SONAR/OREDI: A Tool for Creation and Deployment of Organisation Models

Endri Deliu and Michael Köhler-Bußmeier

University of Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
koehler@informatik.uni-hamburg.de

Abstract. The need for handling the increasing complexity in software systems has allowed the introduction and establishment of an organisational paradigm as an alternative in software modelling and development. Especially within the multi-agent systems community, organisational concepts are enjoying increasing popularity for efficiently structuring multi-agent behaviour. Organisational specifications and their implementation as multi-agent systems lack however a streamlined transition between each other. In this paper we address this problem by introducing a software tool capable of creating and editing organisation models as well as deploying such models as multi-agent systems. The tool is built on Sonar, a formal organisational specification based on Petri nets. By unifying in one tool the organisational specification and deployment process quick reaction cycles to incremental changes of system design become possible.

1 Introduction

Important influxes from sociology and organisation theory have begun delineating what may dissolve the trade-off between agent autonomy and multi-agent system reliability and predictability. Between the system and the agents composing it, other levels of control have been introduced which are mainly derived from sociological concepts. The concept of organisation is used as an umbrella term for groups of agents and their dependencies, interaction channels or relationships (cf. [1] for an survey or organisational approaches to agent systems). As a result, an organisational perspective on multi-agent systems has gradually emerged which focuses on organisational concepts such as groups, communities, organisations, etc., in contrast to the former focus of multi-agent systems on the agent's state and its relationship to the agent's behaviour (cf. [2]).

Modelling agent organisations requires a modelling language that is able to express most (possibly all) of the notions that the concept of organisation encompasses in an intuitive and easily understandable way. Petri nets are well suited for use in modelling systems and simultaneously offer a complete formal frame. In this context, a framework for the development of concurrent and distributed software systems has been built as a multi-agent system basing on reference nets [3], a high level Petri net formalism. Our multi-agent architecture Mulan (**m**ulti-**a**gent **n**ets) [4,5] provides the framework's reference architecture used for

J. Dix and C. Witteveen (Eds.): MATES 2010, LNAI 6251, pp. 76–87, 2010.

the the multi-agent system. Mulan is built on Java and reference nets and can be executed in Renew [3], a Petri net editor and simulator.

In this work, Oredi (**or**ganisation **edi**tor), a Petri net based tool will be presented. It enables editing organisation models as well as deploying such models as multi-agent systems, e.g. as Mulan systems. Oredi is built on top of Renew and relies on Sonar (**s**elf **o**rganising **n**et **a**rchitecture **r**eference), a formal organisational specification for electronic institutions based on Petri nets. Section 2 will shortly introduce the main concepts of Sonar which are supported by Oredi. In Section 3, the deployment of Sonar organisation models into agent organisations with Oredi is presented.

2 Sonar: A Formal Model of Organisations

In the following we give a short introduction into our modelling formalism, called Sonar. A detailed discussion of the formalism can be found in [6], its theoretical properties are studied in [7]. A Sonar-model encompasses:

1. A set of interaction models (called *distributed workflow nets*, short: DWF nets) based on a data ontology and a role model;
2. An organisational model that defines a network of organisational sub-units (usually called *positions*) and their resources together within a net model, that describes the team-based delegation of tasks;
3. A stratification model that assigns a hierarchy level n to all components;
4. A set of transformation rules together with a model metric;
5. A refinement/abstraction operation to build nested Sonar-models.

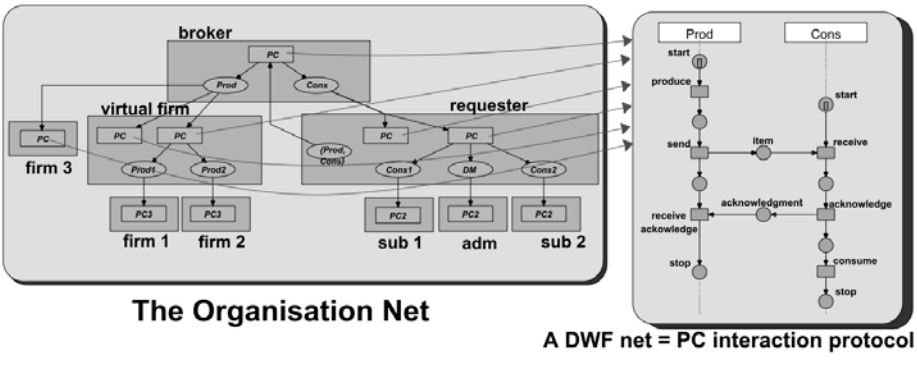

Fig. 1. A Sonar-Model

Figure 1 illustrates the basic relationships between the Sonar interaction model, the role/delegation model and the position network. The example describes the relationship between some positions (drawn as grey boxes: *broker*, *virtual firm*, *requester*, etc.) in terms of their respective roles (*Producer*, *Consumer* etc.) and associated delegation links. In this scenario, we have a requester

