Modelling a sociological case study

Michael Köhler, Daniel Moldt, and Heiko Rölke

University of Hamburg – Computer Science Department
Vogt-Kölln-Straße 30 – 22527 Hamburg
{koehler,moldt,roelke}@informatik.uni-hamburg.de

Abstract. This paper describes the modelling principles and the modelling process that are developed and used in our project group to cope with sociological real world case studies. The modelling process takes place at different levels: individual behaviour and the overall system structure, which are modelled almost independently.
Reference nets, a special form of high-level Petri nets, are used together with their development environment Renew in an iterative rapid prototyping modelling process. The model may be executed showing only the desired properties of the model. It is possible to switch between views on the model even at run time.

Keywords: modelling, multi agent systems, reference nets, sociology, socionics

1 Introduction

Modelling of complex systems is and always will be a difficult task. In the area of computer science the UML (Unified Modelling Language) (see [4]) can be considered to be a standard notation to build models. However, there are several drawbacks that have to be solved. Here we focus on the problem that a technique for a specific application area is needed: for sociology. The specific concepts of this area like autonomy of actors and dynamic system structures need to be supported. Furthermore a modelling technique that generates immediately executable models is necessary.

In this paper we shortly introduce a specific sociological case study that finds its application area in decision theory in the context of universities. This kind of topic is generally considered to be a difficult field. Many facets have to be taken into account when looking for good descriptions or even explanations of some processes and phenomena in this area. In our research project "Acting in social contexts"[1] we have chosen this as our application area. Beside that, we develop an agent oriented Petri net based architecture, platform, and modelling technique and investigate general sociological theories and organisational theories to build

[1] ASKO, short for the German title: "Agieren in sozialen Kontexten". ASKO is part of the socionics project, an interdisciplinary project between sociology and computer science.

a middle range theory for decision making within public service areas. The case study here only illustrates how we capture some basic features in our approach. More details can be found in a local technical report (see [5]). Here some basic principles are introduced and the major focus of our work is presented.

In the next sections we will introduce the case study (section 2), provide some background information about Petri nets (section 3), sketch our agent oriented Petri net architecture (section 4) and central modelling features, the protocols (section 5) and the system structure (section 6). The paper concludes with a general discussion of the central ideas of our approach with respect to the modelling of sociological case studies (section 7).

2 Sociological Case Study

To show how sociological theories can be applied we provide an empirical case study[2]:

There is an university of average size with seven faculties and 140 positions of professors.

- The university is controlled by the research ministry. Its head, the minister of research informs the university that 10 of the 140 positions have to be curtailed.
- The head of the university, the chancellor, decides to curtail two positions at each faculty. Four of these are going back into a pool for innovative approaches of teaching and research. The chancellor delegates the decision which positions have to be given up towards the faculties as well as the decision which faculties will get the four "new" positions. The faculties have to justify their decision.
- The head of the faculty 4, the dean, has a direct private meeting with the minister of research. Following the meeting he and two members of his faculty write a "future paper" which explains why all positions have to remain at the faculty. All other professors and researchers as well as all other faculties do not see possibility to make any suggestions.
- At a dean meeting the decision shall be prepared. The future paper is rejected as being not sufficient.

Our study concentrates on the question: How do the faculties deal with the delegated task to curtail positions and how do they decide?

To complete the case study we present the outcome: The faculties and the university as a group actor do not decide at all. However, the decision is made at the chancellors level. This decision is a random selection.

What are the reasons for this? Anticipatory we provide the solution: The university is not able to mobilise and bundle their potentials and resources towards a constructive power that makes a decision. In opposition to this, the ministry is able to do so. The relatively small ability for self organisation is the reason that the university-rationality loses to the political-administrative one.

[2] The German notions used within universities are translated to English notions whenever possible.

3 Reference Nets

Petri nets are a well established means for the description of concurrent systems. Petri nets allow for an intuitive representation of the notions of causality, alternatives, parallelism, resource, action, and others.

