such a report is received, a new RFQ with the task is sent to ensure that the task is eventually performed.

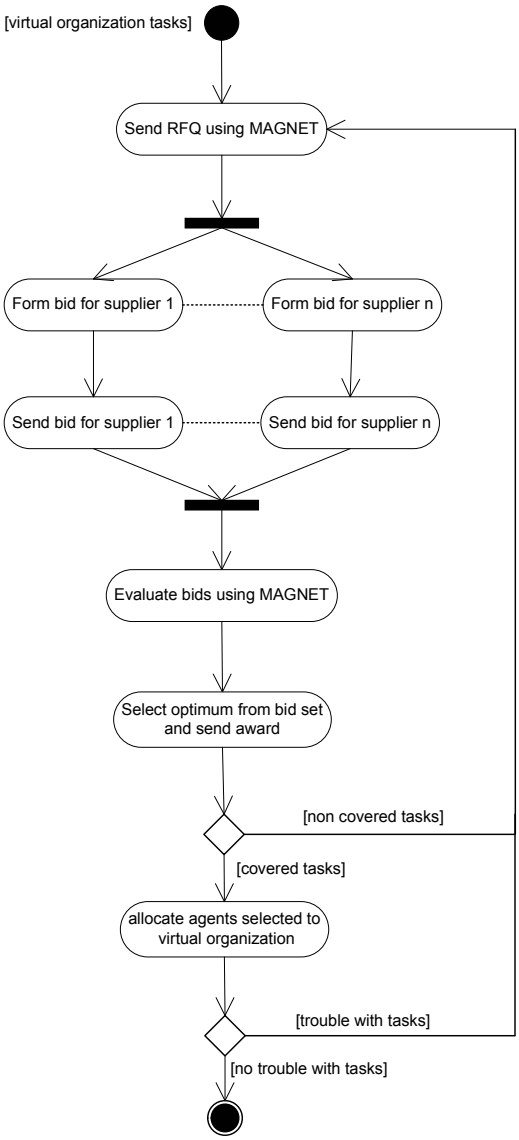

Fig. 1. UML Activity Diagram for the System

3.3 Supplier Agent

The *supplier* agents in the system are assumed to have one particular resource available during a certain time interval. Furthermore, a *supplier* is attributed with a certain preference for particular tasks, for example using the resource for a short time or using it in the beginning of the availability interval. In order to be able to derive

which tasks a *supplier* is to bid on, this Section presents an algorithm which derives which tasks are included in the bid, determines the cost, and finally, determines the time windows to be inserted. The notation used for the algorithm is shown in Table 1.

Table 1. Language used in the pseudo code

Function	**Explanation**
first_task: RFQ → TASK	The function application provides the task with the earliest early start time in the RFQ.
next_task: RFQ x TASK → TASK	Results in the task with the first earliest start time in the RFQ later than the earliest start time of the specified task.
number_of_tasks: RFQ → INTEGER	The number of tasks in the RFQ.
earliest_start: TASK → TIME	Denoting the earliest start time for executing the task.
latest_start: TASK → TIME	The latest possible start time for executing the task.
expected_duration: TASK → DURATION	The expected duration of executing the task.
latest_finish: TASK → TIME	Denoting the latest possible finish time of the execution of the task.
preference: TASK → REAL	The preference value for the task, a value between 0 and 1.
last_task_before: SCHEDULE x TIME → TASK	The last task in the schedule with a latest finish before the specified time.
next_task: SCHEDULE x TASK → TASK	Specifying the next task in the schedule.
switch_time: TASK x TASK → DURATION	The time needed to switch from one task to another.
determine_risk: REAL → REAL	The risk factor taken, based on the preference for the task.

The algorithm is specified in pseudo code below, a current schedule s from the supplier's perspective with tasks already scheduled is assumed to be present in advance.

```
t = first_task(RFQ)
do{
  before   = last_task_before(s, earliest_start(t))
  after    = next_task(s, before)
  chi      = determine_risk(preference(t))
  duration = chi * (expected_duration(t) + switch_time(before, t) + switch_time(t, after))
  if (earliest_start(t) + duration ≤ latest_start(after))
    if (preference(t) > phi ||
        number_of_tasks(RFQ) == 1)){
       // Add the task to the bid and schedule, set the cost using a particular cost function
    }else{
        // Do not include the task
    }
}while(t = next_task(RFQ,t) && t != NULL)
```

As can be seen, the first task to be performed is taken out of the RFQ. Given the current schedule, the task just ending before the early start time of the current RFQ task is obtained as well as the task after that. Furthermore, based on the preference (a value between 0 and 1) for the current RFQ task, the amount of risk to be taken is determined (e.g. I like this task so much, I will be able to perform it faster than average) represented by χ. Now calculate the *expected* duration for performing the task, which includes switching from the previous task, performing the task itself, and switching to the next task in the schedule. The *assumed* duration to be used in the calculation is obtained by multiplying this with the χ factor. In case the duration added to the earliest start time for task t is before the latest start time of the next task, then the task can in principle fit within the schedule. There is however still the preference of the supplier, which is specified by means of ϕ. ϕ is the threshold for the preference value above which a task is preferred. In case a task is preferred and fits within the current schedule, add the task to the bid. Do the same in case the RFQ contains one single task. This reflects the understanding by the *supplier* that this is a task that really needs to be performed and for which it is hard to get somebody. The global result is that unpopular tasks also will be performed. Once a task is added to the bid, the cost for performing the task are added by means of a cost function. Two cost functions are used in this paper, where the first is simply the *assumed* duration for the task. The second cost function used is the *assumed* duration divided by the preference value, which means a higher price for less preferred tasks. One element not addressed in the algorithm is determination of time windows to be included in the bid, which is specified in pseudo code below.

```
if (chi ≤ 1){
    earliest_start = latest_finish(before) + chi * switch_time(before, t)
    if (earliest_start < earliest_start(t)){
        earliest_start = earliest_start(t)
    }
    latest_start = latest_finish(before) + switch_time(before, t)
    if (latest_start < earliest_start(t)){
       latest_start = earliest_start(t)
    }
}else{
    earliest_start = latest_finish(before) + switch_time(before, t)
    if (earliest_start < earliest_start(t)){
        earliest_start = earliest_start(t)
    }
    latest_start = latest_finish(before) + chi * switch_time(before, t)
    if (latest_start < earliest_start(t)){
       latest_start = earliest_start(t)
    }
}
duration = expected_duration(t) * chi
```

In case the χ value is less than or equal to 1 (i.e. the task is *assumed* to go faster than *expected*), then set the earliest start time to either the earliest start time specified for the task or, in case this is not feasible, to the latest finish time of the task before in the

schedule plus the *assumed* switching time. The latest start is set to the latest finish time of the task before plus the *expected* switch time or, in case before the specified earliest start time, the earliest start time specified in the RFQ. If the value of χ is greater than 1, the earliest and latest start times are calculated just opposite from less than or equal to 1. Finally, the duration is set to the assumed duration.

After having sent a bid, a bid award is possibly received, resulting in the task actually being executed. The schedule is therefore replaced by a schedule including the tasks that have been awarded.

In the execution phase, incidents can occur that require replanning by the *customer* agent (or in similar domains, leading to the supplier agent becoming a customer agent that is seeking another supplier agent to solve his task). Three types of incidents are distinguished: (1) A simple task delay, that requires no replanning; (2) A task failure, the task needs to be performed by another *supplier*; (3) A day failure, all tasks for the day need to be re-planned.

4 Case Study

This section presents the results of a case study performed in order to validate the virtual organization formation approach presented in this paper. First, the domain in which the case study has been performed is described, thereafter the results regarding system performance are presented.

4.1 Case Study Description

In order to obtain experimental results, a choice has been made to use real company data instead of randomly generated data. Using company data has as the advantage that it can be determined how well such a system would work in a real environment instead of an artificially created one. The data has been obtained from a company within the field of logistics. This area is particularly interesting for application of the system due to the movement of several companies to so called Fourth Party Logistics (4PL), see e.g., [2]. A 4PL logistics company is an intermediate link within the chain of transporting goods, it closes contracts with large parties to arrange the logistics across the entire supply chain of the organization. 4PL companies have a limit amount, or possible even no trucks of their own (see e.g. [6]). They therefore have contracts with a number of trucking companies which they can call in case they need a truck for a particular order. The price for such a trip is negotiated over the phone. In the case study, the 4PL does not negotiate with the trucking companies through a scheduling officer, but directly with the truck drivers of that company. In this way the truck drivers get a higher responsibility for creating a revenue for the company they work for and they get the opportunity to guard their own preferences. Hence, the 4PL company is the *customer* in the system described in the previous section, and the trucks are the *supplier* agents, where a formation of a virtual organization for the transportation of certain goods is the goal of the negotiations.

The data used for the experiments concerns transportation of containers, of which only one can be carried at the same time by a truck. As a result, trucks can only perform tasks in sequential order and not in parallel. Furthermore, there are different types of containers: 20 feet and 40 feet containers, both of them can only be carried by a truck suitable for that particular type. Each of the tasks contain an intake time (around which the order to transport the container comes in, and thus the time at which an RFQ can already be sent), an early start time (when the container becomes available), a deadline (when the container needs to be delivered), and a start and end location. Precedence constraints are present as well in case a container has to be transported along several locations. The data obtained from the company mainly concerns container transports from one of the container terminals at the port of Rotterdam (there are several such terminals in the port) to a particular customer, after which the container needs to be returned to a certain location. Typically, about 20 orders are received each day, most of which require a pickup early in the morning. For the usage of the system presented in Sections 2 and 3, each truck is seen as a separate *supplier* where the resource is in this case the ability to transport a particular type of container. On average, about 10 trucks are available as *suppliers* per day. Trucks have a start location at which they are located at the beginning of the day (typically close to the port of Rotterdam), and have a start and end time (e.g. the trucks starts at 9 am and stops at 5 pm). Preferences of trucks are found in the different pickup and destination locations, the length of the trip required to perform the task, and the start and end times of the tasks. As a result of interviews with personnel from the data providing company, these preferences have been determined for each truck, based on the driver assigned to it. The real cost for performing a task is set to the travel time in minutes to perform the task (i.e. driving to the pickup location, performing the task, and returning from the destination location). Note that this can differ from the price actually put in the bid for the task.

4.2 Case study Results

In order to evaluate the effectiveness of the system, simulation runs have been performed using the real life data from the trucking domain as described before. For this purpose, the logs of the order system of one representative week has been used. Using this data, the system is evaluated from two perspectives. First, the time needed to evaluate the bids is measured, to see whether this evaluation process itself is not a bottleneck within the virtual organization formation process. The algorithm for the supplier agents can be run in parallel, which is not the case for the customer agent. Another perspective from which the system is evaluated is to see how different cost functions and preference thresholds influence the overall satisfaction of the *supplier* agents within the system.

Algorithm Performance. First, the performance of the evaluation algorithm during the simulation runs is presented. Note that these results are specific results for the characteristics of the data. For more generic results on algorithm performance and a comparison between different algorithms, see [3]. The experiments have been run on a Sun UltraSPARC IIIi 1062 MHz CPU with 2 GB memory. Figure 2 shows the results of the IDA* algorithm used for the case study for RFQs with varying amount of tasks.

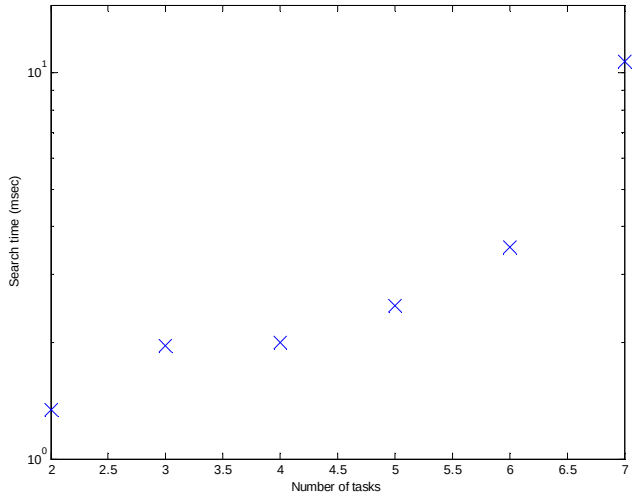

Fig. 2. IDA* search time for different number of tasks

Furthermore, Table 2 shows more detailed characteristics for the evaluation process.

Table 2. Evaluation characteristics

Number of tasks	Average number of bids	Average number of tasks in bid	Search time IDA* (msec)
2	3.59	1.00	1.35
3	6.05	1.07	1.98
4	6.00	1.00	2.00
5	7.25	1.08	2.50
6	8.00	1.25	3.54
7	7.63	1.01	10.69

As can be seen in the table, the average amount of tasks per bid is always close to one, which is due to the fact that an RFQ in the trucking domain typically specifies several tasks which need to be performed in parallel and, as already stated in the introduction of the case study, the trucks cannot execute tasks in parallel. Since only full bids can be awarded, they therefore often only bid for one task. As the graph shows, also for the RFQ's with the largest amount of tasks observed in the data (i.e. seven tasks in one RFQ) the evaluation algorithm generates a solution in just over 10

milliseconds. For a more extensive discussion on the scalability of the IDA* algorithm within the MAGNET system, see [3].

Supplier Satisfaction. Besides the evaluation time, the satisfaction of *suppliers* is another element which has been investigated. The satisfaction of the suppliers is measured in the average preference for the tasks they get awarded. Two parameters can be varied regarding this satisfaction, namely the threshold value ϕ and the function for cost to be included in the bid (i.e. assumed duration or assumed duration divided by the preference). Figure 2 shows the satisfaction of the different agents for both cost functions for varying ϕ values. Note that despite the threshold for bidding on tasks, tasks can still be bid upon in case only one task is included in the RFQ. As a result, the satisfaction can be below the threshold value set.

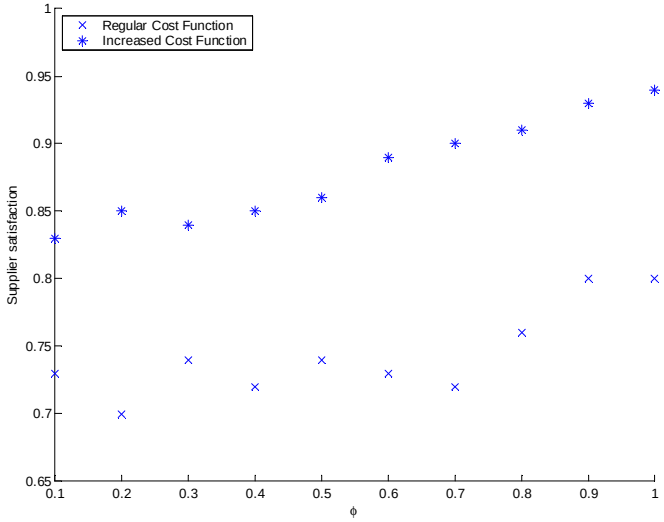

Fig. 3. Driver satisfaction for varying ϕ values

As can be seen in Figure 3, the increase of the price in case a task is not preferred is shown to be effective in the simulation runs of the case study. Having such a preference requires a less strict setting of the preference threshold ϕ for bidding on a task still obtaining a reasonable satisfaction rate. When looking at the regular price option, satisfaction is much lower once the value for ϕ decreases. An additional performance measure is of course the efficiency of the solution found, which in the trucking domain can be measured by means of the amount of effective driving (the amount of driving for a task divided by the total amount of driving). In the simulation runs, no correlation was found between the setting for the preference and the effectiveness of driving. On average, 62% of all driving was effective.

5 Discussion

This paper presents an approach for the formation of virtual organizations in highly dynamic environments which require a low overhead for the formation process of the organization. The approach allows for the formation of such an organization without the different parties needing to have knowledge about each others constraints and profiles. The approach is based upon an existing negotiation system called the MAGNET system which is extended with specific implementations for the *supplier* and *customer* agents for the formation of virtual organizations. The implementation of the *supplier* agent incorporates preferences for tasks as well as schedules specifying the tasks to be performed. In a case study in the trucking domain, the paper shows that the evaluation algorithms incorporated in the MAGNET system scale well, requiring a minimal time for the evaluation process. Furthermore, reflecting the preference of a task in the price bid for that task in the algorithm increases the overall satisfaction of the *supplier* agents.

In the field of virtual organizations, negotiation systems have been introduced and used as well. In [8] a virtual office system is mentioned called *SmartProcurement* which is said to initiate the formation of a virtual organization by means of an electronic or human request for quotation (RFQ). Thereafter, a purchasing agent acquires a list of agents which are known vendors of the requested item and sends the RFQ to the vendors. Subsequently, the bids are evaluated and a bid is selected, informing the vendor agent upon acceptance. The approach is however more meant as a framework to support such negotiation, similar to the MAGNET system, not as a specific implementation of the agents themselves.

Besides the MAGNET system, more negotiation systems have been developed. The advantage of the MAGNET system is the market infrastructure in between the *supplier* and the *customer* agent whereas most other negotiation systems focus on direct agent to agent negotiation [12, 5] (from [1]). Based on the MAGNET system more extensive *supplier* agents have been developed [1], however these agents have not been tested with real life data. Furthermore, [1] does not focus on the formation of virtual organizations.

Team or coalition formation is another related field. Different protocols for the formation of coalitions are compared in [13]. Variations of such protocols go from local to social utility based negotiation systems. The authors show that increased social context can improve system performance. The agents are however required to share meta-level information before they allocate resources. In the trucking domain, however, agents do not want to share such meta-level information, as they might be competitors. Therefore the approach presented in [13] is not feasible in domains in which the agents represent competitors.

Different role-allocation and reallocation algorithms are compared in [10] The comparison is based on for the framework developed for the Role-based Markov Team Decision Problem. In the future the same framework could be applied to compare the approach presented in this paper with other role-allocation algorithms with respect to the corporate data for the trucking domain.

Acknowledgements

The authors would like to thank Maria Gini from the Department of Computer Science and Engineering at the University of Minnesota for the fruitful discussions. Furthermore, the authors wish to thank the anonymous reviewers for their useful comments that helped to further improve the paper.

References

[1] Botman, S., Hoogendoorn, M., Bud, V., Jaiswal, A., Hawkins, S., Kryzhnyaya, Y., Pearce, J., Schoolcraft, A., Sigvartsen, E., Collins, J., and Gini, M., Design of supplier agents for an auction-based market. In: Giorgini, P., Giorgini, P. Lesperance, Y., Wagner, G., Yu, E. (eds.), Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002 @ AAMAS-02), July 2002.

[2] Briggs, P., The hand-off: the future of outsourced logistics may be found in the latest buzzword [Fourth Party Logistics], Canadian Transportation Logistics 102(5), pp. 18, 1999.

[3] Collins, J., Solving Combinatorial Auctions with Temporal Constraints in Economic Agents. PhD thesis, University of Minnesota, June 2002.

[4] Collins, J., Gini, M., and Mobasher, B., Multi-agent negotiation using combinatorial auctions with precedence constraints. Technical Report 02-009, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, Minnesota, February 2002.

[5] Fatima, S.S. and Wooldridge, M.. Adaptive task resources allocation in multi-agent systems. In Proc. of the Fifth Int'l Conf. on Autonomous Agents, pp. 537-544, 2001.

[6] Foster, T., 4PLs: The next generation of supply chain outsourcing? Logistics Management and Distribution Report 38(4), pp. 35, 1999.

[7] Goldman, S., Nagel, R., Preiss, K., Agile Competitors and Virtual Organizations, Van Nostrand Reinhold, New York, 1995.

[8] O'Leary, D.E., Kuokka, D., and Plant, R., Artificial Intelligence and Virtual Organizations, Communications of the ACM 40(1), pp. 52-59, 1997.

[9] Miles, E.R., Snow, C.C., Mathews, J.A., Miles, G., and Coleman, H.J., Organizing in the knowledge age: Anticipating the cellular form, Academy of Management Executive 11(4), pp. 7-20, 1997.

[10] Nair, R., Tambe, M., Marsella, S., Role Allocation and Reallocation in Multiagent Teams : Towards a Practical Analysis, In: Proceedings of the Second Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2003), pp. 552-559, ACM Press, 2003.

[11] Nisan, N., Bidding and allocation in combinatorial auctions. In 1999 NWU Microeconomics Workshop, 1999.

[12] Sandholm, T.W., Negotiation Among Self-Interested Computationally Limited Agents PhD thesis, Department of Computer Science, University of Massachusetts at Amherst, 1996.

[13] Sims, M., Goldman, C.V., and Lesser, V., Self-Organization through Bottom-up Coalition Formation, In: Proceedings of the Second Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2003), pp. ACM Press, 2003.

Chapter 12

Decentralized Task Allocation using MAGNET: An Empirical Evaluation in the Logistics Domain

Decentralized Task Allocation using MAGNET: An Empirical Evaluation in the Logistics Domain

Mark Hoogendoorn[1], Maria L. Gini[2], and Catholijn M. Jonker[3†]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
mhoogen@cs.vu.nl
[2]University of Minnesota, Department of Computer Science and Engineering
200 Union Street SE, Minneapolis, United States
gini@cs.umn.edu
[3]Delft University of Technology, Department of Man-Machine Interaction
Mekelweg 4, 2628CD Delft, The Netherlands
catholijn@mmi.tudelft.nl

Abstract. This paper presents a decentralized task allocation method that can handle allocation of tasks with time and precedence constraints in a multi-agent setting where not all information needed for a centralized approach is shared.

In our MAGNET-based approach agents distribute tasks via first-price reverse combinatorial auctions, where the auctioneer is whatever agent has tasks to be allocated. The choice of MAGNET is based on its uniqueness to handle auctions for allocation of tasks which include time windows and precedence constraints.

Empirical evaluations based on real data obtained from a logistics company show that the system performs well. The costs of the allocations obtained by our approach are within 5% from the optimal allocation. The computation time is linear in the number of tasks, while computing the optimal allocation is an NP-hard problem.

1 Introduction

There are many real-world problems in which agents need to plan in advance and schedule multiple tasks, think of logistics, hospital schedules that have to be changed with new patients coming in, manufacturing on demand, and design of complex systems. We are interested in situations where an agent recruits other agents to carry out tasks for which precedence and time constraints are a common phenomenon, such as in logistics, hospitals, and manufacturing on demand.

The field of planning has contributed several centralized heuristic algorithms for optimal task allocation. For example, algorithms have been created for the Vehicle Routing Problem and its instances (see e.g. [9]), and the Dial-a-Ride problem [17]. The main disadvantage of such algorithms is their centralized nature, since a

† The ordering of the authors is based on the effort put into the article.

centralized allocation of tasks to multiple agents is not always possible. It may be computational unfeasible to find an optimal allocation or agents unwilling to share complete information about their resources and commitments may invalidate the algorithm.

Decentralized task allocation has been a topic of research for quite some time already, see e.g. [28], [29], and [24]. However, so far, the decentralized task allocation literature has not addressed the problem of task precedence relations and time constraints between the tasks. This paper presents a decentralized way of allocating tasks that does deal with precedence and time constraints. The method exploits the unique feature of the MAGNET [6] system that allows autonomous agents to negotiate over complex coordinated tasks, with precedence and time constraints, in an auction-based market environment [4].

In our method MAGNET agents participate in market-mediated first-price reverse combinatorial auctions, where the agent which allocates the tasks to other agents is the auctioneer. Any agent can be an auctioneer, so any agent can, at any moment in time, attempt to allocate its tasks to other agents via auctions.

The method has been thoroughly evaluated by means of empirical analysis using data obtained from a logistical company. This choice of domain allowed us to test specifically the effectiveness of the method to deal with precedence relations and time constraints, while delivering solutions that are near to an optimum.

In logistics, the tasks that require allocation are time based from different perspectives. The transportation devices (ships, trucks, plains, trains) are not cost effective while they are being used as storage room. Furthermore, devices are often not allowed to stay at the same place for long. For example, in the harbor of Rotterdam, ships are assigned specific slots for off loading their goods. Due to the nature of ships and harbors, last minute rescheduling of slots and ships is impossible. Ships and harbor have to know the schedule much longer in advance, see e.g. [27]. The industry and/or companies that need the goods are ever operating from a produce on demand principle instead of keeping a large stock. This implies that the logistics process starts no sooner than when an order comes in, while the customer still expects a speedy delivery.

Furthermore, the goods themselves can put time constraints on the logistics. For example, perishable goods like flowers have to cross the world in hours to be still of value at the point of delivery.

Finally, the logistics process itself can cause precedence constraints; a good cannot be transported from a particular location before it has arrived at that particular location. Similar precedence constraints can be caused by production on demand process requiring different raw materials or half-fabricates.

As a result, transportation tasks typically have time windows specified, stating when particular goods can be picked up, and when they need to be delivered. Other aspects relevant for logistics are the locations (from, to), and some indication of what type of load is to be transported (e.g., the type of container), so that an appropriate transportation device can be selected and scheduled.

This paper is organized as follows. Section 2 presents the MAGNET system itself whereas the application of the MAGNET system in the field of logistics is presented in Section 3. Results of the empirical evaluation using a dataset obtained from a

logistics company are presented in Section 4. Section 5 discusses related work, and finally, Section 6 presents our conclusions and gives directions for future work.

2 The MAGNET System

The MAGNET architecture provides a framework for secure and reliable commerce among self-interested agents. MAGNET shifts the burden of market exploration, auction handling, and preliminary decision analysis from human decision-makers to a network of heterogeneous agents.

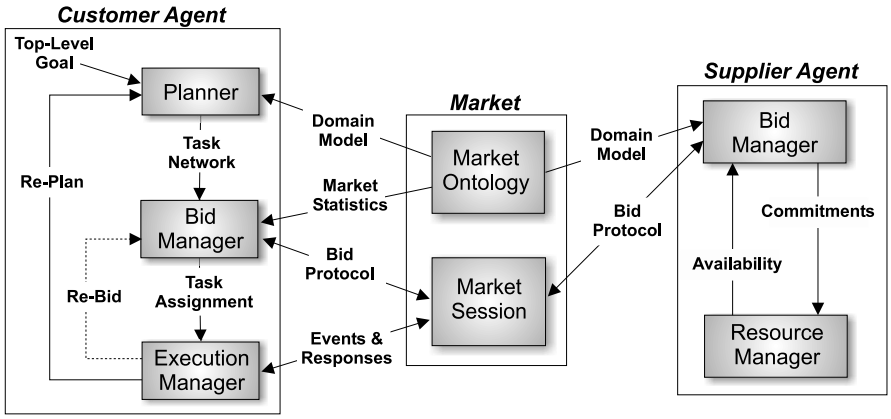

Fig. 1. MAGNET Architecture

The MAGNET system architecture, shown in Figure 1, consists of: (1) a *customer* agent, which allocates tasks to other agents. The tasks have time constraints and other restrictions; (2) *suppliers* agents, which bid on the tasks and execute them when awarded; and (3) the MAGNET market server, which keeps track of the activities of the agents and of the auctions.

The main interactions between agents in the MAGNET system are as follows:

- A customer agent issues a *Request for Quotes* (RFQ) which specifies the tasks, their precedence relations, and a time line for the bidding process. For each task, a time window is specified giving the earliest time the task can start and the latest time the task can end.
- Supplier agents submit bids. A bid includes one or more tasks, a price, the portion of the price to be paid as a non-refundable deposit, and the estimated duration and time window for task execution. Supplier data reflect supplier resource availability and constrain the customer's scheduling process.
- The customer agent decides which bids to accept. Each task needs to be mapped to one bid and the constraints of all awarded bids must be satisfied in the final work schedule. In MAGNET the customer can chose from a collection of winner-

determination algorithms (A*, IDA*[‡] [5], simulated annealing, and integer programming [4]).

- The customer agent awards bids and specifies the work schedule.

3 MAGNET and Logistics

A domain in which task allocation is part of the core operations is the field of logistics [18]. Orders that arrive demand a set of specific transportation tasks to take place. These transportation tasks need to be assigned to a particular resource (e.g. a truck or a ship).

The logistic domain has been a topic of research in classical planning for quite some time (see e.g. [21]), mainly focusing on calculating optimal solutions or approximating them from a centralized perspective. For instance, in [11] the problem addressed is to find optimal routes for transportation orders of a large set of users. Orders have to be picked up and delivered at specific locations, within a given time window, and using a limited number of trucks. The solution proposed is centralized, and it is used to support a human dispatcher.

Distributed planning has been popular in distributed AI applications(see, for instance, [12]), where agents are assumed to be cooperative, but coordinating the plans of individual agents is still a challenging task [8]. When the agents are not cooperative, auction based approaches to allocation of tasks are more commonly used (for instance, [13;10]).

A trend has now emerged in the field of logistics which requires a more distributed setting: Fourth party logistics (4PL) [1]. In fourth party logistics companies arise that sign contracts with large companies to arrange their entire transportation demand. These companies however do not have sufficient resources on their own to arrange all these transports and therefore distribute many of those tasks to other(partner) companies. A rapid assignment of tasks to particular resources is essential for these 4PL companies. Orders typically arrive at the company by phone, and being able to immediately inform the customer on when the task will be performed gives a competitive advantage.

Given this setting, centralized calculation of the optimal solution might no longer be feasible due to the lack of complete information (availability of resources which is too sensitive for a company to communicate) as well as the complexity of calculating this optimal solution within a short period (time is crucial in the business). The latter especially holds due to the fact that constraints such as time windows and precedence constraints are also specified for these tasks, making calculation of the optimum even harder.

The MAGNET system can help overcome these problems since it allows for task allocation in a distributed way where companies can maintain their own schedule. Furthermore, the strength of the MAGNET system is that it is also able to handle time

[‡] Iterative Deepening A* (IDA*) [19] is a variant of A* which uses the same heuristic function in a depth-first search, and which keeps in memory only the current path from the root to a particular node. In each iteration of IDA*, search depth is limited by a threshold on the value of the heuristic function.

windows as well as precedence constraints which is essential in this domain. Other task allocation methods based on auctions assign only the tasks needed for the immediate time period [10]. Because of this, they do not produce optimal allocations. MAGNET avoids this problem by soliciting bids for tasks spanning over time, and accepting the optimal combination of bids that fits the overall schedule.

Given the scenario of 4PL companies presented above, task allocation can be performed as follows: The 4PL company (i.e. the *customer*) issues an RFQ, sends it to a partner company (i.e. the *supplier*) who can bid on one or more tasks included in the RFQ. The price they bid equals the amount of driving required to perform the task (the prices per kilometer of driving for each partner firm is fixed).

Based upon this viewpoint, implementations of both *supplier* and *customer* agents have been created.

3.1 Supplier Agent

The *supplier* agent maintains a schedule for its resources and generates bids based upon that schedule. The schedule specifies when resources are available as well as a start location when the resource becomes available and an end location when the availability slot ends. During that availability time, the schedule consists of entries that specify when tasks are scheduled to be performed (i.e. start and end time), and furthermore what the start and end location of that particular task is.

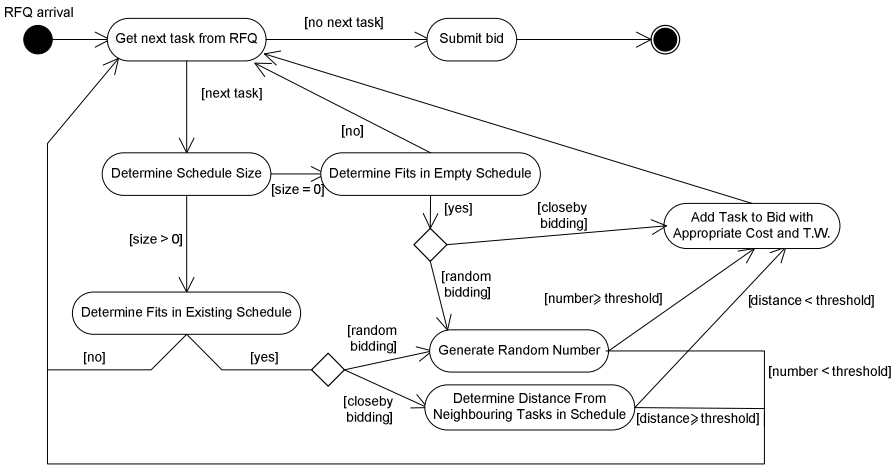

Fig. 2. UML Activity Diagram of Supplier Algorithm

When an RFQ arrives, the *supplier* agent determines what tasks it is able to perform with the resources it controls. Furthermore, it calculates whether it can perform the task in the time window specified given the current schedule of the resources (including the tasks that have been bid upon or included in the bid but have not been awarded nor rejected). Given that a task indeed satisfies these constrains, two possible algorithms have been implemented:

- Random bidding, which includes a task in the bid with a certain probability. Time windows that are feasible according to the current schedule are included in the bid.
- Closeby bidding, which includes only those tasks that are close (given a certain distance measure) to the start or end location of the tasks that are already part of the schedule. In case of an empty schedule, the task is included. Again, time windows that are feasible according to the current schedule are included in the bid.

As a cost, we use the sum of the distance of getting to the start location of the task, performing the transport, and returning from the end location is included. Figure 2 shows the algorithm in the form of a UML activity diagram.

Note that preferences of *suppliers* can also be taken into consideration, which is however not the focus of this paper. In [15] for example, bidding strategies are specified that do take such preference into account.

3.2 Customer Agent

The *customer* agent simply creates RFQ's for tasks matching the orders that have been received, and evaluates the bids that have been received based upon the evaluation algorithms part of the MAGNET system. Since it could happen that certain tasks are not bid upon, dummy bids for each task are added to the bid set for this evaluation process with an extremely high price. In case such a dummy bid gets awarded the task needs to be sent again, possibly attracting some *suppliers* that did not get their bid awarded. Each RFQ which is sent the same day is later referred to as a *cycle*.

4 Empirical Evaluation

To see how the setup within the field of logistics described in the previous Section would work in a real life setting, data has been obtained from a 4PL company. The characteristics of the data are described first. Thereafter, results of using the data as an input for the system are presented as well as comparisons between the solutions found and the optimal solution. In this case, the optimal solution could be calculated as all information is centrally available in one dataset, which is not necessary for the MAGNET algorithm. This does enable us to compare the distributed with the centralized approach, giving us insight in the quality of the distributed solution. Furthermore, the time required for the computations is compared as well.

4.1 Dataset Description

The dataset has been obtained from a company within the field of logistics. It concerns a mid-size company that focuses on transport of various types of goods, including perishable goods, and containers. Transportation of containers has been a

growing global market over the last decades [25], and 4PL companies need to transport many containers as part of the contracts they have with their customers.

The dataset we have obtained from the company concerns these container transports. Each morning the company receives a set of tasks concerning transportation of a containers at that specific day from a specific pickup location (for instance a container terminal), to a specific destination location where the container is either unloaded or loaded, thereafter transporting the container to a third location where it is left behind. Besides these locations, time points are also specified, indicating after which time point the container becomes available at the pickup location and when the container needs to be returned to the third location. The amount of orders received upon a day is on average just above 20 of such transportation tasks. The size of the dataset concerns 100 such days, totaling to approximately 2000 tasks that need allocation.

Note that the complexity of this scheduling for this company is clearly not in the amount of tasks to be scheduled. The main problem here is speed and incompleteness of information. As described above 4PL companies do not have the trucks themselves, they have to negotiate with the companies that do have trucks. In fact the situation is that different 4PLs have to compete with each other for work. They can only be effective if their interaction with truck owning companies is time effective. Similarly, the truck owning companies have to compete with each other for work, and, again, time is of the essence.

Besides tasks that need to be performed, the company of course also has particular resources to which these tasks can be allocated. The dataset also includes the trucks that can be used as a resource on a particular day. For each of these trucks, an availability slot is given, including a start time when the resource is available, and an end time after which the truck is no longer available. The capacity of such a truck is that it can carry one container at the same time. Each truck starts at the headquarters of the company at the beginning of the availability slot, and needs to end at the headquarters at the end of the slot. In the dataset on average half the amount of trucks are available compared to the number of tasks that need to be performed. This is more than sufficient to perform all transports while still meeting the requirements that have been set for these orders.

Given this dataset, both types of companies are represented, namely the truck owning companies (the trucks in the dataset) and the 4PL company (the orders in the dataset). Each truck is represented by exactly one *supplier* agent within the MAGNET system. Furthermore, a distance measure is included in the *suppliers* that is able to calculate the distance between different locations. For each *supplier*, this distance function is the same. Finally, each *supplier* uses the same definition of locations considered to be close to each other (in case of the Closeby algorithm), which is based upon a definition given by planners within the company.

4.2 Results

The results reported in this Section concern usage of the full dataset(i.e. 100 days of operations with on average 20 orders, meaning approximately 2000 orders). Since we are interested in how well our algorithms scale up, we want to vary the amount of

tasks that require allocation upon a day. This means that we perform runs over the full 100 days and for each of such runs the number of tasks that require allocation upon one day is kept the same (e.g. 5 tasks per day). Therefore, each day selections are made of the total number of orders that are available from the dataset, where the size of the selection equals the number of tasks we want to investigate. We've performed runs using 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, and 20 tasks. Also, selections of resources have been made to make the runs as realistic as possible. As has already been mentioned, on average half the amount of resources are available compared to the number of tasks that require allocation. This means that the number of resources available upon a day during such a run is set to 1, 1, 2, 2, 3, 3, 4, 4, 5, 6, 7, and 10 respectively. The experiments have been conducted on a Sun UltraSPARC-IIIi 1062 MHz CPU with 8 GB of memory. Calculation of the optimal result is performed by means of a brute force algorithm which does not scale up well with the number of tasks that require allocation upon a day. Theoretical results show that the type of problem, called the capacitated dial-a-ride problem is NP-hard to solve [2]. As a result of this, such calculations could only be performed up till 10 tasks per day. Regarding the MAGNET system, the IDA* algorithm has been used for evaluation of the bids that have been submitted by the trucks. IDA* is an optimal, memory-bounded, heuristic search algorithm. Its time complexity is hard to characterize, since it depends on how good the heuristic used is [19]. Its space complexity is linear in the depth of the solution. This makes IDA* a good choice when an optimal solution is needed in a large state-space where A* would run out of memory.

4.2.1 Comparison to optimal solution

Figure 3 shows the average deviation over running the algorithm on the full 100 days of the solution found by the distributed MAGNET algorithm using the Closeby bidding versus calculation of the optimal result from a centralized perspective. A result of 1 means that the average solution found is equal to the optimum (which is the lowest cost for performing the tasks) whereas 1.05 for instance means the average

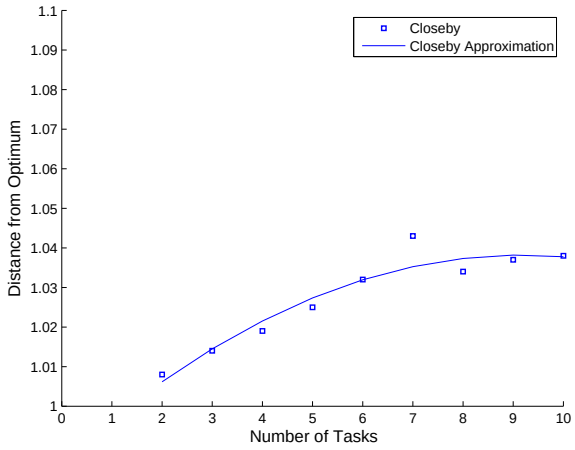

Fig. 3. Closeness of the MAGNET algorithm to the optimal result

result found is 5% above optimal. The results are presented for varying number of tasks that require allocation upon a day. As can be seen, the deviation of the solution found compared to the optimum initially increases with the number of tasks. However, the steepness of this increase in deviation from the optimal result decreases as the number of tasks that need allocation increase. This decrease is due the fact that more tasks increase the probability of the trucks finding a task which nicely fits within their schedule, avoiding large driving distances from one task to another. The Random bidding algorithm simulation runs have also been performed but for the sake of clarity these are not shown in the Figure. The results are significantly worse compared to the Closeby bidding algorithm. For 5 tasks for example, the deviation from the optimum is 1.17 and increasing.

Besides comparing the quality of the solution found, the difference in search time is also a crucial element within the field of logistics. As already mentioned before, being able to immediately inform customers over the phone gives a competitive advantage. In Figure 4 the average total evaluation time(i.e. the sum of the evaluation time for all *cycles* upon a day in the case of the MAGNET algorithm) over 100 days for varying number of tasks is shown. Again, only the Closeby bidding algorithm is shown as the Random algorithm scales in a similar fashion. As can be seen, the algorithm for optimal performance does not scale well, whereas the MAGNET algorithm scales very well, it can even be approximated by a linear function. Note again that IDA* has been used here. When considering a maximum waiting time of approximately 1 minute on the phone, no more than 8 orders can be placed in case of the centralized algorithm. For the decentralized MAGNET algorithm however, 20 orders can certainly be handled which is currently the maximum number of orders received by the company.

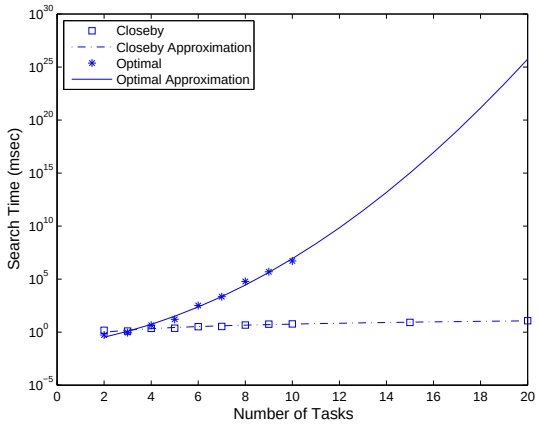

Fig. 4. Total evaluation time needed for optimum and MAGNET algorithm (i.e. sum of evaluation time of all cycles needed). Note the logarithmic scale

4.2.2 MAGNET Bidding Strategy Characteristics

Besides comparing the quality and search time of the solution found by the MAGNET based system with the centralized approach, the characteristics of the two different

bidding algorithms (i.e. Closeby and Random) have been compared as well. As already mentioned, the Closeby algorithm finds solutions of a much higher quality than the Random algorithm. Furthermore, the search times scale approximately the same for both bidding strategies. A third measure for comparison is the number of *cycles* needed (i.e. how many times an RFQ with tasks needs to be sent to have a fully covered task allocation for a day).

