Generating Executable Multi-Agent System Prototypes from SONAR Specifications

Michael Köhler-Bußmeier, Matthias Wester-Ebbinghaus, Daniel Moldt

University of Hamburg, Department for Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
(koehler,wester,moldt)@informatik.uni-hamburg.de

Abstract. This contribution presents the MULAN4SONAR middleware and its prototypical implementation for a comprehensive support of organisational teamwork, including aspects like team formation, negotiation, team planning, coordination, and transformation. Organisations are modelled in SONAR, a Petri net–based specification formalism for multi-agent organisations. SONAR models are rich and elaborated enough to automatically generate all necessary configuration information for the MULAN4SONAR middleware.

1 Introduction

Organisation-oriented software engineering is a discipline which incorporates research trends from distributed artificial intelligence, agent-oriented software engineering, and business information systems (cf. [1, 2] for an overview). The basic metaphors are built around the interplay of the macro level (i.e. the organisation or institution) and the micro level (i.e. the agent). Organisation-oriented software models are particularly interesting for self- and re-organising systems since the system's organizing principles (structural as well as behavioral) are taken into account explicitly by representing (in terms of reifying) them at run-time.

The following work is based on the organisation model SONAR (Self-Organising Net Architecture) which we have presented in [3, 4]. In this paper we turn to a middleware concept and its prototypical implementation for the complete organisational teamwork that is induced by SONAR.

First of all we aim at a rapid development of our middleware prototype. Therefore we need a specification language that inherently supports powerful high-level features like pattern matching and synchronisation patterns. The second requirement is a narrow gap between the specification and implementation of the middleware prototype. Ideally, middleware specifications are directly executable. As a third requirement, we are interested in well established analysis techniques to study the prototype's behaviour. As a fourth requirement we want the middleware specifications to be as close as possible to the supported SONAR-model of an organization. Related to this, the fifth requirement results as the possibility to be able to directly generate the middleware specifications from the SONAR-model automatically. The sixth requirement is that we want an easy translation of the prototype into an agent programming language.

Since SONAR-models are based on Petri nets we have chosen high-level Petri nets [5] as the specification language for our middleware prototype. This choice meets the requirements stated above: We can reuse SONAR-models by enriching them with high-level features, like data types, arc inscription functions etc. Petri nets are well known for their precise and intuitive semantics and their well established analysis techniques, including model checking or linear algebraic techniques. We particularly choose the formalism of *reference nets*, a dialect of high-level nets which supports the nets-in-nets concept [6] and thus allows to immediately incorporate ("program") micro-macro dynamics into our middleware. Reference nets receive tool support with respect to editing and simulation by the RENEW tool [7]. Additionally, RENEW has been extended by the agent-oriented development framework MULAN [8, 9], which allows to program multi-agent systems in a language that is a hybridisation of reference nets and Java. We make use of MULAN and provide a middleware for SONAR-models. Consequently, our middleware is called MULAN4SONAR and we present a fully-functional prototype in this paper.

The paper is structured as follows: Section 2 briefly sketches our formal specification language for organisational models, called SONAR. Section 3 addresses our MULAN4SONAR middleware approach on a rather abstract and conceptual level. It illustrates the structure of our target system: SONAR-models are compiled into a multi-agent system consisting of so called position agents, i.e. agents that are responsible for the organisational constraints. Section 4 describes our implemented middleware prototype in detail. It is generated from SONAR-models. The middleware serves integration and control of all organisational activities, like team formation, negotiation, team planning, coordination, and transformation. We consider related work in Section 5 before we close the paper with a conclusion in Section 6.

2 The Underlying Theoretical Model: SONAR

In the following we give a short introduction into our modelling formalism, called SONAR. A SONAR-model encompasses (i) a data ontology, (ii) a set of interaction models (called *distributed workflow nets, DWFs*), (iii) a model, that describes the team-based delegation of tasks (called *role/delegation nets*), (iv) a network of organisational positions, and (v) a set of transformation rules A detailed discussion of the formalism can be found in [3], its theoretical properties are studied in [4].

In SONAR a formal organisation is characterised as a delegation network of sub-systems, called *positions*. Each position is responsible for the execution or delegation of several tasks. Figure 1 illustrates the relationship between the SONAR interaction model, the delegation model and the position network – i.e. the aspects (ii) to (iv).[1] The left side of the figure describes the relationship between the positions (here: *broker*, *virtual firm*, *requester*, etc.) in terms of their respective roles (here: *Producer*, *Consumer* etc.) and their associated delegation links. In

[1] To keep the model small we we have omitted all data-related aspects and transformation rules – i.e. the aspects (i) and (v) – in this figure.