A net is assembled from places and transitions. Places represent resources that can be available or not or conditions that may be fulfilled. Places are depicted in diagrams as circles or ellipses. Transitions are the active part of a net. Transitions are denoted as rectangles or squares. A transition that fires (or occurs) removes resources or conditions (for short: tokens) from places and inserts them into other places. This is determined by arcs that are directed from places to transitions and from transitions to places.

It is assumed throughout this text that the reader is familiar with Petri nets in general as well as coloured Petri nets. Reisig [12] gives a general introduction, Jensen [6] describes coloured Petri nets. Generally speaking, coloured Petri nets permit a more compact representation while offering the same computational power compared to simpler net formalisms.

Reference nets [10] are so-called higher (coloured) Petri nets, a graphical notation that is especially well suited for the description and execution of complex, concurrent processes. As for other net formalisms there exist tools for the simulation of reference nets [11]. Reference nets show some expansions related to "ordinary" coloured Petri nets: nets as token objects, different arc types, net instances, and communication via synchronous channels. Beside this they are very similar to coloured Petri nets as defined by Jensen. The differences now are shortly introduced.

Nets as tokens Reference nets implement the "nets within nets" paradigm of Valk [15]. This paper follows his nomenclature and denominates the surrounding net *system net* and the token net *object net*. Certainly hierarchies of net within net relationships are permitted, so the denominators depend on the beholders viewpoint.

Arc types In addition to the usual arc types reference nets offer *reservation arcs*, that carry an arrow tip at both endings and reserve a token solely for one occurrence of a transition, *test arcs*, and *inhibitor arcs*. Test arcs do not draw-off a token from a place allowing a token to be tested multiple times simultaneously, even by more than one transition (test on existence). Inhibitor arcs prevent occurrences of transitions as long as the connected places are marked.

Net instances Net instances are similar to objects of an object oriented programming language. They are instantiated copies of a template net like objects are instances of a class. Different instances of the same net can take different states at the same time and are independent from each other in all respects.

Synchronous channels One synchronous channel [1] permits a fusion of transitions (two at a time) for the duration of one occurrence. In reference nets (see

[10]) a channel is identified by its name and its arguments. Channels are directed, i.e. exactly one of the two fused transitions indicates the net instance in which the counterpart of the channel is located. The other transition can correspondingly be addressed from any net instance. The flow of information via a synchronous channel can take place bi-directional and is also possible within one net instance. It is possible to synchronise more than two transitions at a time by inscribing one transition with several synchronous channels.

4 Multi Agent System

This section gives a short introduction to a multi agent system (MAS) modelled in terms of "nets within nets" (see figure 1). This survey is given to make the general ideas visible that are prerequisite to the understanding of the concepts that follow in later sections of this paper.[3] It is neither an introduction to multi agent systems nor the assets and drawbacks of dividing the system into platforms is discussed here. For a broad introduction to multi agent systems see for example [16], the special view taken in our work is related to the standard proposal of the "Foundation for Intelligent Physical Agents" (FIPA) [2, 3].

Take a look at figure 1: The grey rounded boxes enclose nets (net instances) of their own right (entitled multi agent system, agent platform, agent structure, and protocol). The ZOOM lines enlarge object nets that are tokens in the respective system net.[4] The upper left net of the figure is an arbitrary multi agent system with places containing agent platforms and transitions modelling communication channels between the platforms. This is just an illustrating example, the number of places and the form of interconnection has no further meaning.

The first zoom leads to a closer view of a simplified agent platform. The central place agents contains all agents that are currently hosted on the platform. New agents can be generated (or immigrate from other platforms) by transition new, agents can be destroyed or migrate to another platform (transition destroy). Internal message passing differs from the external case so it is conceptually separated: The internal communication transition binds two agents (the sender and the receiver of a message) and allows them to hand over messages via calls of synchronous channels. External communication involves only one agent of the actual platform (and an agent of a distant platform). For external communication as well as for agent migration the communication transitions of the top level agent system net are needed. The interaction of the multi agent system and the agent platform is made possible by inscribing the transitions with synchronous channels connecting for example the transition external communication of an agent platform with that of another one via the communication structure transition of the multi agent system. These inscriptions are not visible in the figure. The agent platform net is the system net for the agents that are hosted on the platform. The object nets are net instances of the next zoomed net: agent.

