Automatic Generation of Distributed Team Formation Algorithms from Organizational Models

Michael Köhler-Bußmeier and Matthias Wester-Ebbinghaus

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, D–22527 Hamburg
{koehler,wester}@informatik.uni-hamburg.de

Abstract. Software systems are subject to ever increasing complexity and in need of efficient structuring. The concept of organization as an expressive and abstract real-world reference presents a promising starting point. Organizational concepts have particularly been studied within the multi-agent systems community. However, there exists a conceptual gap between organizational specifications and their multi-agent implementation. We address this problem by presenting an integrated approach to formalize organizational models with Petri nets and to directly deploy these specifications in a multi-agent system. The operational semantics of Petri nets establishes a close link between organizational specification and deployment that eases system development and maintenance. As an important example, we are able to describe the formation of multi-agent teams in an organizational scenario in terms of Petri net dynamics.

1 Introduction

Inspiration from organization theory serves to handle phenomena in multi-agent systems that carry a *supra-individual* character. Especially in the past few years multi-agent system researchers have begun to comprehend the organizational metaphor as the central instrument to combine local agent autonomy with reliability and predictability on the system level (we provide a detailed overview of recent and current work concerning modelling approaches, development methodologies and middleware in [1]). This combination is achieved by imposing "organizational facts" onto the system. Boissier [2] identifies several organizational dimensions that different approaches incorporate to varying degrees. Among these dimensions, the most prominent are the structural, functional and interactional (dialogical) dimensions. They respectively carry specifications concerning the structure of the collective level of a multi-agent system, its global functioning and the interactions between agents in terms of communication.

One central questions is how to close the conceptual gap between high-level organizational specifications of a software system to be and its multi-agent system implementation. The above mentioned organizational dimensions are not mutually independent. Only in concert they draw a coherent picture. Thus agents

are embedded in a system of interrelated organizational specifications and left with the task of operationalizing them. One possibility to deal with this problem is to introduce organizational dimensions with the specific purpose to establish an explicit link between other dimensions. For example, $\mathcal{M}$OISE$^+$ [3] and ISLANDER [4] exhibit *deontic* and *normative* dimensions respectively. These enrich an agent's structural embedding with obligations and permissions towards functional and dialogical specifications. Some approaches with a particular emphasis on *teamwork* go even further. They link agents to an organizational framework in terms of the agents' (joint) *mental states* [5, 6]. Depending on the structural embedding of the agents (which roles they occupy) team and team plan formation are carried out by establishing mutual beliefs, joint commitments and joint intentions concerning the following joint team activity.

In this paper we present a related but distinctive approach. We propose an approach to model organizations based on Petri nets. It integrates a structural, a functional and an interactional dimension. However, the peculiarity of using Petri nets as a modelling technique renders additional dimensions or mechanisms for paving the way to operationalization obsolete. The operational semantics of Petri nets allows them to be directly utilized in a software implementation. We exemplify this idea by defining teams solely in terms of Petri net dynamics and specifically elaborate on the case of team formation in an organizational setting. In Section 2 we present our formal Petri net model of organizations. In Section 3 we demonstrate how the formal specifications may be instantiated in the form of a multi-agent system and how they directly lead to a distributed algorithm for team formation to be carried out by the agents. In Section 4 we give full particulars on the role of each individual agent in the course of the team formation procedure and thus link the organizational coordination plans with individual action plans. We conclude our results in Section 5 and relate them to the wider context of our current work.

2 Formal Model of Organizations

We present a Petri net model of organizations. It is intentionally lean and open to be extended or specialized for specific purposes. Our model focuses on the two fundamental (and opposing) requirements for every organized activity postulated by Mintzberg [7], the *division of labour* into various tasks to be performed and the *coordination* of carrying out these tasks in order to accomplish the overall activity. Accordingly, our model basically includes *tasks*, *roles* included in the accomplishment of these tasks and (hierarchical as well as vertical) *task-based relationships between roles* for the coordination. Our Petri net-based approach to formalize these concepts allows to elegantly combine both the structural (through the structure of a Petri net model itself) and the behavioural (through the operational semantics of Petri nets) properties of organizational models.

In this section we address the *formal organization* without reference to the organization's members [8]. It is not until the next subsection that we turn

to multi-agent system deployment and thus to the *informal organization* with reference to agents as members of the organization.

Petri nets offer both a graphical representation and formal semantics. In [9] we present our model in more detail together with formal theorems and proofs. In this paper we just assume a general understanding of Petri nets in terms of places, transitions and arcs as their basic elements. For a thorough introduction into Petri nets we refer to [10].