and two suppliers of some product. Coupling between them is provided by a broker.[1] From a more fine-grained perspective, the requester and one of the suppliers consist of delegation networks themselves. For example, in the case of the *virtual firm* supplier, we can identify a management level and two subcontractors. The two subcontractors may be legally independent firms that integrate their core competencies in order to form a virtual enterprise (e.g. separating fabrication of product parts from their assembly). The coupling between the firms constituting the virtual enterprise is apt to be tighter and more persistent than between requester and supplier at the next higher system level, which provides more of a market-based and on-the-spot connection.[2]

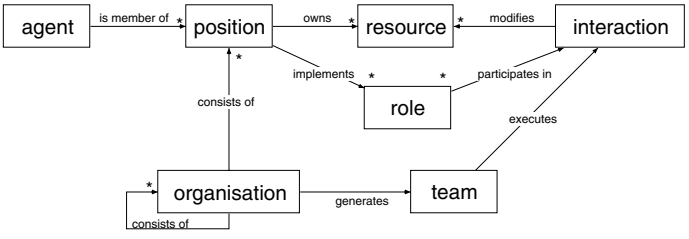

Fig. 2. Basic SONAR Concepts and their Relationships

In general an *organisation* consists of (sub)organisations and atomic organisations, called positions. *Positions* are staffed by agents. Staffed agents have access to *resources* owned by the position and they implement the *roles* assigned to the position. Agents interact in *teams* in which they play roles. Interaction protocols describe the possible interaction processes between roles. As a byproduct of their interaction resources, including data or knowledge are modified. These basic concepts and their relationships are illustrated in Figure 2.

A *distributed workflow net* (DWF net) is a multi-party version of the well-known workflow nets [8] where the parties are called *roles*. Roles are used in DWF nets to abstract from concrete agents. For example, the two roles *Producer* and *Consumer* have the same form of trading interaction no matter which agent is producing or consuming. The right side of Figure 1 shows the DWF net *PC* that describes the interaction between both roles: First the producer executes the activity *produce*, then *sends* the produced *item* to the consumer, who *receives* it. The consumer sends an *acknowledge* to the producer before he *consumes* the item.[3] Technically speaking roles are some kind of type for an agent

[1] Note that for this simplified model brokerage is an easy job, since there are exactly two producers and one consumer. In general, we have several instances for both groups with a broad variety of quality parameters making brokerage a real problem.

[2] This coupling is usually expressed using the refinement/abstraction operator of SONAR, but is omitted here for simplicity reasons.

[3] To simplify the presentation we have omitted all data-related aspects in our discussion of distributed workflow nets. In SONAR each DWF net uses data object based on the model's ontology. Cf. [9,10] for details.

describing its behaviour. Note that agents staffed to positions usually implement several roles.

An organisation net is a Petri net $N = (P, T, F)$ where each task is modelled by a place p and each task implementation (delegation/execution) is modelled by a transition t. Each place p is labelled by a role $R(p)$ and each transition t with a DWF net $D(t)$. Each place and each transition is assigned to the position O it belongs to. This is illustrated by surrounding boxes in Figure 1.

In general a delegation t comes along with a behaviour refinement. In our example, the position *requester* implements the role *Cons* by generating subtasks for the roles *Cons 1*, *DM*, and *Cons 2*. These subtasks are handled by the positions *sub 1*, *adm*, and *sub 2* that implement their respective roles according to the DWF PC_2 (not shown here) which decomposes the behaviour of role *Cons* into the composition of *Cons 1*, *DM*, and *Cons 2*. In well formed organisations it is guaranteed that the service generated from the refined DWF PC_2 and the roles *Cons 1*, *DM*, and *Cons 2* has the same communication behaviour as the service generated from the original DWF *PC* and the role *Cons*.