The number of *cycles* needed, averaged over the 100 days within a run, is shown in Figure 5 for a varying number of tasks that require allocation upon a day. As can be seen, the Closeby algorithm needs fewer cycles for the lower amount of tasks whereas the Random algorithm needs fewer for the higher amount of tasks. This can be explained by the amount of trucks being present: The more trucks, the higher the probability that a task will be bid upon. Furthermore, since the initial location of the trucks is the same, the Closeby bidding strategy initially results in trucks bidding upon the same tasks, causing more cycles.

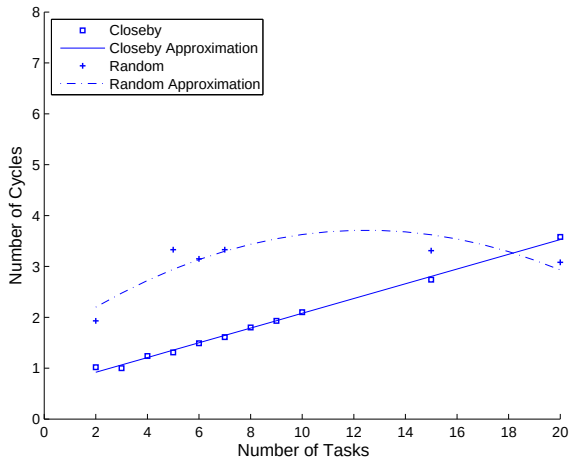

Fig. 5. Cycles needed by Closeby and Random bidding strategy

4.2.3` MAGNET Evaluation Algorithm Characteristics

Finally, results are shown on the average performance of the MAGNET evaluation algorithm (IDA* in this case) within one *cycle*. Figure 6 shows the performance for varying number of tasks in the RFQ. The algorithm scales very well and can be approximated by a linear function.

The characteristics of the bids that are evaluated are shown in Table 1, including detailed average evaluation times. The table shows that as the number of tasks increases, so does the average number of bids that have been received. This is logical because more tasks are presented, and therefore the probability of trucks being able to perform at least one of the tasks increases. Note that the average number of bids for a certain number of tasks can exceed half the amount of tasks (i.e. the number of trucks available when starting with that task size) as this concerns averages over all *cycles* and all amount of tasks that need to be scheduled upon a day (i.e. 2 to 20 tasks). It might for instance be the case that for a run with 20 tasks, multiple *cycles* are needed

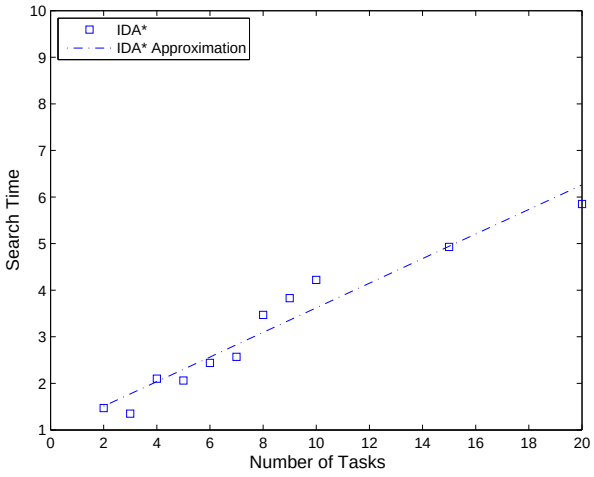

Fig. 6. Individual evaluation time needed by MAGNET

in which the last *cycle* only concerns 2 tasks whereas 10 trucks can still bid. Furthermore, the average number of tasks per bid increase with the number of tasks as well, which is due to the fact that tasks can more easily be combined.

Table 1. MAGNET evaluation characteristics

Number of Tasks	Avg. number of Bids	Avg. Tasks per Bid	Avg. Search Time
2	4.15	1.30	1.47
3	5.20	1.43	1.35
4	7.11	1.79	2.10
5	7.81	1.97	2.06
6	9.68	2.26	2.44
7	10.98	2.45	2.57
8	12.88	2.71	3.47
9	13.53	2.69	3.83
10	14.65	2.65	4.22
15	22.11	3.33	4.93
20	30.00	3.99	5.85

5 Related Work

Work done in centralized task allocation or planning involves finding efficient algorithms for solving (or approximating a solution for) specific problems. One specific family of problems is that of vehicle routing problems (VRP). A variant of

the VRP that is close to the task allocation problem used as an empirical evaluation in this paper include the capacitated VRP with pick-up and deliveries and time windows (CVRPPDTW). Furthermore, the dial-a-ride problem (DARP) generalizes a number of such vehicle routing problems [7] and when including capacities maps to the problem addressed in this paper. This problem is known to be an NP-hard problem to solve [2]. See for example [2], [17], and [11] for algorithms that solve such problems from a centralized perspective. Solving the vehicle routing problem from a centralized perspective might however not always be feasible, resulting in research focusing on decentralized task allocation as well.

Distributed constraint optimization algorithms have been proposed for task allocation (see, for instance, ADOPT [23] and OptAPO [22]). These algorithms are appropriate in domains where optimality is essential, but have high communication costs. [26] proposes an approximate algorithm for distributed task allocation which trades off optimality for reduced communication costs and which is specially suited for large teams in simulated search and rescue.

Auctions [20] have been suggested for allocation of computational resources since the 60's. The Contract Net [28] is perhaps the most well known and widely used bidding protocol for distributed problem solving. Many multi-agent and distributed systems use some form of auction to allocate resources. Auction-based methods for allocation of tasks are becoming popular as an alternative to other allocation methods, such as centralized scheduling [3], distributed planning [12;8], or application-specific methods, which do not easily generalize. An advantage of auctions is they are a distributed mechanism and draw on a large body of analytical results from economics. In addition, one-shot auctions are efficient in the case of low bandwidth and unreliable communications.

Scheduling plays an important role in task allocation, since before accepting a task an agent has to find how to fit it into its existing schedule. In [16] combinatorial auctions are used for the initial commitment decision problem, which is the problem an agent has to solve to decide whether to accept or refuse a new task. In [14] scheduling decisions are made not by the agents, but instead by a central authority, which has insight into the states and schedules of the agents. In MAGNET, there is no central authority; the market is used only as a repository of statistical information.

Despite the abundance of work in auctions, limited attention has been devoted to auctions over tasks with complex time constraints and interdependencies, as in MAGNET. Auctions for decentralized scheduling have been studied extensively by Wellman. The emphasis of their work is in the supply-chain construction, more than dealing with time, and in analyzing strategies using game-theoretic techniques. A protocol for combinatorial auctions for supply chain formation is proposed in [29]. Complex task networks are allowed, but they do not include time constraints. A protocol for decentralized scheduling is proposed in [31]. The study is limited to scheduling a single resource, while we are interested in multiple resources. In [30] agents bid for individual time slots on separate, simultaneous markets.

6 Conclusions and Future Work

In this paper, we present an approach to perform decentralized task allocation using the MAGNET system. There is already a vast amount of literature on performing such task allocation using negotiation, see e.g. [28] and [29], however, the unique feature of the system presented here concerns the negotiation about complex tasks including time window and precedence constraints. In a variety of domain such constraints are vital for task allocation, such as for the field of logistics. Implementations are created for both the *supplier* and *customer* agent where for the former two different bidding strategies are implemented, namely one which takes the distance to tasks into account (i.e. only bidding on tasks that are close to a task you already perform) called Closeby, and a Random bidding algorithm

To evaluate the proposed approach, a comparison is made to a central task allocation scheme which is able to calculate the optimal solution. Such an evaluation could be performed on a randomly generated dataset, in this paper however, the choice is made to use empirical data. This choice results in a dataset with characteristics that indeed occur in the real world, giving more insight in the usability of the approach in real life.

The evaluations show that the approach using Closeby bidding comes very close to the optimal result. The maximum deviation found is just over 4% of the optimal result, whereas the trend is that this deviation from the optimum is not (or hardly) increasing for greater amount of tasks. The Random bidding does not perform that well, showing that taking distances into account when bidding is very effective for the quality of the solution found. When looking at the computation time needed to come to the solution found, the MAGNET algorithm scales very well (linear), whereas calculation of the optimal solution does not (NP-hard). For 20 tasks, the maximum observed in the dataset, the MAGNET algorithm took a total of just under 12 msec.

For future work, we want to investigate what would be the influence of giving the *supplier* agents a preference for tasks would be upon the distance of the optimal solution. In the logistical domain for example, drivers of trucks tend to have particular preferences for tasks which is often taken into consideration by human planners. Furthermore, we want to investigate the scaling of the algorithms for very large datasets, consisting of for instance thousands of tasks that need to be allocated.

Acknowledgements

The authors would like to thank the logistics company for providing the data set.

References

[1] Briggs, P., The hand-off: the future of outsourced logistics may be found in the latest buzzword [fourth party logistics]. *Canadian Transportation Logistics*, 102(5):18, 1999.

[2] Charikar, M. and Raghavachari, B. The finite capacity dial-a-ride problem. In *39th Annual Symposium on Foundations of Computer Science*}, page 458, Los Alamitos, CA, USA, 1998. IEEE Computer Society.
[3] Chien,S., Barrett, A., Estlin, T., and Rabideau, G. A comparison of coordinated planning methods for cooperating rovers. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 100--101. ACM Press, 2000.
[4] Collins, J. Solving Combinatorial Auctions with Temporal Constraints in Economic Agents, PhD thesis, University of Minnesota, June 2002.
[5] Collins, J., Demir, G., and Gini, M. Bidtree ordering in IDA* combinatorial auction winner-determination with side constraints, In J. Padget, O. Shehory, D. Parkes, N. Sadeh, and W. Walsh, editors, *Agent Mediated Electronic Commerce IV*, volume LNAI2531, pages 17—33, Springer-Verlag, 2002.
[6] Collins, J., Ketter, W. and Gini, M., A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *Int'l Journal of Electronic Commerce*, 7(1):35--57, 2002.
[7] Cordeau, J. and Laporte, G., The dial-a-ride problem (darp): Variants, modeling issues and algorithms, *4OR*, 1, 2003.
[8] Cox, J.S., Durfee, E.H., and Bartold, T., A distributed framework for solving the multiagent plan coordination Problem, In *Autonomous Agents and Multi-Agent Systems*, pages 821--827, 2005.
[9] Desrochers, M., Desrosiers, J., and Solomon, M., A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research*, 40(2):342--354, 1992.
[10] Dias,M.B., Zlot, R.M., Kalra, N., and Stentz, A.T., Market-based multirobot coordination: A survey and analysis, Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2005.
[11] Dorer, K. and Calisti, M., An adaptive solution to dynamic transport optimization, *Proc. of AAMAS05*, pages 45--51, 2005.
[12] Durfee, E.H., Scaling up agent coordination strategies, *IEEE Computer*, 34(7):39--46, July 2001.
[13] Gerkey, B.P. and Mataric, M.J., Sold!: Auction methods for multi-robot coordination, *IEEE Trans. Robotics and Automation*, 18(5):758--786, October 2002.
[14] Glass, A. and Grosz, B.J., Socially conscious decision-making, In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 217--224, June 2000.
[15] Hoogendoorn, M. and Jonker, C.M., Formation of virtual organizations through negotiation, In *Proceedings of the Fourth German Conference on Multiagent Technologies (MATES 2006)*, pages 135--146. Springer, 2006.
[16] Hunsberger, L., and Grosz, B.J., A combinatorial auction for collaborative planning, In *Proc. of 4th Int'l Conf on Multi-Agent Systems*, pages 151--158, Boston, MA, 2000, IEEE Computer Society Press.
[17] Jaw, J.J, Odoni, A., Psaraftis, H., and Wilson, N., Heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows, *Transportation Research Part B*, 20B:243--257, 1986.
[18] Kasilingam, R., Logistics and Transportation: Design and Planning}, Springer, 1999.
[19] Korf, R. E., Depth-first iterative deepening: An optimal admissible tree search, *Artificial Intelligence*, 27:97--109, 1985.
[20] Krishna, V., *Auction Theory*, Academic Press, London, UK, 2002.
[21] Magnanti, T., Combinatorial optimization and vehicle fleet planning: Perspectives and prospects, *Networks*, 11:179--214, 1981.
[22] Mailler, R. and Lesser, V., Solving distributed constraint optimization problems using cooperative mediation, In *Proc. of AAMAS04*, 2004.
[23] Modi, P.J, Shen, W.-M., Tambe, M., and Yokoo, M., An asynchronous complete method for distributed constraint optimization, In *Proc. of AAMAS03*, 2003.

[24] Moehlman, T.A., Lesser, V.R., and Buteau, B.L., Decentralized negotiation: An approach to the distributed planning problem, *Group Decision and Negotiation*, 1:161--191, 1992.
[25] Notteboom, T., Container shipping and ports: An overview. *Review of Network Economics*, 3(2):86--106, 2004.
[26] Scerri, P., Farinelli, A., Okamoto, S., and Tambe, M., Allocating tasks in extreme teams, In *Proc. of AAMAS05*, pages 727--734, 2005.
[27] Schut, M.C., Kentrop, M., Leenaarts, M., Melis, M. and Miller, I., Approach: Decentralised rotation planning for container barges., In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 755--759, 2004.
[28] Smith, R.G., The contract net protocol: High level communication and control in a distributed problem solver, *IEEE Trans. Computers*, 29(12):1104--1113, December 1980.
[29] Walsh, W.E., Wellman, M., and Ygge, F., Combinatorial auctions for supply chain formation, In *Proc. of ACM Conf on Electronic Commerce (EC'00)*, pages 260--269, October 2000.
[30] Wellman, M., MacKie-Mason, J., Reeves, D., and Swaminathan, S., Exploring bidding strategies for market-based scheduling, In *Proc. of Fourth ACM Conf on Electronic Commerce*, 2003.
[31] Wellman, M.P., Walsh, W.E., Wurman, W.R. and MacKie-Mason, J.K., Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271--303, 2001.

Part V:
Organizational Change Process:
Mixed Change Processes

Chapter 13

Automated Evaluation of Coordination Approaches for Component-based Software Systems

Part of this chapter appeared as: Bosse, T., Hoogendoorn, M., and Treur, J., Automated Evaluation of Coordination Approaches. In: Ciancarini, P. and H. Wiklicky, H. (eds.), *Coordination Models and Languages: Proceedings of COORDINATION 2006*, LNCS 4038, Springer Verlag, 2006, pp. 44-62. The original publication is available at www.springerlink.com.

Automated Evaluation of Coordination Approaches for Component-based Software Systems

Tibor Bosse, Mark Hoogendoorn, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
{tbosse, mhoogen, treur}@cs.vu.nl, http://www.cs.vu.nl/~{tbosse, mhoogen, treur}

Abstract. How to coordinate the processes in a complex component-based software system is a nontrivial issue. Many different coordination approaches exist, each with its own specific advantages and drawbacks. To support their mutual comparison, this paper proposes a formal methodology to automatically evaluate the performance of coordination approaches. This methodology comprises (1) creation of simulation models of coordination approaches, (2) execution of simulation experiments of these models applied to test examples, and (3) automated evaluation of the models against specified requirements. Moreover, in a specific case study, the methodology is used to evaluate some coordination approaches that originate from various disciplines.

1 Introduction

Coordinating the processes in a complex software system is a nontrivial issue. In a nutshell, the problem of coordination comes down to *deciding when the different processes involved can be performed*. For more precise definitions, see, e.g., [9, 21]. By a component-based approach to software systems, a divide and conquer strategy can be used to address the various aspects involved. This may lead to a possibly large number of components, which each can be analyzed and designed independently. However, a designer may still be left with the problem how all these fragments can be combined into a coherent system. To solve such a problem, many different coordination approaches have been proposed, each having its advantages and drawbacks. Important questions when choosing such a coordination approach are the suitability, correct functioning, and efficiency of the approach for the particular component-based system.

This paper presents a methodology to enable a comparison of such factors for the different coordination approaches in a series of test examples. First of all, this methodology allows for the creation of simulation models for each of the coordination approaches. Secondly, it comprises an engine which simulates the different coordination approaches for a variety of test examples. Finally, the methodology consists of an automatic evaluation of the outcome of the simulations against specified requirements (e.g. successfulness and efficiency).

The problem of coordination of component-based software systems has crucial aspects in common with the problem of coordination in natural (biological), cognitive

(human and animal mind) or societal systems (organizational structures). Evolution processes over long time periods have generated solutions for the coordination problem in these areas. Therefore, it may make sense to analyze in more detail how these solutions work. Some literature is available that describes theories for coordination in these areas. This literature can be used as a source of inspiration to obtain new approaches to coordination of complex component-based software systems. As a first step, this paper evaluates a number of such approaches in a specific case study, to see to what extent they provide satisfactory solutions.

First, in Section 2 the methodology and supporting software tools are described. In Section 3 a number of coordination approaches obtained from the literature in various disciplines are briefly introduced. Section 4 describes a set of test examples that can be used as input for the evaluation of the coordination approaches. In Section 5 the simulations that were undertaken to evaluate the usefulness of the coordination approaches for the test examples are briefly discussed. Section 6 presents the results, and Section 7 is a final discussion.

2 Evaluation Method

To explore possibilities to address the coordination problem, an evaluation methodology, supported by a software environment, has been created, which consists of the following steps: (a) a number of *coordination approaches* are selected, (b) a number of *test examples* representing specific software component configurations are chosen, (c) based on each of these coordination approaches a *simulation model* is formally specified, (d) related to the test examples, relevant *requirements* are formally specified in the form of relevant *dynamic properties*, (e) *simulations* are performed where selected coordination approaches are applied to the chosen test examples, resulting in a number of *simulation traces*, and (f) the simulation traces are *evaluated* (automatically) for the specified requirements. Figure 1 gives a graphical overview of the evaluation methodology.

To evaluate a given coordination approach, adequate test examples of component-based software configurations are needed (step b). One may be tempted to use a real-life component-based software system as a test example, e.g., consisting of hundreds of components. However, such type of testing for one case would take a lot of effort, and to get a reasonable idea it should be repeated for a representative number of software systems at least. For this stage of the exploration this would not be appropriate. Instead, a number of smaller but representative test examples have been identified. As a source, the library of workflow patterns described in [1] has been used. The examples given there have been extended with input and output data and information flow channels.

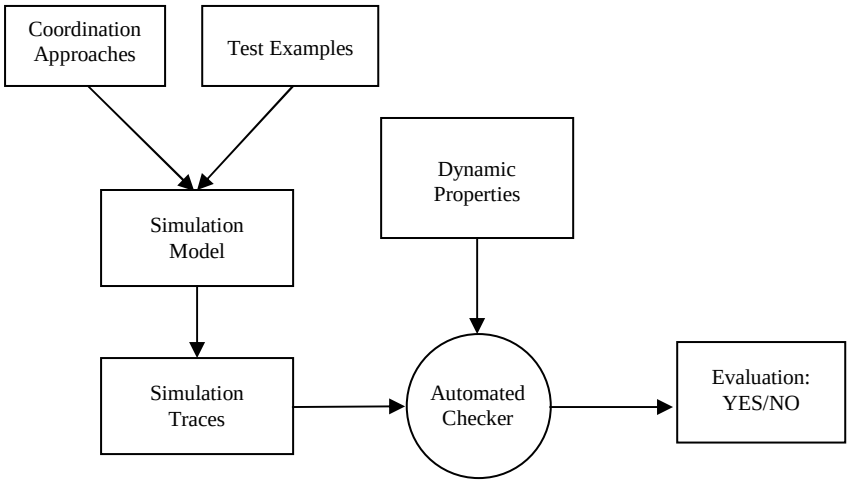

Fig. 1. General research methodology

To test the selected coordination approaches on the chosen examples, implementations have to be made. One way to do this would be to create specific implementations for each of the (abstract) test examples, by explicitly defining the internal functioning of the components involved. Next, one would add to these implementations - one by one - implementations of the coordination approaches, and then run each of these implementations. The resulting log data, which should include a registration of the processing time, for example, in terms of processor time or number of computation steps, can then be evaluated. Such an evaluation at an implementation level, however, has some drawbacks: the specific implementations chosen may affect the results, and the specific underlying software/hardware combination may affect the processing times measured; e.g., think of aspects of concurrency that within a software/hardware environment may have to be mapped onto a form of interleaving of processes. Therefore a different approach is chosen: all the testing is done within one given simulation environment. Within this environment, one by one the processing of a software system based on one test example and one coordination approach is simulated. In that case, the examples are defined at an abstract level (i.e., only in terms of input-output relations, ignoring the internal functioning). The measured time then is simulated time, not processing time. In simulated time, processes can easily be active in parallel. The simulation environment chosen is logic-based, so that the simulation models and the resulting simulation traces can be logically analyzed, supported by another software environment.

To evaluate the resulting simulation traces, in the first place it is needed to identify the relevant properties, serving as requirements, on which such an evaluation should be based. A number of aspects can be covered in such requirements. A first aspect is effectiveness or *successfulness* to provide the desired output for the example system.

When a coordination approach does not involve the right components at the right times, and therefore is not able to generate the desired output, then it is not effective. A second aspect to evaluate is *efficiency*: to what extent time is wasted in the process to obtain the eventual goals. A third aspect is to what extent the coordination approach is able to generate the possible *activation traces* one has in mind for the given example. Such properties can be formally specified and automatically checked for the simulation traces.

To support the evaluation method described, a software environment is used. By means of this software environment, one can logically specify simulation models, execute these models in order to get simulation traces, specify relevant dynamic properties, and check such properties against simulation traces. For the simulation part, the language LEADSTO is used [6], based on a variant of Executable Temporal Logic [4]. The basic building blocks of this language are causal relations of the format $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, which means:

if	state property α holds for a certain time interval with duration g,
then	after some delay (between e and f) state property β will hold for a certain time interval of length h.

where α and β are state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. For the analysis part, the language TTL is used [7]. This predicate logical language supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. TTL is built on atoms referring to states, time points and traces. A *state* of a process for (state) ontology Ont is an assignment of truth values to the set of ground atoms in the ontology. The set of all possible states for ontology Ont is denoted by STATES(Ont). To describe sequences of states, a fixed *time frame* T is assumed which is linearly ordered. A *trace* γ over state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$. A special software environment has been developed for TTL, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against a set of (simulated or empirical) traces.

3 Coordination Approaches

As mentioned earlier, the coordination problem in software systems has crucial aspects in common with the problem of coordination in natural (biological), cognitive

(human and animal mind) or societal systems (organizational structures). Therefore, a large body of literature is available that describes coordination approaches in these areas. In this section, some of the most well-known approaches are discussed. Section 3.1 focuses on the *behavior networks* approach by Pattie Maes [19]. Section 3.2 describes Selfridge's *pandemonium* model [25], and Section 3.3 addresses the decision-making techniques known as *voting methods* [20]. These approaches were chosen for two reasons. First, because they are well-known approaches in the (wider) literature in various disciplines on coordination. Second, because together they more or less cover the area of different coordination approaches: the behavior networks use a rather *global* and sequential strategy (i.e., the approach determines which component is activated based on global information concerning all components), whereas voting methods and (especially) the pandemonium model use a *local* and possibly nonsequential strategy (i.e., the components involved only use information about themselves or their direct neighbors to determine which component is activated).

3.1 Behavior Networks

Behavior networks have been introduced by Pattie Maes in 1989. She distinguishes *competence modules* within a system, where each module is specified by a tuple containing four elements: (1) a list of preconditions to be fulfilled before a competence module can become active; (2) the competence module's action in terms of an add list; (3) the competence module's actions in terms of a delete list; (4) a level of activation. A competence module is said to be executable in case the list of preconditions is fulfilled. A network of competence modules is created via three types of links: successor links (a link from x to y for every element on the add list of x which is on the preconditions list of y), predecessor links (a link from x to y for every element on the precondition list of x which is on y's add list), and conflictor links (a link from x to y for every element on the precondition list of y which is on x's delete list). Through these links the competence modules activate and inhibit each other, so that "after some time the activation energy accumulates in the modules that represent the 'best' actions to take given the current situation and goals" [19]. The patterns of these spreading activations among modules, as well as the input of new activation energy into the network, is determined by the state of the environment and goals via three ways: activation by state (add activation to modules that (partially) match the current state), activation by goals (add activation to modules which (partially) achieve the goals), and inhibition by protected goals (remove activation from modules that (partially) remove the protected goals). Thereafter, activation spreads through the network via activation of successors, activation of predecessors, and inhibition of conflictors. After having spread the activation, a decay phase makes sure the overall activation remains constant within the network. Once performed, a competence module fires in case it is executable, the activation is over the threshold that has been set, and it is the competence module with the highest activation. In case the module indeed fires, its activation goes to 0, and all thresholds return to their normal value. In case no module fires, the threshold is reduced by 10%. For more mathematical details, see [19].

3.2 The Pandemonium Model

In 1958, Selfridge proposes an approach he calls pandemonium, to enable pattern recognition [25]. This is a system composed of primitive constructs called *demons*, each representing a possible pattern. Once an image is presented, each of the demons computes the similarity of the image with the pattern it represents, and gives an output depending monotonically on that similarity. Finally, a decision demon selects the pattern belonging to the demon whose output is largest.

Jackson [16] extends this idea to a theory of mind. Besides demons involved in perception, he also identifies demons that cause external actions and demons that act internally on other demons. Jackson pictures the demons as living in a stadium. Almost all of them are the crowd, cheering on the performers. The remainder of the demons are down on the playing field, exciting the crowd in the stands. Demons in the stands respond selectively to these attempts to excite them. Some are more excited than others; some shout louder. The demon in the stands that shouts loudest replaces one of the currently performing demons, which is sent back to the stands. The loudness of the shouting of a demon is dependent upon being linked with the demon that must excite. Stronger links produce louder responses. The system starts off with initial built-in links between the demons. New links are made between demons, and existing links are strengthened in proportion to the time they have been together on the field, plus the gain of the system (i.e., when all is going well, the gain is higher).

3.3 Voting Methods

The concept of *voting* refers to a wide collection of techniques that are used to describe decision-making processes involving multiple agents. Although originating from political science, voting methods are currently used within a number of domains, including game theory (where they are used as methods for conflict resolution) and pattern recognition (where they are used to combine classifier outputs).

The general idea of voting methods is rather intuitive, and is comparable to the techniques used in elections. Consider a set of agents N, and a set of possible outcomes S of an election. Each agent $i \in N$ has preferences over the outcomes: $\leq_i \subseteq S \times S$. The voting approach uses a function F that selects a candidate outcome S, given the preferences of the voters. A simple instance of F would be to count all votes, and to select the outcome with the highest amount of votes. However, a large number of (more complex) voting approaches exist. These can roughly be divided into three classes: *unweighed voting methods* in which each vote carries equal weight, *confidence voting methods* in which voters can express a degree of preference for a candidate, and *ranked voting methods* in which the voters are asked for a preference ranking over the candidates. See [20] for an overview of different voting methods.

As mentioned above, voting methods are currently used in many different domains, such as game theory and pattern recognition. In this paper it will be explored whether they are of any use to solve coordination problems in complex (component-based) software systems. To this end, the electorate will be filled in by certain components, and the candidates by the possible activations of components.

4 Test Examples

Test examples have been identified to test the different coordination approaches. The examples were inspired by the workflow patterns defined by van der Aalst et al. [1]. These patterns can be seen as building blocks for more complex patterns occurring in real-life component-based systems. In total, seven test examples have been described, which are discussed below. A test example consists of a number of components, called {C1, C2,..}, and several types of data, called {d1, d2,..}. Different components need different data as input, and create different data as output.

Pattern 1 - Sequence

The first workflow pattern defined by [1] is straightforward: it involves three components. After completion of the first component, the second component is activated, and after completion of the second, the third component is activated.

On the basis of this pattern, a next step was to create a corresponding test example. In principle, this means defining an example (in terms of components and data) in such a way that, if provided as input to a coordination approach, pattern 1 will come out. A visualization of such an example is given in Figure 2. In this case component C1 needs data d1 as input, and creates data d2 as output. Moreover, as indicated in the box on the right, the *input data* (the data that is initially available to the system) is d1, and the *goal data* (the data that the system needs to create in order to be successful) is d4. Given this situation, the expectation is that if any coordination approach is applied to the example, the result will be a trace in which the components are activated in sequence (i.e., first C1, then C2, and then C3). Note that it is assumed that data is shared, i.e., whenever a component generates output data, this data is immediately available to all other components in the system. This could be implemented, for example, by incorporating a *shared repository*, where all components store their output data and read their input data from. Another assumption is that data cannot be removed. Thus, once data is written to the shared repository, it will stay there. Other approaches such as explicit communication channels can however easily be incorporated into the methodology.

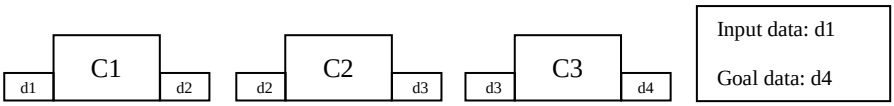

Fig. 2. Test example 1 – Sequence

Pattern 2 – Parallel Split

The second example, the parallel split, is depicted in Figure 3. In order to translate the actual pattern to the test example in the figure, the same approach as described for the

first pattern has been used. Here, the components C2 and C3 can be executed either simultaneously or in any order.

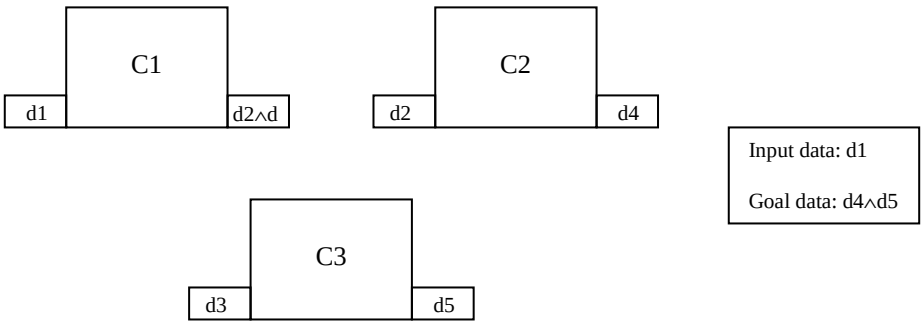

Fig. 3. Test example 2 - Parallel Split

Note that in this case the ∧ stands for the conjunction of two data types. For example, the output data of component C1 is d2 and d3. Likewise, the goal data is d4 and d5.

Pattern 3 - Synchronization

The synchronization pattern is depicted in Figure 4. Here, C1 and C2 can be executed either simultaneously or in any order.

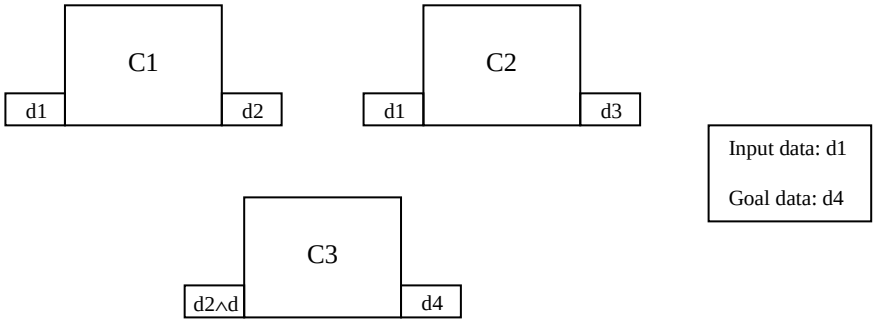

Fig. 4. Test example 3 – Synchronization

Note that in this case it is assumed that a component cannot reason with “partial” data (this would be the case when, e.g., component C3 starts reasoning with d2 only, whilst its input data is d2 and d3).

Pattern 4 – Exclusive Choice

The exclusive choice pattern, is depicted in Figure 5. Here, either component C2 or component C3 may be activated, but not both.

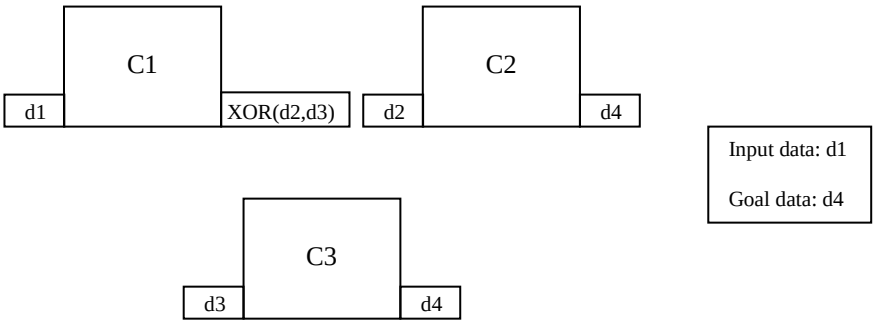

Fig. 5. Test example 4 - Exclusive Choice

Note that in this case the XOR stands for the exclusive disjunction of two data types. For example, the output data of component C1 is either d2 or d3, but not both. The specific output generated by the component may differ in different simulation runs.

Pattern 5 – Simple Merge

The simple merge is depicted in Figure 6. Here, either component C1 or component C2 may be activated, but not both.

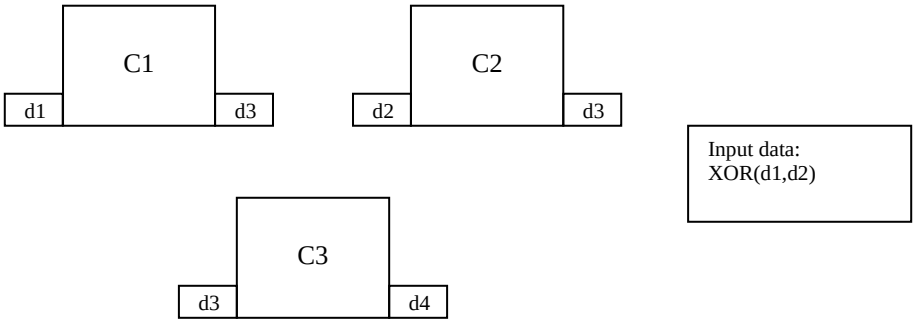

Fig. 6. Test example 5 - Simple Merge

Note that in this case the input data is the exclusive disjunction of d1 and d2, i.e., in some simulation runs it is d1, and in others it is d2.

Pattern 6 – Multi-Choice

The sixth pattern, the multi choice, is depicted in Figure 7. Here, either component C2, or component C3, or both components may be activated.

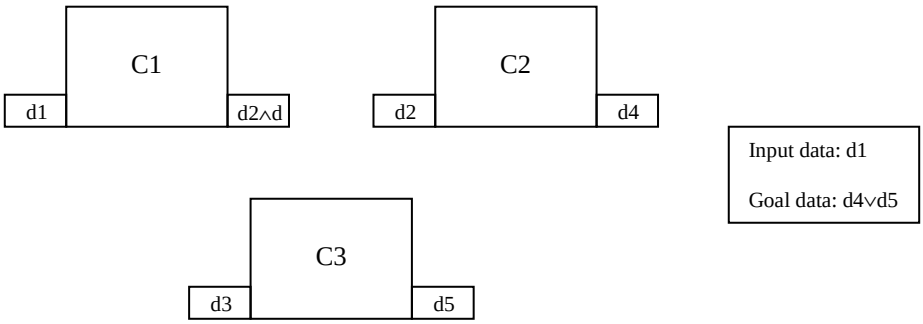

Fig. 7. Test example 6 - Multi Choice

Note that in this case the ∨ stands for the standard disjunction of two data types. Thus, in this case the goal data of the system is d4 or d5 or both.

Pattern 7 - Synchronizing Merge

Pattern 7 involves four components. After completion of the first component, there is a choice between the second and third component: either one of them can be activated, or both. In case one of them is activated, the fourth component is activated after this component has completed. In case both of them are activated, the fourth component is activated after both have completed.

The test example that was created on the basis of this pattern is shown in Figure 8. As can be seen in the figure, in this example both a conjunction in a component's output data and a disjunction in a component's input data occur. Furthermore, note that, when formalizing this example in LEADSTO, the disjunction on the input side of C4 is modeled by defining three separate variants of C4: one variant (called C4) with d4 as input, one variant (called C5) with d5 as input, and one variant (called C6) with d4 and d5 as input.

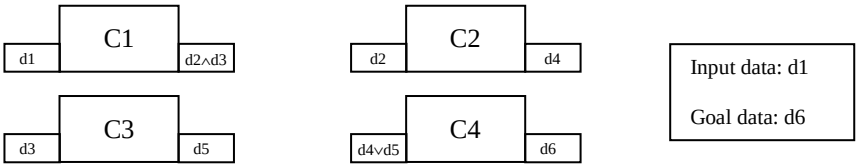

Fig. 8. Test example 7 - Synchronizing Merge

In the next section, these seven test examples will be used as a basis for simulation experiments. For more details about the original workflow patterns, the user is referred to [1]. For an explanation of how these workflow patterns can be related to the area of business process modeling, see [10].

5 Simulation

To compare the coordination approaches described in Section 3 against the test examples shown in Section 4, a number of simulation experiments have been performed. First, the three selected coordination approaches have been implemented in the LEADSTO simulation language (see [5] for implementation details). Next, the implemented simulation models have been applied to the test examples. The simulation models for the behavior networks, the pandemonium, and the voting method, are addressed, respectively, in Section 5.1, 5.2, and 5.3. For each simulation model, two example simulation traces (resulting from applying the model to test example 1 and 7) are provided. The complete set of simulation traces can be found in [5].

5.1 Behavior Networks Simulation

The simulation model for Maes' behavior networks is created on the basis of the mathematical model as presented in [19]. There is one difference: within the simulation model, the lowering of the threshold is not performed, as the available data does not change due to external influences (i.e., the highest executable component will remain the highest until a component has been activated). Therefore, the highest executable component is simply selected, avoiding unnecessary computation. The LEADSTO specification for the approach roughly corresponds to the description in Section 3.1. Table 1 shows the ontology used in the simulation model.

Table 1. Ontology used within the behavior networks simulation model

Relation	Description
input_from_state: TIME x COMPONENT x VALUE	At the time point the component gets the value for activation through the state at that time point.
input_from_goals: TIME x COMPONENT x VALUE	At the time point the component gets the value for activation through the goals that have been set.
spreads_fw: COMPONENT x COMPONENT x TIME x VALUE	At the specified time point the specified activation spreads forwards from the first component to the second
spreads_bw: COMPONENT x COMPONENT x TIME x VALUE	At the specified time point the specified activation spreads backwards from the first component to the second
executable: TIME x COMPONENT	This specifies that the component is executable at the particular time point.
decay: TIME x COMPONENT x VALUE	The component has the specified decay value at the particular time point.

alpha: TIME x COMPONENT x VALUE	The component has the specified alpha value at the particular time point.
active: TIME x COMPONENT x VALUE	This relationship specifies whether or not a component was active at a particular time point. In case VALUE is 1 this is the case, in case of a 0 this is not the case.
activated: COMPONENT	The component is activated.

Pattern 1

Figure 9 shows the simulation trace that resulted from applying the behavior networks approach to the first test example. The left side of the figure shows the state properties that occur during the simulation, whereas the right side shows a time line where a dark box indicates the state property being true and a light box the state property being false.

Initially, the data present is set to d1: data(d|1). Furthermore, the goal is set to d4 for this particular scenario: goal(d|4). Before executing the model several initial values are set to enable a proper functioning. First of all, the activation values (referred to as the alpha values) of the components currently present in the system are set to 0 for the time point before the current time point (i.e. time point 0): alpha(0, c|1, 0), alpha(0, c|2, 0), and alpha(0, c|3, 0). Furthermore, the components' activity at time point 0 is set to 0 as well: active(0, c|1, 0), active(0, c|2, 0), and active(0, c|3, 0). Now the model is executed. First of all, it is determined that only component C1 is executable given the current data available: executable(1, c|1). Then, calculations are performed to determine the activity within the different component. To enable these calculations, several intermediate steps are taken. First of all, the input from the current state is calculated (i.e. given the current data available what is the activation caused for the different components). Since component C1 is the only component that has its preconditions fulfilled, it is the only component to have activation from this source: input_from_state(1, c|1, 0.1). Another intermediate step is to calculate the input from the goals. Since only C3 has a goal as an output, this component is the only one to receive activation through this source: input_from_goals(1, c|3, 0.3). Due to the fact that the previous alpha value is 0, no activation is spread around the network, so the decay can be calculated for the three components present in the system by simply summing up the input from the goals and state per component: decay(1, c|1, 0.1), decay(1, c|2, 0), and decay(1, c|3, 0.3). Calculating the alpha value entails normalizing these numbers. The maximum activation is set to 1 in this example, resulting in the following alpha values: alpha(1, c|1, 0.25), alpha(1, c|2, 0), and alpha(1, c|3, 0.75). As a result, component C1 is activated as this is the executable component with the highest alpha value: active(1, c|1, 1), active(1, c|2, 0), and active(1, c|3, 0). Due to the activity of component C1 its output data is generated, which shows in the trace by means of the presence of data d2: data(d|2).

After that, a new round of the model is performed; both components C1 and C2 are now derived to be executable, since the data is assumed to remain present permanently. The input from the goals remains the same as these have not changed. The input from the current state however changes due to the additional data d2 being present, resulting in an input from state for component C2 as well: input_from_state(2, c|2, 0.1). Since C3 was not active at the previous time point, its activation spreads back through the network, resulting in a backwards spread from C3 to C2: spreads_bw(c|3, c|2,

2, 0.75). Calculation of the decay can now be performed: decay(2, c|1, 0.1), decay(2, c|2, 0.85), and decay(2, c|3, 1.05). Normalization takes place and eventually C2 is selected, resulting in data d3 being present. In the last cycle, C3 is selected with by far the highest alpha value, resulting in the overall goal being reached: data(d|4).