Definition 1. *A* Petri net *is a tuple $N = (P, T, F)$ where P is a set of places, T is a set of transitions, disjoint from P, i.e. $P \cap T = \emptyset$, and $F \subseteq (P \times T \cup T \times P)$ is the flow relation.*

The preset *of a node y is ${}^\bullet y := (\_\ F\ y)$ and* postset *is $y^\bullet := (y\ F\ \_)$. For a set $A \subseteq P \cup T$ we define ${}^\bullet A = \bigcup_{a \in A} {}^\bullet a$ and $A^\bullet = \bigcup_{a \in A} a^\bullet$.*

The minimal nodes are ${}^\circ N = \{x \in P \cup T | {}^\bullet x = 0\}$, the maximal are $N^\circ = \{x \in P \cup T | x^\bullet = 0\}$

A finitely branching Petri net $N = (P, T, F)$ is called a causal net *iff the transitive closure F^+ is acyclic and for all $p \in P$ the conditions $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$ hold.*

Turning to Petri net dynamics, we have to consider markings and transition firing. We define the mappings $\mathbf{pre}, \mathbf{post} : T \to (P \to \mathbb{N})$ by $\mathbf{pre}(t)(p) := |F \cap \{(p,t)\}|$ and $\mathbf{post}(t)(p) := |F \cap \{(t,p)\}|$. A marking of a Petri net (P, T, F) is a multiset (or: a vector) of places: $m \in \mathbb{N}^P$. A transition $t \in T$ of N is *enabled* in marking m iff $m(p) \geq \mathbf{pre}(t)(p)$ holds for all $p \in P$. The successor marking m' obtained after firing t is defined $m'(p) = m(p) - \mathbf{pre}(t)(p) + \mathbf{post}(t)(p)$. Firing is denoted $m \xrightarrow{t} m'$. The notation extends to firing sequences the usual way. The set of all *reachable markings* is $RS(m) = \{m' \mid m \xrightarrow{*} m'\}$.

2.1 Roles and Services

We begin by introducing roles and services in the context of our model.

Roles describe specific expectation structures with respect to the behaviour of agents. Technically speaking, roles are some kind of type for an agent describing its behaviour. Given a set of *atomic roles Rol* the set $\mathcal{R} := 2^{Rol} \setminus \emptyset$ is the role universe. Each $R \in \mathcal{R}$ is called a *role*. The structure of roles is modelled by the partially ordered set $(\mathcal{R}, \subseteq)$. The singletons sets $\{r\}$ are identified with r itself. Whenever $R_1 \subseteq R_2$ for $R_1, R_2 \in \mathcal{R}$, we say that R_1 is more specialized than R_2.

Services in our model are accompanied by *service nets* that represent the model's interactional dimension. A service net is a Petri net that describes the interactions that take place between agents in order to carry out the corresponding service. Each transition models a task and places connect tasks in order to form life lines and message exchanges. However, service nets abstract away from particular agents and identify the interacting roles instead. We assume a unique assignment of roles to service nets.

Definition 2. *A* service net *$D = (N, r)$ is a Petri net $N = (P, T, F)$ with a role assignment $r : T \to Rol$. Define $R(D) := r(T)$. A set of service nets $\mathcal{D}$ is called* service set*, whenever $R(D_1) \cap R(D_2) = \emptyset$ for all different $D_1, D_2 \in \mathcal{D}$.*

Given a service net D and a role $R \subseteq R(D)$ we can restrict D to the subnet $D[R] = (P_R, T_R, F_R)$ of D defined by the nodes related to R: $T_R := r^{-1}(R)$ and $P_R := ({}^\bullet T_R \cup T_R{}^\bullet)$.

Figures 2 and 3 show examples of service nets (details concerning the content of this services are delayed until the following subsection). Each transition t of a service net D is assigned to the atomic role $r(t) \in \mathit{Rol}$ with the meaning, that only agents that implement this role r are able to execute the task t. In Figure 2 all transitions drawn below one of the role names (*producer*, *consumer*) are assigned to it. The subnet $PC[\mathit{producer}]$ is indicated by the filled nodes.

Service nets are very similar to agent UML (AUML) interaction diagrams. Whenever a place p connects two transitions t_1 and t_2 with $r(t_1) = r(t_2)$ the transitions are drawn vertically on the so called *live line* of the role. Whenever $r(t_1) \neq r(t_2)$ then the place p models a *message exchange* which is drawn horizontally (here: *item* and *acknowledgement*). For details cf. [11].

2.2 R/D Nets and Formal Organizations

Based on the former specifications of roles and services we define R/D nets, which are used to describe formal organizations and embody both the structural and functional dimension of our model. R/D nets are Petri nets where each place models a role and each transition models a task. Each place p is labelled with a role $R(p)$ and each transition t with a service net $D(t)$ that describes how the corresponding task is to be carried out. For the net models it is a natural restriction to allow exactly one place in the preset of a task since each task is related to the role it can be used by.

Definition 3. *A* R/D net *is the tuple (N, R, D) where*

1. *$N = (P, T, F)$ is a Petri net with $p^\bullet \neq \emptyset$ for all $p \in P$ and $|{}^\bullet t| = 1$ for all $t \in T$.*
2. *$\mathcal{R}$ and $\mathcal{D}$ are a role universe and a service set respectively.*
3. *$R : P \to \mathcal{R}$ is the role assignment with $\forall t \in T : \forall p, p' \in t^\bullet : p \neq p' \implies R(p) \cap R(p') = \emptyset$.*
4. *$D : T \to \mathcal{D}$ is the service net assignment.*

An organizational structure in our model is built up by *organizational positions* (OP). OPs roughly correspond to a positions in real-world organizations. An OP may encompass several roles by being responsible for several tasks. In addition, it is possibly allowed to delegate role parts for the accomplishment of these tasks to other OPs. OPs are modelled as disjoint subsets of $P \cup T$ of an R/D net. We request the connectivity property. Whenever a task t belongs to an OP, then all used roles $p \in t^\bullet$ belong to it, whenever p belongs to an OP, so do all using tasks $t \in {}^\bullet p$.