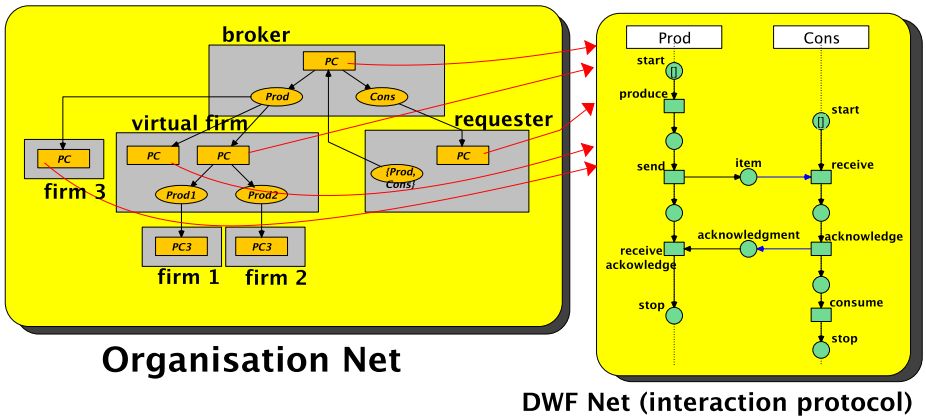

Fig. 1. A simplified SONAR-Model

this scenario, we have a requester and two suppliers of some product. Coupling between them is provided by a broker.[2] 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: *firm 1 firm 2*. 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.

SONAR relies on the formalism of Petri nets. Each task is modelled by a place p and each task implementation (delegation/execution) is modelled by a transition t. Each task place is inscribed by the set of roles which are needed to implement it, e.g. the set $\{Prod, Cons\}$ for the place in the position *requester*. Each transition t is inscribed by the DWF net $D(t)$ that specifies the interaction between the roles. In the example we have two inscriptions: PC and $PC3$ where the former is show on the right of Figure 1. Positions are the entities which are responsible for the implementation of tasks.[3] Therefore, each node in $P \cup T$ is assigned to one position O.[4]

[2] Note that for this simplified model brokerage is an easy job, since there are only 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.

[3] The main distinction between roles and position is that positions – unlike roles – are situated in the organisational network, they implement roles and are equipped with resources.

[4] Organisation nets can be considered as enriched organisation charts. Organisation nets encode the information about delegation structures – similar to charts – and also about the delegation/execution choices of tasks, which is not present in charts.

So far we have used only the static aspects of Petri nets, i.e. the graph structure. But SONAR also benefits from the dynamic aspects of Petri nets: 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 [4]). It generates a *team net* (the team's structure) and a *team DWF*, i.e. the team's behavior specification.

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 [10]. As a minimal requirement the rules must preserve the correctness of the given organisational model. In SONAR transformations are not performed by the modeller – they are part of the model itself. Therefore we assume that a SONAR model is *stratified* by models of different levels. The main idea is that the activities of DWF nets that belong to the level n are allowed to modify those parts that belong to levels $k < n$ but not to higher ones.

3 Organisational Position Network Activities

We now elaborate on the activities of a multi-agent system behaving according to a SONAR-model.

3.1 Conceptual Overview

The basic idea is quite simple: With each position of a SONAR-model we associate one dedicated agent, called an *organisational position agent* (OPA). This is illustrated in Figure 2 where the OPAs associated with a SONAR-model together embody a middleware layer.

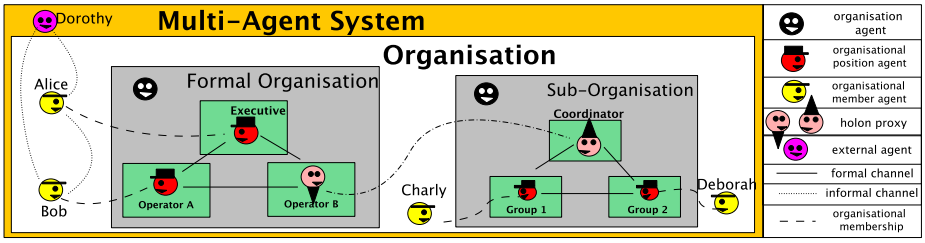

Fig. 2. An Organisation as an OPA/OMA Network

An OPA network embodies a formal organisation. An OPA represents an organisational *artifact* and not a *member/employee* of the organisation. However,

If one fuses all nodes of each position into one single node, one obtains a graph which represents the organisation's chart. Obviously, this construction removes all information about the organisational processes.

each OPA represents a conceptual connection point for an *organisational member agent* (OMA). An organisation is not complete without its OMAs. Each OMA actually interacts with its OPA to carry out organisational tasks, to make decisions where required. OMAs thus implement/occupy the formal positions.[5] 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 (like Alice and Bob in Figure 2) or even to agents completely external to the organisation (like Alice and Dorothy). 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 the complete 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. [11]).

3.2 Organisational Teamwork

SONAR-models of organisations induce *teamwork activities*. We distinguish between organisational teamwork activities of first- and of second-order. First-order activities target at carrying out "ordinary" business processes to accomplish business tasks.