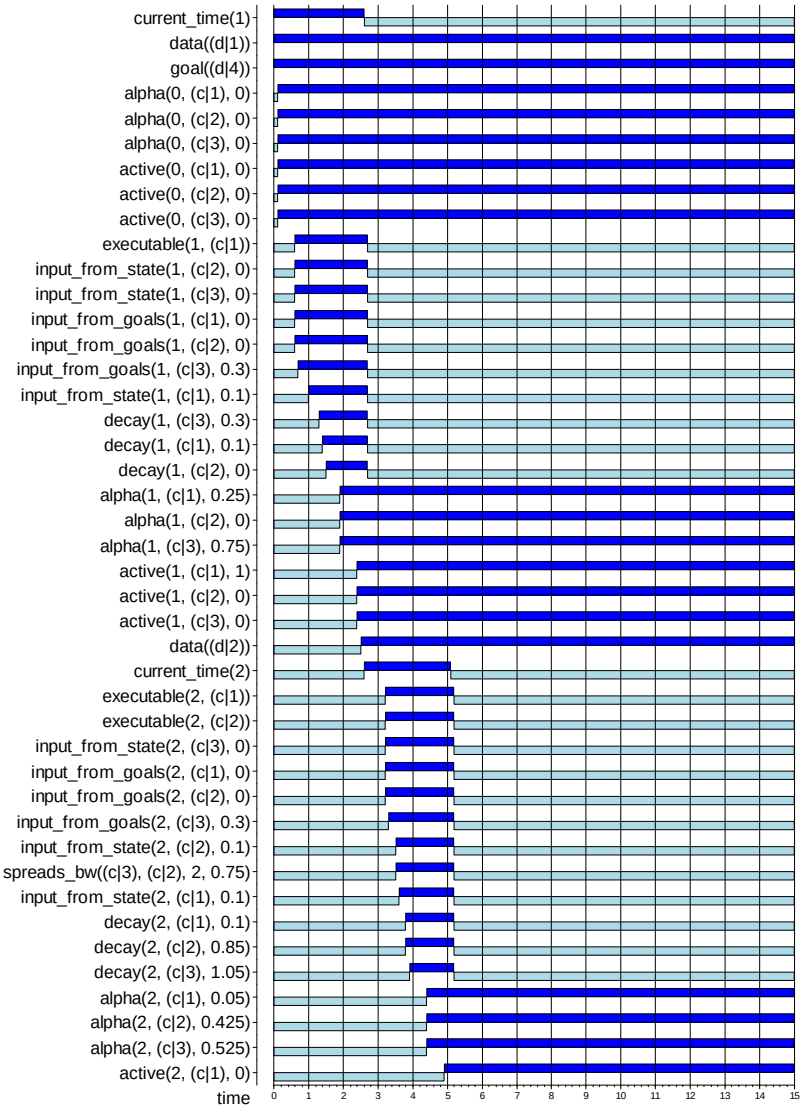

Fig. 9. Simulation Trace - Behavior Networks against Test Example 1
(continued on next page)

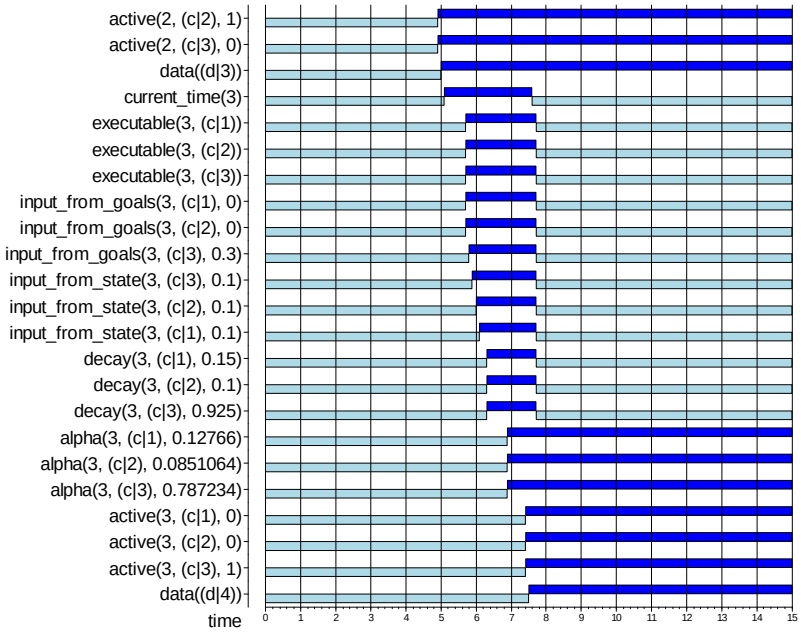

Fig. 9(contd). Simulation Trace - Behavior Networks against Test Example 1

Pattern 7

Figure 10 presents a simulation trace that has resulted from executing the approach on test example 7. Initially, the data present is set to d1: data(d|1). Furthermore, the goal is set to d6 for this particular scenario: goal(d|6). Before starting, the alpha values are set to 0 for the time point before the current time point (i.e. time point 0): alpha(0, c|1, 0), alpha(0, c|2, 0), alpha(0, c|3, 0), alpha(0, c|4, 0), alpha(0, c|5, 0), and alpha(0, c|6, 0). Thereafter calculations are performed to determine the activity within the different components: The input from the current state is calculated (i.e. given the current data available, calculate the activation caused for the different components) as well as the input from the goals. Since only C4, C5, and C6 have a goal as an output, these components are the only ones to receive activation through this source. Due to the fact that the previous alpha value is 0, no activation is spread around the network. The next alpha value for the six components present in the system is therefore obtained by simply summing up the input from the goals and state per component, and normalizing it to 1: alpha(1, c|1, 0.25), alpha(1, c|2, 0), alpha(1, c|3, 0), alpha(1, c|4, 0.25), alpha(1, c|5, 0.25), and alpha(1, c|6, 0.25). As a result, component C1 is activated, as this is the executable component with the highest alpha value: activated(c|1). Due to the activity of component C1, its output data is generated, which is shown in the trace: the presence of data d2 and d3: data(d|2) and data(d|3).

After that, a new round of computation is performed; the input from the goals remains the same, as these have not changed. However, the input from the current state changes, due to the additional data d2 and d3 being present. Furthermore,

activation is now spread through the network, since the previous alpha values are non-zero. After calculation and normalization the following alpha values are the result: alpha(1, c|1, 0.0425532), alpha(1, c|2, 0.255319), alpha(1, c|3, 0.255319), alpha(1, c|4, 0.148936), alpha(1, c|5, 0.148936), and alpha(1, c|6, 0.148936). Since both C2 and C3 are executable and have the highest alpha value, one of them is randomly selected; in Figure 10 this is component C2.

As can be seen in the figure, after activation of C2, component C3 is activated. Finally, C4 is activated, outputting the goal data, which results in termination.

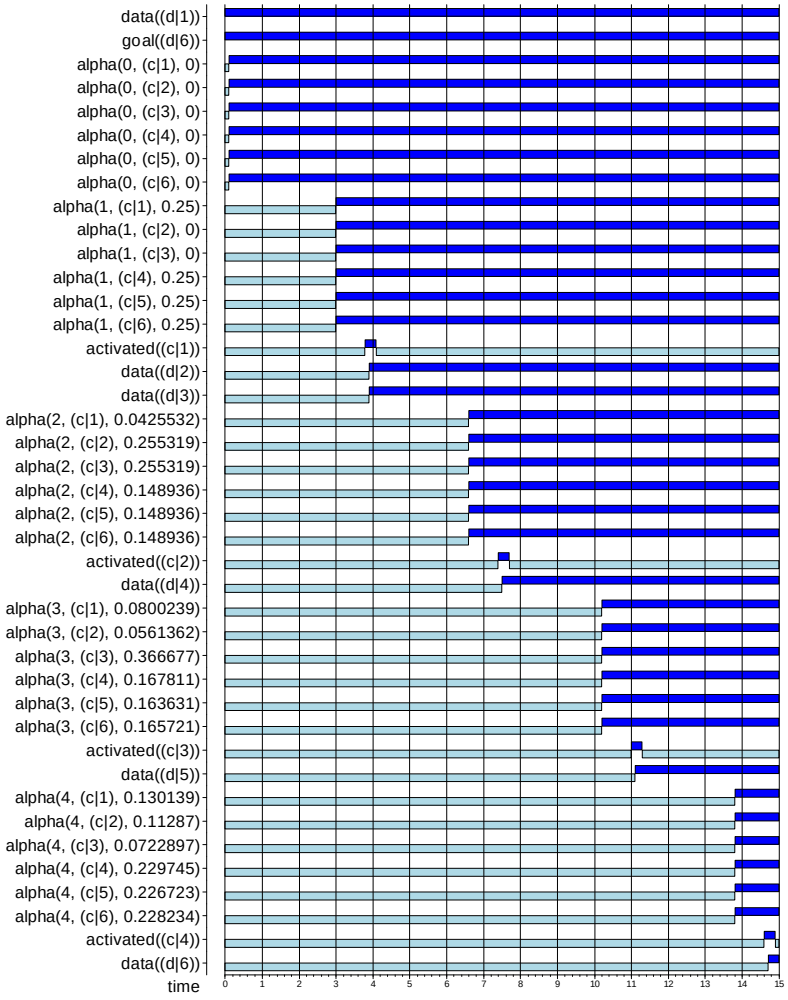

Fig. 10. Simulation Trace - Behavior Networks against Test Example 7

5.2 Pandemonium Simulation

The pandemonium is used as described in Section 3.2, but modified with some simplifying assumptions. In particular, the following procedure is assumed: at the beginning of the process, only the initial data is placed at the shared repository. Whenever new data has been added to the repository, a new round starts in which all components can *shout*. The idea is that, the more urgent a component thinks it is for him to be activated, the louder it will shout. The component that shouts loudest will be allowed to start processing. In case two components shout with exactly the same strength, then either the first component, or the second component, or both are activated (this decision is made randomly, with equal probabilities). When a component is activated, this results in the component adding its output data to the *shared repository* (see Section 4), and the start of a new round.

To determine how loud they will shout, the components make use of a shout function. For different variants of the pandemonium model, different shout functions may be used. In the current model, each component uses the following types of information in its shout function at time point t:

- the amount of data it needs as input (represented by $i1$)
- the amount of its input data that is available at t (represented by $i2$)
- the amount of data it produces as output (represented by $o1$)
- the amount of its output data that is already present at t (represented by $o2$)
- the highest $i1$ for the set of components (represented by max_i)
- the highest $o1$ for the set of components (represented by max_o)

Given these elements, the shout value (i.e., the strength with which a component shouts, represented by sv) is modeled as follows:

$$sv = (i2/i1)^{\beta 1} * (1 - o2/o1)^{\beta 2} * (i1/max\_i)^{\beta 3} * (o1/max\_o)^{\beta 4}$$

Here, $\beta 1$, $\beta 2$, $\beta 3$, and $\beta 4$ are real numbers between 1 and 1.5, indicating the importance of the corresponding factor. Several settings have been tested for these parameters. In the examples shown here, $\beta 1$=1.4, $\beta 2$=1.3, $\beta 3$=1.1, and $\beta 4$=1.2. Since the factors can never exceed 1, the shout value sv will be a value between 0 and 1. The ontology used in the pandemonium simulation is shown in Table 2.

Table 2. Ontology used within the pandemonium simulation model

Relation	Explanation
data: DATA	This specifies that a certain type of data is present in the repository.
shout: COMPONENT x VALUE	A component shouts with a certain (real) value.
active_component: COMPONENT	A component is activated.

Pattern 1
Figure 11 shows the simulation trace of the pandemonium approach for pattern 1.

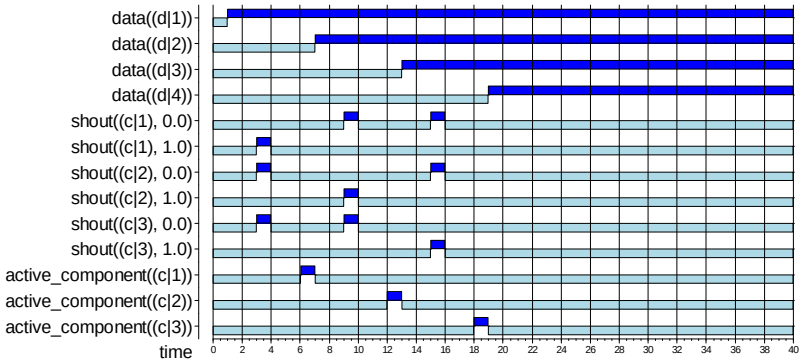

Fig. 11. Simulation Trace - Pandemonium against Test Example 1

As can be seen in Figure 11, initially the only data that is present is d1: data(d|1). Based on these data, every component starts shouting. Component C1 shouts loudest (with strength 1.0, whilst the others shout with strength 0.0): shout(c|1, 1.0), shout(c|2, 0.0), and shout(c|3, 0.0). Thus, component C1 is selected to become active: active_component(c|1). As a result, component C1 creates data d2, which is stored at the repository as well: data(d|2). Again, every component starts shouting. Component C2 shouts loudest (with strength 1.0, whilst the others shout with strength 0.0): shout(c|1, 0.0), shout(c|2, 1.0), and shout(c|3, 0.0). Next, component C2 is selected to become active: active_component(c|2). Next, component C2 creates data d3, which is stored at the repository as well: data(d|3). Again, every component starts shouting. Component C3 shouts loudest (with strength 1.0, whilst the others shout with strength 0.0): shout(c|1, 0.0), shout(c|2, 0.0), and shout(c|3, 1.0). Next, component C3 is selected to become active: active_component(c|3). Eventually, component C3 creates data d4, which is stored at the repository as well: data(d|4). Since d4 is the goal data, at this point the process terminates.

Pattern 7

Figure 12 depicts the simulation trace that has resulted from applying the pandemonium approach to test example 7. As the figure shows, initially the only data that is present is d1: data(d|1). Based on these data, every component starts shouting. Component C1 shouts loudest (with strength 0.47, whilst the others shout with strength 0.0): shout(c|1, 0.466516), shout(c|2, 0.0), ..., shout(c|6, 0.0). Thus, component C1 is selected to become active: active_component(c|1). As a result, C1 creates data d2 and d3, which are stored at the repository as well: data(d|2), data(d|3). Then again, every component starts shouting. This time, both component C2 and C3 shout loudest (with strength 0.20, whilst the others shout with strength 0.0): shout(c|1, 0.0), shout(c|2, 0.203063), shout(c|6, 0.0). As a result, both component C2 and C3 are selected to become active: active_component(c|2), active_component(c|3). Note that this selection is based on the assumption that multiple components may be activated at the same time. If this is not allowed, the approach would select one of the components at random. Next, component C2 creates data d4, and component C3 creates data d5. These data are stored at the repository: data(d|4), data(d|5). Again, every component starts shouting.

Component C6 (which is a specific variant of C4, see the description of the example) shouts loudest (with strength 0.44): shout(c|1, 0.0), shout(c|2, 0.0), shout(c|6, 0.435275). Thus, component C6 is selected to become active: active_component(c|6). Eventually, component C6 creates data d6, which is stored at the repository: data(d|6). Since d6 is the goal data, at this point the process terminates.

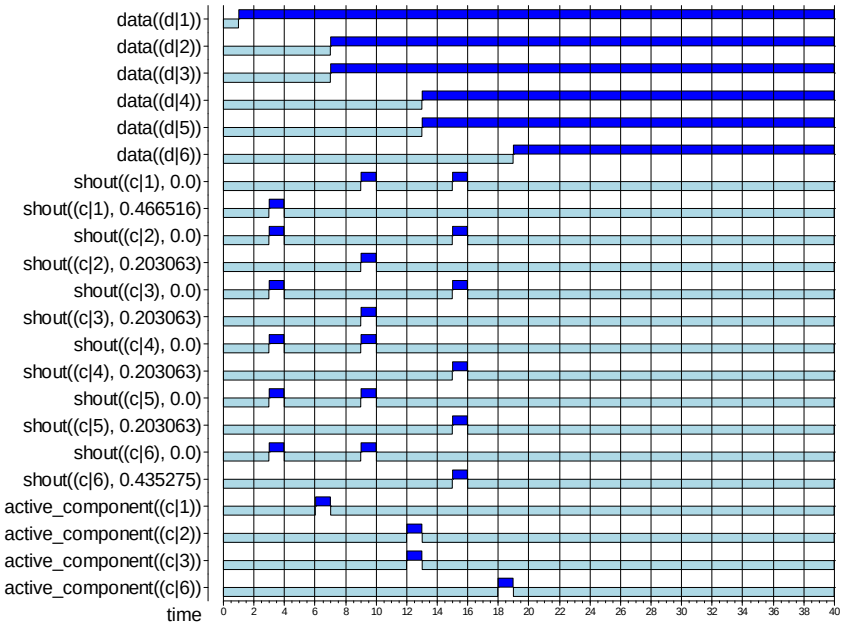

Fig. 12. Simulation Trace - Pandemonium against Test Example 7

5.3 Voting Simulation

The simulation of the voting method uses the same assumptions as the pandemonium method, with one difference: instead of shouting, all components can *vote*. The idea is that each component can vote on only one component (possibly on itself). After all components have voted, the votes are counted, and the component with most votes will be allowed to start processing. To determine on whom they will vote, the components make use of a *voting procedure*. For different variants of the voting method, different voting procedures may be used. In the current model, each component follows the following procedure:

1. if my input is present, and my output is not, then I vote for myself
2. if my input is not present, and this input is generated by one other component, vote for that component
3. if my input is not present, and this input is generated by n>1 other components, vote for one of those components (at random)
4. if my output is present, and this output is used by one other component, vote for that component
5. if my output is present, and this output is used by n>1 other components, vote for one of those components (at random)

6. if my output is present, and this output is used by no other components (i.e., it is part of the goal data), do not vote

Note that this approach assumes a *local* perspective of the components. This means that each component only has knowledge about itself and its direct neighbors. For example, each component knows which other components need the data that it produces as input, but does not know which data these other components produce as output. The ontology used in the simulations is shown in Table 3.

Table 3. Ontology used within the voting simulation model

Relation	Explanation
data: DATA	This specifies that a certain type of data is present in the repository.
vote_for: COMPONENT x COMPONENT	A component votes for a certain (other) component.
active_component: COMPONENT	A component is activated.

Pattern 1

As can be seen in the simulation trace of the first pattern, shown in Figure 13, initially the only data that is present is d1: data(d|1). Based on these data, every component starts voting: vote_for(c|1, c|1), vote_for(c|2, c|1), and vote_for(c|3, c|2). Component C1 receives 2 votes, component C2 receives one vote, and component C3 receives no votes. Thus, component C1 is selected to become active: active_component(c|1). As a result, component C1 creates data d2, which is stored at the repository as well: data(d|2).

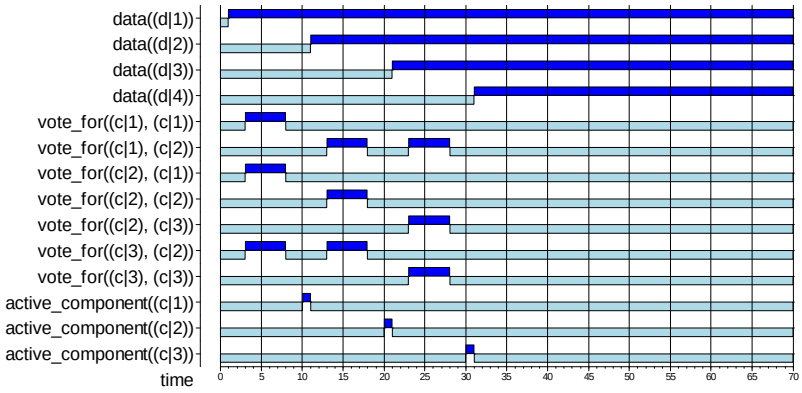

Fig. 13. Simulation Trace - Voting against Test Example 1

Again, every component starts voting: vote_for(c|1, c|2), vote_for(c|2, c|2), and vote_for(c|3, c|2). Component C2 receives all 3 votes and is thus selected to become active: active_component(c|2). Next, component C2 creates data d3, which is stored at the repository as well: data(d|3). Again, every component starts voting: vote_for(c|1, c|2), vote_for(c|2, c|3), and vote_for(c|3, c|3). Component C3 receives 2 votes, component C2 receives one vote, and component C1 receives no votes. Thus, component C3 is selected to become active: active_component(c|3). Eventually, component C3 creates data

d4, which is stored at the repository as well: data(d|4). Since d4 is the goal data, at this point the process terminates.

Pattern 7

Figure 14 depicts the simulation trace that has resulted from applying the voting approach to test example 7. Initially the only data that is present is d1: data(d|1). Based on these data, every component starts voting: vote_for(c|1, c|1), vote_for(c|2, c|1), vote_for(c|3, c|1), vote_for(c|4, c|2). Component C1 receives 3 votes, component C2 receives one vote, and the other components receive no votes. Thus, component C1 is selected to become active: active_component(c|1). As a result, C1 creates data d2 and d3, which are stored at the repository as well: data(d|2), data(d|3). Then again, every component starts voting: vote_for(c|1, c|3), vote_for(c|2, c|2), vote_for(c|3, c|3), vote_for(c|4, c|3). Component C3 receives 3 votes, component C2 receives one vote, and the other components receive no votes. Thus, component C3 is selected to become active: active_component(c|3). Next, component C3 creates data d5, which is stored at the repository: data(d|5). Voting starts again: vote_for(c|1, c|2), vote_for(c|2, c|2), vote_for(c|3, c|5), vote_for(c|4, c|2). Component C2 receives 3 votes, component C5 (which is a specific variant of C4) receives one vote, and the others receive no votes. Thus, component C2 is now selected to become active: active_component(c|2). Component C2 creates data d4, which is stored at the repository: data(d|4). In the next round, the components vote as follows: vote_for(c|1, c|2), vote_for(c|2, c|6), vote_for(c|3, c|6), vote_for(c|4, c|6). Component C6 (which is a specific variant of C4) receives 3 votes, component C2 receives one vote, and the others receive no votes. Consequently, component C6 is selected to become active: active_component(c|6). Eventually, component C6 creates data d6, which is stored at the repository: data(d|6). Since d6 is the goal data, at this point the process terminates.

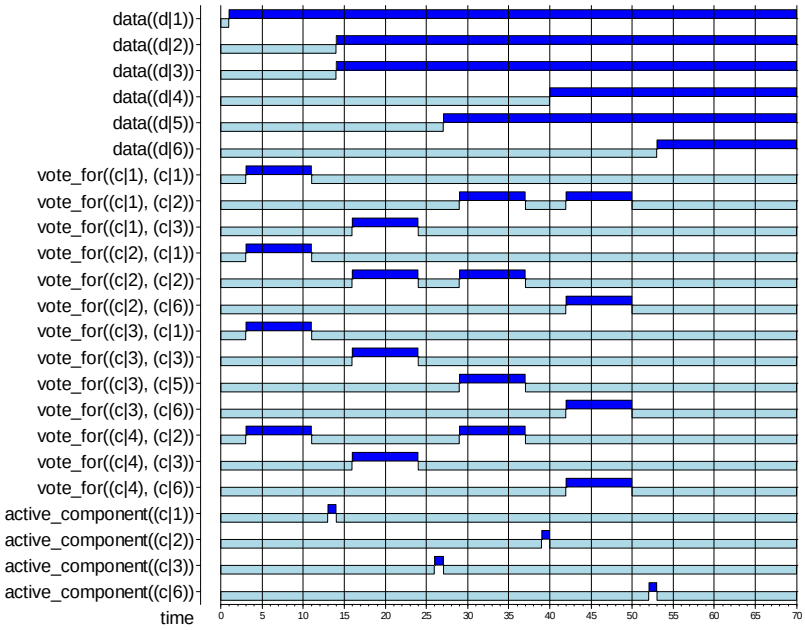

Fig. 14. Simulation Trace - Voting against Test Example 7

6 Evaluation

This section addresses the evaluation of the performance for the different approaches that have been simulated as described above. This evaluation can be performed from multiple perspectives. First of all, the achievement of the goals that have been set for the system are an important evaluation criterion. This criterion is worked out in Section 6.1. Secondly, an element in the evaluation is the efficiency of the approach (see Section 6.2). Finally, patterns can be specified which are (allowed) to occur in the component configurations used as test examples, and it can be checked whether a coordination approach indeed identifies these patterns (see Section 6.3). To enable automated checking of the results of the approaches, a formal specification of these three different types of properties is required. For this purpose, the language TTL introduced in Section 2 is used. After a formal description has been obtained, the automated TTL-checker [7] can be used to see how well the approach performs.

6.1 Successfulness

The first property to be checked is called successfulness. Informally, this property states that in the trace γ all goal data d will eventually be derived. Formally:

successfulness(γ:TRACE) ≡
$\forall$t:TIME, d:DATA [state(γ, t) = goal(d) $\Rightarrow$
$\exists$t2:TIME [t2 $\geq$ t $\wedge$ state(γ, t2) |= data(d)]]

The results of automatically checking this property against the traces that were generated in the simulation show that all approaches eventually find the solution for the examples that have been used. Prerequisite is that there must exist at least one path to the solution.

6.2 Efficiency

Efficiency can be viewed from multiple perspectives. First, one can look at the efficiency of the solution path found by the approach. For now, it is assumed that each component takes an equal amount of time to obtain its output. Therefore, the most efficient solution is simply the solution in which the least amount of components have been activated. Another way to describe efficiency is the efficiency of the approach itself, i.e., the amount of computation time the approach needs to generate a solution. The approach taken in this section is to check whether the shortest activation path is used to reach the goals that are set. For the formalization of this property, it is assumed that the length of the shortest path is known for the particular example being checked:

efficiency(γ:TRACE, shortest_path:INTEGER) ≡
successfulness(γ) $\wedge$ component_activations(γ, shortest_path)

To enable a definition of the amount of activations of a component, first the activation of one component is defined, including its interval:

has_activation_interval(γ:TRACE, c:COMPONENT, tb:TIME, te:TIME) ≡

tb < te ∧ state(γ,te) |≠ activated(c) ∧
[∀t tb≤t<te ⇒ state(γ,t) |= activated(c)] ∧
∃t1<tb [∀t2 t1≤t2<tb ⇒ state(γ,t2) |≠ activated(c)]

An example of a definition for a trace with one component activation is shown below.

component_activations(γ:TRACE, 1) ≡
∃c:COMPONENT, tb:TIME, te:TIME
has_activation_interval(γ, c:COMPONENT, tb:TIME, te:TIME) ∧
[∀c2:COMPONENT, tb2:TIME, te2:TIME
[has_activation_interval(γ, c2:COMPONENT, tb2:TIME, te2:TIME) ⇒ c = c2 ∧ tb = tb2 ∧ te = te2]]

Table 4 shows the outcome of checking the property efficiency in the TTL Checker for the generated traces. A plus indicates that in all generated traces the efficient solution was found; a minus indicates that no efficient solution is found in at least one of the generated traces.

Table 4. Efficiency of the different approaches on the examples

Example	Behavior Networks	Pandemonium	Voting
Sequence	+	+	+
Parallel Split	+	+	-
Synchronization	+	+	+
Exclusive choice	+	+	+
Simple Merge	+	+	+
Multi Choice	-	-	+
Synchronizing merge	-	-	-

For the first five examples, both the behavior networks and the pandemonium always find the optimal path to the solution. For voting, the optimal solution for the parallel split is not always found: apparently, there are situations when this approach is not efficient. This is mainly due to the fact that the voting components have only *local* information. As a result, their voting behavior is not always fully rational. This problem could be solved by allowing a more global perspective for the components.

For the synchronizing merge and the multi-choice (which can be described as the synchronizing merge without component C4), the behavior networks approach fails to find the optimal solution in some cases. For the first, it activates both C2 and C3 whereas only one of the components is required to obtain the goal data. Adapting the parameters of the approach could probably prevent this from occurring. Furthermore, in the synchronizing merge case, both C2 and C3 are activated whereas C4 only needs one input to generate output.

Also the pandemonium model is not always efficient for the multi-choice and synchronizing merge. For the multi-choice, this is the case because the model sometimes generates traces where first C1 is activated, and then C2 and C3 are activated simultaneously. Although this solution is efficient in terms of activation rounds (i.e., only two rounds), it is not efficient in terms of component activations: three components are activated in total, where two activations would have been sufficient (i.e., C1 followed by C2, or C1 followed by C3). For the synchronizing merge, in some cases the same situation occurs as with the behavior networks: sometimes both C2 and C3 are activated simultaneously, whilst only one of them is required.

The voting method however succeeds in always finding the efficient solution for the multi-choice. Here, the aforementioned situation that both C2 and C3 are activated

never occurs, because there is always one component that receives more votes than the others. However, like the other approaches, the voting method is sometimes inefficient with respect to the synchronizing merge. Here, again the same situation occurs as with the behavior networks and the pandemonium: sometimes both C2 and C3 are activated, where only one of them is necessary.

6.3 Specifying and Checking Patterns.

As has been mentioned, certain expected patterns can be specified for component configuration examples, and it can be checked whether these patterns are indeed found by the different approaches. For the test examples used in this document, the component configuration specifications originate from workflow patterns. Therefore, the patterns taken for the test examples are precisely the workflow patterns from which these examples have been derived. Specification of patterns can be done from two perspectives: (1) exhaustively summing up all possible outcomes; (2) specifying the constraints between activation intervals of different components. For the second approach, the interval relations as identified by Allen [2] were used and specified in TTL:

before(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ e1 < b2
meets(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ e1 = b2
overlaps(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ b1 < b2 < e1 < e2
equals(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ b1 = b2 ∧ e1 = e2
starts(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ b1 = b2 ∧ e1 < e2
finished_by(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ b1 < b2 ∧ e1 = e2
contains(b1:TIME, e1:TIME, b2:TIME, e2:TIME) ≡ b1 < b2 ∧ e1 > e2

Below, the selected workflow patterns (pattern 1-7) are specified using TTL expressions. For all patterns, all traces are first summed up in an informal fashion (according to perspective 1 above). After that, the formal TTL expressions specifying the constraints between the activation intervals of the different components are shown (according to perspective 2).

Pattern 1 - Sequence

Possible traces: ABC.

Activation interval constraints in TTL:

∃bA,eA,bB,eB,bC,eC:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bA, eA, bB, eB) ∧
before(bB, eB, bC, eC)

Pattern 2 - Parallel Split

Possible traces: A[BC].

Note: [BC] means either simultaneously or in any order (= in theory, any of the possibilities before, meets, overlaps, equals, starts, finished_by, contains. However, in our

current specifications (both Maes and Pandemonium) we do not handle parallelism. Thus, in the case of [BC] we will only generate the traces BC and CB).

Activation interval constraints in TTL:

∃bA,eA,bB,eB,bC,eC:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bA, eA, bB, eB) ∧
before(bA, eA, bC, eC)

Pattern 3 – Synchronization

Possible traces: [AB]C.

Activation interval constraints in TTL:

∃bA,eA,bB,eB,bC,eC:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bA, eA, bC, eC) ∧
before(bB, eB, bC, eC)

Pattern 4 - Exclusive Choice

Possible traces: AB, AC.

Activation interval constraints in TTL:

[∃bA,eA,bB,eB:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
before(bA, eA, bB, eB)]
∨
[∃bA,eA,bC,eC:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bA, eA, bC, eC)]

Pattern 5 - Simple Merge

Possible traces: AC, BC.

Activation interval constraints in TTL:

[∃bA,eA,bC,eC:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bA, eA, bC, eC)]
∨
[∃bB,eB,bC,eC:TIME
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, C, bC, eC) ∧
before(bB, eB, bC, eC)]

Pattern 6 - Multi Choice

Possible traces: AB, AC, A[BC].

Activation interval constraints in TTL:

parallel_split ∨ exclusive_choice

Pattern 7 - Synchronizing Merge

Possible traces: ABD, ACD, ABCD, ABCD, A B|C D.
Here, "B|C" indicates that B and C are activated simultaneously.

Activation interval constraints in TTL:

[∃bA,eA,bB,eB,bD,eD:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, D, bD, eD) ∧
before(bA, eA, bB, eB) ∧
before(bB, eB, bD, eD)]
∨
[∃bA,eA,bC,eC,bD,eD:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, C, bC, eC) ∧
has_activation_interval(trace1, D, bD, eD) ∧
before(bA, eA, bC, eC) ∧
before(bC, eC, bD, eD)]
∨
[∃bA,eA,bB,eB,bC,eC,bD,eD:TIME
has_activation_interval(trace1, A, bA, eA) ∧
has_activation_interval(trace1, B, bB, eB) ∧
has_activation_interval(trace1, C, bC, eC) ∧
has_activation_interval(trace1, D, bD, eD) ∧
before(bA, eA, bB, eB) ∧
before(bA, eA, bC, eC) ∧
before(bB, eB, bD, eD) ∧
before(bC, eC, bD, eD)]

Table 5 shows whether the algorithms have indeed found the patterns (+) or whether there exists a trace in which the patterns was not found (-).

Table 5. Patterns found by the different approaches

Example	Behavior Networks	Pandemonium	Voting
Sequence	+	+	+
Parallel Split	+	+	+/-
Synchronization	+	+	+
Exclusive choice	+	+	+
Simple Merge	+	+	+
Multi Choice	+	+	+
Synchronizing Merge	+	+	+

As indicated in the table, the behavior networks, pandemonium, and voting approaches always find the patterns that have been identified. In the parallel split case, the success of the voting approach however is debatable. The reason for this is that besides the expected patterns (A[BC]) also patterns such as A-B-B-C appear. According to personal communication with van der Aalst this is however not a violation of the pattern. Following his perspective, a trace satisfies a pattern when the components as prescribed by the patterns occur being active in the trace in the specified sequence. It is however allowed for other components (either a different component or activation of the same component at another time point) to be active within the same trace. For checking the more strict version (i.e. exactly the prescribed sequence without other activations) a *closed world assumption* version of the property has been specified as well.

7 Discussion

To conclude, this paper presented a formal methodology to evaluate and compare the performance of different coordination approaches. The methodology comprises the creation of simulation models for the coordination approaches, the execution of simulation experiments of these models applied to test examples, and their automated evaluation against specified requirements. In a specific case study, the methodology was used to evaluate three well-known coordination approaches from the literature. During this case study, the simulation approach turned out quite beneficial. Within a reasonable time, a nontrivial number of approaches have been tested against a nontrivial number of cases: 3 x 7 = 21 combinations have been explored. Furthermore, the automated checks of dynamic properties against generated traces have turned out useful to evaluate the simulations for the different approaches against requirements. Finally, an existing library of workflow patterns [1] turned out an appropriate source for cases to be explored, although their specification also needs to cover data flow aspects. It was not too difficult to add such data flow aspects.

Concerning the specific case study performed, the voting, pandemonium and behavior networks approach have been thoroughly evaluated with respect to a number of relevant performance indicators, namely successfulness, efficiency, and pattern checks. All approaches turned out effective in finding the solution in all cases. However, none of the approaches is always efficient for all patterns. The behavior networks and pandemonium approaches perform equally well; they succeed for the "simple" cases and sometimes fail to be efficient for the two complicated cases (i.e. multi-choice and synchronizing merge). Surprisingly, the voting approach always finds the most efficient solution for one of the complicated cases, namely the multi-choice. It does however fail in the rather trivial case of the parallel split. All approaches also find the patterns specified for each of the component configuration examples.

All in all, when comparing the different coordination approaches, the performance based on the criteria specified above is almost similar. The way in which they find the component activation sequences is however completely different. The behavior networks approach needs a global overview of the system: it needs to know for each component what data it requires as input and what data it generates as output. Such a global view might not always be available or might be inconvenient. On the other hand, for the pandemonium a completely local view is sufficient: each component only needs information about its own input and output data. In between is the voting approach, which needs information about itself and its direct neighbors. When comparing the approaches on required computation time, the behavior networks approach takes far more computation time than the other approaches. This has two causes: first, due to the fact that all global information is used within the approach, it has a lot more information to take into consideration. Second, both for the voting and pandemonium approach the calculations per component can be performed in parallel, which can not be done in the behavior networks approach.

Work related to the approach presented in this paper can, first of all, be found in the field of action selection mechanisms (also called behavior coordination mechanisms) in robotics. Pirjanian [22] presents an overview of several mechanisms used in that particular field, including a classification of these mechanisms. He

identifies two main streams: arbitration and command fusion. In the arbitration approach, one behavior is arbitrarily selected from a group of competing ones, giving it the ultimate control. For command fusion mechanisms however, recommendations are combined from multiple behaviors to form a control action that represents their consensus. The behavior networks approach as presented by Maes [19] is an example of an arbitration mechanism, whereas both voting and the pandemonium model can be placed in the command fusion category. Tyrrell [26] presents a comparison between several mechanisms for action selection, using a simulator of an animal world. The comparison approach is however not formal like the approach presented in this paper. Furthermore, the framework for comparison is not generic, but developed for a specific case study, making it hard to generalize the results obtained. Another related field can be found within multi-agent systems, where coordination models play an essential role to ensure a proper functioning of the system as a whole. These coordination models address types of interactions and agreements between the different agents that were not considered in this paper. For a comparison between different coordination models in agent systems, see for example [8]. Concerning other related work, coordination models and languages for interfacing between components often focus on how different components within a software system can interact, see for example [3]. Due to the assumption of data being available and interpretable for all components, these component interaction models have not been considered in this paper, but can easily be incorporated in the methodology.

The methodology presented in this paper is supported by two software environments: the LEADSTO environment for simulation [6], and the TTL environment for verification of properties [7]. For simulation, various other approaches exist, such as the Dynamical Systems Theory [23], Executable Temporal Logic [4], PLC automata [11], qualitative reasoning (see, e.g., [12]), and stochastic pi-calculus (as used in [14]). For verification of properties, alternative approaches are standard temporal languages such as LTL and CTL [15], and calculi like the situation calculus [24] and the event calculus [17]. See, respectively, [6] and [7] for an extensive comparison of LEADSTO and TTL with these approaches.

Finally, the work as reported has led to a number of ideas for further research. While the specific coordination approaches borrowed from other disciplines were found to have value, no attempts have been made yet to come up with refinements, extensions or improvements of these approaches, or, inspired by these approaches, to design completely new (and possibly better) approaches. Some possible future extensions are allowing *preference* for certain components, allowing a *dynamic environment*, and enabling the components to process *partial data*.

Acknowledgements

This work has been performed as part of a project funded by Force Vision, the software development department for the Royal Netherlands Navy. Moreover, the authors are grateful to Egon van den Broek, Rob Duell, Andy van der Mee, and Bas Vermeulen for various fruitful discussions.

References

[1] Aalst, W.M.P. van der, Hofstede, A.H.M. ter, Kiepuszewski, B., and Barros, A.P. Workflow Patterns. QUT Technical report FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002.

[2] Allen, J. F. Maintaining knowledge about temporal intervals. In: *Communications of the ACM*, 26, 1983, pp. 832-843.

[3] Arbab, F. Reo: A Channel-based Coordination Model for Component Composition, *Mathematical Structures in Computer Science*, Cambridge University Press, Vol. 14, No. 3, pp. 329-366, 2004.

[4] Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. *The Imperative Future: Principles of Executable Temporal Logic*, John Wiley & Sons, 1996.

[5] Bosse, T., Hoogendoorn, M., and Treur, J. Coordination Approaches for Complex Software Systems. Technical report 06-04ASRAI, Vrije Universiteit Amsterdam, Amsterdam, 2006. URL: http://hdl.handle.net/1871/9195.

[6] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., et al. (eds.), *Proc. of the Third German Conference on Multi-Agent System Technologies, MATES'05*. Lecture Notes in Artificial Intelligence, vol. 3550. Springer Verlag, 2005, pp. 165-178.

[7] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A, and Treur, J. A Temporal Trace Language for the Formal Analysis of Dynamic Properties. Technical Report, Vrije Universiteit Amsterdam, Department of Artificial Intelligence, 2006.

[8] Bourne, R., Shoop, K., and Jennings, N. Dynamic evaluation of coordination mechanisms for autonomous agents. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 155-168. Springer, 2001.

[9] Ciancarini, P. and Wiklicky, H. Proceedings of the Eighth International Conference on Coordination Models and Languages, Coordination'06. Lecture Notes in Computer Science, vol. 4038. Springer Verlag, 2006.

[10] Dehnert, J. and Aalst, W.M.P. van der. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, vol. 13, issue 3, 2004, pp. 289–332.

[11] Dierks, H. PLC-automata: A new class of implementable real-time automata. In M. Bertran and T. Rus, editors, Transformation-Based Reactive Systems Development (ARTS'97), volume 1231 of Lecture Notes in Computer Science, pages 111-125. Springer-Verlag, 1997.

[12] Forbus, K.D. *Qualitative process theory*. Artificial Intelligence, vol. 24, no. 1-3, 1984, pp. 85-168.

[13] Franklin, S. Artificial Minds, MIT Press, Cambridge Massachusetts, 1997.

[14] Gardelli, L., Viroli, M., Omicini, A.: On the Role of Simulations in Engineering Self-Organizing MAS: the Case of an Intrusion Detection System in TuCSoN. In: *3rd International Workshop "Engineering Self-Organising Applications" (ESOA)*, 2005, pp. 161-175.

[15] Goldblatt, R. *Logics of Time and Computation*, 2nd edition, CSLI Lecture Notes 7, 1992.

[16] Jackson, J.V. Idea for a Mind, *SIGGART Newsletter*, no 181, pp. 23-26, 1987.

[17] Kowalski, R. and Sergot, M.A. A logic-based calculus of events, *New Generation Computing*, 4, 1986, pp. 67-95.

[18] Lindsay, P. H., and Norman, D. A. *Human Information Processing: An Introduction to Psychology*. Academic Press, Inc., New York, 1977.

[19] Maes, P. How to do the right thing. *Connection Science*, 1989. 1(3): pp. 291-323.

[20] Ordeshook, P. *Game theory and political theory: An Introduction*. Cambridge: Cambridge University Press, 1986.

[21] Papadopoulos, G. and Arbab, F. Coordination Models and Languages. *Advances in Computers*, vol. 46: The Engineering of Large Systems. Academic Press, New York, 1998.
[22] Pirjanian, P. *Behavior coordination mechanisms -- state-of-the-art*. Technical Report IRIS-99-375, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California, October 1999.
[23] Port, R.F. and Gelder, T. van (eds.) *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass, 1995.
[24] Reiter, R. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, Cambridge MA: MIT Press, 2001.
[25] Selfridge, O. G. Pandemonium: a paradigm for learning in mechanization of thought processes. In *Proceedings of a Symposium Held at the National Physical Laboratory*, pages 513-526, London, November 1958.
[26] Tyrrell, T. *Computational Mechanisms for Action Selection*, PhD thesis, University of Edinburgh, 1993.