Definition 4. *An* organization net *is a pair $(N, \mathcal{O})$ where N is a Petri net $N = (P, T, F)$ and $\mathcal{O}$ is a partitioning on the set $P \cup T$ where for all $O \in \mathcal{O}$ the following holds (with $\bar{O} = (P \cup T) \setminus O$):*

$$\forall p \in O \cap P : {}^\bullet p \subseteq O \wedge p^\bullet \subseteq \bar{O} \quad \wedge \quad \forall t \in O \cap T : {}^\bullet t \subseteq \bar{O} \wedge t^\bullet \subseteq O$$

An element $O \in \mathcal{O}$ is called organizational position *(OP). By $O(y)$ we denote the unique OP for each $y \in P \cup T$.*

A formal organization *is the tuple $Org = (N, \mathcal{O}, R, D)$ where (N, R, D) is a R/D net and $(N, \mathcal{O})$ is an organization net.*

Example 1. An organization net is given in Figure 1 (the purpose of the dashed lines will not be addressed until the following subsection).

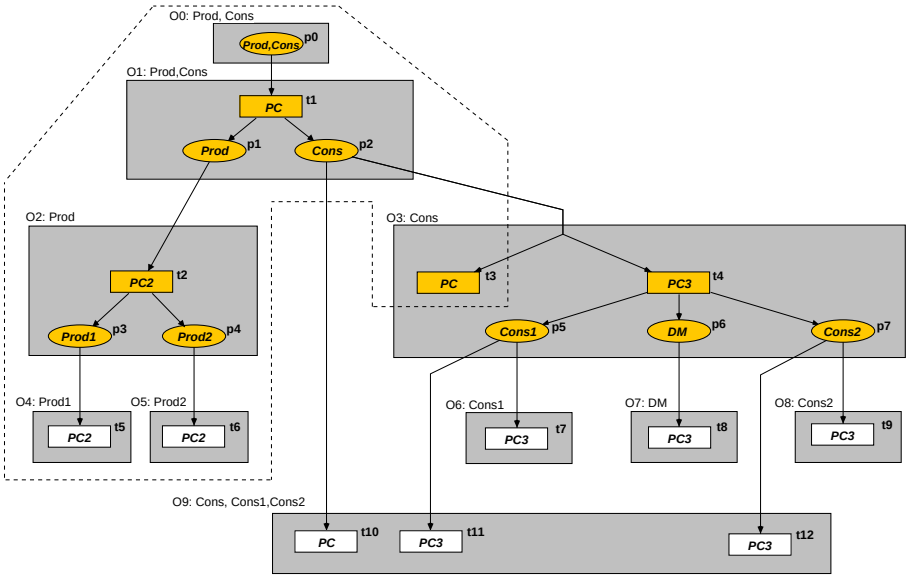

Fig. 1. Producer/Consumer organization net

OPs are depicted as filled rectangles. The OPs O_1, O_2, O_3, O_9 are complex, the OP O_0 is an initial one, and the other OPs are final ones. The organization net models a simple producer/consumer example. The place p_0 is the starting point of activity which has to start with $t_1 \in O_1$. In O_1 the work is divided into its producing and consuming parts, resulting in the roles *Producer* and *Consumer* respectively. The implementation of these roles is delegated to other positions. O_2 is the only OP to implement the role *Producer*. It does so by further dividing the work into two parts and delegating the resulting two roles $Producer_1$ and $Producer_2$ to the OPs O_4 and O_5 respectively, which finally implement their share of the work themselves. Turning to the consuming part of the work, OP O_3 may decide to implement the role *Consumer* itself via task t_3 or to further divide the work and delegate the implementation of the resulting roles. Finally, OP O_9 is able to implement all variants of roles for consuming work. It might for example be considered an emergency OP for the case where the regular OPs responsible for consuming tasks are overloaded with work.

Formal models can be used to analyze structure and behaviour. As an example of this benefit to our approach we refer to *well-formedness* of R/D nets. In

a well-formed R/D net, all service nets and all roles are consistently related to the structure of the organization net. We do not provide a formal definition here (cf. [9]) but give an example. Figure 2 shows the service net PC that is assigned to transition t_1 and describes the interaction between the two roles of *Producer* and *Consumer*. First the producer executes the transition *produce*, then *sends*