[3] For a more detailed presentation refer to [8] or the technical report [13] (in German).

[4] Beware not to confuse this net-to-token relationship with place refinement.

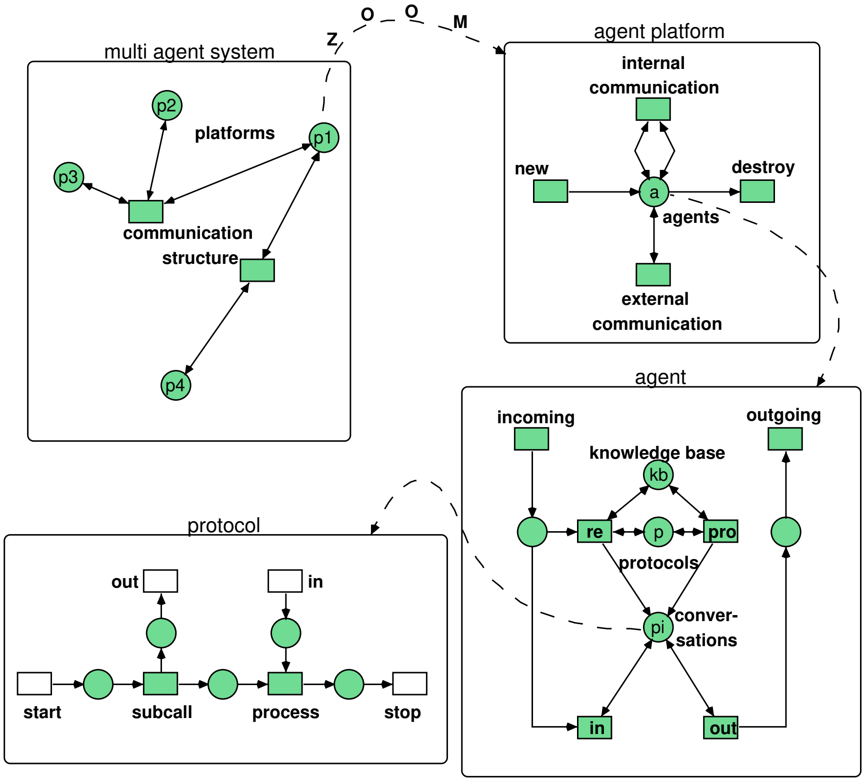

Fig. 1. MAS as nets within nets

The net agent shows the (static) interface net of an agent. All interaction to other agents or the outside world is done via the transitions incoming and outgoing. Incoming messages are processed by the transition re: An appropriate protocol is chosen from the protocol pool of the agent (place protocols) under consideration of the agent's state (place knowledge base). After that the protocol is instantiated and put on the place conversations. The agent takes part in a conversation that spawns several (at least two) agents. Additionally the agent has the opportunity to instantiate its protocols without external cause (messages): Transition pro is the pro-active way to activate new conversations.

Incoming messages that are related to a conversation the agent already takes part of are directly passed to the conversation's protocol via transition in. Transition out fires to pick up outgoing messages generated by the protocol instances (place conversation) to other members of a conversation.

A protocol as in the example net protocol resembles an advanced workflow. Protocols describe pieces of actions that follow strict (syntactical) rules to make the protocols reusable and inter-operable between the agents.

Figure 1 therefore illustrates our implementation and conceptual basis to design MAS. The multi-level structuring is intuitive and straight forward, once an agent oriented design is anticipated. The use of nets allows for a graphical representation and for an execution. Concurrency, nondeterminism, and distribution are directly modelled and explicitly communicated to other people involved in the system design. It is important to notice that places (especially the platform places in the multi agent system net) can also be seen as physical (or logical) places.