Chapter 14

A Specification Language for Coordination in Component-based Software Systems

A Specification Language for Coordination in Component-based Software Systems

Tibor Bosse, Mark Hoogendoorn, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1018a, 1081 HV Amsterdam, The Netherlands
{tbosse, mhoogen, treur}@cs.vu.nl

Abstract. This paper introduces a coordination specification language which is able to handle both pre-specified ways of expressing coordination in component-based software systems, as well as novel more flexible and generic ways of specifying coordination. An iterative process consisting of several steps was taken to define this language and create simulations of such coordination approaches. First of all, useful language elements were defined, after which example coordination approaches were specified using this language. In cases where more language elements were needed to enable specification of such a coordination approach, these have been added to the language. After that, the coordination approaches were simulated using an executable temporal logic and tested using particular test cases. Finally, an evaluation of the coordination approaches was performed by means of formal verification.

1 Introduction

As component-based software systems become increasingly complex, so does the specification of coordination for such a system. In addition, software systems can be dynamic, in the sense that components dynamically enter or leave the system. As a result, for such complex dynamic systems, exhaustively specifying the activation sequences of components, for example in a centralized manner, which is the approach usually taken in more traditional approaches, is no longer an option. This is due to the fact that the components that are available for computation (and their ideal activation sequence) are not known in advance. Furthermore, it is not always desirable to have coordination information available in a global or centralized manner.

As a consequence, more generic and flexible coordination approaches have been proposed, including pandemonium models [13], behavior networks [11], and voting models [12]. In contrast to the more traditional approaches, which are based on qualitative, logical specifications, such alternative methods usually involve quantitative, numerical calculation methods, and often work in a more decentralized manner. In [3] a methodology for the evaluation and comparison of such coordination approaches has been proposed, and a number of such approaches have been evaluated.

The transition of a traditional way of specifying coordination by pre-defined coordination sequences to such more generic and flexible coordination strategies is

not a trivial matter. Current coordination specification languages as, for example, described in [5; 8] are typically unable to specify such coordination approaches. This implies that the transition not only entails moving towards a new approach for coordination specification, but also towards a more expressive coordination language, allowing, for example, more generic types of expressions with variables and quantifiers, and numerical relationships. To address this problem, this paper proposes a coordination language that can express both pre-defined coordination sequences, as well as generic, flexible coordination approaches.

In order to come to such a language, several steps have been performed iteratively. In particular, useful language elements have been identified, example coordination approaches (including the pre-specified, central approaches) have been specified in terms of this language, and the language has been extended in case more elements were found necessary to specify the coordination approach. The example coordination approaches have thereafter been simulated using a temporal logic, and tested using specific test cases that have been identified. Finally, the different coordination approaches have been evaluated by formal verification.

This paper is organized as follows. Section 2 introduces the viewpoint taken in this paper towards coordination in component-based systems. The specification of the language makes a distinction between the coordination level of a system, and the data level at which the actual processing is being done. In Section 3 a reified temporal order-sorted predicate logic language is introduced which allows for the specification of such coordination. In Section 4 this language is shown to be a specialization of a language called TTL, for which both simulation and verification tools are available. Furthermore, Section 4 shown how simulation and verification properties can be specified using the coordination language. Section 5 presents a test example to be used to test such coordination approaches. Section 6 presents several coordination approaches that have been specified using the coordination language and Section 7 shows simulation results based upon these approaches. A formal evaluation of the approaches is presented in Section 8, and finally, Section 9 is a discussion.

2 Viewpoint on Coordination in Component-Based Systems

In Figure 1 the viewpoint taken in this paper on coordination in a component based system is shown. It is shown how from a conceptual perspective the processing done by a component for coordination purposes can be distinguished from the actual processing of data to fulfill a coordination-independent computation. On the top level, above the dashed line, the coordination part of the entire software system is shown. On the coordination level, reasoning takes place about the coordination within the component-based system. This process can be either a centralized or a distributed process. In the latter case, the distribution can for example follow the distribution of the components of the system. Input for this coordination process is coordination information received from the various components and links, whereas the output of this coordination level is coordination information for the components and links within the component-based system. On the data level, the components and links themselves are shown. Each component has two input layers: One for coordination

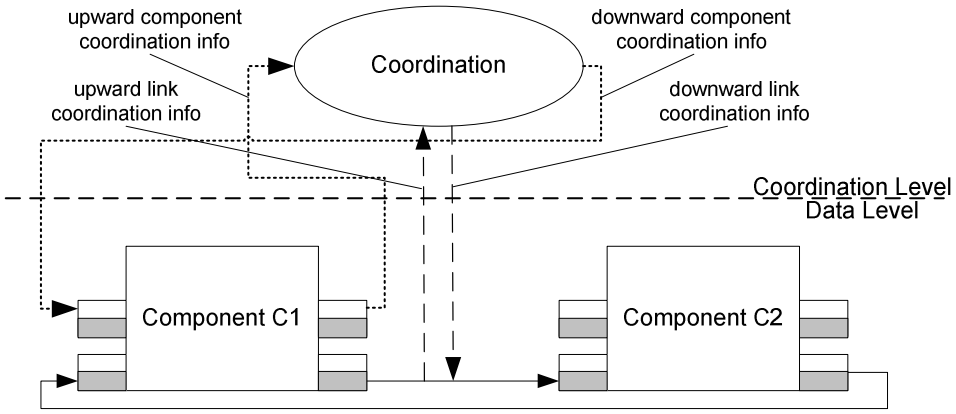

Fig. 1. Different levels of coordination within a component based system

information (the upper square at the left side of the component), and one for data information (the lower square). Furthermore, output is generated on both levels as well, depicted by the squares at the right side of a component. Each link on the data level connects the data output of a component to the data input of another component. Furthermore, each link can receive coordination information, and generate it. Note that this is a conceptual picture at an abstract level. There is no commitment in how far the coordination reasoning process itself is centralized or also distributed over the components. The degree to which it is central or decentralized is a parameter that is left open in this conceptual picture.

3 Coordination Language

Given the viewpoint presented in Section 2, this section presents the actual coordination specification language. The language is a reified temporal order-sorted predicate logic language; cf. [6;7] This means that state ontologies are defined to express state terms, and on top of that a time ontology is used so that by full predicate logic expressivity temporal statements can be specified. Figure 2 provides an overview of the time and state ontologies included in this language.

The main distinction made is in state ontologies for characteristics (static) and for states (dynamic). Moreover, ontologies are distinguished by whether they address coordination information or data information. Furthermore, ontologies are related to their use within input, output or internal states. Finally, in addition to the state ontologies, a generic time ontology is used, to specify temporal relations, and a generic (support) ontology is included for elementary relations and functions such as ordering and calculations for numbers.

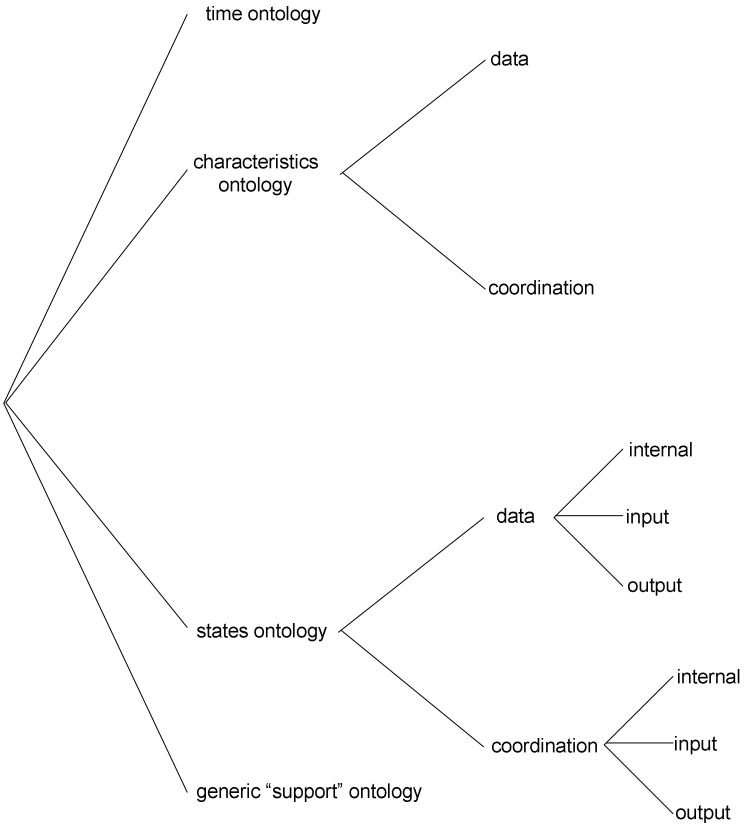

Fig. 2. Partitioned coordination ontology

3.1 Time Ontology

The temporal aspect is important in component coordination, because it sometimes is necessary to express at which time point a particular coordination action was undertaken, how many times in a given interval a specific event has occurred, and whether the reaction of a component has been given in due time in relation to the current time, et cetera. The sorts used in this time ontology are specified in Table 1.

Table 1. Sorts used for time ontology

Sort	**Description**
TIME	Sort indicating time.
REAL	Sort for real numbers.
STATPROP	A sort for terms indicating state properties
STATE	A sort for states
TRACE	A trace indicates a time ordered sequences of states.

The relations specified within the time ontology using the sorts described in Table 1 are shown in Table 2.

Table 2. Relations of the time ontology for component coordination

Relation	**Description**
current_time: TIME	Indicates the current time point.
< (precedes): TIME x TIME	The time point associated with the first argument is mapped to a real value smaller than the one associated with the second argument.
+ : TIME x REAL x TIME	The time point associated with the first argument plus the real value specified in the second argument is the time point specified in the last element.
state: TRACE x TIME → STATE	This function indicates the state of a trace at a specific time point
held_once_since: STATPROP x TIME x TIME x TRACE	PROP has been at least once true in the interval between the first TIME point and the second TIME point in the specified trace.
holds_at:STATPROP x TIME x TRACE	PROP holds at the specified time point within the specified trace.
holds_during: STATPROP x TIME x TIME x TRACE	PROP holds during the time period specified within the specified trace.
holds_just_before: STATPROP x TIME x TRACE	PROP holds just before the time point specified within the specified trace.
holds_just_at: STATPROP x TIME x TRACE	PROP just holds at the time point specified within the specified trace whereas before it was false.

3.2 Ontology of Static Coordination Characteristics

The static relations between components/links, which characterize the immutable relationships between them (for instance, the architectural connections at the data level) or coordination-relevant properties derived from a set coordination states, are grouped in the so-called ontology of static coordination characteristics. First of all, Table 3 shows the sorts that have been used in addition to the sorts specified in Table 1.

Table 3. Sorts used in the static coordination characteristics ontology

Sort	Description
COMPONENT	A component within the component-based system.
LINK	A link between two components within the component-based system.
ARCHITECTURAL_OBJECT	COMPONENT ∪ LINK
INFO_TYPE	A type of information, which can possibly be a grouping of multiple other information types. Furthermore, it can contain information elements that specify a specific value of an element within this information type. An example of an information type could be 'temperature', containing information elements that specify a temperature of 20^{o}C, etc.
INFO_ELEMENT	A specific element of information, such as explained under INFO_TYPE

Using these sorts, Table 4 presents the static coordination level characteristics. These relations can be used on the coordination level to decide upon a component to be used for a particular task. For instance, in case accuracy is required, a very accurate components needs to be chosen. Note that such characteristics can themselves also be dynamic, for instance an average accuracy over time. For now however it is assumed that these are static. Finally, this is by no means an exhaustive list of characteristics, it can be extended with particular characteristics of importance within particular domains.

Table 4. Static coordination-level characteristics

Relation	Description
responsiveness: REAL	Indicates the REAL value (in the interval [0,1]) representing the probability that a component successfully activates.
accuracy: REAL	Indicates the (expected) accuracy of a component (value in the interval [0,1]), i.e. the distance between the computed outputs of the component and the ideal outputs. Is a measure of the correctness of the output produced by a component.
estimated_processing_time: REAL	Indicates the estimated (maximum) processing time required by a component in order to produce outputs from available inputs.
estimated_flops: REAL	Indicates the amount of flops a component needs for its computations (which is a measure per time unit).
estimated_memory_usage: REAL	Indicates the estimated critical resource (memory) usage of the component.

Besides coordination-level characteristics, an ontology for expressing data-level characteristics has been defined as well, and is shown in Table 5. Note that these relations can be applied to components as well as links, unless specifically mentioned otherwise.

Table 5. Static data-level characteristics

Relation	Description
includes: INFO_TYPE x INFO_TYPE	The INFO_TYPE specified includes the INFO_TYPE specified as the second argument.
input_output_type_relation: INFO_TYPE x INFO_TYPE	The information type on the input specified in the first parameter INFO_TYPE is used to generate the output of the type specified in the second parameter INFO_TYPE.
link_type_relation: INFO_TYPE x COMPONENT x INFO_TYPE x COMPONENT	There exists a communication link from the first INFO_TYPE of the output of the first COMPONENT, to the input of the second INFO_TYPE of the second COMPONENT.
input_information_type: INFO_TYPE	The INFO_TYPE is an input information type.
output_information_type: INFO_TYPE	The INFO_TYPE is an output information type.
has_type: INFO_ELEMENT x INFO_TYPE	The information element specified has the information type specified as second argument.

3.3 Ontology of Dynamic Coordination States

Besides an ontology for static coordination states, a dynamic coordination ontology has been defined as well, that changes as the system functions. Again, the division is made between the coordination and data level ontology. In addition, in the tables it is shown of what type the relation is: input, output, or internal for the component. Table 6 presents the additional sorts that have been used within the relations.

Table 6. Sort used for dynamic coordination state ontology

Sort	Description
SIGN	A SIGN indicates whether an INFO_ELEMENT holds (indicated by 'pos'), or whether it does not hold ('neg').
SIGNED_INFO_ELEMENT	An INFO_ELEMENT grouped with a SIGN.
SIGNED_INFO_ELEMENT_CONJUNCTION	A conjunction of SIGNED_INFO_ELEMENTs
TARGET_QUALIFIER	An identifier for a target.
TARGET_EXPRESSION	A target qualifier specifies a target that has been set for an ARCHITECTURAL_OBJECT. For instance to derive all possible information, see Section 3.3.1 for more information.
FOCUS	An identifier of a focus that has been set for a particular architectural object.

The relations that have been specified based upon these sorts are specified in Table 7 and Table 8, where the former gives the coordination-level relations whereas the latter presents the data-level relations. The right column in the Table indicates whether this concerns an input(i), output(o), or internal (int) type of relation.

Table 7. Coordination-level relations for dynamic coordination states

Relation	**Description**	**Type**
awake	Information can be processed	I/O
asleep	Information cannot be processed.	I/O
is_input_focus: FOCUS	The input focus set defined by FOCUS is currently in use.	I/O
is_included_in_focus: INFO_TYPE x FOCUS	The specified INFO_TYPE is part of the focus set defined by FOCUS.	I/O
is_output_focus: FOCUS	The output focus set defined by FOCUS is currently in use.	I/O
info_type_in_focus_has_ qualifier: INFO_TYPE x FOCUS x TARGET_QUALIFIER xTIME	A target identified by TARGET_QUALIFIER has been set for the information type within the focus specified, and this needs to be achieved by the deadline indicated by time point TIME.	I/O/Int
info_type_in_focus_has_ qualifier: INFO_TYPE x FOCUS x TARGET_QUALIFIER x REAL	A target identified by TARGET_QUALIFIER has been set for the information type within the focus specified, and this needs to be achieved within a duration specified by REAL.	I/O/Int
has_expression: TARGET_QUALIFIER x TQ_EXPRESSION	The target qualifier identified by TARGET_QUALIFIER has a particular expression, specified in TQ_EXPRESSION.	I/O/Int
open_to_input_update: INFO_TYPE	Listening to data updates of the types specified in INFO_TYPE.	I/O
busy	The component is currently busy with processing	O
non_busy	The component is not busy with processing.	O
succeeded_with_output_ given_input: SIGNED_INFO_ELEMENT_ CONJUNCTION x SIGNED_INFO_ELEMENT_ CONJUNCTION	The component has succeeded generating all output of the specified SIGNED_INFO_ELEMENT_CONJUNCTION given the input specified in SIGNED_INFO_ELEMENT_CONJUNCTION.	O
failed_with_output_given_ input: SIGNED_INFO_ELEMENT_ CONJUNCTION x SIGNED_INFO_ELEMENT_ CONJUNCTION	The component could not generate all output of the specified SIGNED_INFO_ELEMENT_CONJUNCTION given the input specified in SIGNED_INFO_ELEMENT_CONJUNCTION.	O
available	Indicates whether an ARCHITECTURAL_OBJECT is present in the scenario, and can accept inputs.	O/Int

Table 8. Data-level relations for dynamic coordination states

Relation	Description	Type
currently_needed_input_ for_output: INFO_TYPE x INFO_TYPE	The set of input types in INFO_TYPE is still expected in order to produce an output element of the type INFO_TYPE	Int
accuracy_of_information: SIGNED_INFO_ELEMENT x REAL	The accuracy by which the specified SIGNED_INFO_ELEMENT outputted by COMPONENT is given by the REAL value.	Int
input_provides_output:SIGNED_INFO_ELEMENT_ CONJUNCTION x SIGNED_INFO_ELEMENT_ CONJUNCTION	The first argument SIGNED_INFO_ELEMENT_CONJUNCTION, corresponding to a set of inputs, is used to produce the output indicated by the second SIGNED_INFO_ELEMENT_CONJUNCTION.	I/O
entails: SIGNED_INFO_ELEMENT_ CONJUCTION x TQ_EXPRESSION	The SIGNED_INFO_ELEMENT_CONJUCTION entails that the target specified by TQ_EXPRESSION has been reached.	I/O
information_at_input: SIGNED_INFO_ELEMENT	The SIGNED_INFO_ELEMENT is available at the input.	I
information_at_output: SIGNED_INFO_ELEMENT	The SIGNED_INFO_ELEMENT is available at the output.	O

The expression of targets for particular components, or the system as a whole is presented below.

3.3.1 Target Qualifier Specification

Target qualifier specifications are statements that can be used for assessment of output states. Simple examples of such (output) state properties are:

- at least one signed information element related to a given information type is available
- if a signed information element SIE1 is available at the output, then also signed information element SIE2 is available
- of all non-refinable information types at least one info element with a positive sign needs to be derived.

The language to express target qualifiers is a sublanguage of the state language, based on order-sorted logic.

Atoms used

Within target qualifiers atoms are used that indicate that a certain signed information element is available at the output, expressed using the predicate information_at_output. Moreover, atoms can be used based on predicates has_info_element, has_type, includes, has_sign to express relations between signed information elements, information types, information elements and signs. Furthermore, the predicate is_information_type_in to express relations to a given output focus can be used.

Connectives used
On top of such atoms target qualifier expressions can be built using connectives such as conjunctions (∧, AND), implication (→, IMPLIES), negation (¬, NOT), disjunction (∨, OR) and quantifiers (∀, FORALL; ∃, EXISTS) can be used.

Successfulness with respect to a target qualifier has a simple definition, just expressing that the output state satisfies the target qualifier expression:

is_satisfied(t :TIME, γ:TRACE, T:TQ_EXPRESSION) ≡ state(γ, t) |= T

Sometimes it may be useful to have separate names for target qualifier expressions. These names can be related to the expressions by a predicate has_expression: TARGET_QUALIFIER x TQ_EXPRESSION. The relation

state(γ, t) |= T ⇔ ∃TQE:TQ_EXPRESSION [has_expression(T, TQE) & state(γ, t) |= TQE]

can be used to determine for such a name when the target qualifier is satisfied.

Examples of specific target qualifiers
As an example, the target qualifier 'any' for a particular information type is expressed as follows:

At least for one information element in this information type, output information is available for a signed information element related to the information element.

This can be formally expressed as follows:

any(I: INFO_TYPE) ≡
∃IE:INFO_ELEMENT, SIE:SIGNED_INFO_ELEMENT
[has_type(IE, I) ∧ has_info_element(SIE, IE) ∧ information_at_output(SIE)]]

3.4 Generic Support Ontology

This ontology is not explicitly shown, but contains elements that eases the specification of particular constructs.

Note that all predicates mentioned in this section are represented as *relations*. As such, they can be used at the *object level* of components (i.e., both at the object level of the coordination layer of components, and at the object level of the data layer of components). However, the language should also allow to make statements about these relations, which can be used at the *meta level* of components (i.e., both at the meta level of the coordination layer of components, and at the meta level of the data layer of components), and at the *coordination level*. To this end, a mechanism is needed to translate the relations mentioned in this section to *functions*, which have the same arguments as the corresponding relations, but that also have a destination sort that can be used in meta-statements. For this purpose, the following construct is proposed:

meta_description
relations : DYN_OBJECT_ELEMENT

Here, relations is an information type containing all relations as specified in Table 1 - 8. Furthermore, elements of the sort DYN_OBJECT_ELEMENT can be used in meta-statements. To make this possible, the following meta-predicates are proposed:

selected_control_aspect_for :	DYN_OBJECT_ELEMENT x COMPONENT
monitored_control_aspect_for :	DYN_OBJECT_ELEMENT x COMPONENT
control_aspect :	DYN_OBJECT_ELEMENT
data_aspect :	DYN_OBJECT_ELEMENT

Some examples of meta-statements that can be constructed using these meta-predicates are the following:

selected_control_aspect_for(awake, C1)
monitored_control_aspect_for(accuracy(0.8), C1)
control_aspect(awake)
control_aspect(accuracy(0.8))
data_aspect(input_information_type(d1))

The idea is that selected_control_aspect_for can be used at the output interface of the coordination level, and monitored_control_aspect_for can be used at the input interface of the coordination level. Moreover, control_aspect can be used at the meta level of the coordination input interface of components and at the at the meta level of the coordination output interface of components, and data_aspect can be used at the meta level of the data input interface of components and at the at the meta level of the data output interface of components. To translate statements of the form control_aspect and data_aspect to (and from) object-statements that can be used within the components, upward and downward reflection is used.

4 Expressing Dynamic Properties in the Coordination Language

The coordination language that has been defined in Section 3 can be used both to specify executable simulation models of coordination approaches and to specify properties for verification of traces of systems, for example, simulation results. In this section it is briefly discussed how this can be done (Section 4.2), and how the coordination language relates to the more general languages TTL and LEADSTO (Section 4.1), thus enabling the use of software tools for simulation and verification that have been developed for these languages.

4.1 Relating the Coordination Language to TTL and LEADSTO

In this subsection it is shown how the coordination language can be considered a specialization of the Temporal Trace Language (TTL) [9] by adding certain state ontologies and definable temporal predicates. For the language TTL, verification tools are available that can as a result be used for the coordination language as well.

Moreover, it is shown how an executable sublanguage of the coordination language can be considered a specialization of the language LEADSTO by adding pre-specified state ontologies. As a result, the simulation tools available for the LEADSTO language can be used as well.

In TTL, ontologies for states are formalized as sets of symbols in sorted predicate logic. For any state ontology Ont, the ground atoms form the set of *basic state properties* BSTATPROP(Ont). Basic state properties can be defined by nullary predicates (or proposition symbols) such as incident, or by using n-ary predicates (with n>0) like observes(amount_of_casualties, 7). The *state properties* based on a certain ontology Ont are formalized by the propositions (using conjunction, negation, disjunction, implication, and quantification) made from the basic state properties and constitute the set STATPROP(Ont). For the coordination language the pre-specified state ontologies as presented in Section 3 are available.

In order to express dynamics in TTL, important concepts are *states*, *time points*, and *traces*. A *state* S is an indication of which basic state properties are true and which are false, i.e., a mapping S: BSTATPROP(Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont). Moreover, a fixed *time frame* T is assumed which is linearly ordered. Then, a *trace* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by TRACES(Ont).

The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation, indicated by the infix predicate |=, comparable to the Holds-predicate in the Situation Calculus. Thus, state(γ, t) |= p denotes that state property p holds in trace γ at time t. Likewise, state(γ, t) $|\neq$ p denotes that state property p does not hold in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted predicate logic, using the usual logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and the quantifiers $\forall$, $\exists$ (e.g., over traces, time and state properties). The set DYNPROP(Ont, γ) is the subset of DYNPROP(Ont) consisting of formulae with γ occurring in which is either a constant or a variable without being bound by a quantifier. Note that the predicates of the time ontology introduced in Section 3 can be defined in terms of atoms of the form state(γ, t) |= p. For example,

holds_at(p, t, γ) $\equiv$ state(γ, t) |= p

holds_during(p, t1, t2, γ) $\equiv$ $\forall$t [t1$\leq$t<t2 $\Rightarrow$ holds_at(p, t, γ)]

To model direct temporal dependencies between two state properties, not the expressive language TTL, but the simpler *leads to* format is used. This is an executable format that can be used to obtain a specification of a simulation model in terms of dynamic properties. The format is defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

if *state property* α *holds for a certain time interval with duration* g,
then *after some delay (between* e *and* f*) state property* β *will hold for a certain time interval of length* h.

For a precise definition of the *leads to* format in terms of the language TTL, see [9]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically in a causal graph like style.

4.2 Specification of Dynamic Properties

The coordination language is defined as a reified temporal order-sorted predicate logic language built on top of an order-sorted predicate logic language for state properties. This means that dynamic properties are expressed as temporal order-sorted predicate logic expressions for temporal patterns based on temporal atoms using relations from the time ontology. Within such temporal atoms, terms are used to indicate relevant state properties. As an example, consider the following expression:

$$\forall t1, t2$$
$$[t2 \geq t1 + d \ \& \ \text{holds\_during}(\text{monitored\_control\_aspect\_for}(\text{accuracy}(0.2), c1), t1, t2, \gamma)]$$
$$\Rightarrow \exists t3 \ [\ t2 \leq t3 \leq t2 + 3 \ \& \ \text{holds\_at}(\text{selected\_control\_aspect\_for}(\text{awake}, c2), t3, \gamma)\ ]$$

This statement expresses that if during any time interval with duration d the accuracy of component c1 is 0.2, then an alternative component c2 should become awake within 3 time units. Note that the temporal structure of this formula is:

$$\forall t1, t2$$
$$[t2 \geq t1 + d \ \& \ TA1(t1, t2, \gamma)$$
$$\Rightarrow \exists t3 \ [\ t2 \leq t3 \leq t2 + 3 \ \& \ TA2(t3, \gamma)\ ]$$

with temporal atoms

$$TA1(t1, t2, \gamma) \equiv \text{holds\_during}(p1, t1, t2, \gamma)$$
$$TA2(t3, \gamma) \equiv \text{holds\_at}(p2, t3, \gamma)$$

within these temporal atoms, the following terms indicating state properties:

$$p1 \equiv \text{monitored\_control\_aspect\_for}(\text{accuracy}(0.2), c1)$$
$$p2 \equiv \text{selected\_control\_aspect\_for}(\text{awake}, c2)$$

This shows how temporal atoms are built on top of the state ontologies, and how a dynamic property can be obtained by substituting such temporal atoms in a temporal template representing a (generic) temporal pattern.

The dynamic property can also be expressed in the executable LEADSTO format in the following manner:

monitored_control_aspect_for(accuracy(0.2), c1)

$\twoheadrightarrow_{3,\,3,\,d,\,1}$ selected_control_aspect_for(awake, c2)

More extensive examples of dynamic properties expressed in the coordination language can be found in subsequent sections.

5 Test Example

In order to investigate for particular coordination approaches whether they can be expressed using the coordination language introduced in this paper, a number of test examples are used. Based upon such a test example, these coordination approaches can be shown to work by means of simulations using this given test example.

As an illustrative test example used in this paper, the setting specified in Figure 3 is used. Three regular components C1, C2, and C3 are present within the system. The specific function which is performed by the specific component is shown inside the box of the component. As can be seen, component C1 can actually perform two operations, namely calculate the value of information type d2 based upon the value of d1 and furthermore, and the value of d4 based upon the value of d1 and d3. The components themselves have characteristics as well (as also identified in the coordination language). C1 has an estimated processing time of 4 whereas C2 requires 10. C3 has an estimated processing time of 1. Besides components, links are present between the components, indicated by arrows, and an environment is present as well, which outputs the information type d1. The overall goal of the system is set to outputting an element of the information type d4 which needs to be achieved within 12 time steps after data has been received from the environment.

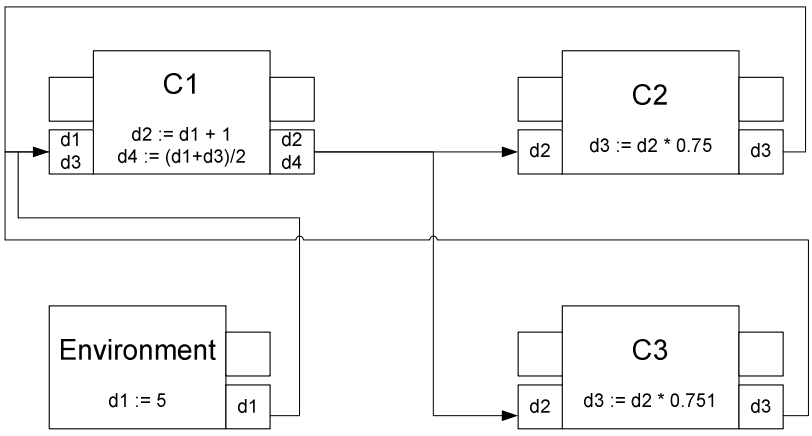

Fig. 3. Test example (note that the distinction between object and meta-level has been omitted in the Figure)

6 Case Studies

To verify whether the language presented above can indeed be used to specify a wide variety of coordination approaches (varying from pre-specified to more generic, flexible approaches), a number of such coordination approaches have been specified using the language. This section presents three of these approaches and shows how they can be specified in the coordination language. Note that in these specifications it is assumed that all links continuously forward the data they receive, and furthermore, that all foci of the components are set to a particular default value, such as "derive all possible information".

6.1 Pre-Specified Coordination

This section shows how a pre-specified coordination sequence for the test example can be expressed in terms of the coordination language introduced in Section 3. The different steps that constitute the coordination approach are described both in an informal notation and in a formal notation, using the LEADSTO format [4]:

PSC1
In case the target set for the system as a whole is to derive one element of information type d4, and no element is present of information type d2, d3, and d4, whereas there is of type d1, then component C1 is selected to become awake and the input update for information types d1 and d3 is to be closed.

∀R:REAL
[[monitored_control_aspect_for(is_output_focus(f1), SYSTEM) &
  monitored_control_aspect_for(info_type_in_focus_has_qualifier(d4, f1, one_element, R), SYSTEM) &
  ¬signed_info_element_present_of(d4) &
  ¬signed_info_element_present_of(d3) &
  ¬signed_info_element_present_of(d2) &
  signed_info_element_present_of(d1)]
 $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
 [selected_control_aspect_for(awake, C1) &
  selected_control_aspect_for(¬open_to_input_update(d1), C1) &
  selected_control_aspect_for(¬open_to_input_update(d3), C1)]]

Note that the signed_info_element_present_of can easily be defined in terms of the coordination language proposed in Section 3. It is meant to improve the readability of the specification.

PSC2
If any component C is awake, and is non-busy, then component C is selected to become asleep.

∀C:COMPONENT
[[monitored_control_aspect_for(awake, C) & monitored_control_aspect_for(nonbusy, C)]
 $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
 [selected_control_aspect_for(asleep, C) & selected_control_aspect_for(¬awake, C)]]

PSC3
In case at least one element of information types d2 and d1 is present, and no element of information type d3 is present, then component C3 is selected to become active.

∀F:FOCUS, R:REAL
[[monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
monitored_control_aspect_for(info_type_in_focus_has_qualifier(d4, F, one_element, R), SYSTEM) &
¬signed_info_element_present_of(d4) & ¬signed_info_element_present_of(d3) &
signed_info_element_present_of(d2) & signed_info_element_present_of(d1)]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
[selected_control_aspect_for(awake, C3) &
selected_control_aspect_for(¬open_to_input_update(d2), C3)]]

Obviously, putting asleep component C3 is performed in a similar fashion as presented for component C1.

PSC4
In case for both information type d3 and d1 at least one element is present, then component C1 is selected to become active.

∀F:FOCUS, R:REAL
[[monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
monitored_control_aspect_for(info_type_in_focus_has_qualifier(d4, F, one_element, R), SYSTEM) &
¬signed_info_element_present_of(d4) &
signed_info_element_present_of(d3) &
signed_info_element_present_of(d2) &
signed_info_element_present_of(d1)]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
[selected_control_aspect_for(awake, C1) &
selected_control_aspect_for(¬open_to_input_update(d1), C1) &
selected_control_aspect_for(¬open_to_input_update(d3), C1)]]

6.2 Backward Goal Propagation

A more generic, flexible way of specifying coordination can for instance be done via the principle of backwards goal propagation. This approach works in a centralized fashion, having knowledge about the entire system. Backward goal propagation starts to reason from the goal to be achieved, looks which components can achieve this goal (if the goal is not already achieved), and what specific input they need for that. Next, it is determined whether the input of these components is present, and in case it is not, other components that generate that specific information are derived. This process continues until a component is reached that can be activated (because its inputs are already present).

Note that the approach described below assumes that, for each information type, the preferred component to derive that information is known. This particular information can for instance be based upon the time used by the various components that can produce the information. A strategy would be to select the one which requires the least processing time. In LEADSTO the approach is specified as follows (note

that, for the sake of conciseness, some rules are not shown, such as rules addressing input and output foci of the components themselves, when an awake component becomes non-busy, when it is set to asleep, et cetera):

BGP1
If for an information type at least one element of a particular information type needs to be derived according to the goal, then this is a required information type for the system to derive.

∀F:FOCUS, R:REAL, IT:INFO_TYPE
[[monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
monitored_control_aspect_for(info_type_in_focus_has_qualifier(IT, F, one_element, R), SYSTEM)
]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
required_information_type(IT)]

BGP2
In case a certain information type is a required type, and a component C is preferred to be used to derive such information, then this component can potentially become active.

∀IT:INFO_TYPE, C:COMPONENT
[[required_information_type(IT) & preferred_component_for(C, IT)]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
potential_activation(C, IT)]

BGP3
In case a component has the potential to become active, and is not missing any information to derive the required output for which it is potentially active, then the component is selected to become awake.

∀C:COMPONENT, IT1:INFO_TYPE
[[potential_activation(C, IT1) &
∀IT2:INFO_TYPE [¬currently_needed_input_for_output(IT2, IT1]]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
selected_control_aspect_for(awake, C)]

Note that the component also needs to be closed to input updates for all information types.

BGP4
In case a component has the potential to become active for information type IT1, and needs information type IT2 as input to derive IT1, then IT2 is required as well.

∀C:COMPONENT, IT1, IT2:INFO_TYPE
[[potential_activation(C, IT1) & currently_needed_input_for_output(IT2, IT1)]
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$
required_information_type(IT2)]

6.3 Pandemonium

Besides generic flexible approaches that require central knowledge of the system, decentralized approaches exist as well. An example of such a decentralized approach is the pandemonium approach, introduced by Selfridge [13]. The approach was

initially meant for pattern recognition. The approach proposed is based on a system composed of primitive constructs called *demons*, each representing a possible pattern. Once an image is presented, each of the demons computes the similarity of the image with the pattern it represents, and gives an output depending monotonically on that similarity. Finally, a decision demon selects the pattern belonging to the demon whose output is largest.

In the context of the component-based software systems proposed in this paper, a variant of the pandemonium approach is used. According to this approach, a new *round* starts after the initial setup of the system or after the components set to awake have performed their task. In such a round, all components can *shout*. The idea is that, the more urgent a component thinks it is for him to be activated, the louder it will shout. The component that shouts loudest will be set to awake. In case multiple components shout with exactly the same strength, then all are set to awake in parallel and as a processing time the maximum of the processing time of each of these components is used. When a component is set to awake, and sufficient input data is present to generate output, this output is indeed generated.

To determine how loud they will shout, the components make use of a shout function. For different variants of the pandemonium model, different shout functions may be used. In the current model, each component uses the following types of information in its shout function at time point t:

- the amount of data it needs as input (represented by $i1$)
- the amount of its input data that is available at t (represented by $i2$)
- the amount of data it produces as output (represented by $o1$)
- the amount of its output data that is already present at t (represented by $o2$)
- the highest $i1$ for the set of components (represented by max_i)
- the highest $o1$ for the set of components (represented by max_o)

Given these elements, the shout value (i.e., the strength with which a component shouts, represented by sv) is modelled as follows:

$$sv = (i2/i1)^{\beta 1} * (1 - o2/o1)^{\beta 2} * (i1/max\_i)^{\beta 3} * (o1/max\_o)^{\beta 4}$$

Here, $\beta 1$, $\beta 2$, $\beta 3$, and $\beta 4$ are real numbers between 1 and 1.5, indicating the importance of the corresponding factor. Several settings have been tested for these parameters. In the examples shown here, $\beta 1$=1.4, $\beta 2$=1.3, $\beta 3$=1.1, and $\beta 4$=1.2. Since the factors can never exceed 1, the shout value sv will be a value between 0 and 1. In case the component has been set to awake in the previous round, another shout function is used which simply sets the shout value to 0, to avoid components from becoming active multiple consecutive times without making progress.

Below, a brief overview of LEADSTO rules that specify such a pandemonium strategy using the language presented in this paper is shown. Note that not all constructs shown below are fully specified in this language, but are definable using the language. The abbreviations are used to improve the readability of the specification.

PM1
In case the component was not previously active, the shout function gets its value according to the multiplication as specified above.

$\forall$C:COMPONENT, I1,I2,O1,O2:INTEGER
[[component_input_number(C, I1) & component_input_present(C, I2) &
component_output_number(C, O1) & component_output_present(C, O2) &
$\neg$previously_active(C)]
$\twoheadrightarrow_{0,0,1,1}$
shout(C, (I2/I1)^1.4 * (1-O2/O1)^1.3 * (I1/max_input)^2.2 * (O1/max_output)^1.2))]

PM2
In case the component was previously active, the shout function gets value 0.

$\forall$C:COMPONENT
previously_active(C) $\twoheadrightarrow_{0,0,1,1}$ shout(C, 0)

PM3
If the shout value of a component is at least as high as the shout value of another component, then this component is better than (i.e., at least as loud as) the other component.

$\forall$C1, C2:COMPONENT, I1, I2:INTEGER
[shout(C1, I1) & shout(C2, I2) & I1 $\geq$ I2] $\twoheadrightarrow_{0,0,1,1}$ better_than(C1, C2)

PM4
If a component is better than all other components, and has sufficient information available to derive output (represented by the component_allowed relation), then the component is selected to become awake.

$\forall$C:COMPONENT, I1,I2,O1,O2:INTEGER
[[$\forall$C2:COMPONENT [better_than(C, C2)] & component_allowed(C1)]
$\twoheadrightarrow_{0,0,1,1}$ selected_control_aspect_for(awake, C)]

7 Simulation Results

In order to investigate how well the coordination approaches presented in Section 6 can be applied to a test example, simulation runs have been performed. The results of these simulation runs for the pre-specified coordination approach, the backward goal propagation approach, and the pandemonium approach are described, respectively, in Section 7.1, 7.2 and 7.3.

7.1 Pre-Specified Coordination

Figure 4 shows a partial trace of the results of applying the pre-specified coordination approach to the test example. In the Figure, the left side denotes the state properties that occur during the simulation, whereas the right side indicates a time line, where a black box indicates that a state property is true at that time point and a grey box indicates that it is false.

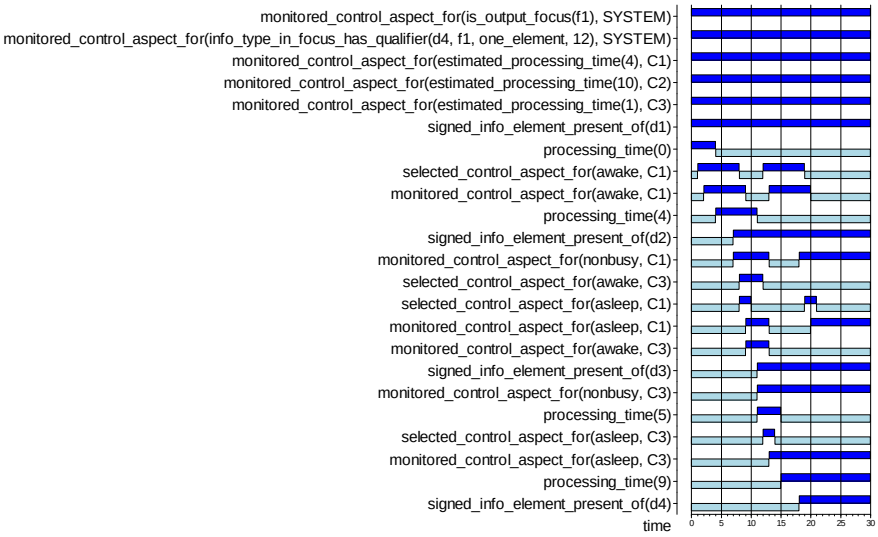

Fig. 4. Partial trace resulting from pre-specified coordination

First of all, the initial goal of the system as a whole is specified:

monitored_control_aspect_for(is_output_foucs(f1), SYSTEM)
monitored_control_aspect_for(info_type_in_foucs_has_qualifier(d4, f1, one_element, 12), SYSTEM)

Furthermore, initially there is a signed info element of the information type d1 (which has been provided by the environment):

signed_info_element_present_of(d1)

As a result, the pre-specified coordination approach starts to reason, which results in the activation of component C1:

selected_control_aspect_for(awake, C1)

This coordination statement results in a monitored control aspect that specifies that the component is indeed awake:

monitored_control_aspect_for(awake, C1)

After the component has indeed been set to awake, the component derives the information it can derive. In this case, information type d2 is derived, which results in an information element being present of that type:

signed_info_element_present_of(d2)

Meanwhile, the total processing time of the components used throughout the simulation is specified. At the moment that component C1 has been activated, the total processing time is 4:

```
processing_time(4)
```

After that, since component C1 has derived its output data d2, C1 becomes nonbusy, which results in this component being set to asleep:

```
selected_control_aspect_for(asleep, C1)
```

The next component activations are performed in a similar fashion (C3 and C1 respectively, following from the pre-specified specification), eventually resulting in the specified target being reached:

```
signed_info_element_present_of(d4)
```

Moreover, the total processing time finally adds up to 9 time steps:

```
processing_time(9)
```

7.2 Backward Goal Propagation

Figure 5 shows a partial trace of the behavior of the backward goal propagation approach as presented in Section 6.2.