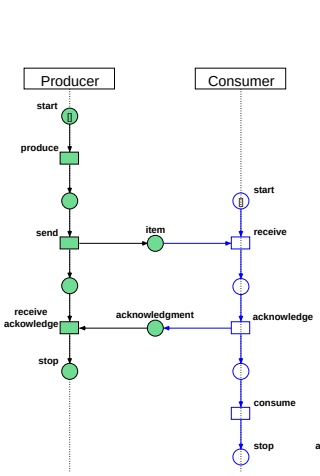

Fig. 2. Producer/Consumer: PC

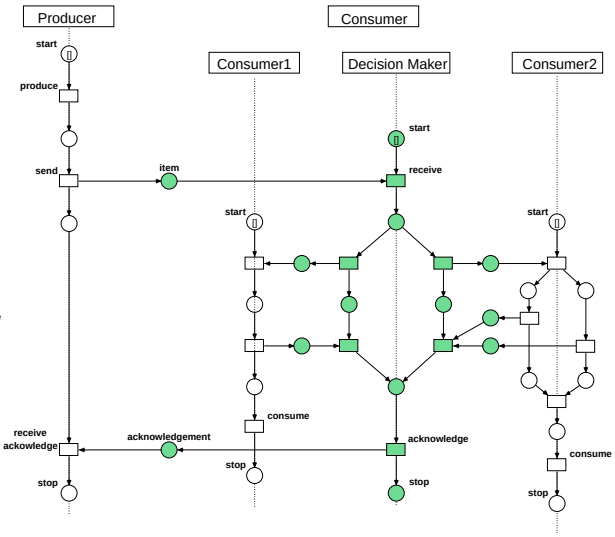

Fig. 3. Refined Producer/Consumer: PC_3

the produced *item* to the consumer, who *receives* it. The consumer sends an *acknowledge* to the producer before he *consumes* the item. Figure 3 presents the service net PC_3 that is assigned to transition t_4 and refines the role *Consumer* in the net PC of Figure 2: The role *Decision Maker* decides whether $Consumer_1$ or $Consumer_2$ receives the item. Basically, well-formedness in this case means that the service net PC_3 (assigned to t_4) is a *refinement* of PC (assigned to t_1) with respect to the former role *Consumer* (assigned to p_2 with $p_2 \in t_1^\bullet \wedge p_2 = {}^\bullet t_4$) by means of the further introduced roles $Consumer_1$, $DecisionMaker$ and $Consumer_2$ (assigned to $\{p_5, p_6, p_7\} = t_4^\bullet$).[1] The refinement concerns communication between the refining roles while input/output behaviour from the perspective of the producer remains the same.

2.3 Teams as Processes of Organization Nets

Organization nets capture all possibilities of task delegation and hence service accomplishment. In order to actually carry out organizational services, temporary structures that are cleansed from ambiguities are required. We call these structures *teams* and formally define them as follows.

[1] Here all roles consist of just one atomic role respectively so the distinction between roles assigned to places and atomic roles in service nets is somewhat artificial.

Definition 5. *A R/D net $N = (P, T, F)$ is called a* team *if N is causal net with exactly one minimal node: ${}^\circ N = \{p_0\}$, $p_0 \in P$, and all maximal nodes are transitions: $N^\circ \subseteq T$.*

In Figure 1 one possible team is framed by the dashed lines. This R/D net is dedicated to the specific purpose modelled by its only minimal node. All delegation ambiguities are resolved, resulting in a fixed allocation of task responsibilities to positions along with a fixed and fully elaborated (composed) interaction pattern for carrying out the corresponding organizational service. In this sense, the participating positions stand ready to act as a team.

In the following we characterize teams as the result of a team formation procedure that is controlled by the organizational structure. In particular, we relate teams to *processes* of organization nets. Petri net processes are a recognized alternative for describing the behaviour of Petri nets by firing sequences [12, 13]. Processes are themselves Petri nets from the class of causal nets. A process of a net N is defined as a causal net K together with a pair of mappings $\phi = (\phi_P, \phi_T)$. For a mapping $f : D \to D'$, let $f^\sharp : \mathbb{N}^D \to \mathbb{N}^{D'}$ denote its extension to a homomorphism on multisets: $f^\sharp\left(\sum_{i=1}^n x_i\right) = \sum_{i=1}^n f(x_i)$.

Definition 6. *Let $N = (P, T, F)$ be a Petri net with the initial marking m, $K = (B, E, \lessdot)$ a causal net and $\phi = (\phi_P : B \to P, \phi_T : E \to T)$ a pair of mappings. Then (K, ϕ) is a* process *of (N, m) if the following holds:*

1. *Preservation of the flow relation: $x \lessdot y \Longrightarrow (\phi(x), \phi(y)) \in F$.*
2. *Representation of the initial marking: $\phi_P^\sharp({}^\circ K) = m$.*
3. *Preservation of localities: ${\phi_P}^\sharp({}^\bullet e) = {}^\bullet(\phi_T(e))$ and ${\phi_P}^\sharp(e^\bullet) = (\phi_T(e))^\bullet$.*
4. *Each node $x \in B \cup E$ has only finitely many predecessors.*

A process has the progress property *iff no transition is enabled in K°, i.e. for each subset $A \subseteq K^\circ$ there is no transition $t \in T$ such that $\phi(A) = {}^\bullet t$. The set of all finite processes with the progress property is denoted $\mathcal{K}(N, m)$.*

Alternatively, a process (K, ϕ) can be constructed from the possible firings, i.e. the enablement of transitions, of the net N. The construction is inductively defined for a process net, by adding transitions according to the enablement condition of the net N. The starting point is given by the initial marking, which defines a simple process without any transitions, but only a place for each token in the initial marking. For the progress property this unfolding is continued until no transition is enabled.

In [9] we prove a practical property: Each process with the progress property introduces a team.

Theorem 1. *Let $Org = (N, \mathcal{O})$ be an organization net with $N = (P, T, F)$. Then for each $p \in P$ and each process $(K, \phi) \in \mathcal{K}(N, \{p\})$ we have that K is a team.*

This theorem allows us to characterize team formation processes in terms of the theory of Petri net processes: Each Petri net process (with the progress property) of an organization net generates a team.