The next sections describe the use of the abstract agent system model in a real world case study. The modeller has to identify the overall system structure (locations, paths from location to location, and communication channels), the involved persons and the person's behaviour. The latter are described in the next section 5, the structure is given in section 6.

5 Modelling Individual Behaviour: Protocols

The models of individual behaviour can be deduced from behaviour primitives similar to workflow schemes. A collection of such behaviour primitives is given in figure 2 to 5.

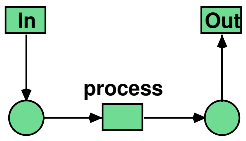

Fig. 2. Simple protocol

The simplest form of a (reactive) behaviour modelling protocol is that of figure 2. All other reactive protocols are refinements of this protocol. The operational semantics is as follows: After starting the protocol (by invocation of transition In via a synchronous channel) and eventually handing over parameters the protocol does some processing without interference with the outside world. The result of the processing – at least a message signalling the termination – is passed to the agent via transition (and synchronous channel) Out.

The result of a protocol may be any possible message – especially answers, new requests or messages that affect the internal state of the agent.

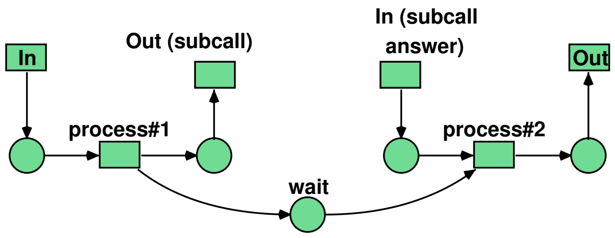

Fig. 3. Protocol depending on sub-call result

If a task depends upon the results of another protocol of the same or another agent, it has to invoke[5] the protocol and wait for the answer message before it is able to proceed. This may be done according to figure 3.

The first phase of this protocol follows the description above. Transition process#1 produces a message containing the sub-call (message of type *request*) and marks the place wait. The sub-call message is handed over to its destination agent via the agent structure, the agent's platform and possibly also the multi agent system net (see figure 1). The answer message enters the protocol as a parameter of the second In transition. It may be further processed by execution of transition process#2 before the protocol finishes.

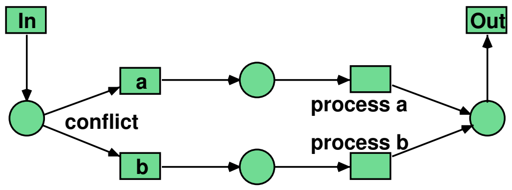

Fig. 4. Protocol with conflict

Other basic building blocks of behaviour are decisions and parallelism. Figure 4 shows two transitions (a and b) that are in conflict: only one of them may fire and determine the protocol's result. This is not to be confused with the situation shown in figure 5 where a fork is performed that enables the parallel execution of both transitions a and b (parallel activities). Their results are joined afterwards. Note that it is often more convenient to model parallel chunks

[5] This is not an invocation as it is known from object oriented languages. More precisely the protocol *asks* for a specific answer message. If and in which way this answer is generated is under total control of the addressee of the request.

of behaviour in independent protocols rather than in one protocol containing independent paths.

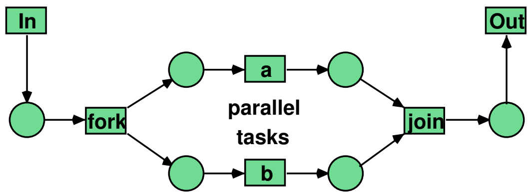

Fig. 5. Protocol with parallel tasks

One example of a behaviour primitive taken from the case study is the decision process the deans go through to react to the demand for curtail propositions of the research ministry. The dean of faculty 4 has the opportunity to choose another action than the other deans, because he knows the research minister personally. The other deans may use the same decision protocol, but in fact they do not have to decide because of resource limitations (behaviour primitives may be regarded as resources, too).

More examples of behaviour primitives, their Petri net representation and annotations to their development may be found in an extensive research paper ([5], in German).