Team formation can be expressed in a very elegant way: If one marks one initial place of an organisation net *Org* with a token, each firing process of the Petri net models a possible delegation process. More precisely, the *token game* is identical to the team formation process (cf. Theorem 4.2 in [7]). Team formation generates a *team net* and a *team DWF*.

As another aspect, SONAR-models are equipped with transformation rules. Transformation rules describe which modifications of the given model are allowed. They are specified as graph rewrite rules [11]. The minimal requirement for rules in SONARÂ is that they must preserve the correctness of the given organisational model.

3 Deploying SONAR Organisations

OREDI contains an editing tool for SONAR formal organisations. It is built as a set of RENEW plugins. OREDI users can create SONAR formal organisations without being directly aware of their underlying constraints. The process of creating formal organisations involves two steps. The first step being the creation of DWF nets and some additional models, that represent refinement relationships between roles. The second step is the creation of organisation nets and the assignment of DWF nets and roles to transitions and places, respectively. The completion of both steps leads users to organisational models. Each step is handled in separate editors. Thus, an editor for modelling DWF nets is used first. Then the results of the first step are loaded in a second editor where the organisation is modelled.

At the end of the modelling process, OREDI supports the deployment of the modelled organisation as an agent organisation consisting of Organisation Agents, Organisation Position Agents (OPA), and Organisation Member Agents (OMA) we describe in the following section.

3.1 Agent Organisations as OPA/OMA Networks

Now that we have obtained a picture of what constitutes a formal organisation according to our approach, we can elaborate on the activities of a multi-agent systems behaving according to a SONAR-model. The basic idea is quite straightforward: With each position of the SONAR-model we associate one dedicated agent, called the *organisational position agent* (OPA).

Figure 3 illustrates our specific philosophy concerning MAS organisations utilising the middleware approach. In SONAR, we describe a formal organisation in terms of interrelated *organisational positions*.

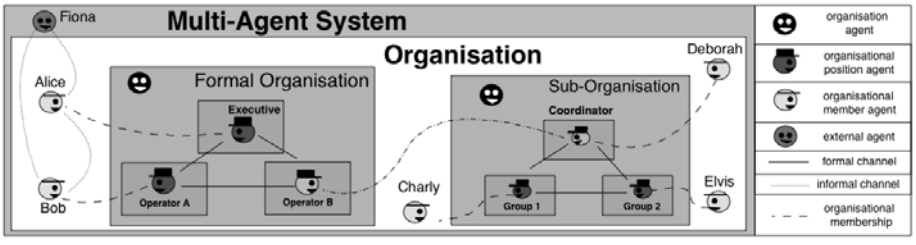

Fig. 3. An Organisation as an OPA/OMA Network

An OPA network embodies a formal organisation. An OPA represents an organisational *artifact* and not a *member/employee* of the organisation. However, each OPA represents a conceptual connection point for an *organisational member agent* (OMA). An organisation is not complete without OMAs. It depends on domain agents that actually carry out organisational tasks, make decisions where required and thus implement/occupy the formal positions. Note that an OMA can be an artificial as well as a human agent. An OPA both enables and constrains organisational behaviour of its associated OMA. Only via an OPA an OMA can effect the organisation and only in a way that is in conformance with the OPA's specification. In addition, the OPA network as a whole relieves its associated OMAs of a considerable amount of organisational overhead by automating coordination and administration. To put it differently, an OPA offers its OMA a "behaviour corridor" for organisational membership. OMAs might of course only be partially involved in an organisation and have relationships to multiple other agents than their OPA (even to agents completely external to the organisation). From the perspective of the organisation, all other ties than the OPA-OMA link are considered as informal connections.

To conclude, an OPA embodies two conceptual interfaces, the first one between micro and macro level (one OPA versus a network of OPAs) and the second one between formal and informal aspects of an organisation (OPA versus OMA). We can make additional use of this twofold interface. Whenever we have a system of systems setting with multiple scopes or domains of authority (e.g. virtual organisations strategic alliances, organisational fields), we can let an OPA of a given (sub-)organisation act as a member towards another OPA of another

organisation. This basically combines the middleware perspective with a holonic perspective (cf. [15]) and is not as easily to be conceptualised in the context of other middleware approaches that take a less distributed/modular perspective. In this paper the aspect of holonic systems is not discussed any further – cf. [16] for an in depth discussion.