3 Multi-Agent System Deployment

In this section we turn from formal organizations (without reference to members of the organization) as defined in Section 2 to their deployment inside multi-agent systems where agents populate an organization as its members. In order to benefit from the operational semantics of our Petri net-based approach, we require that the modelling of a formal organization does not just serve an offline design purpose but is directly operationalized within the software system and takes on steering and controlling responsibility. By this means, the formal specifications along with certain desired (provable) properties are directly mapped onto the resulting software system.

Consequently, our proposal specifically aims at multi-agent system architectures that allow for the direct deployment of Petri net models as software components. One particular candidate that most smoothly satisfies this requirement is the MULAN-architecture [14]. It models agents and multi-agent systems through the higher-order Petri net formalism of reference nets.[2] As reference nets may carry complex Java-instructions as inscriptions and thereby offer the possibility of Petri net-based programming, the Mulan models have been extended to a fully elaborated and running software architecture.

However, we consider our proposal not to be restricted to MULAN-deployment or to other target platforms that allow for the integration of Petri nets as executable software components. Instead, our Petri net models may be regarded as a special form of algorithmic description to be implemented in any desired way. In this case they still retain their character of being directly derived from the properties of the underlying Petri net-based organizational models.

3.1 Multi-Agent System Design

In the following we regard a multi-agent system as an instance of a formal organization $Org = (N, \mathcal{O}, R, D)$. Each organizational position $O \in \mathcal{O}$ is allocated to a *position agent* $Ag(O)$. Figure 4 displays an example.

Position agents integrate the structural and behavioural organizational specifications associated with their respective positions into their local reasoning. Thus, organizational coordination plans and individual action plans are mutually related. The organizational coordination plans imprint the individual action plans and conversely the individual actions of the position agents produce and reproduce the organization's global behaviour. In this sense we arrive at a dualistic relationship between micro and macro perspective within the multi-agent organization.

Position agents are connected through *channels* that are basically the delegation relationships of the underlying organization net $(N, \mathcal{O})$ with $N = (P, T, F)$.

[2] Reference nets show some extensions compared to conventional coloured Petri nets [15]. They implement the nets-within-nets paradigm where a surrounding net (the system net) can have *nets as tokens* (the object nets). Reference semantics is applied, so these tokens are *references* to *net instances*. *Synchronous channels* allow for communication between net instances.

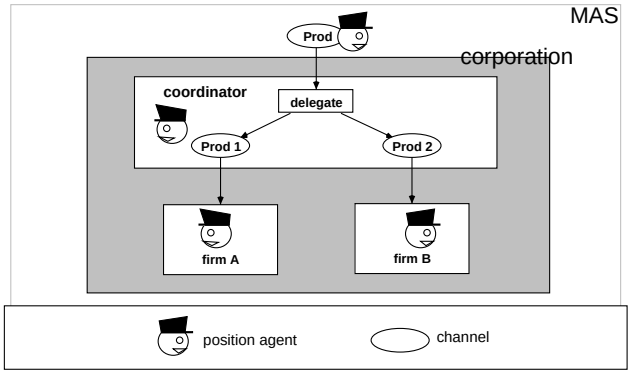

Fig. 4. Multi-agent system design derived from an organization net

As $\mathcal{O}$ is a partition on the set $P \cup T$ each task $t \in T$ is assigned to exactly one position agent $(Ag \circ O)(t)$. If $t^\bullet = \emptyset$ holds, the corresponding position agent carries out the task itself. If $t^\bullet \neq \emptyset$, the corresponding position agent delegates the task. Two agents $Ag(O_1)$ and $Ag(O_2)$ are connected through a directed delegation relationship if $\exists p \in P : p \in O_1 \wedge p \in {}^\bullet O_2$ holds.

In the case of an open multi-agent system environment with agents belonging to different stakeholders entering and leaving on a continual basis, we propose to further distinguish position from member agents as presented in [16]. The organization contracts each position to a member agent that carries out actions and makes decisions. But member agents must connect to the corresponding position agents and use them as (controlling and coordinating) proxies to the organization. This approach distinguishes between the organizational and the domain layer of applications and is an instance of the *common organization implementation architecture for open multi-agent systems* introduced in [2].

3.2 Operationalizing Organization Nets: The Team Formation Case

Besides guiding the design of corresponding multi-agent systems, organization nets may be directly operationalized in the implementation. Organizational processes are basically teamwork processes to carry out organizational services.[3]

Teamwork encompasses the stages of *team formation, team plan formation* and *team plan execution* [17]. The formal Petri net organizational models may guide all three stages and in the paper at hand we exemplarily elaborate on the case of team formation.

In our model, teams are formed through an iterated delegation procedure resting on the underlying organization net $(N, \mathcal{O})$ with $N = (P, T, F)$. Whenever

[3] In the context of this paper, we leave aside further aspects like for example processes for re-organization.

a position agent $Ag(O)$ ($O \in \mathcal{O}$) has been chosen as a team member to implement the role belonging to a place $p \in {}^\bullet O \cap P$ it applies an *internal team formation function* to select one of the implementation possibilities from the set

$$Impl(p, O) = \{t \in T \cap O \mid t \in p^\bullet\} \ .$$

After choosing one element $t \in Impl(p, O)$ and if $t^\bullet \neq \emptyset$ holds the agent $Ag(O)$ further applies an *external team formation function* for all $p \in t^\bullet$. For each p, it selects one of the possible team members from the set

$$TM(p) = \{O \in \mathcal{O} \mid p \in {}^\bullet O\} \ .$$

This delegation process as a whole corresponds to firing sequences of the organization net and thus constructs the net's processes and hence teams.