In the trace, it is initially derived that information type d4 is a required information type:

```
required_information_type(d4)
```

Since component C1 is the only one capable of deriving d4, it is the preferred component, and therefore it is derived that C1 can potentially become active:

```
potential_activation(C1, d4)
```

On the coordination level, it is monitored that C1 needs input of information type d3 in order to derive d4:

```
monitored_control_aspect_for(currently_needed_input_for_ouput(d3, d4), C1)
```

As a result, it is derived that d3 is also a required information type. Now there is a choice: potential activation of C2 or C3, since both can deliver information of type d3. On the system level however, a preference has been specified for component C3. Consequently, C3 can potentially become active:

```
potential_activation(C1, d3)
```

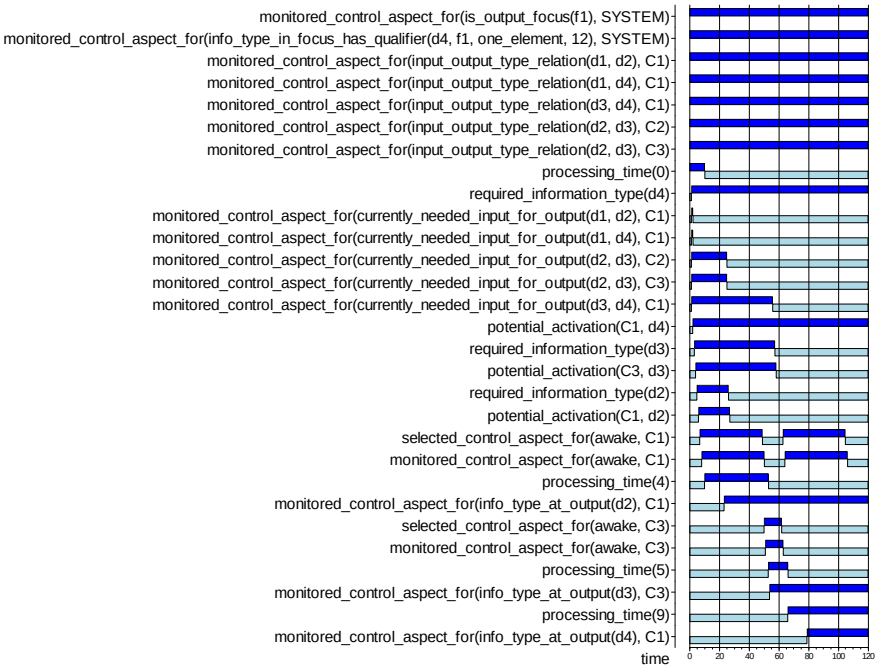

Fig. 5. Partial trace resulting from backward goal propagation coordination

This reasoning continues until a component has been found for which all of its necessary inputs are present. In this case this is component C1, which is therefore set to awake. As a result, d2 is present, which causes component C3 to become awake. Finally, after C3 has derived d3, C1 is activated again, deriving d4 and thereby causing the goal to be achieved. Note that the computation time used by this algorithm is again 9:

```
processing_time(9)
```

This is due to the preference that has been set for derivation of d3. Would this have been C2, then the derivation would have been done in 18 time units.

7.3 Pandemonium

Figure 6 shows a partial trace that results from applying the pandemonium coordination mechanism to the test example described before.

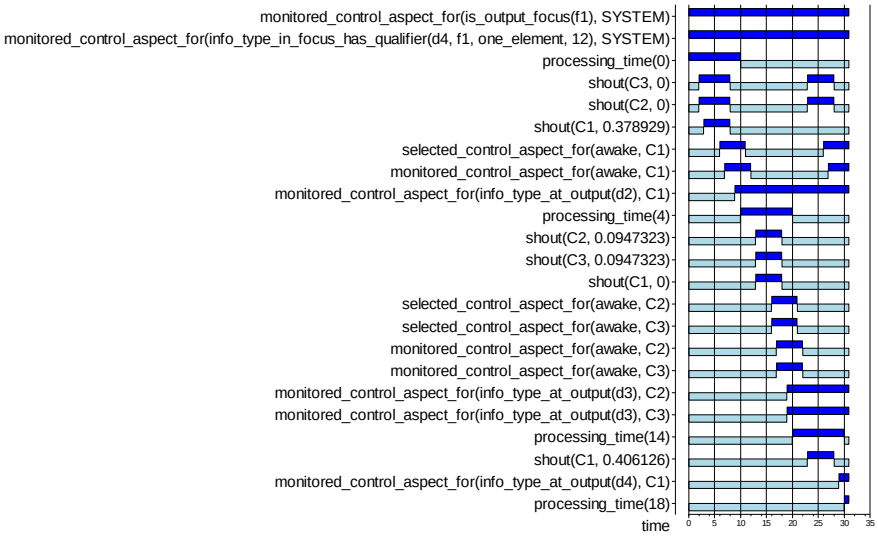

Fig. 6. Partial trace resulting from pandemonium coordination

In the trace, first the initial foci and processing time are set, and thereafter, the first round of shouting is started. Since C2 and C3 do not have any input available which they can use, their shout value is 0:

shout(C2, 0)
shout(C3, 0)

Furthermore, component C1 does have information available at the input, therefore its shout value is greater than 0:

shout(C1, 0.378929)

As a consequence, C1 is the component with the highest shout value that is also executable, and is therefore set to awake:

selected_control_aspect_for(awake, C1)

As a result of the activation of the component, information of the type d2 is derived (not shown in the trace). The activation period of C1 ends, and a new shouting round is started. In this case, component C1 has just been active, which disallows it to become active again, thus its shout value is 0:

shout(C1, 0)

Component C2 and C3 however do shout with a value greater than 0, because the information type d2 (which they need as their input) is now present. Since the components each have the same input and output information type specification, they have the same shout value:

shout(C2, 0.0947323)
shout(C3, 0.0947323)

In case multiple components are present with the same shout value, all of them are to become active. Since C2 and C3 have different processing times, the maximum of the processing times is taken (10 in this case) and is added to the overall processing time (which was 4 before this round):

processing_time(14)

Finally, component C1 is ranked as the highest shouting component again. As a consequence, it is activated, resulting in the overall goal being achieved. The eventual processing time is 18:

processing_time(18)

8 Evaluation

In order to evaluate the performance of the coordination approaches, certain desired properties can be verified against the generated simulation traces. Such properties can be specified in TTL and automatically verified using the *TTL checker* [9]. In order to express such properties in TTL, the ontology specified in Section 3 is used again.

A first check that can be performed is to investigate whether the goal that has been set for the system as a whole has been reached. Two variants of this property can be specified. First of all, a variant is formulated where a deadline has the form of a time-point:

P1: Successfulness with deadline
For all time points t, if at t the system has a particular output focus and target qualifier, then a conjunction of signed info elements exists at the output that entails this target qualifier and furthermore, all of the signed info elements within the conjunction have been derived by a component before the deadline set.

```
∀t1, t2:TIME, F:FOCUS, TQ:TARGET_QUALIFIER, TQE:TQ_EXPRESSION
 [ [ state(γ, t1) |= monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
   state(γ, t1) |= monitored_control_aspect_for(focus_has_qualifier(F, TQ), SYSTEM) &
    state(γ, t1) |= monitored_control_aspect_for(has_expression(TQ, TQE), SYSTEM) &
  ⇒ ∃t3:TIME ≥ t1, SIEC :SIGNED_INFO_ELEMENT_CONJUCTION
    [ t3 ≤ t2 & entails(SIEC, TQE) &
      ∀SIE:SIGNED_INFO_ELEMENT
      [state(γ, t3) |= is_conjunct_of(SIE, SIEC) ⇒
       state(γ, t3) |= monitored_control_aspect_for(information_at_output(SIE), SYSTEM) ]
     ]
  ]
]
```

A similar property can be specified for identified of goals that specify a maximum duration for derivation:

P2: Successfulness with duration

```
∀t1:TIME, R :REAL, F:FOCUS, TQ:TARGET_QUALIFIER, TQE:TQ_EXPRESSION
 [ [ state(γ, t1) |= monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
   state(γ, t1) |= monitored_control_aspect_for(focus_has_qualifier(F, TQ), SYSTEM) &
    state(γ, t1) |= monitored_control_aspect_for(has_expression(TQ, TQE), SYSTEM) &
  ⇒ ∃t2:TIME ≥ t1, SIEC :SIGNED_INFO_ELEMENT_CONJUCTION
    [ t2 ≤ t1 + R & entails(SIEC, TQE) &
      ∀SIE:SIGNED_INFO_ELEMENT
      [state(γ, t2) |= is_conjunct_of(SIE, SIEC) ⇒
       state(γ, t2) |= monitored_control_aspect_for(information_at_output(SIE), SYSTEM) ]
      ]
   ]
]
```

This successfulness property has been verified against the traces that result from the coordination approaches presented in Section 6. Since the goal for the test example presented in Section 5 is specified by means of a maximum duration, variant P2 of the property has been used. The checks pointed out that the pre-specified coordination approach and the backward goal propagation approach satisfy this property. The pandemonium strategy however does not satisfy this property since the deadline is set to 12 time units whereas the pandemonium takes 18 time units to come to a solution.

Besides the successfulness of a system, the efficiency of the outcome can be investigated as well. In order to determine whether the most efficient route has been found, information is needed about what is this most efficient route in the particular test example. In order to calculate this route, for instance, a critical path method can be used. For the specification of this property it is assumed that the most efficient way of derivation of a particular conjunction of signed info elements is known, indicated by most_efficient_derivation_duration: SIGNED_INFO_ELEMENT_CONJUNCTION x REAL. The property expressing that the most efficient derivation has indeed been found can then be expressed as follows:

P3: Efficient derivation

The derivation is efficient in case all elements of a signed info element conjunction have been derived in the most efficient manner, and there is no other signed info element conjunction entailing the target that could have been derived faster.

```
∀t1:TIME, R1:REAL, F:FOCUS, IT:INFO_TYPE, TQ:TARGET_QUALIFIER,
TQE:TQ_EXPRESSION
[ [state(γ, t1) |= monitored_control_aspect_for(is_output_focus(F), SYSTEM) &
   state(γ, t1) |= monitored_control_aspect_for(focus_has_qualifier(F, TQ), SYSTEM) &
   state(γ, t1) |= monitored_control_aspect_for(has_expression(TQ, TQE), SYSTEM)
  ⇒ ∃t2:TIME ≥ t1, SIEC :SIGNED_INFO_ELEMENT_CONJUCTION, R2:REAL
    [ t2 ≤ t1 + R1 & entails(SIEC, TQE) &
      state(γ, t2) |= most_efficient_derivation_duration(SIEC, R2) & t2 ≤ t1 + R2
      ∀SIE:SIGNED_INFO_ELEMENT
      [state(γ, t2) |= is_conjunct_of(SIE, SIEC) ⇒
       state(γ, t2) |= monitored_control_aspect_for(information_at_output(SIE), SYSTEM) ]
      ]
      &
      ¬∃SIEC2:SIGNED_INFO_ELEMENT_CONJUCTION, R3:REAL
      [ state(γ, t2) |= monitored_control_aspect_for(entails(SIEC2, TQE), SYSTEM) &
        state(γ, t2) |= most_efficient_derivation(SIEC, R3) & R3 < R2
      ]
   ]
]
```

This property is satisfied for the trace of the pre-specified and the backwards goal propagation coordination approach. The property is not satisfied for the pandemonium, since the path chosen using the approach was such that the duration summed up to 18 time units of processing required whereas the most efficient solution can be found in 9 time steps.

9 Discussion

This paper presents a coordination specification language that allows for the specification of both pre-defined coordination approaches and more generic, flexible approaches. Using this language, it has been shown how several of such coordination approaches (including a pre-specified as well as several flexible approaches) can be specified in an executable format. The approaches have been tested by applying them to a specified test example, and have been evaluated using formal verification. The pre-specified and backward goal propagation approach both were successful given the goal set, whereas the pandemonium was not (before the deadline) due to an inefficient derivation not limited to the focus set. Furthermore, the first two approaches found the solution in an efficient manner as well.

Besides the approaches that have been analyzed in this paper, other flexible coordination approaches exist, such as behavior networks [11] and voting [12]. It is left for future work to determine whether these approaches can be specified in the language presented in this paper as well.

Already in 1979, Kowalski [10] claimed that the coordination of a component should be separated from the logic component (i.e. the knowledge used in solving problems). In [2] a coordination language for multi-agent systems is proposed. It is stated that one of the contributions of the work is “a conceptualization of the coordination task of multi-agent systems that is able to express a wide range of coordination behaviors”. The language presented there is however not a language which can be used to represent the pre-specified coordination mechanisms such as specified in Section 6 of this paper. In [5] an overview is presented for the usage of coordination models and languages as software integrators. An often discussed coordination model is said Linda [1], which can, according to [5] be seen as a sort of assembly coordination language which can be used for implementing higher-level coordination languages. To see whether the language presented in this paper can also be implemented using Linda is part of future work.

In addition, for future work, a more elaborate evaluation of the coordination approaches presented in Section 6 would be interesting. Such a more elaborate evaluation could result in a characterization of the different approaches, identifying in what situation which approach should be preferred. Furthermore, guidelines on how to specify the coordination of a system will be created to aid the developer in the specification thereof.

Acknowledgments

This work has been performed as part of a project funded by Force Vision, the software development department for the Royal Netherlands Navy. Moreover, the authors are grateful to Joost van den Boom, Rob Duell, Andy van der Mee, and Radu Serban for various fruitful discussions.

References

[1] Ahuja, S., Carriero, N., and Gelernter, D., Linda and friends. *IEEE Computer*, 19(8):26–34, 1986.

[2] Barbuceanu, M., and Fox, M.S., The Design of a Coordination Language for Multi-Agent Systems, In: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, 1996, pp. 341-355.

[3] Bosse, T., Hoogendoorn, M., and Treur, J., Automated Evaluation of Coordination Approaches, In: Ciancarini, P. and H. Wiklicky, H. (eds.), *Proceedings of the 8th International Conference on Coordination Models and Languages (Coordination 2006)*, LNCS 4038, 2006, pp. 44-62.

[4] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), Proc. of MATES'05. LNAI, vol. 3550. Springer Verlag, 2005, pp. 165-178.

[5] Ciancarini, P., Coordination Models and Languages as Software Integrators, ACM Computing Surveys, Vol. 28, No.2, 1996, pp. 300-302.\

[6] Galton, A. (2003). Temporal Logic. *Stanford Encyclopedia of Philosophy*, URL: http://plato.stanford.edu/entries/logic-temporal/#2

[7] Galton, A. (2006). Operators vs Arguments: The Ins and Outs of Reification. *Synthese*, vol. 150 (2006), pp. 415-441.

[8] Gavrila, I.S., and Treur, J., A formal model for the dynamics of compositional reasoning systems. In: A.G. Cohn (ed.), Proc. 11th European Conference on Artificial Intelligence, ECAI'94, Wiley and Sons, 1994, pp. 307-311.

[9] Jonker, C.M., and Treur, J. (2002), "Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness," International Journal of Cooperative Information Systems, 11, pp. 51-92.

[10] Kowalski, R., Algorithm = Logic + Control, Communications of the ACM, Vol. 22, 1979, pp. 424 – 436.

[11] Maes, P., (1989). How to do the right thing. Connection Science, 1989. 1(3): p. 291-323.

[12] Ordeshook, P. *Game theory and political theory: An Introduction*. Cambridge: Cambridge University Press, 1986.

[13] Selfridge, O. G., (1958). Pandemonium: a paradigm for learning in mechanization of thought processes. In *Proceedings of a Symposium Held at the National Physical Laboratory*, pages 513-526, London, November 1958.

Part VI:
Organizational Change Evaluation

Chapter 15

Formal Analysis of Empirical Traces in Incident Management

Part of this chapter appeared as: Abbink, H., Dijk, R. van, Dobos, T., Hoogendoorn, M., Jonker, C.M., Konur, S., Maanen, P.P. van, Popova, V., Sharpanskykh, A., Tooren, P. van, Treur, J., Valk, J., Xu, L., and Yolum, P., Automated Support for Adaptive Incident Management. In: Walle, B. van de, and Carle, B. (eds.), *Proceedings of the First International Workshop on Information Systems for Crisis Response and Management, ISCRAM'04*, May 2004, pp. 69-74.

Furthermore, part of this chapter appeared as: Hoogendoorn, M., Jonker, C. M., Konur, S., Maanen, P.P. van, Popova, V., Sharpanskykh, A., Treur, J., Xu, L., Yolum, P., Formal Analysis of Empirical Traces in Incident Management. In: Macintosh, A., Ellis, R., and Allen, T. (eds.), *Applications and Innovations in Intelligent Systems XII, Proceedings of AI-2004, the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer Verlag, 2004, pp. 237-250.

Formal Analysis of Empirical Traces in Incident Management

Mark Hoogendoorn[1], Catholijn M. Jonker[2],
Peter-Paul van Maanen[1], and Alexei Sharpanskykh[1]

[1]Department of Artificial Intelligence, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {mhoogen, pp, sharp}@cs.vu.nl
URL: http://www.cs.vu.nl/~{mhoogen, pp, sharp}
[2]Nijmegen Institute for Cognition and Information
Radboud University Nijmegen
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
Email: C.Jonker@nici.ru.nl
URL: http://www.nici.ru.nl/~catholj

Abstract. Within the field of incident management decisions have to be made in a split second, usually on the basis of incomplete and partially incorrect information. As a result of these conditions, errors occur in such decisions processes. In order to avoid repetition of such errors, historic cases, disaster plans, and training logs need to be thoroughly analyzed. This paper presents a formal approach for such an analysis, that pays special attention to spatial and temporal aspects, to information exchange, and to organizational structure. The formal nature of the approach enables automation of analysis, which is illustrated by case studies of two disasters.

Keywords: incident management, formal analysis, automated evaluation

1 Introduction

Disasters are unforeseen events that cause great damage, destruction and human suffering. The question that keeps rising is: “Could we have done anything to prevent this?” The key element is the distinction between incidents and disasters. Incidents are disturbances in a system that can lead to an uncontrollable chain of events, a disaster, when not acted on properly.

Incidents cannot be avoided. People make mistakes and nature is unpredictable. Incidents typically lead to chaotic situations and complex problems that have to be solved within limited time. Examples of incidents that took on disastrous proportions because of inadequate human intervention are the crash of a Boeing 747 in an urban area in Amsterdam and the Hercules disaster in Eindhoven in the Netherlands.

Depending on the type of incident many people of various organizations have to cooperate; fire brigade, police, ambulance services, alarm centers, hospitals, coast

guard, local government, national government, army, to name but the most common organizations involved.

From the ICT perspective, research to improve incident management is mainly focused on the design and development of information systems to support both multi-party communication and decision making, thus addressing the multidisciplinary and distributive character of incident management. Systems like IMI [13] and the GIS-based information exchange system for nuclear emergencies in the Netherlands [14] belong to this category.

However, some research projects address the major problem in incident management of adaptively organizing multi-party cooperation in a dynamic context, while minimizing the number of errors. For example, the COMBINED project [5] tries to tackle the problem using adaptive multi-agent technology.

Finally, some projects focus on simulation tools and techniques to support analysis of crisis response and to support the development of training systems. An example of such an approach for simulations of strategic management is presented in [3].

Not addressed in the research mentioned above, is the ability to analyze the progress of incident management after the fact or while the incident unfolds. Such an analysis is useful for two reasons. First of all, it allows for the evaluation of historic cases whereby errors can be detected and learned from in order to avoid them from occurring again. Secondly, if such an analysis could actually be performed at runtime, this would enable the detection of errors before they result in a catastrophe.

This paper shows how such an analysis can be performed using formal modeling and verification techniques. The paper shows how domain specific properties can be specified in a formal language, how a trace (a temporal description of chains of events) can be formally specified, and finally how automated verification can be performed upon such a trace. Properties that can be analyzed include spatial, temporal, information exchange, as well as organization structure properties. These properties can originate from a variety of sources, such as disaster plans, disaster prevention plans, and laws. Two disasters that occurred in the Netherlands have been analyzed in this way, of which the results are presented in this paper.

In Section 2 of this paper an informal analysis of traces of real life case studies is presented. In Section 3 an outline is given of the adopted modeling approach. Section 4 shows a formalization of the trace of one of the case studies. The methodology for validation of such a formalized trace is presented in Section 5. Section 6 discusses a number of essential dynamic properties that have been formalized and automatically checked for the formalized trace from Section 4 using the methodology presented in Section 5. Section 7 is a final discussion.

2 Case Studies

In order to illustrate the functioning of the methodology, a number of cases have been studied. In this paper, two of such studies are presented, namely the Hercules disaster and the Dakota incident which both occurred in the Netherlands. These two examples have been chosen due to the different nature of the plans that are applicable in the particular cases. In the Hercules disaster very detailed plans were available in the

form of disaster prevention plans, whereas for the Dakota incident merely high level disaster plans were applicable.

2.1 Hercules disaster

The informal analysis of the Hercules disaster presented here is based on [8]. A formalization of the occurrences during this disaster can be found in Section 4.

On October 16th, 1996 at 6:03 p.m. a Hercules military airplane crashed at the airport of Eindhoven, in the Netherlands. This disaster involves many examples of miscommunications and lack of communication and is therefore a well known example of a non optimal working disaster prevention organization. An informal description of the events that took place during the rescue phase is presented below.

The Air-Traffic Control Leader on duty anticipated an accident and activated the so-called crash bell at 6:03 p.m. and hereby initiated the alarm phase. Trough the intercom installation he announced that a Hercules plane had landed with an accident and pointed out the location of the plane. The Assistant Air-Traffic Control Leader at the same time contacted the Emergency Centre of the Fire department at the Airbase and reported the situation. The Fire department immediately took action.

The Airbase Fire department must, when reporting to external authority, report which scenario is applicable. There are three different types of scenarios: Scenario 1: A maximum of 2 people involved, Scenario 2: More than 3 and less than or equal to 10 people. Scenario 3: More than 10 people. This all can be found on a checklist and also has consequences for the activities that should take place and the amount of authorities that need to be informed.

The Air-Traffic Control Leader on duty knew that at least 25 people were on board of the plane, this was due to a private source. He called the Emergency Centre of the Fire department at the Airbase around 6:04 p.m. with the order to call 06-11 (the national emergency number at that time).

The Chief of the Airbase Fire department (‘On Scene Commander’, OSC) asked Air-Traffic Control for the number of people on board of the plane at 6:04 p.m. According to this person, the answer was ‘nothing known about that’. Following from this the OSC reported Scenario 2 through the walkie-talkie. The Emergency Centre operator says not to have heard this but does not want to state that this has not been said.

At 6:06 p.m. the Emergency Centre operator calls 06-11 and is connected to the Central Post for Ambulances (CPA). From that point on, the Emergency Centre operator got help from a fire fighter. Together they tried to inform several governmental officials.

At 6:12 p.m. the Regional Emergency Centre of the Fire department (RAC) Eindhoven phoned air-traffic control with the question whether backup was needed, the response was ‘negative’. At 6:12 p.m. the Emergency Centre employee and the aforementioned fire fighter decided to follow Scenario 2 of the disaster plan (there were at least 4 people on board of the Hercules because that is the usual crew for this type of plane). At 6:15 p.m. the first civil fire trucks pulled out.

Besides alarming and informing all parties, actions on the scene were taken during that same period. Immediately after the announcement of the Air-Traffic Control

Leader the Airbase Fire department went to the scene with a Land Rover Command vehicle (LARO) with the OSC and two Major Airport Crash Tenders (MAC's) each manned with a crew of 3 people. The OSC thought that only the crew was on board of the plane and till the moment passengers had been found he handled accordingly.

At 6:05 p.m. the LARO arrived at the scene and directed the MAC's to the plane. At 6:07 p.m. the MAC's took their position of attack, the plane was on fire across the full length of the body. According to the procedures, the extinguishing was aimed at making the body fire-free. At 6:09 p.m. this was the case and the rest of the fire did not spread anymore. In this situation, the survivors could escape from the plane by themselves.

Around 6:10 p.m. one of the MAC's was empty and the other one only had a quarter of the water-supply left. The OSC decided to have a fire fighter switch the empty one for another one that was still full. After 6 minutes the fire fighter was back with a full MAC.

At 6:19 p.m. there was complete control over the fire at the right wing and engine. Thereafter, at 6:25 p.m. the first civil fire trucks arrived on the scene. After their arrival the OSC contacted the chief of the first fire truck who was told that probably four people were on board of the plane. After pumping water to the MAC's at 6:38 p.m. they started extinguishing the left engine.

6:33 p.m. was the exact time point when the decision was made to go inside the plane and use a high-pressure hose to extinguish some small fires inside the plane. After that, at 6:37 p.m. the fire fighters were in the plane for the first time and shortly thereafter the first casualty was discovered. Almost at the same time 20 to 30 other casualties were discovered.

2.2 The Dakota Disaster

The informal analysis of the Dakota disaster presented here is based on [9].

The plane crash of a Dakota PH DDA in 1996 in The Netherlands is another examined disaster. The plane had 6 crew members and 26 passengers on board and crashed into the Wadden Sea.

In the Dakota disaster, other factors are involved in the emergency rescue process. For instance, some officers are not familiar with emergency procedures/protocols for the disaster. The wrong procedures/protocols are picked up. An inefficient rescue procedures/protocols consequently is followed. Another example is that an overload of some of the partners can potentially cause some mistakes during the rescue process. However, miscommunications and inappropriate decisions are also involved in the rescue process.

On September 25, 1996 a Dakota PH DDA of the Dutch Dakota Association (DDA) left Texel International Airport Holland. The plane had 6 crewmembers and 26 passengers on board. Shortly after take off the crew reported engine trouble to Texel International Airport Holland (TIA). Around 4:36 p.m. the crew contacted the Navy airbase The Kooy (MVKK) and stated that it wanted to make an emergency landing on The Kooy. After a short while, The MVKK observed the Dakota disappear from the radar screen.

The MVKK immediately sent a helicopter, initiated a team of rescue helicopters and alarmed the coast guard centre (KWC). At 4:46 p.m. the KWC passed the correct information of the disaster to Regional Alarm Centre northern part of Noord-Holland

(RAC) and asked the RAC to alarm the relevant partners. Unfortunately, the RAC only organised the rescue boats and vessels and did not alarm other parties, that should be warned in the disaster.

At 4:55 p.m., the KWC reported the disaster to Noord Hollands Dagblad (a Dutch newspaper) and RTL TV station. Consequently, the KWC got many requests for information from the ANP (Dutch press office). The KWC is thus under a lot of pressure.

Through the ANP, the National Centre for Coordination (LCC) got the message that the Dakota had crashed. At 5:03 p.m. the LCC contacted the KWC, the KWC asked the LCC to help by providing a trauma team.

Coincidentally, a big drill for ambulances was ready to start. The Drill leader asked the president of the Dutch health service (GGD) whether the drill should still go on. At 5:05 p.m. the president of the GGD called RAC to inquire if the accident is for real. The RAC responded that neither the KWC nor the harbor office (HK) knew what was going on. The GGD even agreed to start the drill.

At almost the same time, the KWC asked the MVKK to take care of the wounded and told the LCC that the trauma team should be sent to MVKK. At 5:07 p.m. the LCC made an appointment with the Ministry of Public Health, Wellbeing, and Sports (VWS), VWS finally arranged the trauma team.

At 5:17 p.m. the first helicopter with casualties landed at Gemini Hospital (Gemini), the Gemini called the RAC to ask what the purpose of this is. The RAC replied that they only knew a plane had crashed and did not know anything more.

At 5:20 p.m. the RAC asked the KWC to get a trauma team from Gemini to MVKK. Meanwhile the centre for ambulances (CPA) of Amsterdam, the mayors of Den Helder and Wieringen, and the commander of the regional fire department are notified. After a while the arrangements of a crisis centre finally set up at the Navy. At 6:44 p.m. all bodies are found and transported. There is only one survivor of the disaster.

3 Modeling Approach

To formally specify dynamic properties that are essential in incident management processes, an expressive language is needed. To this end the *Temporal Trace Language* is used as a tool; cf. [11]. In this paper for the properties occurring in Section 6 both informal or semi-formal and formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly defined as follows.

A *state ontology* is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology

Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of a role r within an incident management process (e.g., mayor, or fire fighter) at time point t is denoted by

state(γ, t, input(r))

analogously

state(γ, t, output(r))
state (γ, t, internal(r))

denote the output state, internal state and external world state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r)) |= p denotes that state property p holds in trace γ at time t in the output state of role r. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as ¬, ∧, ∨, ⇒, ∀, ∃. In trace descriptions, notations such as

state(γ, t, output(r)) |= p

are shortened to

output(r) | p

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, means:

If *state property* α *holds for a certain time interval with duration g,*
then *after some delay (between e and f) state property* β *will hold for a certain time interval of length h.*

For a precise definition of the *leads to* format in terms of the language TTL, see [12]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

4 Formalization of an Empirical Trace

Informal traces of events, such as the trace presented in Section 2 of the Hercules disaster, can be formalized using the formal language TTL as briefly described in Section 5; see also [11]. The translation from an informal trace of events to a formal trace is currently done by hand. However, for the future there are plans to develop a methodology that supports non-expert users in making this translation. When

formalizing a trace four key elements need to be represented within such a trace: (1) temporal aspects; (2) spatial aspects; (3) information exchanges, and (4) organization structure.

Formalizing a trace has several benefits. First of all, specific properties which should hold for a trace can be verified. An example of such a property in the case of an airplane crash is that a fire truck should be at the disaster area within 3 minutes according to the International Civil Aviation Organization (ICAO). Some properties (like the example just mentioned) can often easily be checked by hand, but in more complex cases, a mistake may have been caused by a wrong chain of events. These types of causes are usually difficult to determine, and the formalization can help for this purpose.

Another benefit of the formalization is in the case where the protocol for the disaster prevention organization was incorrect. After the protocol has been rewritten it can be formalized by means of executable properties and the scenario in which the previous protocol failed can be used as an input. Resulting from this, a simulation can be performed which in turn will result in a trace of the functioning of the disaster prevention organization when using the new protocol. By means of this trace the properties that failed with the previous protocol can again be verified to see whether the new protocol has indeed improved the functioning. In case the properties are again not satisfied the cause of this failure can be determined and the protocol can be revised until the desired properties are all satisfied.

An example of a formalization of a trace is shown in Figure 1. It models the events that occurred during the Hercules incident. Only a part of the trace is shown for the sake of brevity. On the left side of the picture the atoms are shown that are present in the trace. All atoms have the format

output('role')|communicated_from_to('src', dst', 'type', 'information')

The 'role' indicates the role that outputs this information, whereas the 'src' and 'dst' model the source and destination role (notice that 'role' = 'src' always holds). A list of all the abbreviations used for the roles is shown in Table 1.

The types of communication are based on speech acts [1]. In the full trace also atoms containing input are present. Behind the atom there is a time line, indicating when the atom is true in the trace.

Table 1. A list of all abbreviations

Abbreviation	Abbreviates
AFD	Airbase Fire Department
ATC	Air Traffic Control
CPA	Central Post Ambulances
MAC	Major Airport Crash tender
OSC	On Scene Commander
OSO	On Scene Operations
MHS	Medical Health Servies
OvD	Officer On Duty
CvD	Commander on Duty
0611	The national emergency number

For example, the first atom
output(ew)|communication_from_to(ew, 'ATC', observe, crash)

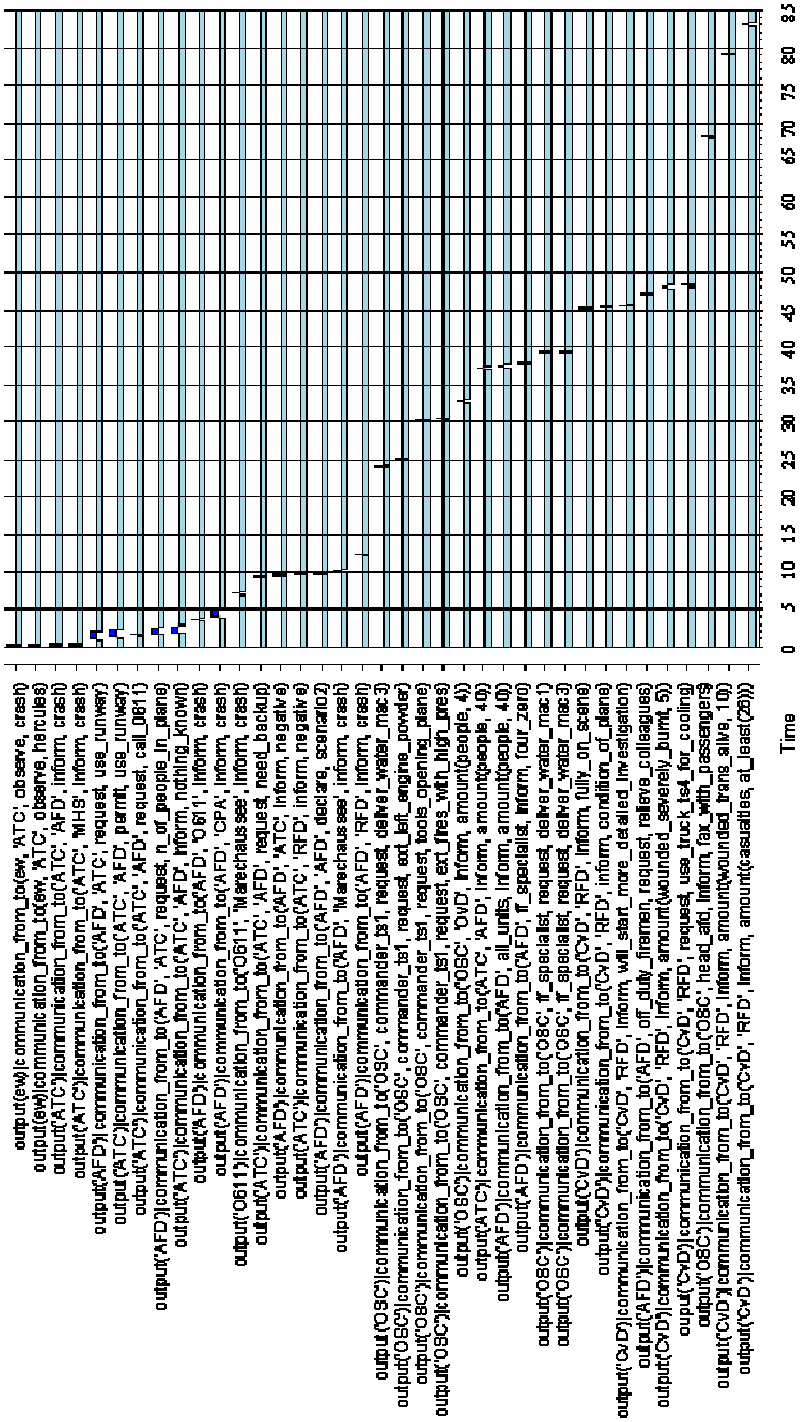

Fig. 1. Formalized empirical trace of the Hercules disaster

which states that the external world outputs a crash of a Hercules to air-traffic control, is true during the first minute after the crash, as he observes the crashed plane during that period.

A verification of properties that should hold for the disaster prevention organization is presented in the next section.

5 Methodology for Validation of a Trace

After having obtained a formalized trace, either by formalization of an empirical trace or by a simulated trace, it is useful to verify essential dynamic properties of an incident management process for the trace. By means of this verification one can determine what precisely went wrong in the example incident management process described by the trace.

Such dynamic properties can be verified using a special software environment *TTL Checker* that has been developed for the purpose of verifying dynamic properties specified in the Temporal Trace Language TTL (cf. [11]) against traces.

The software environment takes a dynamic property and one or more (empirical or simulated) traces as input, and checks whether the dynamic property holds for the trace(s). Using this environment, the formal representation relations presented below have been automatically checked against the trace depicted in Figure 1. Traces are represented by sets of PROLOG facts of the form

```
holds(state(m1, t(2)), a, true).
```

where m1 is the trace name, t(2) time point 2, and a is a state formula in the ontology of the component's input.

It is indicated that state formula a is true in the component's input state at time point 2. The program for temporal formula checking basically uses PROLOG rules that reduce the satisfaction of the temporal formula finally to the satisfaction of atomic state formulae at certain time points, which can be read from the trace representation. Examples of such reduction rules are:

```
sat(and(F,G)) :- sat(F), sat(G).
sat(not(and(F,G))) :- sat(or(not(F), not(G))).
sat(or(F,G)) :- sat(F).
sat(or(F,G)) :- sat(G).
sat(not(or(F,G))) :- sat(and(not(F), not(G))).
```

6 Validation of Hercules Disaster Trace

Below a number of properties are expressed that in particular are relevant for the Hercules case (Section 3.1), are represented in structured semi-formal format, and finally have been formalized using TTL. Such properties are categorized into four types, namely spatial properties, temporal properties, properties concerning the information exchange, and finally, organization structure properties. Properties can originate from disaster plans, disaster prevention plans, but for instance also from laws.

6.1 Spatial Properties

Spatial properties include the movement and position of both people as well as material involved in the disaster prevention process. An example of such a property for the Hercules disaster is shown below.

P1: Fire Truck Moves to Scene
At any point in time t1,
if the AFD is informed about a plane crash on the runway
then at a later point in time t2 a fire truck T will be located on the runway.
Formalization of the property P1:

$\forall$t1, R:ROLE
[state(γ, t1, input('AFD')) |= communication_from_to (R, 'AFD', inform, crash) &
$\Rightarrow$ $\exists$t2 > t1, T:FIRE_TRUCK state(γ, t2)|= located_at(T, 'runway')]

This property is satisfied for the given trace.

6.2 Temporal Properties

Temporal properties include notions such as timely arrival of information and resources. The essence of a property falling into this category is a restriction on the amount of time something is allowed to take. Three properties related to the Hercules disaster have been specified within this category:

P2: Timely Information Delivery
At any point in time t1,
if ATC generates information for AFD about the plane crash,
then at a later point in time t2, t2 $\leq$ t1+2 AFD will communicate this information to RFD.
Formalization of the property P2:

$\forall$t1 [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
$\Rightarrow$ $\exists$t2$\leq$ t1+2 state(γ, t2, output('AFD'))|= communication_from_to('AFD','RFD', inform, crash)]

This property is not satisfied for the given trace.

P3: MAC Timely Arrival at the Disaster Area
At any point in time t1,
if AFD receives information from ATC about the plane crash,
then at a later point in time t2 MAC will join AFD, and at a still later point in time t3 will come to the disaster area in less than 3 minutes upon the plane crash information reception.
Formalization of the property P3:

$\forall$t1 [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
$\Rightarrow$ $\exists$t2>t1 state(γ, t2) |= member_of('MAC', 'AFD') &
$\exists$t3$\leq$ t1+3 & t3>t2 & state(γ, t3)|=member_of('MAC', 'OSO')]

This property is satisfied for the given trace.

P4: Sufficient Number of Ambulances, Called Immediately
At some time point t1,
if CPA generates information about the number of ambulances, sent to the disaster area to RFD,
then at no later point in time t2 CPA will ask for additional ambulances.
Formalization of the property P4:

∀t1, x,y
[state(γ, t1, input('RFD')) |= communication_from_to ('CPA', 'RFD', inform, amount(ambulance_sent, x))
⇒ ¬∃t2>t1
state(γ, t2, output('CPA'))|= communication_from_to('CPA', 'CPA', request, amount(ambulance_needed), y)]

This property is not satisfied for the given trace.
The property P5 is meant to verify if CPA sent a sufficient number of ambulances to the scene immediately.

6.3 Information Exchange Properties

The information exchange properties express what information should be exchanged between the various players in the organization, but also the correctness of this information. An example of such a property is specified below for the Hercules disaster.

P5: Information Correctness
At any point in time t1,
if AFD generates a request for ATC about the number of people on the plane,
then at a later point in time t2 ATC will communicate the correct answer to AFD
Formalization of the property P5:

∀t1 [state(γ, t1, input('ATC')) |= communication_from_to ('AFD','ATC', request, n_of_people_in_plane)
⇒ ∃t2>t1 state(γ, t2, output('ATC'))|= communication_from_to('ATC','AFD', inform, amount(people, 40))]

Automated verification showed that this property is not satisfied in the given trace.

P6: Choice of Protocols
At any points in time t1 and t2, t2≥t1,
if ATC generates information to AFD about the plane crash at t1,
and that the number of passengers is more than 10 at t2,
then at a later point in time t3 AFD declares Scenario 3.
Formalization of the property P6:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]
⇒ ∃t3>t2 state(γ, t3, output('AFD'))|= communication_from_to('AFD','AFD', declare, scenario3)]

This property is not satisfied for the given trace.

6.4 Organization Structure Properties

Final property type is that of properties regarding the organization structure. Such a structure usually defined based upon the severity of the incident. Properties that can therefore be checked against the trace check for the correctness of the organization given the current situation. Two example properties are presented below.

P7: Presence Officer On Duty
At any points in time t1 and t2, t2≥t1,
if ATC generates information to AFD about the plane crash at t1,
and that the number of passengers is more than 10 at t2,
then at a later point in time t3 the OVD role is part of the On Scene Operations
Formalization of the property P7:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]
⇒ ∃t3>t2 state(γ, t3)|= member_of('OvD', 'OnSceneOperations')]

This property is satisfied for the given trace.

P8: Presence Police Units
At any points in time t1 and t2, t2≥t1,
if ATC generates information to AFD about the plane crash at t1,
and that the number of passengers is more than 10 at t2,
then at a later point in time t3 police units are part of the On Scene Operations
Formalization of the property P8:

∀t1, t2, x [t2≥t1 ⇒ [state(γ, t1, input('AFD')) |= communication_from_to ('ATC','AFD', inform, crash)
& x>10 & state(γ, t2, input('AFD'))|= communication_from_to('ATC, 'AFD', inform, amount(people, x))]
⇒ ∃t3>t2, P:POLICE_UNIT state(γ, t3)|= member_of(P, 'OnSceneOperations')]

This property is satisfied for the given trace.

As can be seen from the results of properties verification, given above, 4 from 8 properties are not satisfied over the trace. By analyzing the obtained results one can get insight in which types of errors occurred in the scenario and which points of the disaster plan, disaster prevention plan, and the law were not fulfilled.

7 Discussion

This paper shows how empirical traces in incident management can be thoroughly analyzed in an automated fashion, by means of formal verification techniques. The approach has been illustrated by the analysis of two historic cases, namely the Dakota incident and the Hercules disaster. Properties involving spatial, temporal, information exchange and organizational structure that are of particular importance in incident management have been specified and checked for these case studies. The result of the analysis shows that certain errors occurred in these historic cases which can as a result be avoided in future.