All OPAs share a common structure which we call the *generic OPA (GOPA)*. An OPA O is an instance of this GOPA that is parametrised by that part of the organisational model that describes O, i.e. its inner structure (subtask and delegation/execution activities) and all the surrounding OPAs. The architecture of the GOPA is discussed in [17].

3.2 Deploying SONAR Organisations as OPA/OMA Networks

Oredi supports the deployment of Sonar formal organisations as OPA/OMA networks. The formal organisation net is exported in a XML format. The XML file generated from the organisation net is parsed and the deployment process begins. The generation of the XML file and its subsequent parsing was conceived to provide a platform and application independent solution to deploy Sonar organisations. As such, the generated XML file contains all formal information of the Sonar-model and can be used for deployment in any multi-agent platform. Here, we opted for an implementation of the deployment steps in our Mulan-architecture [4,5] as it supports building multi-agent systems with reference nets. This allowed using Petri nets both as a modelling as well as a programming technology thus easing and streamlining the gap between modelling and development. So, Mulan is our first choice, but of course any agent oriented languages would have done too.

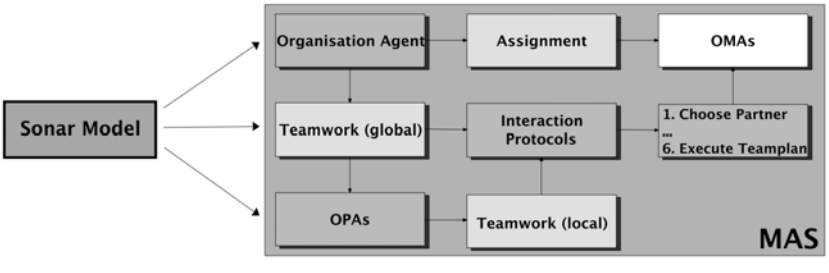

Fig. 4. Deployment of Sonar-Models

Deploying a Sonar organisation as OPA/OMA networks involves

1. the generation of an organisation agent, serving as a platform for the OPAs,
2. the generation all the OPAs, and
3. the assignment of OMAs to OPAs.

After these phases team processes such as team formation, team plan formation, and team plan execution can follow as specified in [6]. The whole compilation is

sketched in Figure 4. In this work, only the generation of the organisation agent and the OPAs and the assignment of OMAs is handled. For a discussion of the teamwork cf. [18,17].

Organisational positions of the organisation net are deployed as OPAs.[4] The generated OPAs know the identity of their neighbour OPAs and communicate with them through a set of encrypted messages. The assignment of OMAs to OPAs is made in a market based fashion with OPAs making open position announcements and interested OMAs competing for the employment for the open positions. The communication between OPAs and OMAs is also encrypted.

Deployment of Organisation Agents and OPAs in Mulan. OREDI deploys formal organisation as MULAN OPAs. MULAN is a FIPA [19] compliant architecture. At first, a MULAN platform is generated where one or more agent organisations can be embedded. The position agents generated for each position in the SONAR organisation net are placed inside the created platform. Additionally, suitable MULAN protocols handle agent conversations. MULAN agents can use protocols proactively or reactively as a response to specific messages. The decision which protocol to use for a specific received message is made in the knowledge base where a mapping between message templates and protocols is consulted.

OPAs have the complete information of their corresponding positions in the SONAR organisation net specified in the XML file. This information includes the position's relative place in the organisation (knowledge about neighbour positions), the roles they are implementing/delegating and their tasks. Generating OPAs out of an organisation net specification is accomplished in agent-oriented fashion by an initial agent. The initial agent is called the *organisation agent* as it has a global view on all positions. The organisation agent is responsible for the generation of the OPAs and their initialisation with information extracted from the formal organisation net specification.