The distributed team formation algorithm is given in pseudo-code in Figure 5. In accordance with Subsection 2.3, the algorithm generates a finite process with the progress property $(K, \phi) \in \mathcal{K}(N, \{p\})$ for $p \in P$ as the starting point. Distribution is denoted by the different selection functions τ^1 and τ^2 where $\tau^1_A(K, \phi, b) \in TM(\phi(b))$ denotes the team member that an agent A selects and $\tau^2_A(K, \phi, b) \in Impl(\phi(b), Ag^{-1}(A))$ denotes the task that an agent A selects.

---

function $K_\tau(N, p)$ is
  $(K, \phi) := (((\{b_0\}, \emptyset, \emptyset), \{(b_0, p)\})$
  while $(K^\circ \cap B) \neq \emptyset$ do
    foreach $b \in (K^\circ \cap B)$ do
      $(K, \phi) := ((B', E', F'), \phi')$ where
        $TM = \tau^1_A(K, \phi, b)$ where $A = (Ag \circ O \circ \phi)(b)$
        $t = \tau^2_{TM}(K, \phi, b)$
        $B'_t = \{b_p \mid p \in t^\bullet\}$; $\phi' = \phi \cup \{(e, t)\} \cup \{(b_p, p) \mid p \in t^\bullet\}$
        $B' = B_K \uplus B'_t$; $E' = E_K \uplus \{e\}$; $F' = F_K \cup \{(b, e)\} \cup (\{e\} \times B'_t)$
  return (K, ϕ)

Fig. 5. The Team Formation Algorithm

4 A Petri-Net Model of Team Formation

The iterated delegation procedure for team formation according to the algorithm from Figure 5 is realized through a conversation between the involved position agents. We propose to utilize *deployment versions* of organization nets for each agent in order that they guide and control the conversation flow. Such a deployment version for a position agent basically takes the corresponding position fragment of the organization net and enriches it with inscriptions and additional net components for the operationalization. Deployment versions need not be modelled manually but can be generated automatically as all information that is needed rests in the initial corresponding organization net.

For the MULAN-architecture this approach is straightforward. In [14], it is described in detail how MULAN-agents participate in conversations. Each agents hosts an instantiated *protocol net* that implements the agent's role in the conversation by receiving and sending messages along a predefined pattern of communication. So deployment versions of organization nets may be integrated in protocol nets as object net tokens in order that they guide and control the conversation. For multi-agent system architectures other than MULAN the integration is likely to be more complicated. However, it does not differ *conceptually* as defining agent interaction patterns and distributing parts of them among agents according to different participating roles is common practice.

4.1 Team Formation: Position-Specific Deployment Version

Deployment fragments of organization nets for the case of team formation will now be examined in more detail. To begin with, we refer to our example from Section 2.2 illustrated in Figure 1. The deployed organization net for the position agent assigned to OP O_3 is shown in Figure 6.[4] It contains the subnet for O_3 as a substructure.

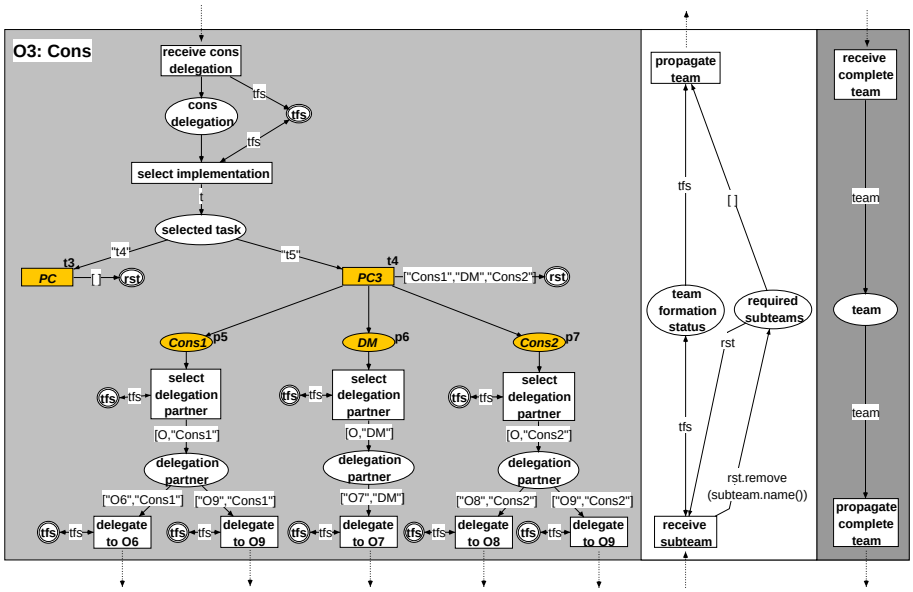

Fig. 6. Team formation: Deployment version for OP O_3 from Figure 1

Upon receiving the delegation call to implement the role *Consumer* (receive cons delegation), the team formation status is initialized (team formation sta-

[4] To avoid overloading the figure with too many details it does not contain all inscriptions that would be necessary for operationalization. This holds especially for the transitions whose function is only given by their name.

tus).[5] The team formation status encapsulates information about delegation partners (both higher and lower in delegation authority), team formation parameters (requirements, constraints), selected tasks and recursively generated subteams. The team formation status is updated when one of the two possible implementations for *Consumer* is chosen (select implementation). Depending on the selected task (t_3 or t_4), different subteams are required (required subteams). If the position agent decides to implement the role itself (t_3) no further subteams are required. If the position agent decides to further delegate the implementation of the task (t_4) the work is divided into further roles for delegation ($Consumer_1$, $DecisionMaker$, $Consumer_2$) and each of these requires its own subteam. Delegation partners have to be selected among possibly multiple options (select delegation partner) and the team formation status has to be updated accordingly. Eventually, the roles are delegated to the selected team members (delegate to O_x).