- *Team Formation:* Teams are formed in the course of an iterated delegation procedure in a top-down manner. Starting with an initial organisational task to be carried out, successive task decompositions are carried out and subtasks are delegated further. A team net according to Section 2 consists of the positions that were involved in the delegation procedure.
- *Team Plan Formation/Negotiation:* After a team has been formed, a compromise has to be found concerning how the corresponding team DWF net (cf. Section 2) is to be executed as it typically leaves various alternatives of going

[5] Note that from a technical point of view, the OPA network *is* already a complete MAS. This MAS is highly non-deterministic since a SONAR-model specifies what is allowed and what is obligatory, so many choices are left open. Conceptually, the OPA network represents the *formal organisation* while the OMAs represent its *informal* part which in combination describe the whole organisation.

one way or the other. A compromise is found in a bottom-up manner with respect to the team structure. The "leaf" positions of the team net tell their preferences and the intermediary, inner team positions iteratively seek compromises between the preferences/compromise results of subordinates. The final compromise is a particular process of the team DWF net and is called the team plan.
- *Team Plan Execution:* As the team plan is a DWF net process that describes an interaction between team positions, team plan execution follows straightforward.[6]
- *Hierarchic propagation:* If a holonic approach as illustrated in Figure 2 is chosen, team activities that span multiple organisations are propagated accordingly.

Second-order activities reorganisation efforts.

- *Evaluation:* Organisational performance is monitored and evaluated in order to estimate prospects of transformations. To estimate whether an organisational transformation would improve organisational performance, we introduce *metrics* that assign a multi-dimensional assessment to a formal organisation. In addition to the Petri net-based specifications of the previous section, there may exist additional teamwork constraints and parameters that may be referred to. How to measure the quality of an organisational structure is generally a very difficult topic and highly contingent. We will not pursue it further in this paper.
- *Organisational Transformations:* As described in Section 2, transformations can either be applied to a formal organisation externally or be carried out by the positions themselves as transformation teams (cf. exogenous versus endogenous reorganisation [12]). In the latter case, transformations are typically triggered by the above mentioned evaluations. But it might also be the case that a new constraint or directive has been imposed and the organisation has to comply.

3.3 Organisation Agents

As shown in Figure 2 all the OPAs of an organisation are within the context of an *organisation agent* which represents the OPA network as a whole. The organisation agent is responsible for the management of the organisational domain data (e.g. customer databases etc.) but also for the management of the organisational *meta data* which includes the data ontology, the interaction protocols (i.e. the process ontology), and also a representation of the SONAR-model itself. This is illustrated in the top half of Figure 3.

Additionally, the organisation agent is responsible for the network wide framing of the organizational teamwork efforts, i.e. team formation, negotiation, and

[6] For the time being, we do not address the topic that team plan execution might fail and what rescue efforts this might entail.

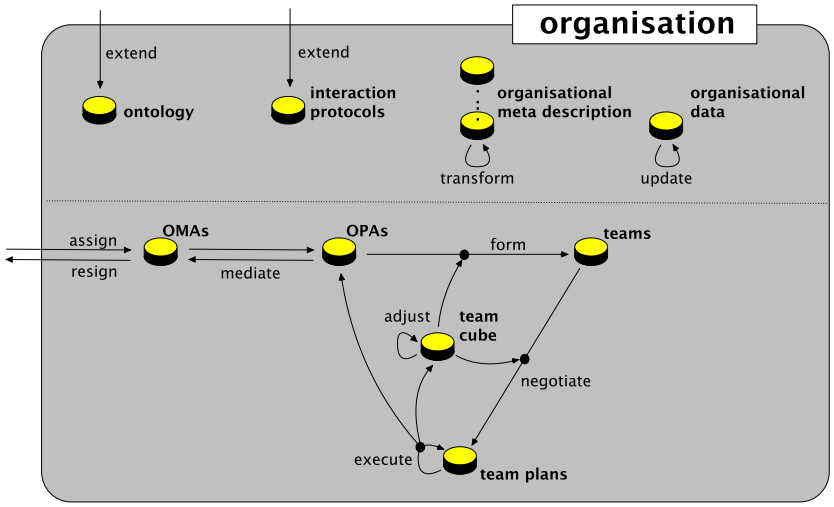

Fig. 3. The Organisation Agent

team plan execution (as illustrated in the bottom half of Figure 3). The organisation agent is responsible for monitoring the *abstract* aspects on the teamwork (i.e. the OPA network perspective), while the OPAs are responsible for the *concrete* decisions (i.e. the OPA perspective).[7] For example, the organisation agent abstractly specifies that during the team formation the OPA O may delegate some task to another agent which must belong to a certain set of OPAs,[8] but the concrete choice for a partner is left to the OPA O which in turn coordinates its decision with its associated OMA.