6 System Structure Modelling

The system structure consists of the locations present in the system, the paths between the locations enabling the agents to migrate, and the communication channels allowing the exchange of messages from one location to another. There are several possible limitations: Certain locations may only be visited by certain agents, some paths may not be known to all agents, the use of some communication channels may be restricted and so on.

The creation of the system model of a real world scenario is straight-forward (see figure 6): locations of the scenario are the office of the chancellor of the university (Präsidium), the conference hall (Dekanekonferenz, dean meeting), the ministry (Ministerium), the offices of the deans (Fachbereiche), an external university (externe Universität) and a restaurant (Restaurant "Konspirativo"). The locations are depicted by large places coloured in light gray.

Paths between the locations that are known (and therefore may be used) by the participants of the case study are denoted as uncoloured transitions. There is for example no direct connection between the ministry and the deans' offices, only (indirect) communication can take place (using the dark message path transitions modelling a message exchange mechanism). So the dean of faculty 4 is only able to meet the minister face-to-face at an informal place (the restaurant).

There are additional places and transitions between the locations representing a store-and-forward asynchronous message exchange system (internal letter mail). Direct communication is only possible for agents sharing a location.

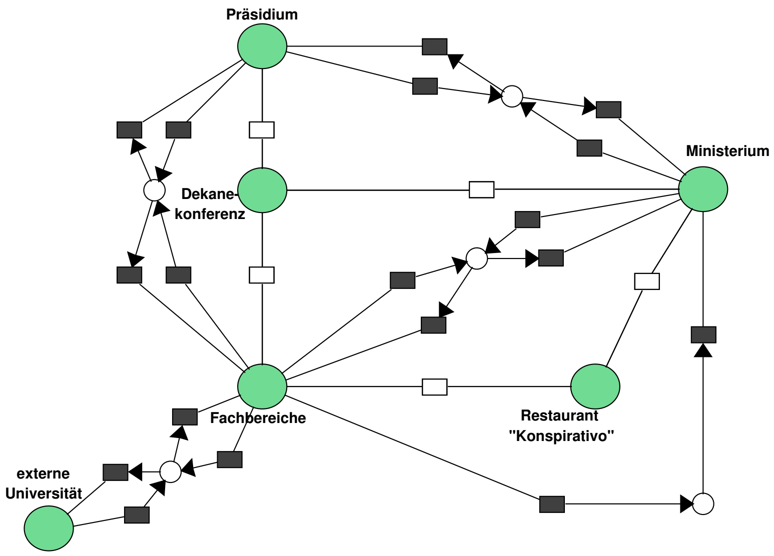

Fig. 6. System structure net

The system structure net is enhanced by adding pictorial representations of some of the tokens and transition occurrences. Figure 7 shows a snapshot of the simulation. The piggy banks show the exchange of messages related to money shortcuts (administration messages), the molecule in the lower left corner stands for a message related to scientific dispute. The two chatting faces in the lower right corner show that two agents in the restaurant are directly communicating – in this case the dean and the chancellor having an informal meeting.

7 Aspects of Modelling Sociological Case Studies

Even the modelling of basic aspects shows principle problems and requirements when dealing with sociological case studies. The general aspects of modelling are still present: functional, behavioural, and structural views as one is usually aware of. Often languages such as UML[6] (see [4]) are far too complex to be

[6] Unified Modelling Language, consisting of many different and not completely integrated modelling techniques.