Research continues to make the approach applicable for the runtime analysis of incident management, possibly intervening in the incident management process in case severe errors are detected that could potentially have dramatic consequences. Such interventions could for example be in the form of a personal agent for each person involved in incident management which provides feedback to its user in the form of advice.

To enable such an analysis, usually one can specify dynamic properties at different aggregation levels, from global properties, to more local properties, and establish hierarchical inter-level relations between the properties. If one of the global properties does not hold, then verification of properties at intermediate levels can follow to identify were the cause of the problem can be found. The verification process can be continued up to the lowest level, consisting of the simplest local properties. See [10] for more details of this diagnostic approach. The approach put forward in this paper can be extended to include such a hierarchical diagnostic approach as well.

Acknowledgements

The authors would like to thank Hans Abbink, Roel van Dijk, Tamas Dobos, Savas Konur, Viara Popova, Peet van Tooren, Jan Treur, Jeroen Valk, Lai Xu, and Pinar Yolum for the fruitful discussions.

References

[1] Austin, J.L. How to do things with words. Oxford University Press, 2nd edition, 1976.
[2] Brazier, F.M.T., Treur, J. Compositional modelling of reflective agents. International Journal of Human-Computer Studies, vol. 50, 1999, pp. 407-431.
[3] Breuer, K., Satish, U. Emergency Management Simulations-An approach to the assessment of decision making processes in complex dynamic environments. In Jose J. Gonzalez (eds), From modeling to managing security: A system dynamics approach, HoyskoleForlaget, 2003, pp. 145-156.
[4] Brown, S. M., Santos Jr., E., Banks, S. B., Stytz, M. R. Intelligent interface agents for intelligent environments. In: Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments, 1998, pp. 145-147.
[5] Burghardt, P. Combined Systems: The combined systems point of view. In: Carlé, B., Walle, B. van der (eds.), Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04, Brussels, Belgium. 2004.
[6] Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., Williamson, M., An approach to planning with incomplete information. In: Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning, 1992, pp. 115-125.
[7] Fargier H., Lang J., Martin-Clouraire R., Schiex T. A constraint satisfaction framework for decision under uncertainty. In: Proc. of the 11th Int. Conf. on Uncertainty in Artificial Intelligence, 1995, pp. 167-174.
[8] Inspectie Brandweerzorg en Rampenbestrijding, Vliegtuigongeval Vliegbasis Eindhoven 15 juli 1996, SDU Grafische Bedrijf, The Hague, 1996.
[9] Inspectie Brandweerzorg en Rampenbestrijding, Dakota-incident Waddenzee 1996, SDU Grafische Bedrijf, The Hague, 1997.
[10] Jonker, C.M., Letia, I.A., Treur, J. Diagnosis of the dynamics within an organisation by trace checking of behavioural requirements. In: Wooldridge, M., Weiss, G., and Ciancarini, P. (eds.), Agent-Oriented Software Engineering, Proc. of 2nd Int Workshop AOSE'01. LNCS vol. 2222. Springer Verlag, 2002, pp. 17-32.
[11] Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
[12] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. Cognitive Systems Research Journal, vol. 4, 2003, pp. 191-210.
[13] Lee, M.D.E. van der, Vugt, M. van. IMI – an information system for effective multidisciplinary incident management. In: Carlé, B., Walle, B. van der (eds.), Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04, Brussels, Belgium. 2004.
[14] Ridder, M. de, Twenhöfel, C. The design and implementation of a decision support and information exchange system for nuclear emergency management in the Netherlands. In: Carlé, B., Walle, B. van der (eds.), Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04, Brussels, Belgium. 2004.

Chapter 16

Agent-Based Analysis and Support for Incident Management

Part of this chapter appeared as: Hoogendoorn, M., Jonker, C.M., Treur, J., and Verhaegh, M., Agent-Based Analysis and support for Incident Management. In: Klusch, M., Rovatsos, M., and Payne, T. (eds.), *Cooperative Information Agents X: Proceedings of the 10th International Workshop on Cooperative Information Agents (CIA 2006)*, LNCS 4149, Springer Verlag, 2006, pp. 109-123. The original publication is available at www.springerlink.com.

Agent-Based Analysis and Support for Incident Management

Mark Hoogendoorn[1] , Catholijn M. Jonker[2], Jan Treur[1], and Marian Verhaegh[3]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands,
tel. +31 20 598 7772, fax. +31 20 598 7653
{mhoogen, treur}@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute of Cognition and Information
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl
[3]Quartet Consult, Jaap Edenlaan 16, 2807 BR Gouda, The Netherlands
info@quartetconsult.nl

Abstract. This paper presents an agent-based approach for error detection in incident management organizations. The approach consists of several parts. First, a formal approach for the specification and hierarchical verification of both traces and properties. Incomplete traces are enriched by enrichment rules. Furthermore, a classification mechanism is presented for the different properties in incident management that is based on psychological literature. Classification of errors provides insight in the functioning of the agents involved with respect to their roles. This insight enables the provision of dedicated training sessions and allows software support to give appropriate warning messages during incident management.

Keywords: Error detection, incident management, formal analysis, handling incomplete information, agent-based support.

1 Introduction

The domain of incident management is characterized by sudden events which demand immediate, effective and efficient response. Due to the nature of incident management, those involved in such processes need to be able to cope with stress situations and high work pressure. In addition to that, cooperation between these people is crucial and is not trivial due to the involvement of multiple organizations with different characteristics (e.g. police, health care, fire department). As a result of these difficulties, often errors occur in an incident management process. If such errors are not handled properly, this may have great impact on the successfulness of incident management.

Research within the domain of computer science and artificial intelligence is being performed to see whether automated systems can improve the current state of affairs in incident management (see e.g. [10; 13]). One of the problems is that the information available is incomplete and possibly contradictory and unreliable. As a

result, more advanced techniques are needed to enable automated systems to contribute an improvement of the incident management process.

This paper presents an agent-based approach to monitor, analyze and support incident management processes by detecting occurring errors and providing support to avoid such errors or to limit their consequences. The approach is tailored towards the characteristics of incident management. First of all, the approach includes a method which deals with incomplete information. In addition, a diagnostic method based on refinement within the approach can signal whether certain required properties of the incident management organization are not satisfied, and pinpoint the cause within the organization of this dissatisfaction. The approach is based on the organizational paradigm nowadays in use in agent systems [1; 4] which allows the abstraction from individual agents to the level of roles. Such an abstraction is useful as typically specification of the requirements in this domain is done on the level of roles (e.g. the police chief should communicate a strategy for crowd control). In case errors are observed in role behavior, they are classified to have more insight in what kind of errors are often made by a particular agent participating in the organization, in order to propose a tailored training program for this agent. In the future the approach as a whole can be incorporated in cooperating software agents for monitoring and providing feedback in training sessions, and software agents which can even monitor incident management organizations on the fly, giving a signal as soon as errors are detected, and providing support to avoid their occurrence or to limit their consequences.

Section 2 introduces the domain of incident management and, more specifically, the situation in the Netherlands. Thereafter, Section 3 introduces the formal language used to specify traces and behavior. Section 4 presents an approach for handling incomplete information by means of enrichment rules whereas Section 5 presents a simple example of a specification of properties in the form of a hierarchy. Section 6 presents such properties for incident management organizations. Furthermore, Section 7 presents the classification scheme for errors, including specific incident management decision rules. Results of a case study are presented in Section 8 and finally, Section 9 is a discussion.

2 The Domain of Incident Management

In this Section, a brief introduction to the domain of incident management in the Netherlands is given. In the Netherlands four core organizations are present within incident management: (1) the fire department; (2) the police department; (3) health care, and (4) the municipalities involved. The first three parties mentioned each have their own alarm center in which operators are present to handle tasks associated with the specific organization.

A trigger for starting up an incident management organization is typically a call to the national emergency number, which is redirected to the nearest regional alarm center in which all three parties have their own alarm center. The call will be redirected to the most appropriate alarm center of the three parties. In case the operator of that alarm center considers the incident to be severe enough to start up the

full incident management organization, he informs the alarm centers of the other organizations as well. Initially, the three alarm centers will send the manpower they think is appropriate for the incident reported. After the manpower has arrived on the scene, each part of the organization in principle acts on its own, each having a different coordinator of actions. In the case of the fire department this is the commander of the first truck to arrive, for health care it is the paramedic of the first ambulance and for the police there is no such coordinator as they have a supporting role. Each of the coordinators are in charge until the dedicated operational leaders of the organization arrive at the scene. The responsibilities of the organizations are briefly described as follows: the fire department takes care of the so called “cause and effect prevention”, the health care organization is in charge of providing medical care, and the police takes care of routing of the various vehicles and crowd control. After the initial phase without structural coordination, an organization is formed in order to coordinate all actions of the individual organizations in case this is still necessary. The fire department is usually in charge of the operational side of this organization and the mayor of the municipality is in charge of the policy part. The mayor is responsible for the formation of the disaster staff for coordinating policy decisions, and is therefore informed of the situation. The operational coordination structures are formed after deliberation between the various parties on the scene has resulted in a mutual demand for such a coordination structure. In case it is decided to form the operational and/or disaster staff, the operators of the alarm centers start warning the relevant people.

In case the full coordination structure is in place, the organization resembles the structure shown in Figure 1. This is a partial picture, as the full picture would be too complex to explain in a brief manner. For more details on the full coordination structure, see [9].

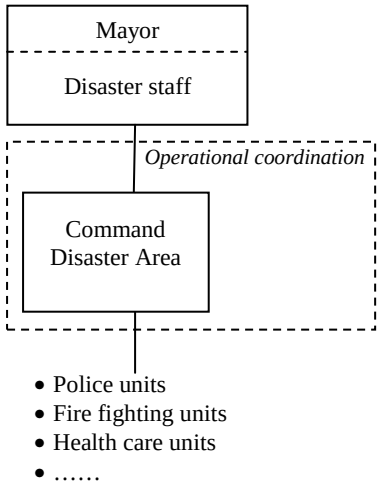

Fig. 1. Full coordination structure for incident management

3 Modeling Method Used

This section describes the language TTL (for Temporal Trace Language) [6] used for expressing dynamic properties as well as the expression of traces. Furthermore, the language meta-TTL is introduced for second-order dynamic properties.

3.1 The Language TTL for Dynamic Properties

To formally specify dynamic properties that are essential in an incident management organization, an expressive language is needed. To this end the Temporal Trace Language is used as a tool; cf. [6]. For the properties occurring in the paper informal, semi-formal or formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly described as follows; for more formal details, see Appendix A.

A state ontology `Ont` is a specification (in order-sorted logic) of a vocabulary. A state for ontology `Ont` is defined as an indication of which state properties expressed in ontology `Ont` hold in the state and which do not hold. The set of all states is modeled by the sort `STATE`. A fixed *time frame* `T` is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology `Ont` and time frame `T` is an indication of which state occurs at which time point, for example if a discrete time frame based on natural numbers is taken, a trace is a sequence of states γ_t ($t \in T$). The set of all traces over state ontology `Ont` is modeled by the sort `TRACE`. Depending on the application, the time frame `T` may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. A *dynamic property* over state ontology `Ont` is a temporal statement that can be formulated with respect to traces based on the state ontology. Such temporal statements can express, for example, a temporal relationship between the fact that in a given trace a certain state property holds at a certain time point and another state property holds at some other time point. For more formal details, see Appendix A.

3.2 The Language Meta-TTL for Second-Order Dynamic Properties

The formalizations of the properties sometimes take the form of second-order dynamic properties, i.e., properties that refer to dynamic properties expressed within TTL. Such second-order dynamic properties are expressed in meta-TTL: the meta-language of TTL. Again, for more formal details, see Appendix A.

4 Handling Incompleteness of Information by Enrichment Rules

The trace of occurrences as logged during or reported from an incident management process usually is incomplete and therefore difficult to analyze. To overcome this incompleteness problem, additional assumptions have to be made on events that have

occurred but are not explicitly mentioned in the logged trace. Such assumptions are addressed in this section. These extra assumptions enrich the trace with elements that are derived from the information in the trace itself, for example at later time points in case an analysis is performed afterwards. An example is the assumption that if at some time point an estimation of the situation is communicated, then at previous time points the necessary information to make that assessment was received or observed by the communicating role.

Addition of such elements to enrich a trace are based on rules which express that given certain trace elements, an additional element can be assumed. These rules in principle can be of two forms: Strict rules which can always be applied and provide conclusions that are certain, and defeasible rules which are used in case strict rules are insufficient to obtain a trace with a reasonable amount of information. However, it is not always possible to claim that a rule is a strict rule. Therefore, such rules are considered premises for the whole analysis.

Examples of such rules are presented below, note that the formal form of these rules can be found in Appendix B. Rule EP1 states that everybody present on the scene is assumed to have an internal judgment about the seriousness of the disaster:

EP1: **Internal judgment at scene**

if	at time t role R is present at the scene
and	situation S is the case
and	S is classified as being a disaster
then	there exists a later point in time t2 < t+d at which R has an internal judgment that this situation is a disaster

Furthermore, in case a role receives a communication that the situation is a disaster and this role does not communicate that he does not believe it being a disaster, then it is assumed that he has the internal judgment that it concerns a disaster:

EP2: **Internal judgment based on communication**

if	at time t R1 communicates to R2 that the current situation S is a disaster
and	there exists no time point at which R2 communicates to R1 he thinks the situation is not a disaster
then	at every time point t2 > t R2 interprets the current state of affairs as being a disaster

5 Property Hierarchies: A Simple Example

This section shows how, for a simple example, properties to be satisfied within an organization can be represented in the form of a property hierarchy. Specifying properties in such a hierarchy has as an advantage that diagnosis of properties can be done in a top down fashion. Such a diagnostic process starts by checking highest level property, and in case such a property is not satisfied pinpoints the error by gradually going down the tree to the unsatisfied properties, while leaving the satisfied properties and their refinements aside. Property hierarchies obtained from the field of incident management are specified in the section hereafter.

The simple example concerns the evacuation of a building in case of an alarm. The overall goal of such a process is to evacuate all people from the building within a certain duration *d* after the alarm has sound:

SP1(d): **Evacuate building**

if	at time t the evacuation alarm sounds in a building
then	there exists a time point t1 later than t and before t + d such that at t1 there are no more people inside the building.

Again, the formal forms for these properties can be found in Appendix B. Of course such a goal is not simply accomplished by itself: more refined properties can be formulated that together enable reaching this goal. These properties are shown in the tree in Figure 2. Three main, more refined properties constitute the achievement of the goal. First of all, a certain percentage of people will simply leave the building upon their own initiative after hearing the alarm (SP2). People leaving on their own initiative leave the building before αd, where α is between 0 and 1. The percentage of people that do not leave the building by themselves, are told to do so by an appointed person that checks all the rooms to be sure the building is empty (SP3). This results in these people leaving the building as well. Note that the duration *d* to be set, depends on the allowed percentage of people not directly responding. In case of a tight *d* this percentage should be low, otherwise the appointed person can never be done in time. A parameter β with a value between α and 1 is used to specify when these people should have been asked to leave the building (which should be before βd). Finally, the appointed persons themselves leave the building (SP4), due to the parameter β at which the people should have been informed, the appointed persons have a time frame between βd and d to leave the building.

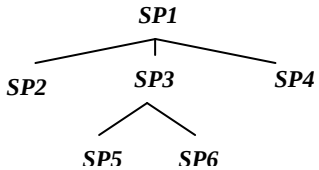

Fig. 2. Property hierarchy for the evacuation of a building upon an alarm

SP2(α, d, p): **Leave immediately**

if	at time t the evacuation alarm sounds in a building
then	there exists a time point t1 later than t and before t + αd (with α between 0 and 1) such that at least *p* percent of all person initially in the building are outside at time point t.

SP3(α, β, d, p): **Leave after correction**

if	at time t the evacuation alarm sounds in a building
and	at time t + αd at least *p* percent of the people are outside of the building already
and	at time t + αd person P is still in the building
and	person P is not responsible for emptying the building
then	there exists a time point t1 later than t + αd and before t + βd (with β between α and 1) and a person AP such that person AP is appointed for emptying the building
and	person P is told at time point t1 by AP to leave the building
and	before t + d this person P is outside of the building

This property again can be refined into two lower level properties. First of all, given the same condition, the communication by the appointed person takes place (SP5),

and secondly, once a person receives this communication he leaves the building (SP6).

***SP5(α, β, d, p):* Communicate correction**

if	at time t the evacuation alarm sounds in a building
and	at time t + αd person P is still in the building
and	at time t + αd at least *p* percent of the people are outside of the building already
and	person P is not responsible for emptying the building
then	there exists a time point t1 later than t + αd and before t + βd and a person AP such that person AP is appointed for emptying the building
and	person P is told at time point t1 by AP to leave the building

***SP6(β):* Leave after receiving communication of correction**

if	before time t + βd person P is told by AP to leave the building
then	there exists a later point in time t1 before t + d at which person P is no longer in the building.

The final property indeed specifies that the appointed person leave the building as well:

***SP4(α, d, p):* Appointed persons leave before deadline**

if	at time t the evacuation alarm sounds in a building
and	person P is an appointed person
and	at time t + αd at least *p* percent of the people are outside of the building already
then	before t + d this person P is outside of the building

6 Property Hierarchies for Incident Management Organizations

This section presents generic properties for incident management organizations in the Netherlands. Only the informal and semi-formal forms are presented here. For the formal form of these properties, see Appendix B.

6.1 Warning of Relevant Parties

The warning of relevant parties by the operator is a high level property stating that: "the operator should alarm all necessary parties in case it is informed of an incident":

***P1(d):* Warn relevant parties**

if	at time t the operator is informed about an incident type I by a role R1,
and	for incident type I role R2 should be informed according to the disaster plan
then	there exists a time t2 later than t and before t + d at which R2 is informed about the incident type I

This property can be refined into a number of similar properties restricted to specific categories of roles that should be informed. For diagnosis, at the highest level property P1(d) can be checked, for example with the result that P1(d) is not satisfied which means that not all relevant parties were informed (but without information on which specific categories were not informed). At one level lower, the diagnosis can be refined by checking the refined properties, resulting in an indication of which of the categories of relevant roles were not informed.

6.2 First Arriving Ambulance

Second, the behavior of the first arriving ambulance is addressed. First, a formal definition of the first arriving ambulance is given:

first_arriving_ambulance(γ:TRACE, t:TIME, A:AMBULANCE)
An ambulance is the first arriving ambulance if:
the ambulance arrives at the scene of an incident at time t
and there does not exist a time t' < t at which another ambulance arrived at the scene of the incident

On the highest level, the first arriving ambulance behavior is described by three important aspects: (1) signaling the green alarm light; (2) communicating a situation report, and (3) presence of at least one person belonging to the ambulance until the officer on duty arrives at the scene:

P2: **First arriving ambulance global behavior**

if	at a time t ambulance A is the first to arrive at the scene
and	at time t3 > t the officer on duty arrives at the scene
then	for all t2 ≥ t and t2 < t3 at least one person belonging to the ambulance should be present at the ambulance
and	for all t4 ≥ t the ambulance is signaling the green alarm light
and	there exists a time t5 later than t at which the driver of that ambulance communicates a correct interpretation of the situation to the operator.

This property can be related to lower level properties as shown in Figure 3. When trying to diagnose why the highest level property is not satisfied, the properties on the lower level can be checked. In case such a property is not satisfied, and it concerns a leaf property, at least one cause for the non-fulfillment of the high-level property has been found. Otherwise, go further down the tree to find the cause. In the tree a number of properties are present to enable satisfaction of P2. First of all, the signaling of the green light, as expressed below.

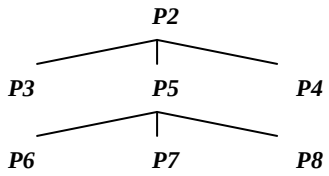

Fig. 3. Property hierarchy for the first arriving ambulance.

P3: **First ambulance green light behavior**

if	at a time t ambulance A is the first to arrive at the scene
then	for all later points in time t2 the ambulance is signaling the green light.

Second, the presence of a person belonging to the ambulance for the time until the officer on duty is present:

P4: **First arriving ambulance personnel presence**

if	at a time t ambulance A is the first to arrive at the scene
and	at time t3 > t the officer on duty arrives at the scene

then for all t2 ≥ t and t2 < t3 at least one person belonging to the ambulance should be present at the ambulance

Finally, a property expressing the communication of the correct situation to the operator:

P5(d): **First arriving ambulance interpretation**
if at a time t ambulance A is the first to arrive at the scene
then at a later point in time t2 < t + d the driver of that ambulance communicates a correct interpretation of the situation

Note that parameter d includes the time to interpret the situation plus the time to start communicating that particular interpretation. Testing whether the interpretation was correct can be performed afterwards (e.g., the amount of casualties). The property P5 can be refined again into three lower level properties. First of all, when arriving at the scene, the paramedic should investigate the current state of affairs:

P6(d): **Paramedic investigation**
if at a time t ambulance A is the first to arrive at the scene
and at time t a paramedic is in the ambulance
then at a later point in time t2 < t + d the paramedic of that ambulance will start an investigation and not be at the ambulance any more

Second, the paramedic will return, communicating the current situation:

P7(d): **Paramedic communication**
if at a time t ambulance A is the first to arrive at the scene
and at time t the paramedic is in the ambulance
and at time t2 the physical position of the paramedic is not inside the ambulance
then at a later point in time t3 < t2 + d the paramedic of that ambulance will communicate a correct interpretation of the situation to the driver

Finally, once the driver has received the communication, he will communicate this to the operator:

P8(d): **Driver communication**
if at a time t the driver of the first ambulance at the scene receives a situation description from the paramedic
then at a later point in time t2 < t + d the driver of that ambulance communicates a correct interpretation of the situation to the operator

6.3 Disaster Staff Activation

Furthermore, properties have been specified for the formation of the disaster staff and activities following from the disaster staff. On the highest level the correctness of these processes in the disaster staff can be described as follows: In case the operator has the internal judgment that the current situation is a disaster, the operational leader will eventually output actions belonging to a strategy communicated by the disaster staff.

P9: **Successful disaster staff**
if at time t the operator judges the current situation as a disaster
then there exists a later point in time t2 at which the disaster staff communicated a strategy
and there exists an even later time at which the operational leader communicates an action appropriate for the strategy according to the disaster plan.

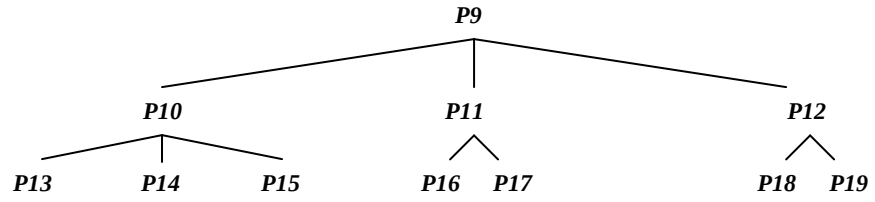

Fig. 4. Property hierarchy for the disaster staff activation and functioning.

Such properties can be related to lower-level properties as shown in Figure 4. On the intermediate level, three properties are present. First, the correct initiation of a disaster staff is expressed:

P10: **Correctly activated disaster staff**
if at time t the operator interprets the current situation being a disaster
then at a later point in time t2 the disaster staff will be informed (and assumed to be present as a result)

Thereafter, in case the disaster staff is formed, it should be active, which is characterized by an output in the form of a strategy:

P11: **Active disaster staff**
if at time t the organizational unit called disaster staff is informed
then at a later point in time t2 > t the organizational unit outputs a strategy S

Finally, such a strategy should lead to actions be taken by the operational leader:

P12: **Active operational leader**
if at time t the operational leader is informed of a strategy S to be applied
then at a later point in time t2 > t the operational leader will command the appropriate actions according to the disaster plan to the roles.

Each of these intermediate properties can again be split up to properties for individual roles within the organization. In order to obtain property P10 a number of properties need to hold. First of all, the mayor should be warned by the operator:

P13(d): **Warn mayor**
if at time t the operator interprets the current situation being a disaster
then at a later point in time t2 > t and t2 < t +d the operator communicates the occurrence of a disaster to the mayor.

Thereafter, the mayor should decide to form the disaster staff:

P14: **Form disaster staff**
if at time point t the mayor interprets the current state of affairs as being a disaster
then at a later point in time t2 > t the mayor forms the organizational unit called disaster staff

Finally, in case the mayor communicates the decision to form the disaster staff, the operator should warn the appropriate parties:

P15(d): **Warn rest disaster staff**
if at time t the operator receives the request of the mayor to form the disaster staff
and role R is part of the disaster staff
then at a later point in time t2 > t and t2 < t +d the operator communicates to role R that the disaster staff is being formed.

Regarding the intermediate property P11 the following properties need to hold for satisfaction of the intermediate property. First, after the mayor has decided to form the disaster staff he will eventually request advice from his disaster staff.

P16: **Start deliberation**
if at time t the mayor decides to form the disaster staff
then at a later point in time t2 > t the mayor starts a deliberation within the disaster staff by requesting advice

After such advice is received, he should choose the appropriate strategy:

P17: **Choose strategy**
if at time t starts a deliberation within the disaster staff by requesting advice
then at a later point in time t2 the mayor communicates a strategy to the operational leader

Finally, the intermediate property P12 is refined to two other properties. First, the operational leader should discuss the strategy with his operational team:

P18: **Choose action**
if at time t the mayor communicates a strategy S to the operational leader
then at a later point in time t2 > t the operational leader requests his operational team for advice how to implement S

Finally, the operational leader communicates actions to be performed, based on the advices obtained in the discussion.

P19: **Communicate action**
if at time t the operational leader request his operational team for advice how to implement S
then at a later point in time t2 the operational leader will communicate actions appropriate for strategy S according to the disaster plan

6.4 Ambulance Routing

Finally, properties are specified regarding ambulance routing. The police should act as follows:

P20: **Route plan includes all wounded nests**
if at time t there are n wounded nests
and at a later time point t2 > t the police communicates details concerning the route to be taken by the ambulances to cpa (the central ambulance post)
then this communication should contain such a route description that ambulances will be sent to all wounded nests.

An alternative property not following standard procedure expresses that the routing is done based explicitly on victim locations:

P21: **Send ambulance to all wounded on the scene**
if at time t there is a wounded person at a position P
then at a later time point t2 an ambulance will be sent to position P
and at an even later time point t3 that ambulance will be at position P

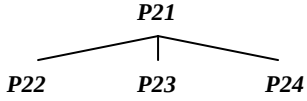

Fig. 5. Property hierarchy for ambulance routing

The property hierarchy for this high-level property is shown in Figure 5 and can be decomposed into several other properties. First of all, a wounded person will result in a communication to the operator of the physical position of this wounded person:

P22: **Communicate wounded location**
if at time t there is a wounded person at a position P
then at a later time point t2 this position will be communicated to the operator

For every communication received by the operator, he eventually communicates the location to an ambulance:

P23: **Send ambulance to wounded**
if at time t a wounded person is communicated to be at a position P
then at a later time point t2 an ambulance will be sent to position P

Finally, once the ambulance gets this communication it will arrive at the location at a later point in time:

P24: **Ambulance arrives at wounded**
if at time point t an ambulance is sent to position P
then at a later time point t2 that ambulance will be at position P

7 Human Error Types

This Section presents a classification scheme for the properties in incident management. Such a classification can help to determine the dedicated training needed. The human error classification presented by James Reason [11] is therefore adopted, who introduces a General Error Modeling approach which identifies three basic error types: (1) skill based slips; (2) rule based mistakes, and (3) knowledge based mistakes. This classification scheme is also used in (Duin, 1992) in which incident management is investigated. Rule based, and knowledge based errors come into play after the individual has become conscious of a problem, which is not the case for skill based slips. In that sense, skill based errors generally precede detection of the problem whereas rule based and skill based mistakes arise during subsequent attempts to find a solution to the problem. Skill based and rule based level error occur when humans use stored knowledge structures whereas knowledge based errors occur when such knowledge structures have been exhausted. Errors are much more likely to occur at the knowledge based level. Table 1 shows how the distinction between the different error types based on several dimensions.

Table 1. Distinctions between different error types (from [11])

Dimension	Skill-based errors	Rule-based errors	Knowledge-based errors
Type of activity	Routine actions	Problem-solving activities	
Focus of attention	On something other than in the task in hand	Directed at problem-related issues	
Control mode	Mainly by automatic processes (schemata)	(stored rules)	Limited, conscious processes
Predictability of error types	Largely predictable (actions)	(rules)	Variable
Ratio of error to opportunity for error	Though absolute numbers may be high, these constitute a small portion of the total number of opportunities for error		Absolute numbers small, but opportunity ratio high
Influence of situational factors	Low to moderate; intrinsic factors (frequency of prior use) likely to exert the dominant influence		Extrinsic factors likely to dominate
Ease of Detection	Detection usually fairly rapid and effective	Difficult, and often only through external intervention.	
Relationship to Change	Knowledge of change not accessed at proper time	When and how anticipated change will occur unknown	Changes not prepared for or anticipated.

For the properties specified for incident management the following classification scheme is used. Skill based properties are those properties that are part of the very basic training of incident management workers. For example, how to start the water pump on a fire truck. A property is classified as a rule based property in case an incident management plan literally includes the property. Finally, a property is called a knowledge based property in case an incident management plan states that a decision needs to be taken, but does not specify how to come to this solution. Using this classification scheme, none of the properties from Section 5 are routine based, whereas properties P1, P3, P5, P6, P8, P13, P14, P15, P16, P19, P22, and P24 are rule based properties. Finally, properties P7, P17, P18, P20, and P23 are knowledge base properties. Note that only the leaf properties are categorized as these are the properties that define the individual role behavior within the organization.

In order to identify which types of error the different participants in the incident management organization are making, the following formula is used (formal form in Appendix B):

if an agent A is allocated to a particular role R in a particular period between t_1 and t_2 in trace γ,
and a situation S occurs in that same period in which property P is relevant for role R whereby the type of property P for role R is of type X (where X is either skill based, rule based or knowledge based)
and the property has a specification which does not hold in the fragment of this trace
then an error of type X is made concerning property P by role R played by agent A.

8 Case Study

As a means to validate the approach presented above, a disaster which has been thoroughly investigated in the Netherlands is taken as a case study. The disaster

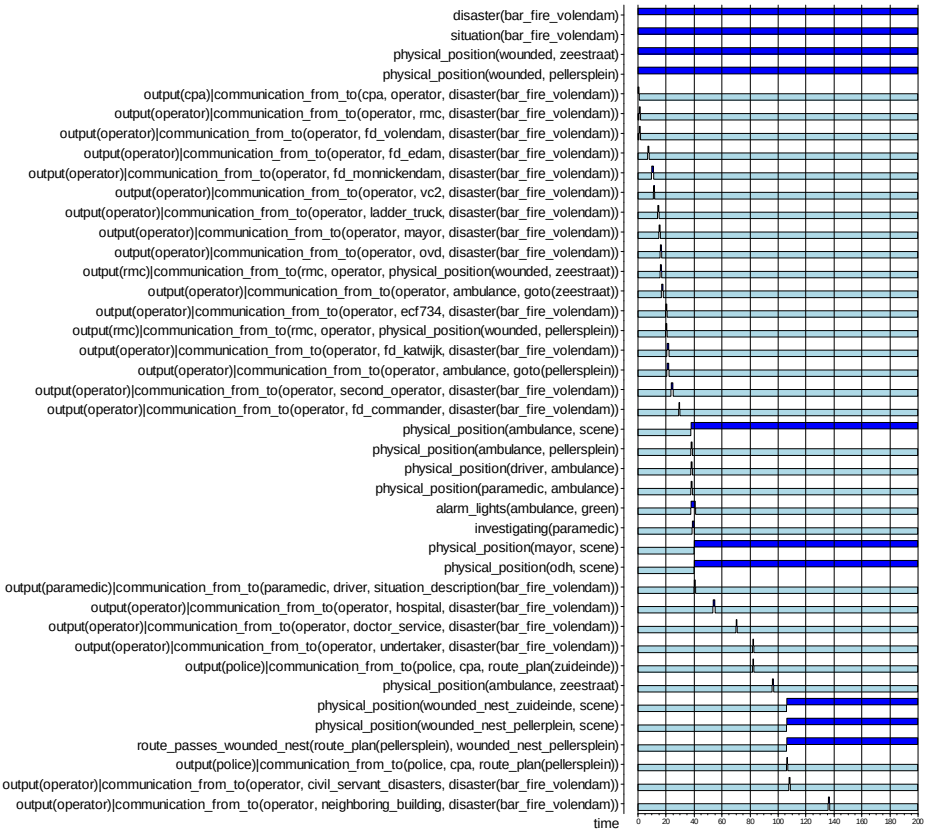

Fig. 6. Partial trace of the Volendam case study

concerns a bar fire which occurred in Volendam, the Netherlands, at New Years Night of the year 2001. The logs of the disaster have been thoroughly described in [8] and have been formalized using the approach presented in Section 3. Thereafter, the trace enrichment rules from Section 4 have been applied. A part of the resulting trace is shown in Figure 6, which uses the same ontology as used for the formalization of the properties in Section 5. On the left side of the Figure, the atoms are shown that occur during the incident management whereas the right side shows a timeline where a dark box indicates an atom being true at that time point and a gray box indicates the atom being false. The trace is used to verify whether the properties as specified in Section 5 indeed hold for the Volendam disaster. The following properties were shown not to hold: P2, P4, P5, P7, P8, P9, P10, P14, and P20. In other words, in the Volendam case study the first ambulance did not comply to the global desired behavior because the information was not communicated properly, and because there exist time points at which nobody was present at the ambulance. Furthermore, the disaster staff was not activated properly because the mayor did not communicate that the disaster staff should be formed, and finally the ambulance routing of the police was incorrect, but luckily the direct routing of the health care services was satisfied. These results

exactly comply to the conclusions in the disaster report [8] which resulted from a thorough investigation of a committee specialized in incident management.

9 Discussion

This paper presents an agent-based approach which can be used for error detection in incident management organizations. The approach consists of several parts. First, a formal approach for the specification of both traces and properties that can be verified against these traces is presented. In domains like incident management, traces might be incomplete. Therefore, enrichment rules for these traces are identified to cope with this incompleteness. Furthermore, the properties that ought to be verified against these traces can be specified in a hierarchical fashion: in case the highest level property is not satisfied, the cause of this dissatisfaction can be determined by looking at the properties one level deeper in the tree, which continues until a leaf property is found which is not satisfied. Finally, a classification mechanism is presented for the different properties based on psychological literature. In case an error is observed such a classification immediately gives insight in the functioning of a particular agent playing a role, which enables performing dedicated training sessions or giving appropriate warning messages.

In the future, the approach presented can be incorporated in personal agents of people involved in incident management. Such agents automatically log all incoming and outgoing information in the form of traces and have knowledge on the property the particular role the agent is playing is required to fulfill. In case properties are observed not to be satisfied, a reminder or warning can for instance be given to the person. Such agents can be useful for training sessions, as it can be observed what kind of mistakes a person typically makes, but could possibly even be used during actual incident management.

Many information systems have been developed or proposed that support processes involved in incident management. Already in the 1980s [14] a decision support system for disaster management has been proposed. In [10], a system is proposed for the support of scaling up an incident management organization. [7] presents the IMI system which can be used as an information source and a communication system, enabling crucial information to be sent to the appropriate people immediately, and information sources such as disaster plans to be widely available and easily usable. The reasoning behind these systems is to minimize the errors that occur in incident management. Despite these efforts, errors will continue to occur in incident management due to the stress, pressure, and incomplete information. Minimizing the consequences of such an error is therefore a necessity. This is exactly what can be established using the system presented in this paper.

In the field of information agents, support systems have also been developed for incident management (see e.g. [13]). In such systems however, the agents again do not check whether errors are made, but simply provide people with information to make sure they are aware of their tasks. This does however not offer a mechanism to detect errors and avoid a chain of unwanted events. Approaches for e.g. detection of protocols (see e.g. [12]), also called overhearing, have been introduced. These

approaches are however more focused on recognizing patterns, not on detection of errors.

Error detection itself is another related research field. In [3] behavioral properties for a parallel computing system can be specified, and can be checked on the fly. The properties are however specified as simple sequences of states, whereas the TTL language as used in this paper has the ability to express timing parameters between these states, often a necessity in incident management. In [5] properties for error detection are specified by means of a finite state machine which again does not allow for time parameter specification.

Acknowledgements

The authors wish to thank the Dutch Ministry of Economic Affairs for funding this research. Furthermore, the authors would like to thank the Netherlands Institute for Fire Service and Disaster Management for sharing their expertise in the domain of incident management.

References

[1] Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), Proc. of the 1st Workshop From Organizations to Organization Oriented Programming in MAS (OOOP), 2005.
[2] Duin, M.J. van, Learning from Disasters (in Dutch), PhD Thesis, Leiden, 1992.
[3] Fromentin, E., Raynal, M., Garg, V.K., and Tomlinson, A., On the Fly Testing of Regular Patterns in Distributed Computations, Information Processing Letters 54:267-274, 1995.
[4] Giorgini, P., Müller, J., Odell, J. (eds.), Agent-Oriented Software Engineering IV, LNCS, vol. 2935, Springer-Verlag, Berlin, 2004.
[5] Jard, C., Jeron, T., Jourdan, G.V., and Rampon J.X. "A general approach to trace-checking in distributed computing systems", In Proc. IEEE International Conference on Distributed Ccomputing Systems, pp. 386-404, Poznan, Poland, June, 1994.
[6] Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
[7] Lee, M.D.E. van der, Vugt, M. van. IMI – an information system for effective multidisciplinary incident management. In: Carlé, B., Walle, B. van der (eds.), Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04, Brussels, Belgium. 2004.
[8] Ministry of the Interior, Investigation Bar Fire New Years Night 2001 (in Dutch), SDU Publishers, The Hague, 2001.
[9] Municipality of Amsterdam, Disaster Plan (in Dutch), 2003.
[10] Oomes, A.H.J., Neef, R.M., Scaling-up Support for Emergency Response Organizations, In: Walle, B. van, and Carle, B. (eds.), Proceedings of ISCRAM 2005, pp. 29-41, 2005.
[11] Reason, J., Human Error, Cambridge University Press, 1990.
[12] Rossi, S., Busetta, P, Towards Monitoring of Group Interactions and Social Roles via Overhearing, In: Klusch, M., Ossowski, S., Kashyap, V., and Unland, R. (eds), Cooperative Information Agents VIII, LNAI 3191, Spinger-Verlag, pp. 47-61, 2004.
[13] Storms, P.A.A., Combined Systems: A System of Systems Architecture, In: Proceedings of ISCRAM 2004, pp. 139-144, May 2004, Brussels.

[14] Wallace, W.A., Balogh, F. de, Decision Support Systems for Disaster Management, Public Administration Review, Vol. 45, Special Issue: Emergency Management: A Challenge for Public Administration, pp. 134-146, 1985.

Appendix A: Temporal Trace Language (TTL)

This appendix presents a formal description of the language TTL and meta-TTL.

A.1 The Language TTL for Dynamic Properties

In TTL [6], ontologies for states are formalized as sets of symbols in sorted predicate logic. For any ontology Ont, the ground atoms form the set of *basic state properties* BSTATPROP(Ont). Basic state properties can be defined by nullary predicates (or proposition symbols) such as hungry, or by using n-ary predicates (with n>0) like has_temperature(environment, 7). The *state properties* based on a certain ontology Ont are formalized by the propositions (using conjunction, negation, disjunction, implication) made from the basic state properties and constitute the set STATPROP(Ont).

In order to express dynamics in TTL, important concepts are *states*, *time points*, and *traces*. A *state* S is an indication of which basic state properties are true and which are false, i.e., a mapping S: BSTATPROP(Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont). Moreover, a fixed *time frame* T is assumed which is linearly ordered. Then, a *trace* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by TRACES(Ont).

The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation, indicated by the infix predicate |=, comparable to the Holds-predicate in the Situation Calculus. Thus, state(γ, t) |= p denotes that state property p holds in trace γ at time t. Likewise, state(γ, t) $\not|=$ p denotes that state property p does not hold in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted predicate logic, using the usual logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, and the quantifiers $\forall$, $\exists$ (e.g., over traces, time and state properties). For example, consider the following dynamic property for a pattern concerning belief creation based on observation:

if at any point in time t1 the agent observes that the situation is a disaster,
then there exists a time point t2 after t1 such that
at t2 in the trace the agent believes that the situation is a disaster

This property can be expressed as a dynamic property in TTL form with free variable γ as follows:

$\forall t{:}T\ [\ \text{state}(\gamma, t) \models \text{observes(itsadisaster)} \Rightarrow \exists t' \geq t\ \ \text{state}(\gamma, t') \models \text{belief(itsadisaster)}\ ]$

The set DYNPROP(Ont, γ) is the subset of DYNPROP(Ont) consisting of formulae with γ occurring in which is either a constant or a variable without being bound by a quantifier. For a more elaborate explanation of TTL, see [6].

A.2 The Language Meta-TTL for Second-Order Dynamic Properties

The language meta-TTL includes sorts for DYNPROP(Ont) and its subsets as indicated above, which contain TTL-statements (for dynamic properties) as term expressions. Moreover, a predicate holds on these sorts can be used to express that such a TTL formula is true. When no confusion is expected, this predicate can be left out. To express second-order dynamic properties, in a meta-TTL statement, quantifiers over TTL statements can be used.

Appendix B: Properties Formally Specified in TTL

This Appendix presents the properties as presented in an informal or semi-formal form in the paper in a formal form using TTL.