In our architecture the concrete choices of the OPAs are framed by the so called *team cube* (cf. Figure 3). The notation *cube* is due to the fact that we have three dimension of teamwork: team formation, negotiation, and team plan execution. For each dimension we can choose between several mechanisms. For example in the team formation phase the delegation of tasks to subcontractors can either be implemented by a market mechanism (i.e. choosing the cheapest contractor), by a round-robin scheduling (i.e. choosing contractors in cyclic order), or even by some kind of "affection" between OPAs/OMAs. Given a concrete situation that initiates a teamwork activities, the organisation chooses an appropriate mechanism for

[7] Note that the existence of a single agent representing the organisation has not to be confused with a monolithic architecture. The main benefit of the existence of an organisation agent is that it allows to provide a network-wide *view* on the team activities.

The abstract aspects could as well be implemented by the OPAs themselves and thus be totally distributed. In fact the concurrency semantics of Petri nets perfectly reflects this aspect: In the mathematical sense the processes of an organisation agent are in fact distributed, even if generated from one single net.

[8] This set of possible delegation partners is calculated from the SONAR-model.

each of the three dimensions. During the execution phase of the team plans the team cube evaluates the process to improve the assignment of mechanisms.

4 The Mulan4Sonar Middleware

Each position of a Sonar-organisation consists of a formal part (the OPA as an organisational artifact) and an informal part (the OMA as a domain member). An organisation together with the OPA network relieves its associated OMAs of a great part of the organisational overhead by automation of administrative and coordination activities. It is exactly the *generic* part of the teamwork activities from Section 3.2 that is automated by the organisation/OPA network: Team formation, team plan formation, team plan execution always follow the same mechanics and OMAs only have to enter the equation where domain actions have to be carried out or domain-dependent decisions have to be made.

4.1 Compilation of Sonar Specifications into Mulan4Sonar

In the following we demonstrate the compilation of an organisational Sonar-model into the Mulan4Sonar middleware layer for automated teamwork support. A Sonar-model is semantically rich enough to provide all necessary information to allow an automated generation/compilation. The aspects of this compilation and the resulting prototypical middleware are discussed using the organisation example introduced above in Figure 1. The prototypical middleware layer generated from this Sonar-model is specified by a high-level Petri net, namely a reference net. This is beneficial for two reasons: (1) the translation result is very close to the original specification, since the prototype directly incorporates the main Petri net structure of the Sonar-model; (2) the prototype is immediately functional as reference nets are directly executable using the open-source Petri net simulator Renew [7] and we can easily integrate the prototype into Mulan [8, 9], our developing and simulation system for MAS based on Java and reference nets. Therefore we have chosen to implement the compiler as a Renew-plugin.

The plugin implements a compiler that is based on graph rewriting. The compiler searches for a net fragment in the Sonar-model that matches the pattern on the left hand side of a rewrite rule and translates it into a reference net fragment which is obtained as the instantiation of the rule's right hand side. An example rule with the parameter n is given in Figure 4: The rule attaches a place for the OPA a to the transition. In the final model this place contains the OPA that represents the position "position name". The rule also adds inscriptions that describe that OPA a is willing to implement the task t (denoted by the inscription a:askImpl("t")) and a list of inscriptions a:askPartner("p_i", O_i) (one for each $p_i, 1 \leq i \leq n$) describing that a delegates the subtask p_i to the OPA O_i. The variable x denotes the identifier of the teamwork process.

We consider teamwork in six phases. For each phase, the original Sonar-model (in our case the one from Figure 1) is taken and transformation rules generate

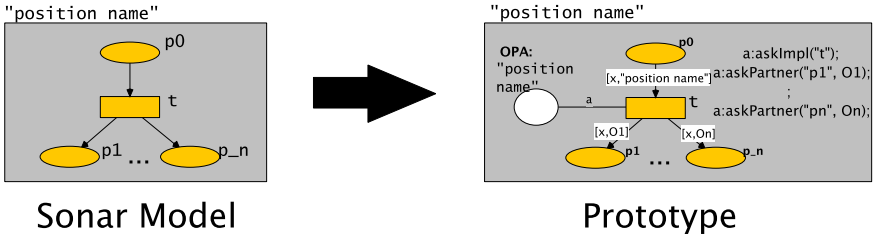

Fig. 4. A transformation rule for Phase 1

an executable reference net fragment. For example, the transformation rule from Figure 4 is used for the first phase, *selection of team members* (see below). Finally, the fragments for the phases 1 to 6 are linked sequentially and the resulting overall net represents the main (organisation-specific) middleware component that is used in the (generic) Mulan4Sonar middleware layer to coordinate the organizational teamwork. The six teamwork phases are the following:

1. *Selection of team members:* By agents receiving tasks, refining them and delegating sub-tasks, the organisation is explored to select the team agents. This way, a team tree is iteratively constructed but the overall tree is not globally known at the end of this phase.
2. *Team assembly:* The overall team tree is assembled by iteratively putting sub-teams together. At the end of this phase, only the root agent of the team tree knows the overall team.
3. *Team announcement:* The overall team is announced among all team member agents.
4. *Team plan formation:* The executing team agents (i.e. the leaves of the team tree) construct partial local plans related to the team DWF net. These partial plans are iteratively processed by the ancestors in the team tree. They seek compromises concerning the (possibly conflicting) partial plans until the root of the team tree has build a global plan with a global compromise.
5. *Team plan announcement and plan localisation:* The global team plan is announced among all team member agents. The executing team agents have to localise the global plan according to their respective share of the plan.
6. *Team plan execution:* The team generates an instance of the team DWF net, assigns all the local plans to it, and starts the execution.