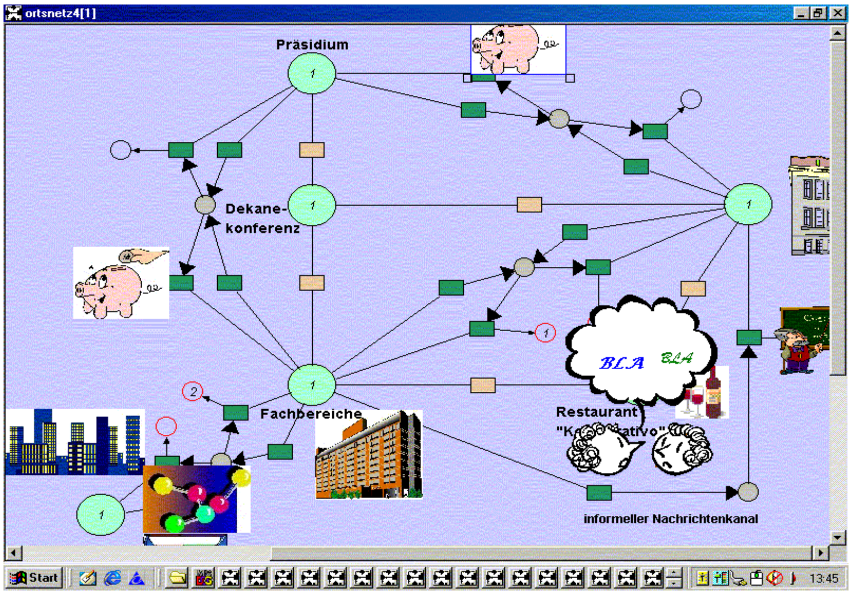

Fig. 7. Enhanced system structure net simulation state

used in this area. Furthermore, special facets of sociological cases are not covered sufficiently: the actor concept, protocols, structure, behaviour, dynamics, causality, etc. More generally the semantics of models often does not really show the possible behaviours that can appear within a model or system respectively.

Therefore, we decided to use Petri nets to build executable models. The central features of the case studies can be shown. To provide an intuitive structure we use agent oriented Petri nets (AOPN) that are based on the "nets within nets" paradigm. This allows us to provide appropriate levels of abstraction according to the interests of sociologists. The nets shown in figure 6 and figure 7 show a very abstract model of the system. Behaviours present in this system are build on the underlying nets which carry more detailed information.[7] If required the details can be made available to sociologists during simulation or when investigating the static structures of the model. On each level the same formalism or drawing technique is used: reference nets. Due to the necessity to include technical information for simulation the models are not yet optimal for sociologists. However, when using animation and partial hiding, the models are the means for discussions. These discussions take place between computer scientists – the modellers – and the sociologists – the users. Furthermore, once the models have been stabilised, they can be used among sociologists to discuss the underlying principles and mechanisms.

[7] Due to place limitations these nets are not shown in the paper.

Experience within our research group could be made over a period of more than one and a half years by now. The sociologists got trained on the reading of Petri net models on an informal basis. They are not trained to build the models themselves at the moment. The specific needs get more and more identified and the technique is adopted accordingly. However, up to now most of the models have to be manipulated to be easily readable. Beside our general agent oriented Petri net based architecture and development platform [8, 13] we are working on a modelling technique that suits the needs of sociologists in a better way. Fundamental concepts like causality, concurrency, conflict, non-determinism, agents (actors), roles, locations, mobility, autonomy, adaptability etc. are directly based on the techniques of Petri nets which allow to provide a clear, precise, and intuitive semantics for all concepts. Once the basic building blocks of sociological case studies has been worked out the modelling process will be much easier. The results gained so far allow us to capture only simple models, since all details have to be modelled explicitly. The goal is to find shortcuts as already used in the case study presented here to build more complex models.

The case study presented here only describes a very specific scenario of the huge range that would have been possible for the actors of the real case. However, the underlying principles of their decisions were explicitly represented in the model. Theoretical investigations are essential for the interpretations of the models and the case studies. This is covered by parallel running working packages within the ASKO project. There the sociological foundations are laid to assure the scientific level. The specific case study here is only illustrating a very small section of the modelling insights reached so far. One of our gained results is that the number of models is relatively large compared to the original texts. The main reason we could find up to now is that the sociologists build quite complex models using natural language constructs. To reach a similar grounded model one needs to build several interacting Petri nets. The advantage is that these models can be formally analysed, executed and animated. Therefore, the validation of such models becomes much easier. And even more, the sociologists gain a deeper insight into their theories due to the preciseness that is required to build the Petri net models.