B.1 Internal Judgment Properties (Section 4)

EP1: **Internal judgment at scene**

∀ R:ROLE, t:TIME, S:SITUATION
[state(γ, t) |= physical_position(R, scene) &
state(γ, t) |= current_situation(S) &
state(γ, t) |= disaster(S)]
⇒ ∃t2>t & t2 < t+d [state(γ, t2) |= internal_judgment(R, disaster(S))]

EP2: **Internal judgment based on communication**

∀R1,R2:ROLE, P:POSITION, t:TIME, S:SITUATION
[state(γ, t) |= communication_from_to(R1, R2, disaster(S)) &
¬∃t'>t [state(γ, t') |= communication_from_to(R2, R1, not(disaster(S)))]]
⇒ ∀t2 > t [state(γ, t2) |= internal_judgment(R2, disaster(S))]

B.2 Building Evacuation Property Hierarchy (Section 5)

SP1(d): **Evacuate building**

∀t:TIME
[state(γ, t) |= alarm_bell_sounds
⇒ ∃t1:TIME > t [t1 < t + d & ¬∃P:PERSON [state(γ, t1) |= in_building(P)]]]

SP2(α, d, p): **Leaving immediately**

∀t:TIME
[state(γ, t) |= alarm_bell_sounds
⇒ ∃t1:TIME > t [t1 < t + αd & ∃I:INTEGER [percentage_out_between(γ, t, t1, I) & I ≥ *p*]]]

The percentage of people getting already outside of the building can be expressed in TTL as

percentage_out_between(γ, t, t1, I) ⇔
∃ I2, I3:INTEGER
[amount_out_between(γ, t, t1, I2) & amount_people_in_building_at(γ, t, I3) &
I2/I3 * 100 ≤ I1 < I2/I3 * 100 +1]

where:

amount_out_between(γ, t, t1, I) ⇔
[$\sum_{\forall P:PERSONS}$ case([state(γ, t1) |= ¬in_building(P) & state(γ, t) |= in_building(P)], 1, 0)] = I

and

amount_people_in_building_at(γ, t, I) ⇔
[$\sum_{\forall P:PERSONS}$ case([state(γ, t) |= in_building(P)], 1, 0)] = I

Here for any formula f, the expression case(f, v1, v2) indicates the value v1 if f is true, and v2 otherwise.

***SP3(α, β, d, p):* Leaving after correction**
∀t:TIME, P:PERSON, I :INTEGER
[[state(γ, t) |= alarm_bell_sounds &
  percentage_out_between(γ, t, t + αd, I) & I ≥ *p* &
  state(γ, t + αd) |= in_building(P) &
  state(γ, t + αd) |= ¬person_for_emptying_building(P)]
 ⇒ ∃t1:TIME > t + αd, AP:PERSON
      [t1 < t + βd &
       state(γ, t1) |= person_for_emptying_building(AP) &
       state(γ, t1) |= communication_from_to(AP, P, leave_building) &
       ∃t2:TIME > t1 [t2 < t + d & state(γ, t2) |= ¬in_building(P)]]

***SP5(α, β, d, p):* Communicate correction**
∀t:TIME, P:PERSON, I:INTEGER
[[state(γ, t) |= alarm_bell_sounds &
  percentage_out_between(γ, t, t + αd, I) & I ≥ *p*
  state(γ, t + αd) |= in_building(P) &
  state(γ, t + αd) |= ¬person_for_emptying_building(P)]
 ⇒ ∃t1:TIME > t + αd, AP:PERSON
      [t1 < t + βd &
       state(γ, t1) |= person_for_emptying_building(AP) &
       state(γ, t1) |= communication_from_to(AP, P, leave_building)]]

***SP6(β, d):* Leaving after receiving communication of correction**
∀t:TIME, P:PERSON, AP:PERSON
[state(γ, t) |= communication_from_to(AP, P, leave_building)
 ⇒ ∃t1:TIME > t [t1 < t + (1-β)d & state(γ, t1) |= ¬in_building(P)]]

***SP4(α, d, p):* Appointed persons leave before deadline**
∀t:TIME, P:PERSON, I:INTEGER
[[state(γ, t) |= alarm_bell_sounds &
  percentage_out_between(γ, t, t + αd, I) & I ≥ *p* &
  state(γ, t) |= person_for_emptying_building(P)]
⇒ ∃t1:TIME > t [t1 < t + d & state(γ, t1) |= ¬in_building(P)]]

B.3 Incident Management Organization Property Hierarchy (Section 6)

***P1(d):* Warn relevant parties**
∀I:INCIDENT_TYPE, t:TIME, R1, R2:ROLE
[state(γ, t) |= communication_from_to(R1, operator, I) &
  state(γ, t) |= according_to_plan_should_be_involved_in(R2, I)]
 ⇒ ∃t2 > t & t2 < t + d [state(γ, t2) |= communication_from_to(operator, R2, I)]]

first_arriving_ambulance(γ:TRACE, t:TIME, A:AMBULANCE)
[state(γ, t) |=physical_position(A, scene) & ¬∃t'< t, [∃B:AMBULANCE [state(γ, t') |=physical_position(B, scene)]]

***P2:* First arriving ambulance global behavior**
∀A:AMBULANCE, t, t2:TIME
[first_arriving_ambulance(γ, t, A) &
state(γ, t2) |= physical_position(officer_on_duty, scene) &
 ¬∃t'''< t2 [state(γ, t''') |= physical_position(officer_on_duty, scene)]]
⇒

∀t3 < t2
    [t3 ≥ t ⇒ [∃R:ROLE [state(γ, t3) |= physical_position(R, A)]]]
 & ∀t4 > t [state(γ, t4) |= alarm_lights(A, green)]
 & ∃t5 > t, X:SITUATION[state(γ, t5) |= communication_from_to(driver, operator, situation_description(X)) &
                         situtation(X)]]

P3: **First ambulance green light behavior**

∀A:AMBULANCE, t:TIME
 [first_arriving_ambulance(γ, t, A) ⇒ ∀t2:TIME > t [state(γ, t2) |= alarm_lights(A, green)]]

P4: **First arriving ambulance personnel presence**

∀A:AMBULANCE, t, t2:TIME
[first_arriving_ambulance(γ, t, A) &
 state(γ, t2) |= physical_position(officer_on_duty, scene) &
 ¬∃t'''< t2 [state(γ, t''') |= physical_position(officer_on_duty, scene)]]
⇒ ∀t3 < t2 [t3 ≥ t ⇒ [∃R:ROLE [state(γ, t3) |= physical_position(R, A)]]]

P5(d): **First arriving ambulance interpretation**

∀A:AMBULANCE, t:TIME
first_arriving_ambulance(γ, t, A)
⇒ ∃X:SITUATION, t2:TIME < t + d & t2>t
      state(γ, t2) |= physical_position(driver, A) &
      state(γ, t2) |= communication_from_to(driver, operator, situation_description(X)) &
      state(γ, t2) |= situtation(X)]

P6(d): **Paramedic investigation**

∀A:AMBULANCE, t:TIME
[first_arriving_ambulance(γ, t, A) &
 state(γ, t) |= physical_position(paramedic, A)]]
⇒ ∃t2:TIME < t + d & t2 > t
     [state(γ, t2) |= not physical_position(paramedic, A) & state(γ, t2) |= investigating(paramedic)]

P7(d): **Paramedic communication**

∀A:AMBULANCE, t,t2:TIME
[first_arriving_ambulance(γ, t, A) &
 state(γ, t) |= physical_position(paramedic, A) & t2 > t &
 state(γ, t2) |= not physical_position(paramedic, A) &
 state(γ, t2) |= investigating(paramedic)]
⇒ ∃t3:TIME < t2 + d & t3 > t2, X:SITUATION
   [state(γ, t3) |= physical_position(paramedic, A) &
    state(γ, t3) |= communication_from_to(paramedic, driver, situation_description(X)) &
    state(γ, t3) |= situtation(X)]

P8(d): **Driver communication**

∀A:AMBULANCE, t,t2:TIME, X :SITUATION
[first_arriving_ambulance(γ, t, A) &
 state(γ, t2) |= communication_from_to(paramedic, driver, situation_description(X))
⇒ ∃t3:TIME < t2 + d & t2 > t [state(γ, t3) |= communication_from_to(driver, operator, situation_description(X))]

P9: **Successful disaster staff**

∀t:TIME
[state(γ, t) |= internal_judgement(operator, disaster)
⇒ ∃t2:TIME > t, S:STRATEGY
     [state(γ, t2) |= communication_from_to(disaster_staff, operational_leader, S) &
      ∃t3:TIME > t2, A:ACTION, R:ROLE
      [state(γ, t3) |= appropriate_action_according_to_plan(S, A) &

state(γ, t3) |= accompanying_role(A, R)] &
state(γ, t3) |= communication_from_to(operational_leader, R, perform(A))]

P10: **Correctly activated disaster staff**

∀t:TIME, R:ROLE
[state(γ, t) |= internal_judgement(operator, disaster) &
state(γ, t) |= part_of(R, disaster_staff)
⇒ ∃t2:TIME > t + d [state(γ, t2) |= communication_from_to(operator, R, form_disaster_staff)]]

P11: **Active disaster staff**

∀t:TIME
[state(γ, t2) |= part_of(R, disaster_staff) &
state(γ, t2) |= communication_from_to(operator, R, form_disaster_staff)
⇒ ∃S:STRATEGY, t2 > t [state(γ, t2) |= communication_from_to(disaster_staff, operational_leader, S)]

P12: **Active operational leader**

∀t:TIME, S:STRATEGY, A:ACTION, R:ROLE
[state(γ, t2) |= communication_from_to(disaster_staff, operational_leader, S) &
state(γ, t) |= appropriate_action_according_to_plan (S, A) &
state(γ, t) |= accompanying_role(A, R)
⇒ ∃t2:TIME > t state(γ, t2) |= communication_from_to(operational_leader, R, perform(A))]

P13(d): **Warn mayor**

∀t:TIME
[state(γ, t) |= internal_judgement(operator, disaster) &
⇒ ∃t2:TIME > t & t2 < t + d [state(γ, t2) |= communication_from_to(operator, mayor, disaster)]

P14: **Form disaster staff**

∀t:TIME
[state(γ, t) |= internal_judgement(mayor, disaster) &
¬∃t'< t [state(γ, t') |= internal_judgement(mayor, disaster)] &
⇒ ∃t2 > t [state(γ, t2) |= communication_from_to(mayor, operator, form_disaster_staff)]

P15(d): **Warn rest disaster staff**

∀t:TIME, R:ROLE
state(γ, t) |= communication_from_to(mayor, operator, form_disaster_staff) &
state(γ, t) |= part_of(R, disaster_staff)
⇒ ∃t2:TIME > t + d [state(γ, t2) |= communication_from_to(operator, R, form_disaster_staff)]]

P16: **Start deliberation**

∀t:TIME
[state(γ, t) |= communication_from_to(mayor, operator, form_disaster)
⇒ ∃t2:TIME > t [state(γ, t2) |= communication_from_to(mayor, disaster_staff, request_advice)]]

P17: **Choose strategy**

∀t:TIME
[state(γ, t) |= communication_from_to(mayor, disaster_staff, request_advice)
⇒ ∃S:STRATEGY, t2:TIME > t state(γ, t2) |= communication_from_to(mayor, operational_leader, S)]

P18: **Choose action**

∀t:TIME, S:STRATEGY
[state(γ, t) |= communication_from_to(mayor, operational_leader, S)
⇒
∃ t2:TIME > t state(γ, t2) |= communication_from_to(operational_leader, operational_team, request_advice(S))]

P19: **Communicate action**
∀t:TIME, S:STRATEGY, A:ACTION, R:ROLE
[state(γ, t) |= communication_from_to(operational_leader, operational_team, request_advice(S)) &
state(γ, t) |= appropriate_action_according_to_plan (S, A) &
state(γ, t) |= accompanying_role(A, R)
⇒ ∃ t2:TIME > t state(γ, t2) |= communication_from_to(operational_leader, R, perform(A))]

P20: **Route plan includes all wounded nests**
∀W:WOUNDED_NEST, R:ROUTE_PLAN, t:TIME
[state(γ, t) |= physical_position(W, scene) &
state(γ, t) |= communication_from_to(police, cpa, R)]
⇒ state(γ, t) |= route_passes_wounded_nest(R, W)

P21: **Send ambulance to all wounded on the scene**
∀W:WOUNDED, P:POSITION, A:AMBULANCE, t:TIME
[state(γ, t) |= physical_position(W, P) &
⇒ ∃t2 > t [state(γ, t2) |= communication_from_to(operator, A, goto(P))] &
∃t3 > t2 [state(γ, t3) |= physical_position(A, P)

P22: **Communicate wounded location**
∀W:WOUNDED, P:POSITION, t:TIME
[state(γ, t) |= physical_position(W, P) &
⇒ ∃R:ROLE, t2 > t [state(γ, t2) |= communication_from_to(R, operator, physical_position(W, P))]]

P23: **Send ambulance to wounded**
∀W:WOUNDED, P:POSITION, R:ROLE, t:TIME
[state(γ, t) |= communication_from_to(R, operator, physical_position(W, P))
⇒ ∃t2 > t, A:AMBULANCE [state(γ, t2) |= communication_from_to(operator, A, goto(P))]]

P24: **Ambulance arrives at wounded**
∀P:POSITION, A:AMBULANCE, t:TIME
[state(γ, t2) |= communication_from_to(operator, A, goto(P))
⇒ ∃t2 > t [state(γ, t3) |= physical_position(A, P)

B.4 Type Error Definition (Section 7)

Type Error ≡
∀γ:TRACE, t_1, t_2:TIME, A:AGENT, R:ROLE, P:DYNPROP, Q:DYNPROPEXPR, S:SITUATION,
X:PROPERTY_TYPE
[holds_in_period(has_role(A, R), γ, t_1, t_2) &
holds_in_period(S, γ, t_1, t_2) &
holds_in_period(relevant_for(P, R, S), γ, t_1, t_2) &
holds_in_period(type_for(P, R, X), γ, t_1, t_2) &
holds_in_period(has_specification(P, Q(R, γ, c_1, c_2)), γ, t_1, t_2) &
¬holds(Q(R, γ, t_1, t_2))]
⇒ makes_error_of_type(A, R, P, X, γ, t_1, t_2)

Chapter 17

Automated Verification of Disaster Plans in Incident Management

This chapter will appear as: Hoogendoorn, M., Jonker, C.M., Popova, V., and Sharpanskykh, A., Automated Verification of Disaster Plans in Incident Management, *Disaster Prevention and Management*, 2007.

Furthermore, part of this chapter appreared as: Hoogendoorn, M., Jonker, C.M., Popova, V., Sharpanskykh, A., Xu, L., Formal Modelling and Comparing of Disaster Plans. In: Carle, B., and Walle, B. van de, (eds.), *Proceedings of the Second International Conference on Information Systems for Crisis Response and Management ISCRAM '05*, 2005, pp. 97-107.

Automated Verification of Disaster Plans in Incident Management

Mark Hoogendoorn[1], Catholijn M. Jonker[2], Viara Popova[1], and Alexei Sharpanskykh[1]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, popova, sharp}@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute of Cognition and Information Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. Every municipality in The Netherlands has its own disaster plan. A disaster plan contains the blueprint of how to handle incidents in the municipality with the aim of preventing incidents to grow into disasters. Given that each municipality has its own organizations, enterprises, infrastructure, and general layout, the disaster plans also differ. On the other hand, the disaster plans have a lot in common. Some municipalities use a common starting point, others develop their own disaster plan from scratch. In this paper two independently developed disaster plans are compared using formal modeling techniques. The analysis reveals that some interesting differences do not stem from a difference in the makings of the municipality. Another question considered in this paper is to which extent disaster plans are followed when incidents occur. To answer this question an automated approach for verifying properties from disaster plans on formalized empirical data by means of dedicated software is proposed.

1 Introduction

Disasters are unforeseen events that cause great damage, destruction and human suffering. The question that keeps rising is: "Could we have done anything to prevent this?" The key element is the distinction between incidents and disasters. Incidents are disturbances in a system that can lead to an uncontrollable chain of events, a disaster, when not acted on properly.

Incidents will keep occurring. People can make mistakes and nature can be unpredictable. Typically this causes chaotic situations and the resulting problems are very complex and have to be solved within limited time. Examples of incidents that took on disastrous proportions because of inadequate human intervention are the crash of a Boeing 747 in an urban area in Amsterdam and the Hercules disaster in Eindhoven in the Netherlands.

In order to cope with such incidents, every municipality in The Netherlands has its own disaster plan. A disaster plan contains the blueprint of how to handle incidents

with the aim of preventing incidents to grow into disasters. The plan describes the relations with all organizations that might possibly be involved, like the mayor, the fire department, police, ambulances, hospitals, other municipalities, provincial government, national government. When comparing municipalities both commonalities and differences stand out. The commonalities encompass such basic elements as a local government, the availability of some kind of police force, fire department, and ambulance services. Small municipalities might not have their own forces of the kind mentioned, but have to share them with other municipalities. Big cities have subdivided their forces in smaller units that predominantly serve specific parts of the city. More fundamental differences involve the infrastructure of the municipality (e.g., forms of public transportation, the road plan, water ways, bridges), but also the enterprises and organizations available within the boundaries of the municipality like airports, factories, restaurants, stadiums and theatres.

Given that each municipality has its own organizations, enterprises, infrastructure, and general layout, it seems self-evident that the disaster plans also differ. On the other hand, the disaster plans form only a blueprint of handling incidents. For every entity in the municipality that carries a predictable risk a more detailed plan has to exist, a so called disaster prevention plan. The advantage of separating disaster plans from disaster prevention plans is that the disaster plan is applicable in all situations and is a relatively compact document. This line of reasoning entails again that the disaster plans of different municipalities should have, and in fact do have a lot in common. On the basis of the above, one might expect that disaster plans are developed from a common template. In general, they are not. Some municipalities use a common starting point; others develop their own disaster plan from scratch. It raises the question how comparable these disaster plans actually are.

Another question is to what extent disaster plans are followed when incidents occur. The identification of differences between the occurrences during an incident and the specification of the disaster plan is of particular importance for the detailed analysis of incidents and, as a result, improvement of incident management (e.g., by performing dedicated training sessions, and possibly by making necessary corrections in disaster plans). The data about the actual events and actions occurring during the incident management process are often available in the form of informal logs. Since the manual analysis of such logs is a time-consuming and error-prone process, tools for the automated analysis would be of use.

This paper presents an approach to support incident management based on disaster plans. The contribution of the approach is threefold: First of all, the paper presents a method to formally describe disaster plans. Using this formal description, disaster plans of different municipalities can be checked for consistency, to avoid problem that could arise once these municipalities have to combine their forces to manage an incident. Furthermore, the formal description of the disaster plans allows for the automated verification of such disaster plans against the empirical data that describe incident management processes occurred in reality.

To illustrate the proposed approach two disaster plans of municipalities of Eindhoven [4] and Uithoorn [5] have been used as a case study for this paper. Eindhoven is a relatively large city in the Netherlands with approximately 200,000 residents. A large scale aviation accident occurred at the airport in 1996 of which logs have been obtained [8]. Uithoorn is a much smaller town than Eindhoven. However,

Uithoorn belongs to a group of municipalities including Amsterdam and 6 surrounding municipalities that base their disaster plans on a common template.

The paper is organized as follows. In Section 2 the formal specification method for disaster plans is presented, whereas Section 3 shows how such formal description of different disaster plans can be compared. Thereafter, Section 4 addresses the verification of formal properties obtained from disaster plans against logs. Finally, Section 5 is a discussion.

2 Formal Specification of Disaster Plans

This section provides some general guidelines for extracting a formal model of the disaster plan from a textual disaster/incident plan and thus bridging the gap between informal and formal representation. In principle, any modeling approach for organizations and any formal language for modeling organizations can be used as a point of departure. For example, a formal language based on description logic for specifying disaster management is introduced in [6]. In this paper, the modeling approach of [3] and [7] based on an order-sorted predicate logic [11] is used for formal modeling of the structure of an incident management organization. Based on the formal structural description from a disaster plan, different scenarios of organizational behavior can be specified and analyzed, using for example the Temporal Trace Language [9].

The formal description of the incident management organization (identified by a name of sort ORGANIZATION) is associated with the disaster plan in which it is specified by the following predicate:

is_based_on: ORGANIZATION x DISASTER_PLAN.

Based on experience in modeling disaster plans the following stages are advocated: phase identification, structure analysis and modeling, task and responsibility analysis, organizational change modeling. Each of these stages is explained in more detail. The comparison of disaster plans is discussed after the modeling steps.

2.1 Phase Identification

In each disaster plan a number of phases of incident management are identified. Typically they are grouped in three general phases depending on the severity of the situation:

- Small incident – no co-ordination between police, fire department and medical forces is needed, the highest level of decision-making and co-ordination only involves functionaries of these three institutions.
- Serious incident – involvement of the mayor is needed at the highest level of decision-making. Typically a disaster management team is formed at the city hall.
- Severe incident involving more than one municipality – co-ordination between the municipalities is needed. Typically the National Coordination Centre is also involved.

The first step in this modeling approach is to identify which particular phases are covered by the disaster plan. The Eindhoven disaster plan identifies five phases: (1) Local incident; (2) Local calamity or disaster; (3) Local incident, calamity, or disaster with use of regional coordination; (4) Inter-local incident; and (5) Inter-local calamity or disaster.

With each phase an organization structure (denoted by its name of sort ORGANIZATION) is associated. For this the following predicate is used:

is_organization_in_phase: ORGANIZATION x PHASE is introduced.

2.2 Structure Analysis and Modeling

Each phase of incident management has its own organizational structure. Therefore, the structure of the organization has to be analyzed and modeled for each phase. Structure analysis aims at identifying all parties involved and their relevant organizational roles and relationships.

- Disaster plans typically contain lists of all parties involved. Institutions like the fire department, ambulance services, police, municipal service and other associated institutions are almost always involved. These institutions exist irrespective of whether an incident occurs or not. However, disaster plans also refer to parties like the operation team, regional coordination centre, and management team, depending on the phase and scope of a disaster/incident and only exist during these phases. The structure can consist of roles that contain other roles and so forth.
- After identifying the roles in the organization at a certain phase, the communications between roles or composite roles need to be identified. For example, a policy team always maintains communication with fire department action centre. With respect to communication and interaction the disaster plans studied by the authors are typically incomplete, making it difficult and in some cases impossible to identify the exact links in the structural model.

The structure of an incident management organization can be described at different aggregation levels, which allows managing the level of complexity and refinement of an organization representation. The aggregation levels refer to a level of the organization consisting of roles and the interaction between those roles. A model of an organization with several aggregation levels also contains a specification of the inter-level relations of those aggregation levels. Therefore, a model of an organizational structure consists of roles, interaction links, interlevel links, and structural properties regarding those elements.

(1) A *role* represents a subset of functionalities, performed by an organization, abstracted from instances of real agents. At the highest aggregation level, the whole organization can be represented as one role. Further, each role can be decomposed into several other roles, until the necessary level of aggregation is achieved. Graphically, a role is represented as an ellipse with white (input interfaces) and black (output interfaces) dots (see Figure 1). A role which is composed of (interacting) subroles, is called a *composite role*. Each role has an input and an output interface, which facilitate in the communication with other roles. Although in this paper the

emphasis is on the organization structure of incident management, an organization is realized by the agents (or sets of agents) fulfilling the roles.
(2) An *interaction link* represents an information channel between two roles. Graphically, it is depicted as a solid arrow, which denotes the direction of possible information transfer. For example, interaction links between roles Fire Department and Police in Figure 1 represent the possibility of communication between them.
(3) An *interlevel link* connects a composite role with one of its subroles. It relates two adjacent aggregation levels. Graphically, it is depicted as a dashed arrow, which shows the direction of interlevel transition (see Figure 1).
(4) Structural properties specify the number of instances of a specified role and the various role-subrole relations. Although the structure of an organization can be specified partly using graphs (see Figure 1), a formal textual language is needed to specify the structural properties. Sorts are introduced for the basic elements of an organization and relations between them (i.e., ROLE, AGENT, ORGANIZATION, INTERLEVEL_LINK, and INTERLEVEL_LINK). Furthermore, a set of relations is defined to specify the structural aspects of the organization. A complete overview is given in [3], here only a few examples are given:

is_role_in: ROLE x ORGANIZATION identifies a role in an organization

has_subrole_in: ROLE x ROLE x ORGANIZATION defines a subrole of a composite role in an organization.

Examples of structural properties are: is_role_in(FD,ORG1), has_subrole_in(FD,VC_FD,ORG1).

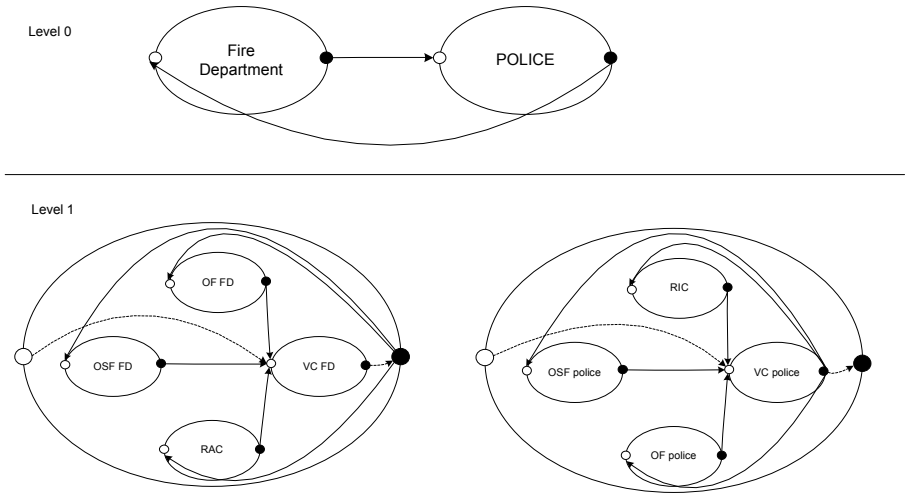

Fig. 1. Example of an organization structure, described at two adjacent aggregation levels

Often, structural properties are valid during the whole period of organization existence and can be considered as static. But in rapidly developing and adapting organizations (e.g., incident management organizations) structural change processes

gain special importance. Structural properties for such organizations will be described later.

For each of the phases identified in the previous step, the structure of the organization has been identified; only the second phase is presented in this paper, see Figure 2.

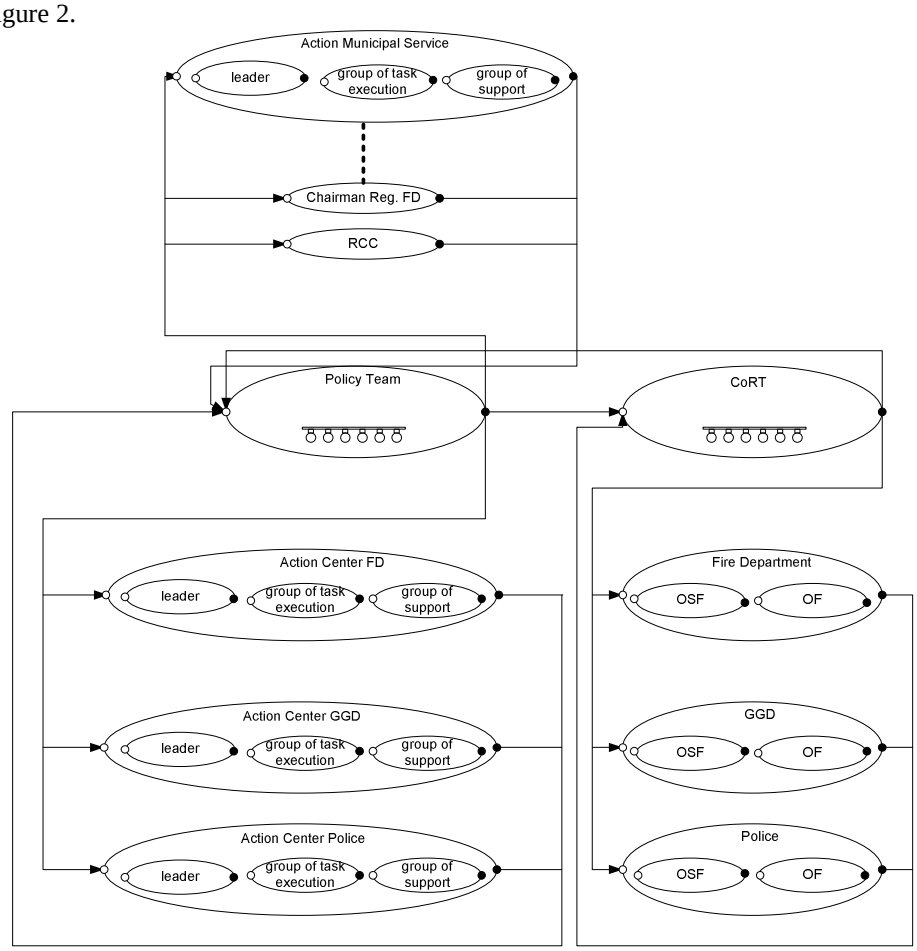

Fig. 2. Structure of the Eindhoven disaster prevention organization in the Local Incident phase.

The abbreviations used in the Figure are the following: OSF stands for On Scene Forces, Off Scene Forces are abbreviated to OF and GGD is an abbreviation for the Medical Services. Finally, CoRT stands for Command Disaster Area. Inter-level connections between composite roles and their subroles are often omitted because the disaster plan does not specify any of these relationships. A partial specification of this Figure in the formal language as presented is shown in Figure 3 in Section 2.5.

2.3 Tasks and Responsibilities Analysis

Having identified the organizational structure in the different phases of incident management, the tasks and responsibilities of the roles have to be determined. Problems at this stage might be vague and unclear formulations of the tasks, no detailed information for the responsibilities per task and per role.

The dynamics of an organization are formed by the execution of tasks by the organization and the change of an organization. To analyze and model the first of these, the tasks and responsibilities of the different structural elements of the organizational model have to be identified. An ontology based on the order-sorted predicate language is introduced that provides a way to express statements describing the hierarchy of tasks, responsibilities of roles for certain tasks in a particular situation and leadership within a composite role. The introduced ontology is useful for any organization that encounters change on a regular basis.

The main sorts are TASK, PHASE, ROLE, and ORGANIZATION. Using these sorts, the language can be extended with a set of relations to specify tasks, responsibilities and the phases of an organization.

- Primary co-ordination of task – which role co-ordinates the execution of the task
- Secondary co-ordination of a task – in some situations the primary co-coordinating role can be replaced by the secondary co-coordinating role. That might happen for example when the particular type of disaster has specifics that can more appropriately be handled by the secondary co-coordinating role.
- Primary execution of a task – the role(s) that execute the task
- Secondary execution of a task – for particular disasters where the emphasis is shifted towards an institution (role) not involved in the primary execution of the task, this institution can also become involved in it.
- Operational leadership within a complex role – the role that takes the leadership of the complex role (group, institution, etc.)

To specify such information the following relations re introduced:

is_subtask_of_in: TASK x TASK x ORGANIZATION, to describe the task-subtask ordering in the organization.

executes_task_primary_in: ROLE x TASK x ORGANIZATION, describes which role is the principle performer of a task in the organization.

executes_task_secondary_in: ROLE x TASK x ORGANIZATION, describes which role is the secondary performer of a task in the organization.

coordinates_task_primary_in: ROLE x TASK x ORGANIZATION, describes which role is the principle coordinator of a task in the organization.

coordinates_task_secondary_in: ROLE x TASK x ORGANIZATION, describes which role is the secondary coordinator of a task in the organization.

operational_leadership_in: ROLE x ROLE x ORGANIZATION, describes which role is the leader in a part of the organization.

From the analysis of the disaster plans considered so far a certain level of similarity in the task and process hierarchy has been discovered. This indicates that it is possible and beneficial to build a general ontology of tasks in disaster situations. A partial one was built on the information available from these two disaster plans and it is considered to analyze more in order to adjust and refine the ontology.

Some examples of structural relations from the Eindhoven disaster plan are: the fire department is in charge of the task of fighting the fire, the police is responsible for evacuating the people, and the medical services are responsible for collecting contaminated goods. These examples are formally represented in Figure 3 in Section 2.5.

2.4 Organizational Change Modeling

Knowing the organizational structures during the different phases of incident management is not enough to model a disaster plan. The last but vital part of the modeling is the specification of organizational change. This entails the identification of all conditions of organizational change. They normally depend on the different incidents/disasters. Typical problems that occur during this phase are lack of information concerning the triggers that cause change. Often the decision to change the organization is left to a deliberation group without stating specific definitions of the triggers.

The modeling process delivers a lot of information concerning how thoroughly a disaster plan is specified. In case some unclear parts are identified, the disaster plan can be improved in a number of ways, e.g., using experts and/or training. Another option is to organize a training dedicated to an unclear part.

The disaster plan of Eindhoven is vague about organizational change: it is left to the mayor and its advisors to decide on the appropriate phase. However, the triggers can be derived by comparing the definitions of each of the phases. For example, going from phase 1 (a local incident) to phase 2 (a local disaster) means that the public is actually seriously threatened. The change of organization involves the following elements: An operational team is added to the organization which is responsible for the action centers of the regional emergency services. Furthermore some of the communication lines are changed.

To formally specify changes to be performed within an organization the language shown is used. The language takes as a basis the structural language as introduced before and the responsibilities and tasks language as defined in the previous section. Sorts used to represent these elements are STRUCT_ELEMENT and RESPONS_TASK_ELEMENT. The sorts are combined into the sort ORG_ELEMENT. Functions are defined for adding, deleting and modifying an organization element (which can also be seen as a combination of add and delete):

- add: ORG_ELEMENT → ORG_CHANGE_ELEMENT, describes an organizational element being added.
- delete: ORG_ELEMENT → ORG_CHANGE_ELEMENT, describes an organizational element being deleted.

modify: ORG_ELEMENT x ORG_ELEMENT → ORG_CHANGE_ELEMENT, describes that the first organization element is modified to the second argument.

Besides the need to specify *what* needs to be changed also the triggers that *cause* the change need to be formally specified. For this the following predicate is introduced:

is_trigger_for_from_to: TRIGGER x ORG_CHANGE_ELEMENT x PHASE x PHASE, describes that when a trigger occurs the phase is changed (if necessary) from the present phase to some other phase, and the organization is changed according to the specification defined in ORG_CHANGE_ELEMENT.

Examples of the use of this ontology are shown in Section 2.5 below.

2.5 Example Formal Description

Figure 3 shows a part of the formal specification of the disaster plan of Eindhoven, covering each of the aspects as addressed in this section.

3 Comparing of Disaster Plans

A comparison of disaster plans consists of the following elements: comparison of phases, comparison of organizational structures in comparable phases, comparison of the task structure in comparable phases, and comparison of the responsibilities scheme in comparable phases. The comparison of phases is a rather straightforward matter. Comparison of the organizational structures entails the identification of comparable and incomparable structures within the organization at each of the phases of incident management, and a comparison of the ontologies used. The comparison of task structures concentrates on the tasks identified in each disaster plan, and discusses comparable and incomparable tasks. Given the comparable tasks, the comparison of responsibilities entails the allocation of responsibilities to roles. This Section presents the results of the comparison of the two disaster plans as introduced before.

For the purpose of comparison of the disaster plans described above a number of relevant properties have been identified. These properties constitute two groups: (1) local municipality properties and (2) regional coordination properties. The first group describes properties that do not influence the incident management organization of other (neighboring) municipalities and can therefore differ between these municipalities. Properties in the second group do influence the incident management organization of other municipalities. In case of an inter-local incident these kind of properties have to be the same to enable a proper functioning of the disaster management organization.

Consider an example of local municipality properties.

Property 1.
Informal form
The command centre of surroundings of the incident area (ComRT) is a part of the incident management organization of municipality X in phase 4.

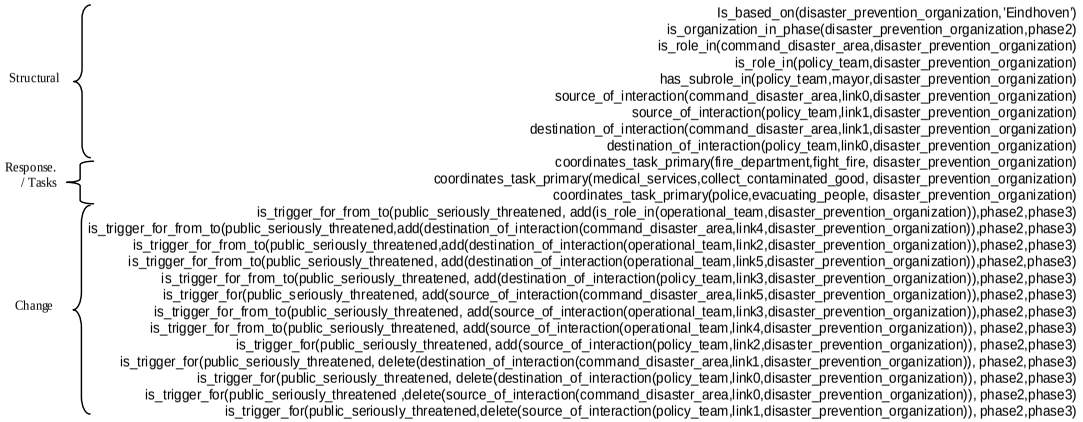

Fig. 3. Part of a formal specification of a disaster plan

Formal form

[is_role_in(ComRT,ORG4) ∧
  is_based_on(ORG4,X) ∧
  is_organization_in_phase(ORG4,PHASE4)]

This property holds for X = Uithoorn and does not hold for X = Eindhoven.
Consider two examples of regional coordination properties.

Property 2.

Informal form

The mayor of the biggest municipality coordinates the work of the Managing Platform Centre (MPC) in the incident management organization of municipality X in phase 4.

Formal form

[coordinates_task_primary_in(biggest_municipality_mayor, regional_collaboration_in_MPC, ORG4) ∧ is_based_on(ORG4, X) ∧ is_organization_in_phase(ORG4,PHASE4)]

This property holds for X = Uithoorn and does not hold for X = Eindhoven.

Property 3.

Informal form

The mayor of the municipality that was the first involved in an incident, coordinates the work of the Managing Platform Centre (MPC) in the incident management organization of municipality X in phase 4.

Formal form

[coordinates_task_primary_in(mayor_involved_first, regional_collaboration_in_MPC, ORG4) ∧
  is_based_on(ORG4,X) ∧ is_organization_in_phase(ORG4,PHASE4)]

This property holds for X = Eindhoven and does not hold for X = Uithoorn.

The formal approach in the comparison of disaster plans allows us to go further and analyze these differences and investigate whether they indeed lead to serious consequences. An example of such analysis is given in the following paragraphs. It is already known (see property 1) that the role ComRT is present in the disaster plan of Uithoorn but not in that of Eindhoven. This role represents the team responsible for activities in the surroundings of the disaster area including traffic regulation, isolation of the area, etc. In both plans the team CoRT is present which co-ordinates the on-scene operations. Is this difference fundamental? Maybe the tasks of ComRT for the case of Uithoorn are actually assigned to CoRT in the case of Eindhoven. This hypothesis is expressed in property 4, and decomposed into properties 5 through 8 to ease the formal proof process, as depicted in Figure 4. The formal relations are:

Property 5 ∧ Property 6 |= Property 4
Property 7 ∧ Property 8 |= Property 6

Only the formal specifications of the leaves of the tree in Figure 4 are given.

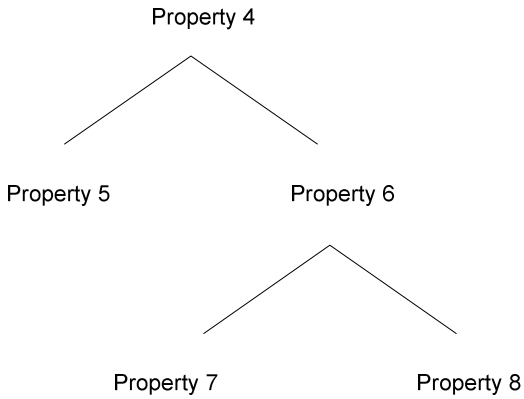

Fig. 4. The decomposition of property 4 represented in an and-tree

Property 4.
Informal form
The set of tasks assigned to CoRT in the disaster plan of Eindhoven is the same as the set of tasks assigned to CoRT or ComRT in the disaster plan of Uithoorn.

Property 5.
Informal form
All tasks of CoRT in the disaster plan of Eindhoven are either tasks of CoRT or of ComRT in the disaster plan of Uithoorn:
Formal form
∀ T:TASK ∀O:ORGANIZATION:
coordinates_task_primary_in(CoRT,T, O) ∧
is_based_on(O,'Eindhoven')
⇒ ∃O':ORGANIZATION [coordinates_task_primary_in(CoRT,T, O') ∨
coordinates_task_primary_in(ComRT,T, O')] ∧ is_based_on(O','Uithoorn')

Property 6.
Informal form
All tasks of CoRT or ComRT in the disaster plan of Uithoorn are also tasks of CoRT in the disaster plan of Eindhoven.

Property 7.
All tasks of CoRT in the disaster plan of Uithoorn are also tasks of CoRT in the disaster plan of Eindhoven.
∀ T:TASK ∀O:ORGANIZATION:
coordinates_task_primary_in(CoRT,T,O) ∧
is_based_on(O,'Uithoorn')
⇒ ∃O':ORGANIZATION coordinates_task_primary_in(CoRT,T,O') ∧ is_based_on(O','Eindhoven')

Property 8.
All tasks of ComRT in the disaster plan of Uithoorn are also tasks of CoRT in the disaster plan of Eindhoven.
∀ T:TASK ∀ O:ORGANIZATION:
coordinates_task_primary_in(ComRT,T,O) ∧ is_based_on(O,'Uithoorn')

$\Rightarrow \exists O'$:ORGANIZATION coordinates_task_primary_in(CoRT,T,O') $\wedge$ is_based_on(O','Eindhoven')

By checking properties 5, 7 and 8, it is discovered that the functions of CoRT in the case of Eindhoven and CoRT and ComRT in the case of Uithoorn indeed overlap. Therefore, while the absence of ComRT is certainly a difference between the two disaster plans, in reality the difference is smaller than expected at first sight.

The comparison between the disaster plans of Uithoorn and Eindhoven revealed two differences in the regional coordination. The first concerns leadership: which mayor is in charge of the disaster management organization in case of an inter-local incident. The Uithoorn plan states that the mayor of the biggest municipality is the leader. The Eindhoven plan states that the mayor of the municipality where the incident started is in charge. Imagine that these are neighboring municipalities and that an incident that affects both municipalities is first discovered in Uithoorn, which is the smallest municipality of the two. According to the Eindhoven disaster plan Uithoorn remains in charge, and therefore does not take any initiative in forming an inter-local incident management organization. Uithoorn however thinks Eindhoven will take the initiative as it is the biggest municipality involved in the incident. To prevent this kind of errors, such differences should be avoided. The second regional coordination difference concerns the incident phases described in the disaster plans. There does not exist a one-to-one mapping between these phases, therefore the municipality that has the lead in the incident management organization might declare a certain phase that cannot be interpreted by the other municipalities involved. For example, in the Uithoorn disaster plan, a phase is present where there is multi-disciplinary coordination without the mayor being involved. In the Eindhoven disaster plan there doesn't exist any phase including multi-disciplinary coordination in which the mayor is not involved in the disaster prevention organization.