Here, we will only discuss first-order organisational teamwork. However, our Mulan4Sonar middleware approach features a recursive system architecture in order to support reorganization, including second-order activities (a presentation of the whole model can be found in the technical report [13]).

Before the six phases are discussed in more detail, we illustrate how a Mulan multi-agent system that incorporates our Mulan4Sonar middleware layer looks like.

4.2 Multi-Agent System with Mulan4Sonar Middleware Layer

In Section 3, we have described our general vision of a multi-agent system that incorporates Sonar organisations: The formal part of each Sonar organization is explicitly represented by a distributed middleware layer consisting of OPAs for each position and one organisation agent as an additional meta-level entity. In our current prototypical implementation of the Mulan4Sonar middleware layer, the organisation agent is actually not yet fully included, at least not as an *agent*. Instead, the organisation agent of a Sonar organisation manifests itself in terms of the generated six-phase reference net explained in the previous subsection (together with possible DWF nets). This concept is illustrated in Figure 5.

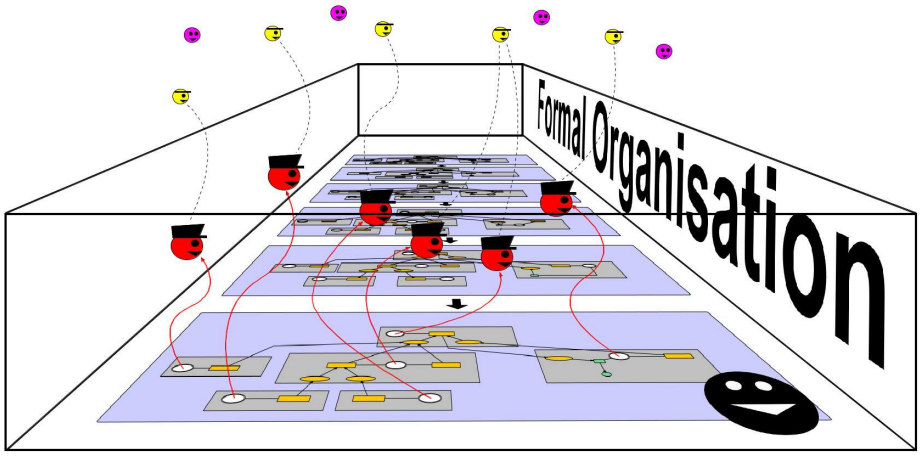

Fig. 5. Mulan4Sonar middleware layer in the current prototype

It is shown that the formal part of a Sonar organisation is embodied by the generated middleware net and the position agents that are hosted on the *agent places* of the net. Here we do not elaborate on the internal structure of the agents as we would have to go into the details of multi-agent system programming with Mulan which is out of the scope of the paper. All OPAs share the same generic OPA architecture (GOPA) that we have presented in [14]. Note that in the current prototype, the OPAs are *directly embedded* on the agent places of the middleware net. This is justified as they are actually reified *parts* of the formal organisation and we assume that the whole middleware (and thus the formal organization) is executed on the same Mulan platform. The OMAs however are external agents that have chosen to act as members of the organization. Consequently, they can be hosted on remote platforms and communicate with their respective OPAs via message passing.

For future developments of our MULAN4SONAR middleware we plan to have the organisation agent to be actually realized as a MULAN agent (see Subsection 4.4).

4.3 Explanation of the Six Teamwork Phases

As explained in Subsection 4.1, a SONAR-model of an organisation is compiled into executable reference nets for each of the six teamwork phases. Afterwards, the reference nets for the phases 1 to 6 are combined in one reference net and linked sequentially. This linkage is achieved via synchronisation inscriptions. Thus, the end of a phase is synchronised with the start of the succeeding phase.

The reference nets for the six phases share the same net structure but have different inscriptions. This reflects the fact that all teamwork is generated from the same organisational SONAR-specification, but in different phases different information is needed. Figure 6 shows the generated reference net for the first phase, *selection of team members*.[9]

Before any teamwork can occur, the system setup has to be carried out. Six position agents (OPAs) – one for each position – are initialized and registered. The position agents are hosted on the agent places of the generated middleware net. After this step the initialisation is finished and teamwork may ensue.