The information needed from an OPA includes which other OPAs are its neighbours. This requires the identity of the neighbours. At least in MULAN, the identity of agents and their location can only be known after the creation of these agents. This means that information about the neighbour positions has to be provided for an OPA only after, not during its creation. Thus, the information about the place of a position agent in an organisation is conveyed through a conversation with the organisation agent. During the conversation, in order to make sure that the messages come from the right parties they are signed with a cryptographic mechanism which requires that parties know their respective keys. In Figure 5, an AUML sequence diagram displays the conversation between OPAs and the organisation agent during which the organisation agent communicates to the positions all relevant information extracted from the organisation net specification. In FIPA terminology, conversations between agents are called *protocols*. MULAN *protocols* describe the behaviour of agents during

[4] In fact, the OPAs are *compiled* into a *complete* specification for the GOPA instance. This specification is complete in the sense that all remaining, open design aspects are gathered in the implementation of the OMAs.

conversations. The AUML diagram in Fig. 5 also serves as an overall sketch of the used MULAN protocols which can be generated automatically from the AUML diagrams. Cf. [5] for an overview of the MULAN tool family.

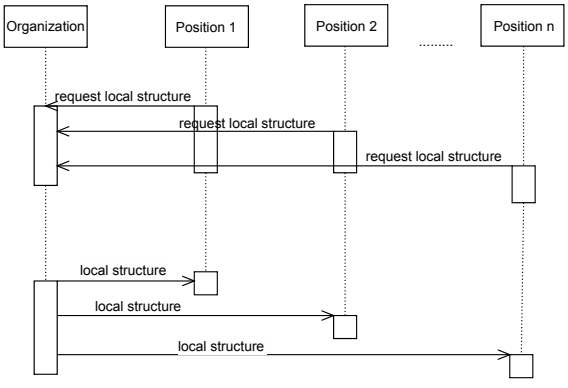

Fig. 5. The organisation agent communicating the information extracted from the respective positions to the generated OPAs

The conversation displayed in Fig. 5 is based on the assumption that the OPAs already know the identity of their organisation agent. However, the organisation agent does not know the identities of its OPAs. OPAs send a message with their identifiers to their organisation agent requesting their local structure which should include all the relevant information extracted from the respective positions in the SONAR organisation net such as the neighbours, the implementing and delegating roles, the tasks, etc. After receiving the requests for the local structure and the identifiers from all the position agents, the organisation agent proceeds and sends the respective local structure to each position agent. The conversation partners know their respective crypto keys so all the messages of the conversation are signed with the private keys of the sending parties. However, the aspect of authentication has been left out from Fig. 5 for simplicity.

Assignment of OMAs to OPAs. After the generation of the OPAs and the communication of the local structures to them, the assignment of OMAs to OPAs is started. The approach for the assignment process is leaned on [20]. As the organisation agent represents some kind of a service provider and logical platform to the OPAs and the potential OMAs, it assumes at this point the management of the assignment of OMAs to OPAs. If an OPA has an open position (either because its OMA resigned or it has been fired), the OPA sends a request to the organisation agent to start the procedure for the occupation of the open position. The organisation agent publishes a job description for the open position to a central registry component named **D**irectory **F**acilitator (DF) – see FIPA Agent Management Specification. A DF is a mandatory component of an agent platform in FIPA that provides a yellow pages directory service to

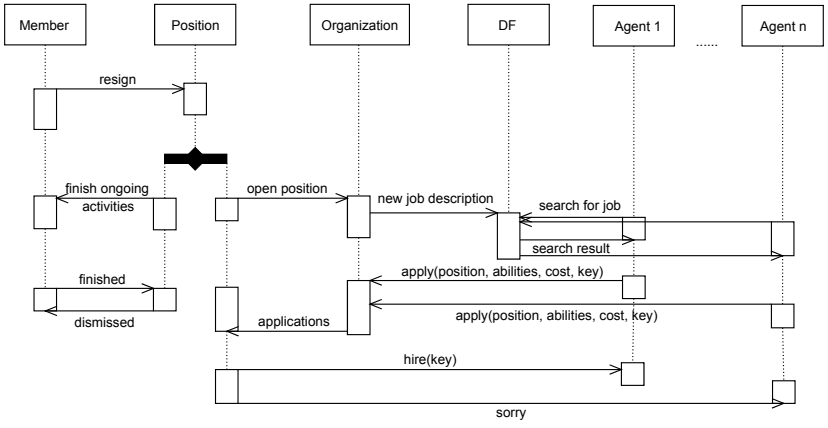

Fig. 6. The assignment of an agent as a member to a position agent

agents. Agents can advertise their services through the DF. In MULAN the DF is also an agent with which the organisation agent can communicate.

The external agents that are interested in occupying open positions in the organisation can search through the DF and apply to the organisation agent for a specific open position. The initial assignment of OPAs with OMAs is a special case where all OPAs have open positions. The diagram in Fig. 6 displays the case when the resignation of a member triggers the start of the procedure to assign a new member for the position. The organisation agent sends a description for a new job to the DF. The job description contains the identifier of the vacant OPA, the requirements that applicants have to fulfil, and a time period during which applications for the job are accepted. Agents that find that job description interesting after a search in the DF, apply to the organisation agent. Their application includes the description of the job for which they are applying, their crypto key, their personal abilities as a response to the requirements specified in the job description, and the costs for their service. The applications are received from the organisation agent.

After the application period for a vacant position expires, the organisation agent sends all received applications to the respective position agent, the OPA selects the new member and lets it know the key for the authentication during their future communication as well as the fact that it has been hired. In the case of resignation from a member, the member is dismissed only after finishing its ongoing activities on behalf of the organisation. Even if a new member may have been hired since the resignation request of the old member, the old member is dismissed only after finishing all its ongoing activities. This means that a position can have more than one member agents for limited time periods. An alternative way for an agent to get the list of open positions within an organisation is to send a message to the organisation agent itself with a request for the open positions that the organisation has.

For handling the communication for both the generation of the OPAs and the assignment of OMAs to OPAs an ontology was developed. The ontology was developed with Protégéand was employed as a domain specific shared vocabulary throughout the conversations between the agents involved in the two phases.

3.3 Related Work

Our approach lies in the research area of organisation centred design of multi-agent systems. There is a recent collection on approaches to agent systems in [1]. Among them we like to mention OPERA [21], MOISE [22], and ISLANDER [23].(Of course there are a lot of approaches on agent-oriented software engineering, like GAIA [24], but those usually do not explicitly rely in the metaphor of the organisation.) These methodologies are equipped with their middleware: OperettA [12], $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [13], and Amelie [14].

We provide a more detailed comparison of our SONAR-approach to MOISE and ISLANDER in [25] where we also derive conclusions concerning best fits between different approaches and application contexts. Compared to other middleware layers, we advocate complete distribution. Instead of introducing one or more middleware managers that watch over the whole organisation (cf. the manager in $\mathcal{S}$-$\mathcal{M}$OISE$^+$) or at least over considerable parts (cf. institution, scene, transition managers in Amelie), we associate each position with its own *organisational position agent* (OPA).

4 Conclusion and Outlook

Oredi, the tool presented in this paper, is a Petri net based software tool for modelling Sonar organisations as well as for deploying these models to agent organisations. By providing a tool that carries out specification as well as deployment of formal organisations a close link between these two phases of system development has been provided. With Oredi users can build Sonar organisations through a combination of graphical interfaces, a set of interaction constraints and context based suggestions without being required to possess active knowledge of the formal rules underlying the models' specifications. The deployment of formal organisation models is based on the decoupling of the elements of the multi-agent system that specify the organisational structure from those that act as members of the organisation. Positions in the formal organisation are deployed as agents (OPAs) and are embedded within the organisation agent, all implemented as Mulan agents. Oredi provides the specification and implementation of the assignment of OMAs to OPAs. Provided the necessary capabilities, any agent, developed in any programming language, can be assigned to an OPA as its OMA and take over the execution of the necessary tasks.

Besides, the deployment of formal organisation nets as agent organisations can also be extended to include the team formation, team plan formation and team plan execution phases as the corresponding theoretical foundations have already been provided in [18,17,26].

In the future extensions of OREDI both the modelling and the deployment phases will be subject to further development. A special focus should be laid on the modularisation of the OREDI models – using the refinement operation of SONAR – as it can allow the distribution of the modelling process. Formal organisations can be partitioned into modules which can be developed and maintained in separate files by different parties.

This modularisation will support the holonic approach taken in SONAR where not only positions are part of an organisation, but also (sub-)organisations. Organisations can be linked to each other by delegation relationships. Linking organisations can be achieved by adding extra graphical components to OREDI that represent the link to other sub-organisations.

References

1. Dignum, V. (ed.): Handbook of Research on Multi-Agent Systems IGI Global. Information Science Reference (2009)
2. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
3. Kummer, O., Wienberg, F., Duvigneau, M., et al.: An extensible editor and simulation engine for Petri nets: Renew. In: Cortadella, J., Reisig, W. (eds.) ICATPN 2004. LNCS, vol. 3099, pp. 484–493. Springer, Heidelberg (2004)
4. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 224–241. Springer, Heidelberg (2001)
5. Cabac, L., Dörges, T., Rölke, H.: A monitoring toolset for Petri net-based agent-oriented software engineering. In: van Hee, K.M., Valk, R. (eds.) ATPN 2008. LNCS, vol. 5062, pp. 399–408. Springer, Heidelberg (2008)
6. Köhler-Bußmeier, M., Wester-Ebbinghaus, M., Moldt, D.: A formal model for organisational structures behind process-aware information systems. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets. LNCS, vol. 5460, pp. 98–114. Springer, Heidelberg (2009)
7. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae 79, 415–430 (2007)
8. Aalst, W.v.d.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
9. Köhler, M., Ortmann, J.: Formal aspects for service modelling based on high-level Petri nets. In: International Conference on Intelligent Agents, Web Technologies and Internet Commerce, IAWTIC 2005 (2005)
10. Köhler, M., Moldt, D., Ortmann, J.: Dynamic service composition: A petri-net based approach. In: Conference on Enterprise Information Systems (ICEIS 2006), pp. 159–165 (2006)
11. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer, Heidelberg (2006)
12. Okouya, D., Dignum, V.: Operetta: a prototype tool for the design, analysis and development of multi-agent organizations. In: AAMAS 2008, pp. 1677–1678 (2008)
13. Hübner, J.F., Sichman, J.S., Boissier, O.: S-Moise: A middleware for developing organised multi-agent systems. In: International Workshop on Organizations in Multi-Agent Systems (OOOP 2005), pp. 107–120 (2005)

14. Esteva, M., Rodriguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: An agent-based middleware for electronic institutions. In: AAMAS 2004, pp. 236–243 (2004)
15. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organization of multiagent systems. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS (LNAI), vol. 2744, pp. 71–80. Springer, Heidelberg (2003)
16. Wester-Ebbinghaus, M., Moldt, D.: Modelling an open and controlled system unit as a modular component of systems of systems. In: International Workshop on Organizational Modelling (OrgMod 2009), pp. 81–100 (2009)
17. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: Sonar: A multi-agent infrastructure for active application architectures and inter-organisational information systems. In: Braubach, L., van der Hoek, W., Petta, P., Pokahr, A. (eds.) MATES 2009. LNCS (LNAI), vol. 5774, pp. 248–257. Springer, Heidelberg (2009)
18. Köhler, M., Wester-Ebbinghaus, M.: Organizational models and multi-agent system deployment. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 307–309. Springer, Heidelberg (2007)
19. FIPA: FIPA 97 Specification, Part 1 - Agent Management. Technical report, Foundation for Intelligent Physical Agents (1998), `http://www.fipa.org`
20. Köhler-Bußmeier, M.: SONAR: Eine sozialtheoretisch fundierte Multiagentensystemarchitektur. In: Selbstorganisation und Governance in künstlichen und sozialen Systemen. Lit Verlag, Münster (2009)
21. Dignum, V., Dignum, F., Meyer, J.J.: An agent-mediated approach to the support of knowledge sharing in organizations. Knowledge Engineering Review 19, 147–174 (2004)
22. Boissier, O., Hannoun, M., Sichman, J.S., Sayettat, C.: MOISE: An organizational model for multi-agent systems. In: 7th Ibero-American Conference on AI, pp. 156–165. Springer, Heidelberg (2000)
23. Esteva, M., de la Cruz, D., Sierra, C.: Islander: an electronic institutions editor. In: Falcone, R., Barber, S.K., Korba, L., Singh, M.P. (eds.) AAMAS 2002. LNCS (LNAI), vol. 2631, pp. 1045–1052. Springer, Heidelberg (2003)
24. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Trans. Softw. Eng. Methodol. 12, 317–370 (2003)
25. Wester-Ebbinghaus, M., Köhler-Bußmeier, M., Moldt, D.: From multi-agent to multi-organization systems: Utilizing middleware approaches. In: Artikis, A., Picard, G., Vercouter, L. (eds.) ESAW 2008. LNCS, vol. 5485, pp. 46–65. Springer, Heidelberg (2009)
26. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: A petri net based prototype for mas organisation middleware. In: Workshop on Modelling, Object, Components, and Agents (MOCA 2009), pp. 29–44 (2009)