Whenever a subteam for a formerly delegated role is received (receive subteam), the team formation status is updated and the subteam is removed from the list of required subteams. If this list is empty, all required subteams have been received. Now the team formation status contains all information that is required to generate the team for the delegation level of O_3 and can be forwarded to the higher level of delegation authority (in this case O_1) as an implementation for the role *Consumer* (propagate team).

Finally, the position agent will receive the complete team description from its higher level of delegation authority (receive complete team) and forwards it to its lower level team members (propagate complete team).

Recall that deployment versions of organization nets are to be utilized inside protocol nets (or some other mechanism to implement agent participation in conversations). It is not at the level of the deployed organization net that issues like mapping of agents to positions, addressing or message generation are dealt with. These are deferred to the next higher level of conversation management. Deployed organization nets only contain information that can directly by derived from the underlying organization net.

It should be obvious that the transitions select implementation and select delegation partner are the most prominent options for refinement. Different scenarios are imaginable. For example a position agent might have to accept all delegations at all odds. Otherwise it might be allowed to negotiate performance conditions depending on its own workload and that of its lower delegation partners. Or it might even be allowed to reject delegation calls.

4.2 Team Formation: General Deployment Version

The deployed organization net from Figure 6 is specifically tailored for OP O_3 from Figure 1. It is a very simple example with just one partner of higher del-

[5] The twofold circuited places are *virtual places*. Reference nets offer virtual places as exact copies of the original ones for the sake of readability, avoiding arcs all across the diagram. In this case the virtual places tfs are copies of the place team formation status and the virtual places rst are copies of the place required subteams.

egation authority and only two implementation variants. Still, the deployment version is a quite large net due to operationalization overhead. More complex organizational positions would lead to very large and probably hard to arrange deployment versions. Instead, Figure 7 shows a coloured version of the former deployment version that can be used for an *arbitrary* organizational position.

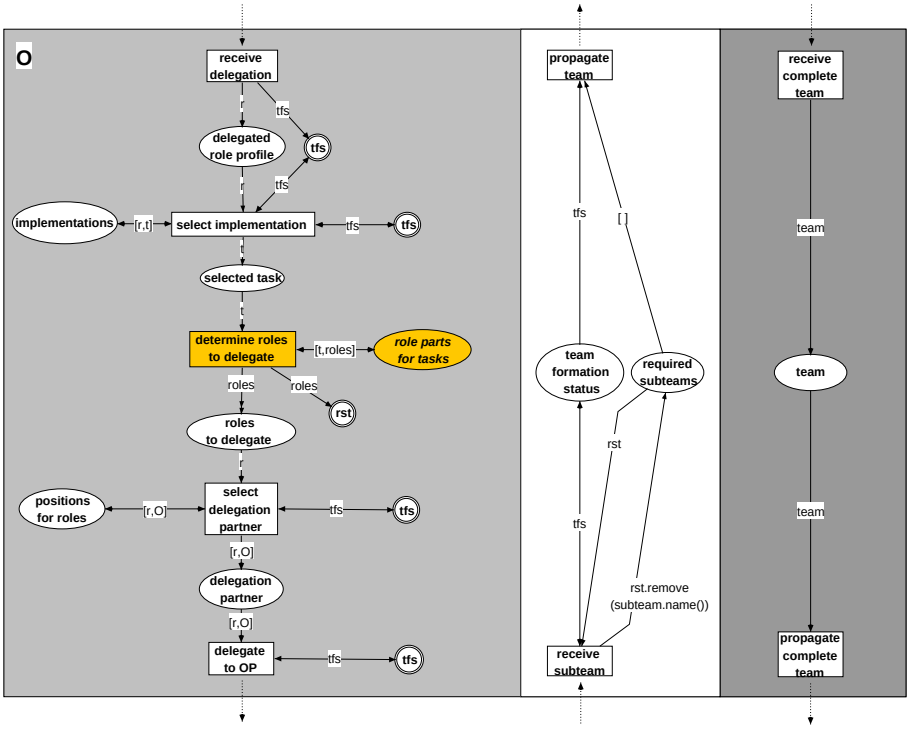

Fig. 7. Team formation: General deployment version for organizational positions

Basically, the generalized version introduces associations between delegated roles and tasks (implementations), tasks and roles to delegate (role parts for tasks), and roles and organizational positions (positions for roles) as side conditions for the selection of implementations, the determination of role parts for an implementation and the selection of delegation partners respectively. Each association is simply a tuple that rests as a token on the corresponding place. Consequently, an organizational position might be arbitrarily complex and still be tackled by the deployment version from Figure 7 for the sake of team formation. A rise in complexity does only result in additional tokens on the places implementations, role parts for tasks and positions for roles.

This deployment version is still only usable for one occurrence of a delegation call. If the tokens related to specific delegations would be arranged as tuples with an identification number as a further element, the net from Figure 7 could even be used as a permanent component that all protocol nets (or some other

mechanism to implement agent participation in conversations) could use jointly and simultaneously.

5 Outlook

We presented an approach to use Petri nets to both specify and deploy organizational models. The result is an integrated approach to develop organization-oriented software systems that especially focuses on closing the conceptual gap between design and implementation. Moreover, the approach explicitly facilitates change. As the specification is immediately used to structure and operationalize the multi-agent system and deployment versions of organization nets can be generated at run time, changing the specifications consequently changes the structure and behaviour of the software system.

In addition, our proposal to distinguish between position and member agents in the case of open multi-agent environments also aims at incorporating our model into existing multi-agent systems. No matter what the predominant structures and relationships are, our model can introduce a new organizational level (or just selected organizational features/domains) by means of a position agent infrastructure to which existing agents connect as members and which they use for coordination. The former structures and relationships of course do not become obsolete but correspond to the informal channels besides the formal ones defined by the organization.

Turning to future work, we consider the result of the paper at hand as one step into the direction of our vision of organization-oriented software engineering that we published in [1]. We present a software architecture based on *organizational units*. Four levels of organizational units are distinguished according to different degrees of abstraction, namely *department, organization, organizational field* and *society*. These levels are analogous to the levels of analysis that Scott proposes in [8] for organizations in human societies. We consider this architecture to accompany a shift in paradigm from agent-oriented to organization-oriented software engineering. It is no longer the agent that represents the central design metaphor and software building block but the organization. However, it is the multi-agent systems as the smallest units of the architecture upon which all other levels of (logical) organizational units are built. In this paper we defined the conceptual and operational connection of positions to their organizational embedding. The next step on our way to arrive at a multi-level architecture is to regard a position as possibly being occupied by a complex organizational unit in itself. This results in a conception of nested organizational units that in the whole form a system with multiple organizational levels. From the perspective of a particular organization, some of these levels are to be regarded as intra-organizational (departments, divisions, offices) while others are to be regarded as inter-organizational (fields, markets, mergers, societies).

References

1. Wester-Ebbinghaus, M., Moldt, D., Reese, C., Markwardt, K.: Towards Organization–Oriented Software Engineering. In Züllighoven, H., ed.: Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings. Volume 105 of LNI., GI (2007) 205–217
2. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open systems. In: Proceedings of the Seventh International Workshop on Engineering Societies in the Agents World (EASW 2006). (2006)
3. Hübner, J.F., Sichman, J.S., Boissier, O.: A model for the structural, functional and deontic specification of organizations in multiagent systems. In: Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02). Volume 2507 of Lecture Notes in Artificial Intelligence., Springer Verlag (2002)
4. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In Meyer, J.J., Tambe, M., eds.: Pre-Proceedings of the 8th International Workshop on Agent Theories, Architectures and Languages (ATAL-2001). (2001) 106–119
5. Pynadath, D., Tambe, M.: An automated teamwork infrastructure for heterogeneous software agents and humans. Autonomous Agents and Multi-Agent Systems **7**(1–2) (2003) 71–100
6. Sonenberg, E., Tidhar, G., Werner, E., Kinny, D., Ljungberg, M., Rao, A.: Planned team activity. In: Artificial Social Systems. Volume 830 of Lecture Notes in Computer Science. Springer Verlag, Heidelberg (1994) 227–256
7. Mintzberg, H.: Structure in Fives: Designing Effective Organizations. Prentice-Hall (1983)
8. Scott, W.R.: Organizations: Rational, Natural and Open Systems. Prentice Hall (2003)
9. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae **79**(3–4) (2007) 415–430
10. Girault, C., Valk, R.: Petri nets for systems engineering: A guide to modelling, verification and applications. Springer Verlag (2003)
11. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring Petri net-based agent interaction protocols. In v. d. Aalst, W., Best, E., eds.: International Conference on Application and Theory of Petri Nets 2003. Volume 2679 of Lecture Notes in Computer Science., Springer-Verlag (2003) 102–120
12. Goltz, U., Reisig, W.: The non-sequential behaviour of Petri nets. Information and Control **57** (1983) 125–147
13. Best, E., Fernández, C.: Nonsequential processes: a Petri net view. Springer-Verlag (1988)
14. Rölke, H., Moldt, D., Köhler, M.: Modelling the structure and behaviour of Petri net agents. In Colom, J.M., Koutny, M., eds.: Application and Theory of Petri Nets 2001: 22nd International Conference, ICATPN 2001. Volume 2075 of Lecture Notes in Computer Science. Springer Verlag (2001) 224–241
15. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
16. Köhler, M., Wester-Ebbinghaus, M.: Petri net-based specification and deployment of organizational models. In Moldt, D., Kordon, F., van Hee, K., Colom, J.M., Bastide, R., eds.: Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'07), Siedlce, Poland, Akademia Podlaska (June 2007) 67–81
17. Wooldridge, M., Jennings, N.: The cooperative problem-solving process. Journal of Logic and Computation **9**(4) (1999) 563–592