For our given SONAR-model we have only one position that is able to start a team, namely O_4 since it is the only position having a place with an empty preset (i.e. the place p_0). Whenever the position agent O_4 decides to begin teamwork, it starts the first phase, *team member selection*. The only possibility for task p_0 is to delegate it to O_1. Here, O_1 has only one implementation possibility for this task, namely t_1. This entails to generate the two subtasks p_1 and p_2. O_1 selects the agents these subtasks are delegated to. For p_2 there is the only possibility O_4 but for p_1 there is the choice between O_2 and O_3. Partner choices occur via the synchronisation a:askPartner(p, O) between the middleware net and the position agents: Agent a provides a binding for the partner O when the task p has to be delegated. Assume that the agent O_1 decides in favour of O_3, then the control is handed over to O_3 which has a choice how to implement the task: either by t_2 or by t_5. This decision is transferred between position agents and middleware net via the synchronisation a:askImpl(t) which is activated by the agent a only if t has to be used for delegation/implementation.

After this iterated delegation has come to an end – which is guaranteed for well-formed SONAR-models – all subtasks have been assigned to team agents and the first phase ends. At this point the agents know that they are team members, but they do not know each other yet. To establish such mutual knowledge the second phase starts.

We cannot cover every phase in detail. The general principle has been shown for phase one, namely enriching the original SONAR model of an organisation with (1) connections to position agents and (2) execution inscriptions along the purpose of the respective teamwork phase.

[9] Note that the rule from Figure 4 has been applied several times.

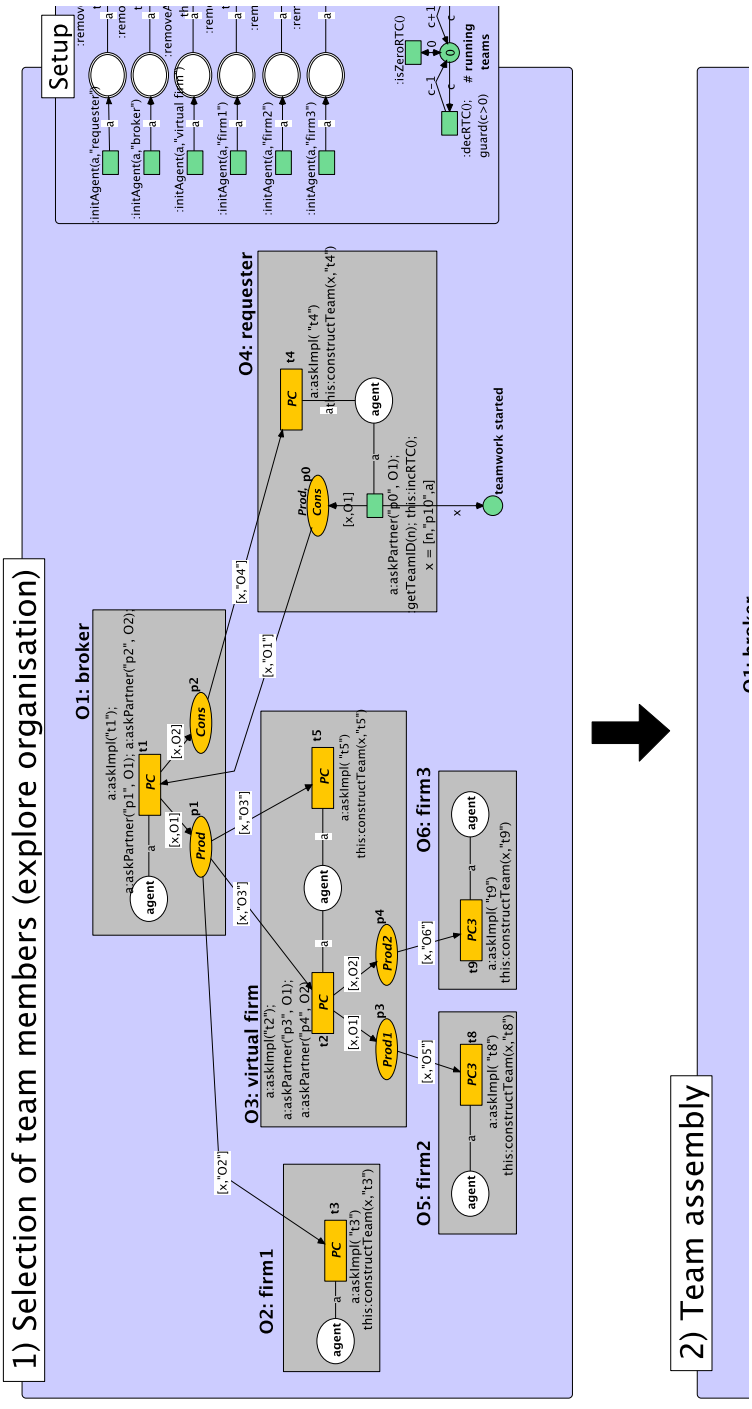

Fig. 6. Zoom: First Phase of the MULAN4SONAR-middleware

The purpose of the remaining five phases has been covered in Subsection 4.1. Here, we want to cover one technical aspect specifically. The description of the first phase has made clear that it is a top-down phase. Following the delegation relationships of the original SONAR-model, a team tree is built from the root down to the leaves. It is also clear that the second phase has to be a bottom-up phase. The overall team is not yet known to any position agent. Thus, beginning with the leaves of the team tree and the corresponding "one-man sub-teams", sub-teams are iteratively assembled until the complete team is finally known at the root node. Consequently, for the second phase, the direction of the arrows has to be reversed compared to the original SONAR model. Analogous observations hold for the remaining four phases. Phases 3 and 5 are top-down phases while phases 4 and 6 are bottom-up phases.

4.4 Strengths, Weaknesses and Future Work

In this subsection, we give a brief qualitative evaluation of the approach taken in this paper. SONAR is a formal model of organisations based on Petri nets. It is often difficult to initially come up with an approach to deploy formal specifications in a software environment. In the case of the Petri net specifications, one can take advantage of the inherent operational semantics. In this sense, Petri nets often allow for a rapid prototyping approach to go from abstract models (requiring only simple Petri net formalisms) to fully functional, executable models (requiring high-level Petri net formalism, in our case reference nets). Consequently, our first approach was to take a SONAR-specification of an organisation and derive an executable prototype by manually attaching inscriptions and add some auxiliary net elements.

While manually crafting an executable reference net for each specific SONAR-model is of course not worthwhile in the long run, it provided us with very early lessons learned and running systems from the beginning on. The work presented in this paper was the next step. Based on our experiences from the handcrafted prototypes we were able to clearly denominate and devise the transformation rules that were needed for automated generation of executable reference net fragments from SONAR-models.

Consequently, we see the conceptual as well as operational closeness between an underlying SONAR-model and its generated middleware net as a crucial advantage for our fast progress in deploying SONAR-organisations. In addition, formal properties like well-formedness (cf. [3, 4]) of a SONAR-model directly carry over to the implementation level.

However, there are also problems associated with our current approach. Firstly, organisational specifications at run-time are only available in terms of the reference net generated from the underlying SONAR-model. This format is not very suitable for being included in an agent's reasoning processes. Secondly, reorganization efforts are only achieved via a workaround. Changing only particular elements of a reference net at runtime is not inherently supported by our environment. Thus, for a reorganization of an organization, the whole middleware net has to be replaced.

Because of the mentioned problems, we are working on further improving the MULAN4SONAR middleware. Current efforts target at keeping the organizational specification as a more accessible and mutable data structure at the level of the middleware layer. Although it is no longer necessarily represented as a reference net itself, the organisational activities and dynamics allowed by the middleware layer are still directly derived from the underlying Petri net semantics of SONAR.

5 Related Work

Our work is closely related to other approaches that propagate middleware layers for organisation support in multi-agent systems like $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [15], AMELI [16] or TEAMCORE/KARMA [17]. The specifics of each middleware layer depends on the specifics of the organizational model that is supported. What all approaches have in common is that domain agents are granted access to the middleware layer via *proxies* that constrain, guide and support an agent in its function as a member of the organisation, cf. *OrgBox* in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, *Governor* in AMELI, *Team Wrapper* in TEAMCORE/KARMA. Our organisational position agents, the OPAs, serve a similar purpose. The are coupled with organisational member agents, the OMAs, which are responsible for domain-level actions and decisions.

However, in the case of $\mathcal{S}$-$\mathcal{M}$OISE$^+$ and AMELI, management of organisational dynamics is mainly taken care of by middleware manager agents (the *OrgManager* for $\mathcal{S}$-$\mathcal{M}$OISE$^+$ and the *institution, scene* and *transition managers* for AMELI). The proxies mainly route communication between the domain level agents and the middleware managers. Consequently, middleware management is to some degree centralised.[10] In our case, the OPAs are both proxies and middleware managers. They manage all six phases of organisational teamwork in a completely distributed way. This is quite similar to the function of the Team Wrappers in TEAMCORE/KARMA. The KARMA middleware component can be compared to the organisational agent in our approach. It is a meta-level entity that is responsible for setting up the whole system and for monitoring performance.

In [19], we additionally study the conceptual fit between different middleware approaches (in combination with the organisational models they support) and their application on different levels of a large-scale system of systems.

6 Conclusion

In this paper, we have built upon our previous work SONAR on formalising organisational models for MAS by means of Petri nets [4, 3]. In particular, the paper is dedicated to a prototypical MULAN4SONAR middleware layer that supports the deployment of SONAR-models. As SONAR-specifications are formalised with

[10] However, in the case of $\mathcal{S}$-$\mathcal{M}$OISE$^+$, the new middleware approach ORA4MAS [18] (*organizational artifacts for MAS*) has been devised, resulting in a more decentralised approach.

Petri nets, they inherently have an operational semantics and thus already lend themselves towards immediate implementation. We have taken advantage of this possibility and have chosen the reference net formalism as an implementation means. Reference nets implement the nets-in-nets concept [6] and thus allow us to deploy SONAR-organisations as nested Petri net systems. The reference net tool RENEW [7] offers comprehensive support, allowing us to refine/extend the SONAR specifications into fully executable prototypes.

This leaves us with a close link between a SONAR specification of an organisation and its accompanying MULAN4SONAR middleware support. The structure and behaviour of the resulting software system is directly derived and compiled from the underlying formal model. For example, we have explicitly shown how the organisation net of a formal SONAR-specification can be utilised for the middleware support of six different phases of teamwork. In each phase, the original net is used differently (with different inscriptions and arrow directions). This approach of deploying SONAR-models does not only relieve the developer of much otherwise tedious programming. It also allows to preserve desirable properties that can be proven for the formal model and that now carry over to the software technical implementation.

Finally, although we have introduced the idea of SONAR-organizations acting in the context of other SONAR-organizations, we have not addressed the topic in detail here. We study this subject in [20, 21], but on a more abstract/generic level than SONAR offers. Nevertheless, we have already begun to transfer the results to SONAR.

References

1. Carley, K.M., Gasser, L.: Computational organisation theory. In Weiß, G., ed.: Multiagent Systems. MIT Press (1999) 229–330
2. Dignum, V., ed.: Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global (2009)
3. Köhler-Bußmeier, M., Wester-Ebbinghaus, M., Moldt, D.: A formal model for organisational structures behind process-aware information systems. Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems **5460** (2009) 98–114
4. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae **79** (2007) 415 – 430
5. Girault, C., Valk, R., eds.: Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications. Springer (2003)
6. Valk, R.: Object Petri nets: Using the nets-within-nets paradigm. In Desel, J., Reisig, W., Rozenberg, G., eds.: Advanced Course on Petri Nets 2003. Volume 3098 of LNCS, Springer (2003) 819–848
7. Kummer, O., Wienberg, F., Duvigneau, M., Schumacher, J., Köhler, M., Moldt, D., Rölke, H., Valk, R.: An extensible editor and simulation engine for Petri nets: Renew. In Cortadella, J., Reisig, W., eds.: International Conference on Application and Theory of Petri Nets 2004. Volume 3099 of LCNS, Springer (2004) 484 – 493
8. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In Colom, J.M., Koutny, M., eds.: International Conference on Application and Theory of Petri Nets. Volume 2075 of LNCS, Springer (2001) 224–241

9. Cabac, L., Dörges, T., Duvigneau, M., Moldt, D., Reese, C., Wester-Ebbinghaus, M.: Agent models for concurrent software systems. In Bergmann, R., Lindemann, G., eds.: Proceedings of the Sixth German Conference on Multiagent System Technologies, MATES'08. Volume 5244 of LNAI, Springer (2008) 37–48
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer (2006)
11. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organization of multiagent systems. In: Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS). Volume 2744 of LNCS, Springer (2003) 71–80
12. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open organizations. In O'Hare, G., Ricci, A., O'Grady, M., Dikenelli, O., eds.: Engineering Societies in the Agents World VII. Volume 4457 of LNCS, Springer (2007) 86–105
13. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: A Petri net based prototype for MAS organisation middleware. In Moldt, D., ed.: Workshop on Modelling, object, components, and agents (MOCA'09), University of Hamburg, Department for Computer Science (2009) 29–44
14. 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.: Conference on Multi-Agent System Technologies, MATES 2009. Volume 5774 of LNAI, Springer (2009) 248–257
15. 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: From Organizations to Organization-Oriented Programming (OOOP 2005). (2005) 107–120
16. Esteva, M., Rodriguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: An agent-based middleware for electronic institutions. In Sierra, C., Sonenberg, L., Tambe, M., eds.: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004). (2004) 236–243
17. Pynadath, D., Tambe, M.: An Automated Teamwork Infrastructure for Heterogeneous Software Agents and Humans. In: Autonomous Agents and Multi-Agent Systems, 7(1–2). (2003) 71–100
18. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents: Giving the organisational power back to the agents. In: Autonomous Agents and Multi-Agent Systems, 20(3). (2010) 369–400
19. Wester-Ebbinghaus, M., Köhler-Bußmeier, M., Moldt, D.: From Multi-Agent to Multi-Organization Systems: Utilizing Middleware Approaches. In: Alexander, A., Picard, G., Vercouter, L., eds.: Engineering Societies in the Agents World IX. Volume 5485 of LNCS, Springer (2008) 46–65
20. Wester-Ebbinghaus, M., Moldt, D.: Modelling an open and controlled system unit as a modular component of systems of systems. In Köhler-Bußmeier, M., Moldt, D., Boissier, O., eds.: International Workshop on Organizational Modelling (OrgMod'09), University of Paris (2009) 81–100
21. Wester-Ebbinghaus, M., Moldt, D.: Structure in threes: Modelling organization-oriented software architectures built upon multi-agent systems. In: Proceedings of the 7th International Conference an Autonomous Agents and Multi-Agent Systems (AAMAS'2008). (2008) 1307–1311