This paper mainly reflects the part of the work that is done on the computer science side. Unfortunately the only well-balanced publication treating both the sociological case study and the modelling in equal rights, is a German technical report [5]. This is due to the prototypical character of the work. Nevertheless there were some interesting insights gained by the sociologists in our team, primarily concerning the completeness of a case study[8]. To model a complete scenario of a case study one has to clarify aspects that sociologists (apparently) do not give their best attention when focusing at their field of interest: locations/whereabouts, used/unused communication channels, properties of single actors when looking at groups or the whole picture and vice versa.

[8] These insight does not only apply to case studies. We have found similar gaps when modelling various sociological theories as well. See for example [7]. A general introduction to sociological modelling in terms of Petri nets is given in [9].

Beside our goals regarding the architecture and platform, one main focus is on the improvement of an adequate modelling technique that a) allows to reduce the size of the models, b) allows to express details more condensed, c) keeps the executability and formal background, d) can easily be related to animations, e) provides easy navigation between different levels of abstraction, and f) allows to cover the formerly named fundamental concepts in a form that is easily understood by sociologists without longer training periods.

References

1. S. Christensen and N.D. Hansen. Coloured Petri nets extended with channels for synchronous communication. In Rober Valette, editor, *Application and Theory of Petri Nets 1994, Proc. of 15th Intern. Conf. Zaragoza, Spain, June 1994*, LNCS, pages 159–178, June 1994.
2. FIPA. Homepage. `http://www.fipa.org`.
3. FIPA. FIPA 97 Specification, Part 1 - Agent Management. Technical report, Foundation for Intelligent Physical Agents, `http://www.fipa.org`, October 1998.
4. Object Management Group. OMG UML Homepage. `http://www.uml.org`.
5. Daniela Hinck, Michael Köhler, Roman Langer, Daniel Moldt, and Heiko Rölke. Stellenstreichungen an der Universität Mitteldorf. Mitteilung/Technical report 303, Universität Hamburg, Fachbereich Informatik, 2001.
6. K. Jensen. *Coloured Petri nets, Basic Methods, Analysis Methods and Practical Use*, volume 1 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1992.
7. Michael Köhler, Roman Langer, Daniel Moldt, and Heiko Rölke. Combining the sociological theory of Bourdieu with multi agent systems. In C. Jonker, A. Letia, G. Lindemann, and T. Uthmann, editors, *Modelling Artificial Societies and Hybrid Organisations (MASHO'00), Workshop at the ECAI 2000*, 2000.
8. Michael Köhler, Daniel Moldt, and Heiko Rölke. Modelling the structure and behaviour of Petri net agents. In J.M. Colom and M. Koutny, editors, *Proceedings of the 22nd Conference on Application and Theory of Petri Nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 224–241. Springer-Verlag, 2001.
9. Michael Köhler and Heiko Rölke. Petrinetze als Darstellungstechnik zur Modellierung in der Soziologie. *Sozionik aktuell*, 1(2), 2001.
10. Olaf Kummer. Simulating synchronous channels and net instances. In J. Desel, P. Kemper, E. Kindler, and A. Oberweis, editors, *Forschungsbericht Nr. 694: 5. Workshop Algorithmen und Werkzeuge für Petrinetze*, pages 73–78. Universität Dortmund, Fachbereich Informatik, 1998.
11. Olaf Kummer, Frank Wienberg, and Michael Duvigneau. *Renew - User Guide*. University of Hamburg, Vogt-Kölln-Straße, Hamburg, 1.5 edition, May 2001.
12. Wolfgang Reisig. *Petri nets: An introduction*. Springer, 1985.
13. Heiko Rölke. Die Mulan Architektur. Technical report, Universität Hamburg, 2001.
14. Sozionik aktuell. http://www.sozionik-aktuell.de/.
15. Rüdiger Valk. Petri nets as token objects: An introduction to elementary object nets. In Jörg Desel and Manuel Silva, editors, *Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 1–25, June 1998.
16. Gerhard Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.