Differences in local municipality properties were also observed in the comparison of the disaster plans. These differences include elements such as splitting up the command of the disaster area in the disaster plan of Uithoorn while this remains one group in the Eindhoven disaster plan. These difference can however be formally mapped to each other, and are therefore not as crucial.

4 Verification of Disaster Plan Properties Against Logs

In order to determine to which extent disaster plans are followed in reality, when incidents occur, an automated verification method is proposed. By means of this method, the formal specification of a disaster plan is checked automatically on formalized empirical data concerning an incident. This empirical data are usually represented in the form of informal logs (also called *traces*) that contain events. Such informal logs can be formalized using the formal language TTL [9]. The translation from a log of events to a formal trace is currently done by hand. However, for the future there are plans to develop a methodology that supports non-expert users in making this translation. After such a formalization of a log has been created, the formal properties extracted from the disaster plan can be automatically verified against the formalized trace. This Section first of all shows what such a formalized

trace looks like, and thereafter presents results of checking the properties obtained from the Eindhoven disaster plan to the logs of the Hercules airplane crash in 1996.

4.1 Formalizing an Empirical Trace

An example of a formalization of a trace is shown in Figure 5. It shows the most relevant parts of the occurrences during the Hercules incident. The ontology used in the trace is identical to the one introduced in Section 2 on formally describing a disaster plan. In the left side of the figure, the relevant so called *atoms* in the trace are shown whereas the right part represents a time line. In the time line a black box indicates that the atom is true whereas a grey box indicates that it is false.

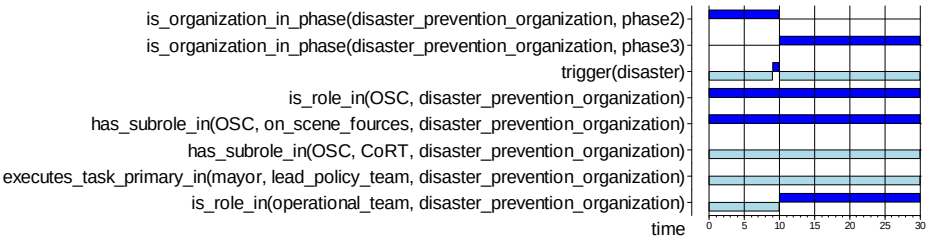

Fig. 5. Partial Empirical Trace of the Eindhoven Plan Crash

As can be seen in the trace, from time point 0 to 10 the phase declared is phase 2 whereas between 10 and 30 phase 3 holds. Furthermore, at time point 9 a trigger is observed for changing the organization, namely that the current situation has been declared a disaster. The partial structure of the organization at different time points is shown in the figure as well. During the entire incident, the OSC (for On Scene Commander) role is part of the organization and of the On Scene Forces group. Furthermore, the OSC is never part of the Command Disaster Area (abbreviated to CoRT in the trace). Finally, the operational team role is added to the organization from time point 10 and on.

4.2 Verification of Properties Against a Formalized Trace

After having obtained a formalized trace, properties extracted from the disaster plans can be verified against such a trace. By means of this verification one can determine what part in the example incident management process described by the trace did not follow the disaster plan.

For such verification, based on the formal representation of the disaster plan, a set of facts is defined in the form follows_from_disaster_plan(X), where X is a relation from the formalized disaster plan. Then, based on the identified facts dynamic properties are specified that can be verified on the formalized empirical trace by means of the dedicated software environment *TTL Checker*. To enable automated verification, dynamic properties should be expressed by formulae in the Temporal Trace

Language. The software environment takes a TTL formula and one or more traces as input, and checks whether the formula holds for the trace(s).

Below, a number of dynamic properties in the form of TTL formulae are considered, based on the disaster plan for the Eindhoven municipality. These properties have been checked automatically on the formalized empirical trace, a part of which is depicted in Figure 5.

First of all, it is checked whether the organizational structure in the different phases indeed corresponds to the disaster plan. Note that this property only concerns the subrole relationship, similar properties can be specified for the other structural relationships.

Property 9.

Informal form

For all time points t in trace γ, if the phase at time point t is P, and the disaster plan specifies that a particular role R2 should have a subrole R1 in organization O in phase P, then role R1 is indeed a subrole of role R2 in organization 0 at time t.

Formal form

∀t:TIME, ∀R1,R2:ROLE, ∀P:PHASE, ∀O, O':ORGANIZATION:
[[state(γ, t) |= is_organization_in_phase(O, P) &
follows_from_disaster_plan(has_subrole_in(R1, R2, O'))) &
follows_from_disaster_plan(is_organization_in_phase(O', P))]
⇒ state(γ, t) |= has_subrole_in(R1, R2, O)]

The relation state(γ, t) |= p denotes that within the state state(γ, t) at time point t in trace γ the state property p holds. This property is not satisfied in the given trace, because the OSC role should be part of the CoRT role in both phase 1 and 2 according to the disaster plan, whereas it is not in the trace.

A second property concerns the checking whether the tasks and responsibilities mentioned in the disaster plan are indeed performed. Again, this property just shows an example of how to check one relationship for the tasks, the rest of the relationships can be checked in a similar fashion.

Property 10.

Informal form

For all time points t in trace γ, if the phase at time point t is P, and the disaster plan specifies that a particular role R should be the primary executer of a task T in phase P, then role R is indeed the primary executer of this task T at time t.

Formal form

∀t:TIME, R:ROLE, ∀P:PHASE, ∀T:TASK ∀O, O':ORGANIZATION:
[[state(γ, t) |= is_organization_in_phase(O, P) &
follows_from_disaster_plan(executes_task_primary_in(R, T,O')) &
follows_from_disaster_plan(is_organization_in_phase(O', P))]
⇒ state(γ, t) |= executes_task_primary_in(R, T, O)]

This property is again not satisfied, as the mayor role should be the primary executer of the task to lead the policy team, whereas he does not perform that task.

A final property which has been checked against the trace is to investigate whether the organizational change processes in the organization have been successful, as shown in property 11.

Property 11.
Informal form
For all time points t in trace γ, if the phase at time point t is P and a trigger T holds, and furthermore the disaster plan specifies that in phase P given trigger T a new phase P2 should hold, and roles should be added, then at a later point in time t2 phase P2 will be the case, and the organizational element will have been added.

Formal form
∀t:TIME, ∀OL:ORG_ELEMENT, ∀P1,P2:PHASE, ∀T:TRIGGER ∀O, O':ORGANIZATION:
[[state(γ, t) |= is_organization_in_phase(O, P1) &
  state(γ, t) |= trigger(T) &
  follows_from_disaster_plan(is_organization_in_phase(O', P1)) &
  follows_from_disaster_plan(is_trigger_for_from_to(T, add(OL:ORG_ELEMENT), P1, P2))]
⇒ ∃t' t'>t [state(γ, t') |= ORG_ELEMENT & state(γ, t') |= is_organization_in_phase(O, P2)]]

This property is satisfied in the trace. The phase transitions do go according to the disaster plan. The initial organization however is, as has already been stated, not correct. Since the change is only concerned with transitions between phases, this property does hold.

5 Discussion

In this paper a formal framework for modeling and comparing disaster plans and checking disaster plans on empirical traces is presented and applied to a number of case studies. The framework extends earlier work of [7] and [3] with specific constructs and reusable patterns for the domain of incident management, in specific for disaster plans. The approach uses formal graphical, and textual languages, in casu sorted first-order predicate logic and TTL (see [9]). More specifically, sorted first-order predicate logic is used for formalizing structural properties in disaster plans and TTL is used for expressing causal temporal properties for the automated verification on formalized empirical traces by means of the dedicated software.

When compared with the work of [6] the framework presented in this paper is more generic from several perspectives. The first advantage is that the framework allows modeling on different levels of abstraction, and is, therefore, capable of modeling the Dutch disaster plans, which are on a highly abstract level of abstraction when compared to the plans that [6] modeled. The second advantage is that simulation of the models in different situations is possible. The third advantage is the software support for checking the model against simulation and transcribed real traces.

[12] introduce an approach for the verification of properties against simulation traces of an agent-based system which models human behavior in incidents. They do however not address using empirical logs from the incident management field within their work. Furthermore, the paper work does not concern the formal specification of disaster plans and automated verification of the properties described in such plans, which is one of the main contributions of this paper.

With respect to incident management this work contributed by proposing a formal approach for the modeling and comparison of disaster plans. The approach is

explained in detail and tested in two case studies. The main results are the classification of differences into local differences and inter-local differences. The local differences effect only incident management in the municipality itself. The local differences can be fundamental or not when comparing the actual incident management. For example, two disaster plans differed in having only one or two zones around the epicenter of the incident. This difference has clear effects on the organizational structure prescribed in the disaster plans. However, the tasks associated with the zones are comparable. The same holds for the associated responsibilities. In other words, the organizational structure differs, but the dynamics are comparable. The inter-local differences are counterproductive when municipalities have to cooperate in case inter-local incidents. Comparing two disaster plans in this manner revealed a possible conflict regarding leadership. The consequence is clear: all neighboring municipalities should use the same rules for determination of leadership. Therefore, all municipalities in The Netherlands should share those rules.

In the future, systems such as the IMI system [10] will contain many disaster plans. Making sure that these disaster plans are consistent with each other is of crucial importance for inter-local incident management. The plans in the system can be formalized, and verifying whether a new plan is consistent with the plans currently in the database would simply entail formalizing that plan and performing verification. In case the plan is indeed consistent the plan can be added to the database, including the formal description. On the long run an entirely different approach can be followed. Instead of taking an informal disaster plan as a point of departure, in future disaster plans should be first and foremost formal plans, from which an informal plan that is readable for human beings is automatically generated.

References

[1] Abbink, H., Dijk, R. van, Dobos, T., Hoogendoorn, M., Jonker, C.M., Konur, S., Maanen, P.P. van, Popova, V., Sharpanskykh, A., Tooren, P. van, Treur, J., Valk, J., Xu, L., Yolum, P., Automated Support for Adaptive Incident Management. In: Walle, B. van de, and Carle, B. (eds.), *Proc. of the First International Workshop on Information Systems for Crisis Response and Management, ISCRAM'04.* Brussels, 2004.

[2] Breuer, K., Satish, U. Emergency Management Simulations-An approach to the assessment of decisionmaking processes in complex dynamic environments. In Jose J. Gonzalez (eds), From modeling to managing security: A system dynamics approach, HoyskoleForlaget, 2003, pp. 145-156.

[3] Broek E. L. van den, Jonker, C.M., Sharpanskykh, A., Treur J., and Yolum, P., Modeling and Analyzing Multi-Agent Organizations (Submitted to Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'05)

[4] Gemeente Eindhoven, Rampenplan, Eindhoven, the Netherlands, May 1993.

[5] Gemeente Uithoorn, Rampenplan, Uithoorn, the Netherlands, October 2003.

[6] Grathwohl, M., de Bertrand de Beuvron, F., and Rousselot, F., *A new application for description logics: Disaster management.* In Proc. of the International Workshop on Description Logics '99, Linkoping, Sweden, 1999

[7] Hoogendoorn, M., Jonker, C.M., Schut, M., and Treur, J., Modelling the Organisation of Organisational Change. In: Proc. of the Sixth International Workshop on Agent-Oriented Information Systems, AOIS'04.

[8] Inspectie Brandweerzorg en Rampenbestrijding, *Vliegtuigongeval Vliegbasis Eindhoven 15 juli 1996*, SDU Grafische Bedrijf, The Hague, 1996.
[9] Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
[10] Lee, M.D.E. van der, Vugt, M. van. IMI – an information system for effective multidisciplinary incident management. In: Carlé, B., Walle, B. van der (eds.), Proceedings of the International Workshop on Information Systems for Crisis Response and Management '04, Brussels, Belgium. 2004.
[11] Manzano, M. Extensions of First Order Logic, Cambridge University Press. 1996.
[12] Narzisi, G., Mysore, V., Nelson, L., Rekow, D., Triola, M., Halcomb, L., Portelli, I., and Mishra, B., Complexities, Catastrophes and Cities: Unraveling Emergency Dynamics. In: International Conference on Complex Systems (ICCS 2006), Boston, MA, USA June 25-30, 2006.

Part VII
Discussion

Chapter 18

Conclusion

Conclusion

The research presented in this thesis covers a broad spectrum of the issues involved in organizational change. The spectrum ranges from the analysis of the performance of an existing organization, to the evaluation of an organizational change process, and of the steps in between.

The *analysis* of the performance of an existing organization is addressed from various perspectives, ranging from an analytic approach based upon labeled graph for the performance of organizations (Chapter 2), to models that identify the precise problems in the organization and suggest a possible new organization (Chapter 3, 4, and 5). To specify models that determine a possible new organization, a number of techniques are used, inspired by meta-reasoning (Chapter 3), the domain of organizational design (Chapter 4), and the algorithmic domain of graph theory and max flow networks (Chapter 5). Each technique presents a particular strength in the analysis. Chapter 2 focuses mainly on an extensive analysis of the detailed aspects of task performance, but does not address how the organization can be changed to perform correctly (if necessary). Chapter 5, using a graph based approach, abstracts away from the specific tasks in the analysis, to suggest an update to an organizational model based upon the problems identified. Specification of such an update mechanism using the extensive analysis method presented in Chapter 2 would certainly be possible using the approach presented in Chapter 5 by considering a separate graph for each specific task. In Chapter 3 the possible changes in the organization are assumed to be pre-specified, thereby mainly focusing on the choice between various possibilities to change the organization. Finally, Chapter 4 focuses on the specification of requirements of an organization based upon changing environmental conditions, It presents a specific component-based model for specifying knowledge on how such changes within the environment can be refined to a more operational level and how an organization can be changed in order to fulfill such requirements. The broad spectrum of techniques offered is shown to be effective in several domains, such as incident management, the naval domain, and large scale manufacturing organizations. The analysis techniques can easily be reused due to the abstraction level that has been used in the specification of these techniques. The combination thereof provides a basic set of reusable tools that enable analysis of the current organization and an investigation of potentially new organizations before going into the actual change process. As a result, these techniques contribute to a more effective analysis of the problems in organizations, resulting in more effective change processes as well.

Of course, a description of a potential new organization does not immediately result in that new organization being in place. The process of moving from one organization to another has therefore been addressed in this thesis as well. Again, it is addressed from several perspectives based upon the way the change process is directed. First of all, *centralized change processes* are addressed which focus on some central entity that directs the change process. Centralized change models are specified without addressing the specific agents, which makes them highly generic and reusable. For the creation of centralized models, inspiration was found in organization theory, due to the body of research that exists there. As a result, a model is created

based upon a popular theory for moving from one organization to another (Chapter 6). In order to evaluate the usefulness of the change model it has been evaluated in a number of case studies. For the naval domain, the model is used to describe and analyze changes in fleet formation (Chapter 7), whereas the model is also used in the domain of government organizations (Chapter 8). The evaluation of the model in these specific domains shows that an analysis using the model is useful, which was also acknowledged by for instance government employees that were involved in the modeling of their organizational change processes. Some domains however, do not have a central authority that can direct change, therefore, *decentralized change processes* are addressed in this thesis as well. In domains such as biology, such change processes are frequently observed, therefore, a model has been created for decentralized change drawing inspiration from honeybee colonies (Chapter 9). To make this model even more generic and reusable, a higher abstraction level is introduced, and specializations thereof, including both quantitative and qualitative specializations (Chapter 10). The model is shown to describe human organizations as well, namely in the field of incident management. Besides the specification of a reusable model for decentralized organizational change, the issue of populating such an organization is addressed as well by means of negotiation (Chapter 11 and 12). In these negotiation processes and strategies that are specified, both the preference (Chapter 11) as well as the efficiency of the solution (Chapter 12) are addressed. The different strategies are shown to be effective, and are thoroughly analyzed by using actual company data. As a result, both a model is specified that can be used by organization modelers who want to specify an organization exhibiting decentralized organization change, as well as approaches for the formation such an organization in an effective manner. Finally, also approaches are defined in this thesis that are generic in the sense that they can be used for both centralized and decentralized organization change; *mixed change processes*. First of all, an approach is presented which is able to evaluate centralized and decentralized change processes (Chapter 13), which is of crucial importance when designing an organizational model. Strategies that have been tested include well known approaches for coordination in an organization. In order to specify such coordination approaches, an extensive language is presented as well (Chapter 14).

The final topic addressed in this thesis is the *evaluation* of the effectiveness of organizational change. This has been addressed from the viewpoint of verifying traces of the functioning of the changed organization. Actually, this topic has been addressed as well in other parts of this thesis. For the domain of incident management however, dedicated approaches have been created that enable an analysis of critical event occurrences in such organizations. First of all, two approaches are presented that specify properties an incident management organization should satisfy (Chapter 15 and 16). The verification of such properties against empirical logs results in the identification of specific errors that need to be addressed. Chapter 15 thereby mainly focuses on the formalization of an empirical trace and the identification of what kind of properties to verify against an empirical trace. Chapter 16 uses the approach presented in Chapter 15 and extends it with verification using hierarchies of properties. Furthermore, Chapter 16 also identifies the types of errors that can be made by agents when they are playing a particular role. The domain of incident management is a suitable domain to perform these evaluations due to the logging of

information (i.e. disaster reports), as well as the description of properties that should hold within such organizations, which is done in great detail (i.e. disaster plans and disaster prevention plans). Finally, also plans for organizational change are evaluation, in the form of the evaluation of disaster plans (Chapter 17). The evaluation approaches presented are generic in the sense that they can easily be applied in other domains as well. Using such approaches after a change has been performed gives a good insight on the effectiveness of such a change, and can as a result be used by organizational experts and modelers as well.

To summarize, the contribution of this thesis is the introduction of models and techniques to describe and improve change within organizations. A broad spectrum of such models and techniques has been presented in this thesis, both focusing on the specification of the models and techniques themselves, as well as on the analysis thereof. To demonstrate the variety of applicability of these models and techniques, domains including social insects, business organizations, computer organizations, and government organizations have been used as case studies. The models and techniques have all been specified in a generic manner allowing for reuse of these models and techniques, and furthermore, they have been analyzed in some depth. Practitioners in the field of organizational change can use such models and techniques to improve the effectiveness of organizational change.

Chapter 19

Related Work

Related Work

The work presented in this thesis is of a multi-disciplinary nature. As a result, the work is related to the disciplines involved, i.e., economics, social science, computational and mathematical organization theory, biology, computer science/artificial intelligence. The discussion in this related work section only addresses work related to the thesis as a whole. For related work concerning the specific cases studied, the reader is referred to the discussion or related work section of that specific chapter.

This chapter is organized as follows. First of all, organizational change literature originating within human organizations is addressed. Thereafter, literature in the domain of organizational change within biology is presented and related to the work presented in this thesis. Furthermore, Section 3 concerns related work in computer science, and finally, in Section 4 the approach as a whole is compared to other organizational modeling approaches.

1 Change in Human Organizations

In human organizations, change is a part of everyday life. Due to rapid developments in society, change of organizations has become inevitable. Hence, some organizations are continuously undergoing change. Change in organizations is however not always successful. Both Hall *et al.* [1] and Bashein *et al.* [3] state that over 70% of all change processes do not achieve the intended goal. Boonstra [4] criticizes typical explanations given for failure of such change processes; he states that insufficient attention is paid to the change process itself. This thesis tries to address both the processes before and after change of an organization as well as the process of change itself, addressed in parts iii to v.

In general, three main types of organizational change are identified in literature (see e.g. [4]): *planned organizational change*, in which the problems and solutions of change are known. In *organizational development* the problems are known but not entirely clear in organizational development, whereas the solutions are not known, but there is at least an idea about the direction of search for solutions. Finally, *transformational change* is defined by Ackerman [1] as *"emergence of a totally new state of being out of the remains of the old state"*. As can be observed, the clarity of problems and solutions of change becomes more vague with the type of change. The types of change addressed in this thesis are both planned organizational change as well as organizational development. In this related work section, first of all the evaluation of organizations and the search for the optimal organization given the circumstances is addressed, thereafter the process of organizational change is discussed.

Evaluating an Organization. Evaluation of an organization can be performed by measurement of the effectiveness of an organization. Cunningham [9] for example introduces seven alternative strategies for assessing organization effectiveness and provides criteria when each one of these approaches can be used. Triggers for a change of an organization, due to a loss of effectiveness can for instance be found in Jaffee [21]. Once it is observed that an organization is indeed becoming less effective, a new organization needs to be designed, and such a design needs to be evaluated as well. These problems have for example been addressed in the field of contingency theory, which aims at finding the best organization given the environmental conditions under which it is functioning (see e.g. [12]). Furthermore, Mintzberg [28] for instance specifies the characteristics for which particular organizational forms are best suitable. In operations research (see e.g. [16]), an emphasis is put upon design of organizations in the most efficient manner. Work from these fields has been taken as a source of inspiration for the models presented in this thesis. The work in this thesis can also contribute to a more formal analysis of the functioning of an organization.

Organizational Change Processes. The organizational change processes addressed in this thesis (i.e. planned organizational change and organizational development) both concern a process of change that moves from one stable state to another. A vast amount of research has been performed concerning such change processes, see, e.g., [2;23;24;27].

From a centralized organizational change perspective, the theory used as a source of inspiration for a lot of this research is the three step change model introduced by Lewin [23]. Lewin identifies three phases within an organizational change process: *unfreezing*, *movement*, and *refreezing*. He states that there are two opposing forces at work when changing an organization: forces that resist the change, and forces that drive towards the newly desired organization. The unfreezing phase begins at the moment that change becomes necessary and consists of the process of changing the resisting and driving forces in such a way that change becomes possible (i.e., the driving forces outweigh the resisting forces). The actual change of the organization is contained in the movement phase in which the organization is moved from the current state to the desired stated. The refreezing phase involves freezing the newly formed organization so that there is no possibility to return to the former status quo or to continue changing in another unwanted direction. The whole re-organization process is completed when all phases have been completed. The unfreezing can be performed by increasing the driving forces and/or by decreasing the resisting forces. An example of an extension of Lewin's theory is that of Lippitt *et al.* [24], who identify seven steps based upon the three step model of Lewin. The centralized change model of Lewin is in this thesis taken as a basis for modeling centralized organizational change processes, and has been described in the modeling approach used throughout this thesis. Furthermore, it has been analyzed by means of logical simulation and verification. Such an analysis of the theory of Lewin has not been found in any other related work.

When looking at decentralized organizational change processes, the actual process of convincing people to move to a new organization does not exist, since the people themselves decide when and how to change. Examples of organizations exhibiting decentralized organizational change are the cellular organization [27], and the organic

organization [2]. This thesis presents models for such decentralized change processes that can potentially be used when modeling these types of organizations. In such decentralized organizational change process, negotiations are sometimes used to form an organization. In part iv of this thesis, two chapters are devoted to the analysis of the results found using the MAGNET negotiation system [7], both addressing satisfaction of the participants of the organization as well as the optimality of the solution found. Regarding the satisfaction of participants of the organization, new bidding algorithms have been proposed for negotiation that emphasize on the preference for particular tasks. When looking at the optimality of the solution found, new, efficient algorithms have been proposed as well.

2 Organizational Change in Biological Systems

Change in biology is studied for a number of aspects. Organizations are sometimes studied on a low level, such as the organization of chemical processes and their relation in a cell. Chemical processes in the *E.Coli* bacteria are, for example described by means of mathematical differential equations [31]. On a higher level however, change is also addressed from a social biological perspective. Wilson for example, focuses on colonies in social inspects, see, e.g., [17], and the roles and tasks that can be distinguished in such organizations. For honeybee colonies, the division of labor between worker bees, and the change thereof over time, has been of particular interest, see, e.g., [32;36]. Many other social insects have also been intensively described. Researchers in the field of computational biology use such mechanisms observed in social insects as source of inspiration for algorithms for self-organization (see, e.g., [35]). This thesis has aimed at describing the high level socio-biological processes, such as labor division on an organizational modeling level. In particular, the part covering decentralized organizational change processes has described and analyzed organizational models that draw inspiration from biological organizations and allows for an analysis of such processes. The approach used in this thesis that differs from the approaches used in for instance computational biology is the logical nature of the models presented in this paper, contrary to the models expressed by means of differential equations such as in use in computational biology.

3 Organizational Change in Software Systems

Due to the increase in complexity of software organizational views of software systems are used more often nowadays in computer science. Using the organizational views as an initial step, allocation of software agents (or components) to perform particular roles is a next step to be taken, also when changing an organization. A method that allows for specification on the organizational level with software engineering in mind is for instance GAIA [37].

A next step after having specified the organizational level of a software system is the allocation of agents to particular roles within a (changed) organization. This raises a number of research questions. First of all, which agent should be allocated to what

particular roles. Different approaches for role-allocation and reallocation algorithms are compared in [29]. The comparison is based on a framework developed for the Role-based Markov Team Decision Problem. Multi-agent negotiation is one particular discipline that can also be used to determine which agent is best to be allocated to a particular role. Just as in human organizations, auctions have also been proposed for allocation of computational resources [22]. One of the most popular paradigms in use is the contract-net protocol [34]. Other fields, in which allocation is addressed are centralized scheduling [6] and distributed planning [8]. Moreover, coordination algorithms have been proposed that allow for an effective allocation of agents to particular tasks. Maes [25], for instance, proposes a centralized algorithm that determines which agent is to become active, given the current state of the world. Other approaches such as voting [30] and the pandemonium [33] try to achieve the same, but are specified from a more decentralized perspective.

In this thesis the aim is to model and analyze such allocation mechanisms. An analysis methodology for coordination strategies as well as a language for the specification of such strategies has been presented in part v. The language allows for both the specification of centralized strategies as well as decentralized strategies, which makes the language unique. For the comparison of such strategies, a completely new approach has been developed to evaluate strategies. In addition, the work presented in part iv concerning negotiations introduces an allocation mechanism. The approach is however more aimed at supporting the formation of human organizations.

4 Organizational Modeling Approaches

A number of organizational modeling approaches exist besides AGR [13] extended with dynamic models [14]. GAIA [37] distinguishes five main elements for describing an organization: (1) the environment; (2) roles; (3) the interaction between roles; (4) organizational rules, and (5) organizational structures. As has been stated before, GAIA is motivated from the perspective of being a first step in the development of the implementation of a multi-agent system. It does however also allow for an analysis at the organizational level. Besides defining organizational structure, MOISE [20] specifies missions for roles, which can include concepts such as goals, plans, actions, and resources. Furthermore, authority links between roles can also be specified. OperA [10] identifies three models: the organizational model, the social model, and the interaction model. The approach describes the behavior of the organization as a whole, and the distribution of these objectives between different roles. Agent behavior is regulated by social contracts describing the role the agent is playing. In addition, interaction contracts describe actual interactions between agents.

The description of an organizational model by means of norms (see e.g. [5]) is used in several of these approaches. In the modeling approach used throughout this thesis, such norms can also be expressed. The following norm, for example (from [26]]) “Students are prohibited from sitting the exam if they have not completed the assignment” can easily be formulated in terms of a dynamic property for the student

role. In this way, the approach used in this thesis is suitable for modeling organizations in terms of norms.

In organizational modeling, organizational change has been addressed before. In [26] a framework is introduced which enables verification and analysis of organizational change. In the framework, changes in the organizational structure are allowed, however the process of organizational change itself is not addressed nor modeled, contrasting it to the work performed in this thesis. Their framework enables verification on the static organizational model to check whether the organizational model is workable. Furthermore, analysis is performed on simulations of possible outcomes of the organizational model, which is meant to see how the organization will act when populated by different societies of agents. The simulations used throughout this thesis abstract from the population of agents, and allows for the simulation of the organizational model itself. This avoids the development of agents to investigate the effectiveness of such organizational models. Dignum et al. [11], emphasize the necessity for multi-agent organizations to have the ability to reorganize, and state that additional requirements are needed for agents that have the ability to change. This thesis makes those requirements more concrete in the form of providing several models and templates for such organizations. In MOISE+ [19] reorganization is addressed as well. Four phases in a reorganization process are identified for controlled organizational change: (1) monitoring phase; (2) design phase; (3) selection phase, and (4) implementation. The reorganization is being controlled by an organization manager within a reorganization group. This architecture however restricts the analysis possibilities using this approach, since decentralized change processes can for instance not be addressed, whereas these approaches are addressed in this thesis. In [18] a general diagnosis engine is presented which drives adaptation processes within multi-agent organizations using the TAEMS modeling language as the primary representation of organizational information. In the design of the diagnostic engine three distinct layers are identified: symptoms, diagnosis, and reactions. The redesign of organizations is however not addressed from an organizational modeling perspective, which is the case in the approach presented in this thesis. The approach presented in [18] is specified purely on the agent level. This makes it harder to reuse such models.

References

[1] Ackerman, L.S., Development, transition, or transformation: the question of change in organization, OD Practitioner, 18(4) , 1986, pp. 1-9.

[2] Aiken, M., and Hage, J., The Organic Organization and Innovation, *Sociology* 1:63-82, 1971.

[3] Bashein, M.L., Marcus, M.L., and Riley, P., Business Process Reengineering: preconditions for success and failure, Information Systems Management 9, 1994, pp. 24-31.

[4] Boonstra, J.J. (editor), Dynamics of Organizational Change and Learning, Wiley, 2004.

[5] Castelfranchi, C., 1998, Modelling Social Action for AI Agents. *Artificial Intelligence*, 103: pp.157 - 182.

[6] Chien,S., Barrett, A., Estlin, T., and Rabideau, G. A comparison of coordinated planning methods for cooperating rovers. In *Proc. of the Fourth Int'l Conf. on Autonomous Agents*, pages 100--101. ACM Press, 2000.
[7] Collins, J., Ketter, W. and Gini, M., A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *Int'l Journal of Electronic Commerce*, 7(1):35--57, 2002.
[8] Cox, J.S., Durfee, E.H., and Bartold, T., A distributed framework for solving the multiagent plan coordination Problem, In *Autonomous Agents and Multi-Agent Systems*, pages 821--827, 2005.
[9] Cunningham, J.G., Approaches to the Evaluation of Organizational Effectiveness, *Academy of Management Review 2*: 463-474, 1977.
[10] Dignum, V., A Model for Organizational Interaction: Based on Agents, Founded in Logic, PhD thesis, 2003.
[11] Dignum, V., Sonenberg, L., Dignum, F., 2004, Dynamic Reorganization of Agent Societies, In: *Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004*.
[12] Donaldson, L., *The Contingency Theory of Organizations*, Sage Publications, 2001.
[13] Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.
[14] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems, in Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organizations. Proceedings of MAAMAW'01*, 2001.
[15] Hall, G., Rosenthal, T., and Wade, J. How to make reengineering really work, *Harvard Business Review*, 71(6), 1993, pp. 119-131.
[16] Hillier, F.S. and Lieberman, G.J., Introduction to Operations Research, McCraw-Hill, SF, 2002.
[17] Hölldobler, B., and Wilson, E.O., The Ants, Harvard University Press, 1990.
[18] Horling, B., Benyo, B, and Lesser, V., Using Self-Diagnosis to Adapt Organizational Structures, In: Muller, J.P., Ander, E., Sen, S., and Frasson, C., Proceedings of the Fifth International Conference on Autonomous Agents, ACM Press, 2001, pp. 529-536.
[19] Hübner, J.F. and Sichman, J.S., Using the MOISE+ model for a cooperative framework of MAS reorganization, Boletim Técnico BT/PCS/0314, Escola Politécnica da USP, São Paulo, 2003.
[20] Hübner, J.F., Sichman, J.S., and Boissier, O., A Model for the Structural, Functional and Deontic Specification of Organizations in Multiagent Systems. In: Proc. 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), Porto de Galinhas, Brasil, 2002. Extended abstract in: C. Castelfranchi and W.L. Johnson (eds.), Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02. ACM Press, 2002, pp. 501-502.
[21] Jaffee, D. (2001), *Organization Theory – Tension and Change*, McGraw-Hill Companies.
[22] Krishna, V., *Auction Theory*, Academic Press, London, UK, 2002.
[23] Lewin, K. (1951), *Field Theory in Social Science*, Harper & Row, New York.
[24] Lippit, R., Watson, J., Westley, B., The Dynamics of Planned Change, Harcourt, New York, 1958.
[25] Maes, P., How to do the right thing. *Connection Science*, 1989. 1(3): pp. 291-323.
[26] McCallum, M., Vasconcelos, W.W., and Norman, T.J., 2005, Verification and Analysis of Organisational Change. In: Boissier, O., Dignum, V., Matson, E., Sichman, J. (eds.), *Proc. 1st OOOP Workshop*.
[27] Miles, R.E., Snow,, C.C., Mathews, J.A., Miles, G., Coleman, H.J. (1997), "Organizing in the knowledge age: anticipating the cellular form", *Academy of Management Executive*, Vol. 11 No.4, pp.7-24.

[28] Mintzberg, H., Structure in Fives: Designing Effective Organizations, Prentice Hall, 1992.
[29] Nair, R., Tambe, M., Marsella, S., Role Allocation and Reallocation in Multiagent Teams : Towards a Practical Analysis, In: Proceedings of the Second Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2003), pp. 552-559, ACM Press, 2003.
[30] Ordeshook, P. *Game theory and political theory: An Introduction*. Cambridge: Cambridge University Press, 1986.
[31] Rohwer, J.M., Meadow, N.D., Roseman, S., Westerhoff, H.V., and Postma, P.W. (2000). Understanding glucose transport by the bacterial phosphoenolpyruvate:glycose phosphotransferase system on the basis of kinetic measurements in vitro. J Biol Chem. 2000 Nov 10; 275(45): 34909-21.
[32] Schultz, D.J., Barron, A.B., Robinson, G.E., 2002, A Role for Octopamine in Honey Bee Division of Labor, *Brain, Behavior and Evolution*, vol. 60, pp. 350-359.
[33] Selfridge, O. G. Pandemonium: a paradigm for learning in mechanization of thought processes. In *Proceedings of a Symposium Held at the National Physical Laboratory*, pages 513-526, London, November 1958.
[34] Smith, R.G., The contract net protocol: High level communication and control in a distributed problem solver, *IEEE Trans. Computers*, 29(12):1104--1113, December 1980.
[35] Theraulaz, G., Bonabeau, E., and Deneubourg, J.L., 1998, Response thresholds reinforcement and division of labor in insect societies. Proceedings of the Royal Society of London Series B-Biological Sciences, 265: 327-332.
[36] Winston, M.L. and Punnet, E.N., 1982, Factors determining temporal division of labor in honeybees, *Canadian Journal of Zoology*, vol. 60, pp. 2947-2952.
[37] Zambonelli, F., Jennings, N., Wooldridge, M., 2001, Organizational Rules as an Abstraction for the Analysis and Design of Multi-agent Systems, *Journal of Software and Knowledge Engineering*, Vol. 11, pp. 303-328.

Chapter 20

Future Work

Future Work

Many directions exist in which the research presented in this thesis can be further extended. One topic which has not been addressed is that of organizational learning. An organization can learn from past experiences, and change based upon those experiences. It could be argued that the decentralized organizational model presented for honeybee colonies exhibits learning from the past due to the threshold mechanism (which causes thresholds to decrease over time), however in organization theory many more advanced strategies are presented (see e.g. [2]). It would be interesting to see how such learning mechanisms can be modeled. Besides learning from occurrences of events in the past, an approach could also be to "learn" the optimal organization given the current circumstances. In [3] a first attempt towards such learning is presented based upon genetic algorithms. This domain would certainly be worth exploiting.

The approaches presented in this thesis are based upon AGR extended with dynamic models [4]. An open question is how the models and techniques presented in this thesis could be used in other organizational modeling approaches, such as presented in the related work section. Such a survey would help modelers who prefer other organization modeling approaches to apply the models and techniques presented here.

As organizations are becoming more dynamic, so is the structure of those organizations. Nowadays, organizations that are of the cellular [5] or organic type [1] are occurring more frequently. The investigation of how well existing human organizations that exhibit such an organizational structure can be described and analyzed using the models for organizational change presented in this thesis would certainly be interesting.

In the specific domains of application throughout this thesis, a lot of potential for future work exists as well. For the domain of incident management for example, several approaches for evaluating the effectiveness of change have been proposed in this thesis. These approaches have been evaluated by means of the investigation of historic cases. It would be interesting to see how effective these evaluation approaches could be at runtime. Extensive case studies can be performed to see whether the effectiveness of change in incident management organizations can indeed be improved. In the domain of logistical organizations, addressed in the negotiation chapters of this thesis, evaluations of the proposed allocation strategies can be evaluated for more complex cases as well, such as multiple goods that can be combined within one truck.

In a more generic sense, in order to make the models and techniques presented in this thesis more accessible for practitioners, a support environment can be created that integrates them. Such a support environment could as a result facilitate more effective organizational change. Of course, whether the effectiveness is indeed improved would need to be investigated after such a system has been used by these practitioners.

References

[1] Aiken, M., and Hage, J., The Organic Organization and Innovation, *Sociology* 1:63-82, 1971.

[2] Argyris, C., On Organizational Learning, Blackwell Publishing, 1999.

[3] Bou, E., Lopez-Sanchez, M., and Rodriguez-Aguilar, J.A., Self-Configuration in Automatic Electronic Institutions, In: Coordination, Organization, Institutions and Norms in Agent Systems (COIN@ECAI 2006), 2006, pp.1-7.

[4] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems, in Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organizations. Proceedings of MAAMAW'01*, 2001.

[5] Miles, R.E., Snow,, C.C., Mathews, J.A., Miles, G., Coleman, H.J. (1997), "Organizing in the knowledge age: anticipating the cellular form", *Academy of Management Executive*, Vol. 11 No.4, pp.7-24.

Samenvatting: Modelleren van Verandering in Multi-Agent Organisaties

Organisatie kan gedefinieerd worden als een systematische ordening van elementen die samen een bepaald doel willen bereiken. Organisatie komt niet alleen bij mensen voor, maar ook in biologische systemen binnen de informatica komt organisatie voor. Binnen het veld van de computationele en mathematische organisatietheorie probeert men theorieën die bestaan over organisaties te ontwikkelen en testen met behulp van computationele en mathematische modellen. Eén van de disciplines binnen de computationele en mathematische organisatietheorie is de discipline van de multi-agentsystemen. Een multi-agentsystem is een systeem dat bestaat uit diverse interacterende agenten die een bepaald proces of doel nastreven. Zulke systemen worden met name gebruikt om het collectieve gedrag binnen een organisatie op basis van het individuele gedrag van agenten te onderzoeken. Hierbij worden deze systemen voornamelijk vanuit een abstract, organisatieperspectief beschreven. Het voordeel van zo'n abstracte manier van representeren is dat complexere systemen gemakkelijker beschreven kunnen worden.

Een belangrijk aspect binnen organisaties is de verandering van zulke organisaties. Onderzoek heeft uitgewezen dat 70% van de organisatieveranderingen in bedrijven het van tevoren gestelde doel niet haalt. Gegeven de ontwikkelingen in het veld van computationele en mathematische organisatietheorie en in het bijzonder op het gebied van multi-agentsystemen is de vraag of deze nieuwe aanpakken gebruikt kunnen worden om organisatieveranderingsprocessen te beschrijven, en zo mogelijk te verbeteren. Dit is precies waarover dit proefschrift gaat. Het doel van het proefschrift is om organisatieveranderingsprocessen te analyseren en modelleren met behulp van aanpakken uit het veld van multi-agentsystemen. Verder is het doel om modelleurs van organisaties te voorzien van blauwdrukken en hulpmiddelen om hen in staat te stellen ook organisatieveranderingen te modelleren.

Om dit doel te bereiken is er gekozen om een bestaande aanpak te gebruiken. Deze aanpak bestaat uit twee delen, te weten een deel waarmee de structuur van de organisatie op verschillende aggregatieniveaus gerepresenteerd kan worden en een deel waarmee het gedrag van deze organisatie, ook op verschillende aggregatieniveaus, beschreven kan worden. De methode om het gedrag mee te beschrijven is een formele logische methode waarmee zowel kwantitatieve als kwalitatieve eigenschappen uitgedrukt kunnen worden.

In dit proefschrift worden aan de hand van deze methode diverse facetten van organisatieverandering onderzocht. Deel ii van dit proefschrift richt zich op de analysefase binnen organisatieveranderingsprocessen. Hierin wordt gekeken naar de huidige organisatie en geanalyseerd of en, zo ja, welke zaken er niet juist verlopen binnen een organisatie. Op basis van deze analyse worden vervolgens methodes gepresenteerd om deze organisaties te verbeteren. In deel iii, iv en v wordt het proces van organisatieverandering zelf gemodelleerd en geanalyseerd. Hierbij gaat het dan om de verandering van de huidige organisatie naar de nieuwe organisatie. Deze veranderingen kunnen plaatsvinden op een gecentraliseerde manier (deel iii), een decentrale manier (deel iv), of een tussenvorm (deel v). Nadat een verandering heeft

plaatsgevonden dient deze ook geëvalueerd te worden om te onderzoeken of de verandering inderdaad succesvol gebleken is. Hoe zo'n evaluatie uitgevoerd kan worden met behulp van de gebruikte aanpak wordt in deel vi gepresenteerd. Tenslotte presenteert deel vii conclusies, geeft het een overzicht van gerelateerd werk, en presenteert het mogelijke vervolgstappen in het onderzoek.

SIKS Dissertation Series

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup, (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)
Knowledge Management: The Role of Mental Models in Business Systems Design

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated Representation

2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building Web Service Ontologies

2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling

2006-07 Marko Smiljanic (UT)
XML schema matching -- balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web

2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor's Opinion

2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems

2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types

2006-12 Bert Bongers (VU)
Interactivation - Towards an e-cology of people, our technological environment, and the arts

2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents

2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change

2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain

2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks

2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device

2006-18 Valentin Zhizhkun (UVA)
Graph transformation for Natural Language Processing

2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach

2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the Web

2006-22 Paul de Vrieze (RUN)
Fundaments of Adaptive Personalisation

2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web

2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings