Analysing SONAR Model Transformations

Michael Köhler-Bußmeier

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

Abstract. In this paper we study the space of organisation models that are reachable via model transformation in our SONAR-framework.
The space of organisation models is defined as a Petri net, where each reachable marking represents one SONAR-organisation model and each transitions represent a model transformation.
Keywords: multi-agent systems, organisation centred design, model transformation, Petri net, SONAR

1 Introduction

In virtual enterprises different partners, called agents, cooperate within an organisational setting. A major research focus for multi-agent systems (MAS) is the coordination of self-interested agents. Recent work puts emphasis on the organisation, that enables the teamwork of agents (cf. [1, 2] for an overview). Teamwork includes aspects like team formation, team planing (distributed problem solving), and coordinated plan execution.

The interdisciplinary field *socionics* [3], situated between sociology and computer science, emphasises the *interplay* of the macro level (i.e. the organisation) and the micro level (i.e. the agent), i.e. an organisation is not only a passive structure that provides a guiding frame for the teamwork – with the same right one could define it the other way around: The organisation is the dynamic entity that evolves in the context of the interaction of agents. In analogy to the notion of *teamwork* we like to coin this dynamics as *orgwork*.

Our in-depth discussion of the interplay of agents and organisations shows that both are active entities (cf. [4]), which influence each other at the same time. The conceptual background is a little bit different from the mainstream in organisation-centred MAS: While both agree on the fact that organisations are entities that are not static, but evolve, organisation-centred MAS motivate this aspect from the desire to allow some kind of adaption at the organisation level, while in socionics agents and organisations are instantiations of the *same* concept, which naturally implies that organisation plan, learn, adapt, etc. like agents do. Since agents and organisations are essentially the same, we are free to describe a system either from the agent-perspective, from the organisation-perspective, or from their interplay-perspective. We will concentrate on the interplay-perspective in the following, i.e. the orgwork.

In this presentation we show how the orgwork is modelled in our SONAR-framework (short for: Self-Organising Net ARchitecture). The teamwork aspects of SONAR have been studied in [5]. The orgwork aspects of SONAR are devoted to *distributed* model transformations. Note, that in SONAR teamwork and orgwork are entangled, i.e. model transformations are not independent from agent interactions – they are the other side of the coin. Each teamwork is an orgwork, as it generates a transformation. From the organisation perspective one could say that the organisation *learns* during the model transformation.

We already have a rich theoretical basis for the teamwork, which is based on Petri nets [6]: Teams are unfoldings of delegation nets, plans are unfoldings of multi-party workflow nets, etc. In this paper, we demonstrate that model transformations could also be handled within the Petri net theory. For this purpose, we introduce so called *meta-organisation nets*. The marking of a meta-organisation net describes an organisation model, the firing of a meta-transition describes a model transformation. This might seem a little bit unusual, since most approaches specify model transformations within a graph rewriting context [7]. Here, we advocate for a Petri net based approach due to the following reasons: (i) We like to have only one type of formalism in SONAR to obtain an integrated, lean formal setting and (ii) we like team- and orgwork to be executed by the same engine (here: RENEW). It turns out, that many interesting properties of transformations could be formulated as natural net properties. Therefore, we could rely on well established analysis tools to investigate the space of all transformations.

The paper has the following structure: Section 2 introduces the SONAR-framework. Section 3 presents how the team-/org-work is generated from a SONAR-model. Analogously, Section 4 defines meta-organisation nets, which are used to specify the org-work. In Section 5 we define a simple logic to define organisation policies, i.e. properties, which have to be fulfilled by an organisation model and have to be preserved by the transformations. We show how meta organisation nets can be analysed to check properties of the space of organisation transformations.

2 The SONAR Framework

In this section 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*), (iii) a model, that describes the team-based delegation of tasks (called *role/delegation nets*), (iv) a network of organisational positions, and (v) transformation rules.

2.1 Distributed Workflows, Roles and Services

In the following we ignore the colour of of workflow tokens and restrict ourselves to black tokens and skip the discussion of the data ontology. In the following we fix a set of roles $\mathcal{R}$. A *distributed workflow net* (DWFN) $D = (P, T, F, r : T \to \mathcal{R})$ is a multi-party version of the well-known workflow nets [8] where the parties

are called *roles*. Each transition of a distributed workflow net is mapped by r to a role with the meaning that a transition t can executed only by an agent that implements the role $r(t)$.

Let $R(D)$ be the set of roles used by D, i.e. $R(D) := r(T_D)$. For each set of roles $R \subseteq R(D)$ we can construct the subnet $D[R] = (P_R, T_R, F_R)$ of $D = (P_D, T_D, F_D)$ (called the role-component generated by R) by setting $T_R := r^{-1}(R)$, $P_R := ({}^\bullet T_R \cup T_R{}^\bullet)$, and $F_R := F_D \cap (P_R \cup T_R)^2$. All message places become places at the border of $D[R]$. Each partition $R_1, \ldots, R_k$ on the set of roles in D also decomposes D into its role-components: $D = D[R_1] \| \ldots \| D[R_k]$. The role-component $D[R]$ defines the *service* provided by D w.r.t. the roles R. For singletons $D[\{r\}]$ we write $D[r]$.

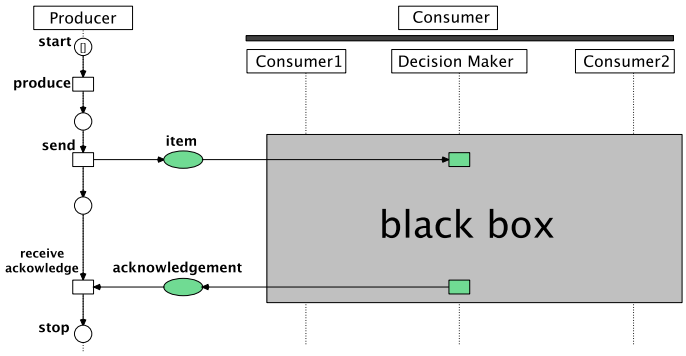

Fig. 1. Refined Distributed Workflow

$D[R] \simeq D'[R']$ denotes the fact that a component $D[R]$ cannot distinguished from another component $D'[R']$ with the same interface. This is formalised as a bisimulation with respect to the input/output behaviour at the message interface. The DWFN PC_2 in Figure 1 shows such a refinement, where the role *Consumer* of another DFWN PC (not shown here) has been refined by the interaction of the three roles $Consumer_1$, *Decision Maker*, and $Consumer_2$. The fact that the consumer part $PC[Consumer]$ is i/o-bisimlar to the part $PC_2[Consumer_1, Decision\ Maker, Consumer_2]$ is denoted:

$$PC[Consumer] \simeq PC_2[Consumer_1, Decision\ Maker, Consumer_2]$$

Let $\mathcal{D}$ be a set of DWF nets. Then, $(\mathcal{R}, \mathcal{D}, \simeq)$ is called a DWFN *repository*.

2.2 The Formal Organisation

Assume that $\mathcal{A}$ is the set of agents. On the conceptual level we define the tasks the organisation is responsible for and how they are handled. In SONAR, organisation is a net, where each place is of the form $p = \mathsf{task}^a_{D[R]}$, which describes a

task for the agent a to establish the service $D[R]$. Each transition t is either a delegation, a split, a refinement, or an execution operation (cf. Fig. 2):[1]

1. Delegate: The task to implement DWFN $D[r]$ is delegated from agent a to b. Only the delegation operation delegates the ownership of a task.
2. Split: The DWFN $D[r_1, \ldots, r_n]$ is split into the component $D[r_1], \ldots, D[r_n]$. Note, that this operation does not alter the interaction behaviour since $D[r_1, \ldots, r_n] \simeq (D[r_1] \| \cdots \| D[r_n])$.
3. Refinement: The DWFN $D[r]$ is replaced by $D'[r_1, \ldots, r_n]$, which has to be a refinement, i.e. $D[r] \simeq D'[r_1, \ldots, r_n]$ must hold.
4. Execution: The DWFN $D[r]$ is executed by the agent that is responsible for the task.

The set of all tasks and operations is defined as follows:

$$\mathcal{P} := \{\mathsf{task}^a_{D[R]} \mid D \in \mathcal{D} \wedge R \subseteq R(D), a \in \mathcal{A}\}$$

$$\mathcal{T}_{deleg} := \left\{\mathsf{d}(\{\mathsf{task}^a_{D[r]}\}, \{\mathsf{task}^b_{D[r]}\}) \;\middle|\; D \in \mathcal{D} \wedge r \in R(D) \wedge a, b \in \mathcal{A} \wedge a \neq b\right\}$$

$$\mathcal{T}_{split} := \left\{\mathsf{s}(\{\mathsf{task}^a_{D[r_1,\ldots,r_n]}\}, \{\mathsf{task}^a_{D[r_1]}, \ldots, \mathsf{task}^a_{D[r_n]}\}) \;\middle|\; D \in \mathcal{D} \wedge \{r_1, \ldots, r_n\} \subseteq R(D) \wedge n > 1 \wedge a \in \mathcal{A}\right\}$$

$$\mathcal{T}_{refine} := \left\{\mathsf{r}(\{\mathsf{task}^a_{D[r]}\}, \{\mathsf{task}^a_{D'[R]}\}) \mid D, D' \in \mathcal{D} \wedge D \neq D' \wedge D[r] \simeq D'[R] \wedge a \in \mathcal{A}\right\}$$

$$\mathcal{T}_{exec} := \left\{\mathsf{e}(\{\mathsf{task}^a_{D[r]}\}, \emptyset) \mid D, D' \in \mathcal{D} \wedge a \in \mathcal{A}\right\}$$

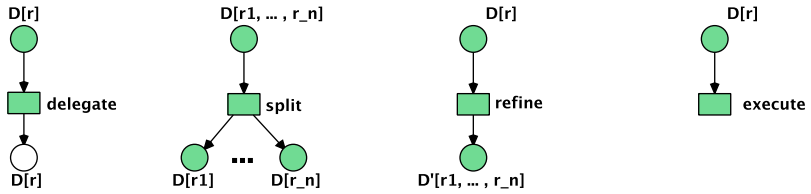

Fig. 2. Delegation, Split, Refinement, and Excution

We define $\mathcal{T} := \mathcal{T}_{deleg} \cup \mathcal{T}_{split} \cup \mathcal{T}_{refine} \cup \mathcal{T}_{exec}$. Note, that the sets $\mathcal{T}_{deleg}$, $\mathcal{T}_{split}$, $\mathcal{T}_{refine}$, and $\mathcal{T}_{exec}$ are pairwise disjoint.

Let $t = \mathsf{op}(X, Y) \in \mathcal{T}, \mathsf{op} \in \{d, s, r, e\}$, then we define the flow relation $\mathcal{F}$ by ${}^\bullet t = X$ and $t^\bullet = Y$,

The mapping $\alpha : \mathcal{P} \to \mathcal{A}$ returns the owner of a task: $\alpha(\mathsf{task}^a_{D[R]}) := a$. For each t we define its ownership equal to the owner of the place in the preset. Then all conflicts are agent-internal: $\forall p \in P : \forall t \in p^\bullet : \alpha(t) = \alpha(p)$.

[1] This is a slight modification of the definition in [6, 5], which allows that a transition t is a delegation, split, and refinement at the same time. We have chosen this simplified version for presentational purposes.

The places $\mathcal{P}$ and transitions $\mathcal{T}$ encode implicitly several aspects: the operation type, the ownership of tasks, and the agency of operations.

A SONAR-model is stratified in the sense that each node of organisation net belongs to a specific level n. Assume we have $(\mathcal{P}, \mathcal{T}, \mathcal{F})$ as given above. We assume that each DWFN $D \in \mathcal{D}$ belongs to exactly one level $n = n(D)$ and for each $p = \mathsf{task}^a_{D[R]} \in \mathcal{P}$ we set its level to $n(p) = n(D)$. Let $\mathcal{P}_n$ be the set of all place with level n. Furthermore, we assume that for each $t \in \mathcal{T}$ the level does not change, i.e. all the places p in ${}^\bullet t \cup t^\bullet$ belong to the same level. We define the level of $t \in \mathcal{T}$ as the level of the surrounding places. Let $\mathcal{T}_n$ be the set of all place with level n.

An *organisation* is a Petri net that contains some transitions of $\mathcal{T}$ and for each transition the complete pre- and postset.

Definition 1. *A* organisation *is a Petri net $N = (P, T, F)$ with $T \subseteq \mathcal{T}$, $P = {}^\bullet T \cup T^\bullet$, and $F = \mathcal{F} \cap (P \cup T)^2$.*

The places in $P^0 := {}^\circ P := \{p \in P \mid {}^\bullet p = \emptyset\}$ are those tasks that the organisation is responsible for, i.e. tasks that are generated externally.

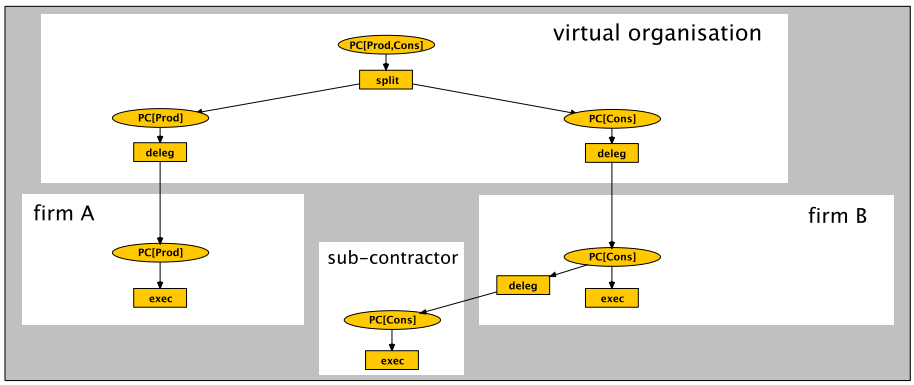

Fig. 3. An Example Organisation Net

Figure 3 shows an example organisation net, where the agents (*firm A, firm B* etc.) are indicated by the named boxes.

The stratification partitions the sets $\mathcal{P}$ and $\mathcal{T}$, i.e. $\mathcal{P} = \bigcup_{n\in\mathbb{N}} \mathcal{P}_n$ and $\mathcal{T} = \bigcup_{n\in\mathbb{N}} \mathcal{T}_n$. Similarily, each organisation $N = (P, T, F)$ is decomposed into $N := \bigcup_{n\in\mathbb{N}} N_n$, where each $N_n = (P \cap \mathcal{P}_n, T \cap \mathcal{T}_n, F \cap (\mathcal{P}_n, \cup\mathcal{T}_n)^2)$ is the organisation of level n.

The transitions in T are those operations that are explicitly allowed by the organisation N. This does not mean that the operations in $\mathcal{T} \setminus T$ are forbidden – some operations may become allowed whenever the organisation transforms. To specify which operations are permitted or forbidden we define organisation policies in Section 5.

2.3 Basic Transformations

We define two basic transformations: add_t and del_t where add_t adds the transition $t \in \mathcal{T}$ to the organisation $N = (P, T, F)$ provided that the preset of t is already part of N. Analogously, del_t removes t and its postset:

$$\mathsf{add}_t(P,T,F) := \begin{cases} (P \cup t^\bullet, T \cup \{t\}, F \cup {}^\bullet t \times \{t\} \cup \{t\} \times t^\bullet) & \text{if } t \notin T \wedge {}^\bullet t \subseteq P \\ \text{undef.} & \text{otherwise} \end{cases}$$

$$\mathsf{del}_t(P,T,F) := \begin{cases} (P \setminus t^\bullet, T \setminus \{t\}, F \setminus ({}^\bullet t \times \{t\} \cup \{t\} \times t^\bullet)) & \text{if } t \in T \\ \text{undef.} & \text{otherwise} \end{cases}$$

Then $ATF := \{\mathsf{add}_t, \mathsf{del}_t \mid t \in \mathcal{T}\}$ is the set of all atomic transformations. A transformation is the composition $\tau = \tau_1; \cdots ; \tau_n$ of atomic transformations.

The *level* of a basic transformation add_t or del_t is defined as the level of t, i.e. $n(\mathsf{add}_t) = n(t)$ and $n(\mathsf{del}_t) = n(t)$.

We now come to the org-work part: Each transition t_D of a DWFN D is labelled with a basic transformation: $\lambda(t_D) = \mathsf{add}_t$ or $\lambda(t_D) = \mathsf{del}_t$ – or with $\lambda(t_D) = \bot$ to indicate the absence of a transformation. The intended meaning is that the execution of the DWFN transition t_D also executes the basic transformation $\lambda(t_D)$. We will come back to this point in Section 3.

We require that the transformation inscription of each transition in D has a level less than the level $n(D)$ of the DWFN D itself. Therefore, each firing sequence $w = t_1 \cdots t_n$ of the DWFN generates a sequence of transformations $\lambda(w) = \lambda(t_1) \cdots \lambda(t_n)$ and due to the level restriction on the $\lambda(t_i), i = 1..n$ we obtain the property that the DWFN transforms only lower organisation levels.

3 Team-Work and Org-Work

In the following we give a short explanation of the teamwork derived from a SONAR-model: team formation, the team-DWFN, team planning via negotiation, and organisational transformations as show in Fig. 4. This is just a short summary - cf. [5] for details.

The processes described below are implemented by a specific middleware, called MULAN4SONAR, which is parametrised by a concrete SONAR-model. The generic part of the MULAN4SONAR-middleware is specified by a high-level Petri net, namely a reference net. This is beneficial for two reasons: (1) 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 [9] and we can easily integrate the prototype into MULAN [10, 11], our development and simulation system for MAS based on Java and reference nets.

Team Formation: Processes of the Organisation Net Team formation for a given task $\mathsf{task}^a_{D[R]}$, i.e. the assignment to a to implement $D[R]$, is then expressed as an execution sequence w from the initial marking $m_0 = \mathsf{task}^a_{D[R]}$

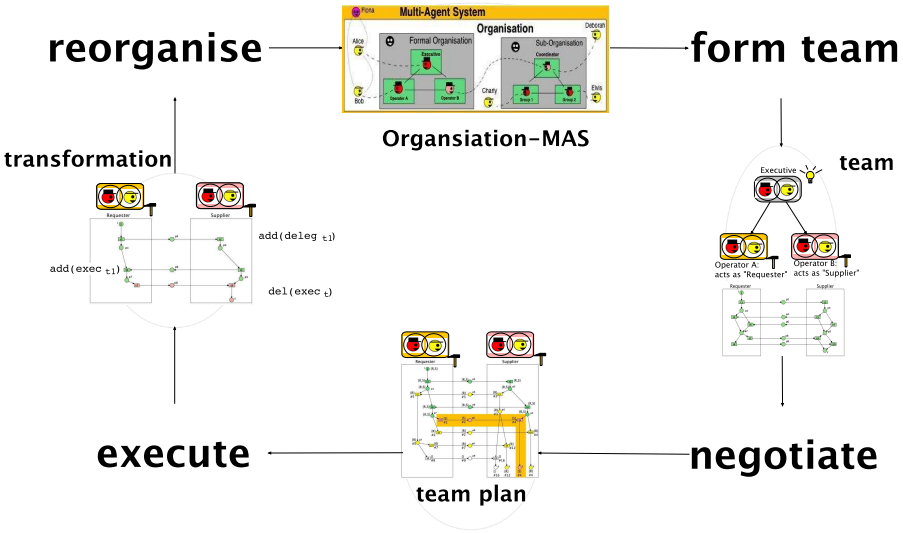

Fig. 4. Interplay of team- and org-work in SONAR

to the empty marking $m = \mathbf{0}$, i.e. $m_0 \xrightarrow{w} \mathbf{0}$ assigns an executing agents to each (sub-)tasks p.

Note, that the following refinement property holds for all task assignments: Whenever we put a token on the place $\mathsf{task}^a_{D[R]}$ then each reachable marking m describes a refinement of $D[R]$.

If Petri net processes (i.e. partially ordered runs) are used instead of sequences, then the net structure of the process can be used as the team's interaction structure (for the formal definitions of "teams-as-processes" cf. [6]). Here, only maximal processes are considered (i.e. we assume the progress property).

Assume that we have an organisation team OT, i.e. a process of the organisation net for the initial marking $m_0 = \mathsf{task}^a_{D[R]}$ together with the process morphism ϕ. For each reachable place-cut C of OT define the DWFN $D(C)$ as the composition of the role-components of the final transitions:

$$D(C) := \|_{b \in C} D(\phi(b))[R(\phi(b))]$$

Since each transition in the organisation N actually refines $D[R]$, we obtain that for each reachable place-cut C in a team OT the distributed workflow net $D(C)$ is a refinement of $D[R]$: $D[R] \simeq D(C)$.

The Team-DWFN Each team OT generates the team-DWFN $D(OT)$ that is derived from the executing transitions, which are the maximal transitions in the process net.

Each group member starts with its individual partial plan, which is an unfolding of the team-DWFN $D(OT)$ with the property that all different branches

of the unfolding lead to the final state. The set of all partial plans of a workflow N is denoted $\mathbb{PP}(N)$.

Assume that we have the team OT and its team-DWFN $D(OT)$. From OT we can deduce which agents implements which role of the team-DWFN: $\alpha_{OT} : R(D(OT)) \to \mathcal{A}$ maps each role r in the team-DWFN to the agent of the team that is assigned to r.

Team Planning: Group Plan Negotiation Each team generates a *team-plan* via negotiation. In our formal setting, Petri net unfoldings are used since a compromise can be easily characterised in terms of intersection of Petri nets – which is not that easy for a *sequential* formalisation of partial plans.

A team OT defines a tree-like structure, i.e. a team consists of sub-team, etc. Note, that the same agent can occur several times at different positions within a team. We denote this tree as nested sets, which we denote by G_{OT}.

Let G be some sub-group, i.e. a subset of G_{OT}. During the negotiation each group member $g \in G$ (which is a group again) recursively calculates its local partial plan π_g. The intersection $\bigcap_{g \in G} \pi_g$ of all these partial plans is not a partial plan in general: If all group members reach the final state of the workflow via different processes the intersection does not contain the final state at all.

For each G we define a *group-plan* π_G as a partial-plan with minimal distance to the intersection $\bigcap_{g \in G} \pi_g$ of all the local plans: Let $\mathbb{PP}(N, (\pi_g)_{g \in G}, d)$ denote the set of all partial plans such that we have to expand the intersection $\bigcap_{g \in G} \pi_g$ by at most d nodes.

The negotiation protocol is roughly the following: The hierarchical structure of the team G induces a the set of sub-groups. The negotiation protocol selects a sub-team G' and an initial distance d. Then the agents within G' compute a non-empty approximating subset of $\mathbb{PP}(N, (\pi_g)_{g \in G'}, d)$. Iteratively, these sets of group-plans are combined to obtain an approximation for a "bigger" sub-group. It is allowed to extend the distance d whenever this seems appropriate.

Finally, the group $G = G_{OT}$ is considered and we obtain the team-plan $\pi_{OT} := \pi_{G_{OT}}$ for the whole team.

Group Plan Effect: Transformations Each group plan π_{OT} induces a transformation $\lambda(\pi_{OT})$ on the organisation. Therefore, we have a dynamics of the organisation, i.e. org-work.

Each sequence $w = t_1 \cdots t_n$ of transition in the team-DWFN $D(OT)$ generates the organisation transformation $\lambda(w) : ORG \to ORG$:

$$\lambda(w) := \lambda(t_1); \ldots; \lambda(t_n) := \lambda(t_n) \circ \cdots \circ \lambda(t_1)$$

We have already seen that due to the level restrictions on each $\lambda(t)$ the transformation $\lambda(w)$ transforms only *lower* levels of the organisation.

There is another constraint for transformations that arises from the ownership within the team OT: From a given team OT we know the agent $a = \alpha_{OT}(r(t_D))$ that executes the team-DWFN transition t_D in the team plan. In

general, the executor can be different from the owner of a transformation, where the owner is the agent that owns the manipulated t: $\alpha(\mathsf{add}_t) := \alpha(t)$ and $\alpha(\mathsf{del}_t) := \alpha(t)$. But since agents are autonomous, agents cannot "manipulate" each other. If one agents likes to transform t (i.e. add or delete it), but does not own it, it has to negotiate with the owner about it. Therefore, we require that the executor $\alpha_{OT}(r(t_D))$ of a transformation $\lambda(t_D)$ is also its owner $\alpha(\lambda(t_D))$:

$$\forall t_D \in T_D : \lambda(t_D) \neq \bot \Longrightarrow \alpha_{OT}(r(t_D)) = \alpha(\lambda(t_D))$$

Definition 2. *Assume that N is an organisation and OT is a team.*

*A firing sequence $w = t_1 \cdots t_n$ of the team-*Dwfn $D(OT)$ *is a* team-transformation *if each transformation is executed by the agent that owns it:* $\forall 1 \leq i \leq n : \lambda(t_D) \neq \bot \Longrightarrow \alpha_{OT}(r(t_i)) = \alpha(\lambda(t_i))$.

A firing sequence is called applicable to the organisation N *if the transformation $\lambda(w)$ is defined for N.*

We can extend these notions to partially ordered runs of a Dwfn. Whenever we have two concurrent events e_1 and e_2 in the run, then we require that the transformations $\lambda(\phi(e_1))$ and $\lambda(\phi(e_2))$ are independent and owned by the right agent.

4 Formalisation of Org-Work by Meta-Organisations

The main problem of the negotiation process is to ensure that the transformation generated by a team plan is applicable to the current organisation. The approach taken here is to define *meta-orgaisations*. A meta-organisation net encodes in its marking the current organisation and can "decide" which transformations (i.e. add_t and del_t) are currently applicable. Therefore, a firing sequence of the team-Dwfn is applicable iff it is enabled in the meta-organisation.

We can guarantee that the transformation generated by the team plan is applicable to the organisation if we *synchronise* the team-Dwfn $D(OT)$ with the meta-organisation $\hat{N}_{P^0}$. And during the negotiation, we do not construct a partial plan for the team $D(OT)$, but for the synchronous product of $D(OT)$ and the meta-organisation $\hat{N}_{P^0}$. Then each team plan (more precisely: the subnet that is obtained by restricting the process to nodes belonging to the team-Dwfn $D(OT)$) fulfils the transformation constraints of Def. 2 by construction.

Here, we see the benefit to have an integrated model for team- and org-work: Both are Petri nets, which allows a very simple definition of the synchronisation as a net product.

In the following we define transformations as processes of another Petri net $\hat{N}_{P^0}$, called the *meta-organisation*. Assume a fixed universe $(\mathcal{P}, \mathcal{T}, \mathcal{F})$. We also assume a set of places $P^0 \subseteq \mathcal{P}$ that contains all tasks that the organisation is responsible for.

For each $n \in \mathbb{N}$ we define the *meta-organisation of level n* as $\hat{N}_n = (\hat{P}_n, \hat{T}_n, \hat{F}_n)$ to describe the possible transformation processes.

– We define the set of meta-places and the set of meta-transitions:

$$\hat{P}_n := \{\hat{p} \mid p \in \mathcal{P}_n\} \cup \{\mathsf{on}_t, \mathsf{off}_t \mid t \in \mathcal{T}_n\}$$
$$\hat{T}_n := \{\mathsf{activate}_t, \mathsf{deactivate}_t \mid t \in \mathcal{T}_n\}$$

– The meta-arcs for $\mathsf{activate}_t$ are defined by:

$${}^\bullet\mathsf{activate}_t := \{\hat{p} \mid p \in {}^\bullet t\} \cup \{\mathsf{off}_t\} \quad \text{and} \quad \mathsf{activate}_t{}^\bullet := \{\hat{p} \mid p \in {}^\bullet t \cup t^\bullet\} \cup \{\mathsf{on}_t\}$$

A token on $\hat{p}$ is used to "activate" each transformation on $t \in p^\bullet$, i.e. each transition $\mathsf{activate}_t$.
Each transition $\mathsf{deactivate}_t$ reverts the activation transition:

$${}^\bullet\mathsf{deactivate}_t := \mathsf{activate}_t{}^\bullet \quad \text{and} \quad \mathsf{deactivate}_t{}^\bullet := {}^\bullet\mathsf{activate}_t$$

– Since all the $\mathcal{P}_n$ and $\mathcal{T}_n$ are disjoint, so are the meta-organisations $\hat{N}_n$ and we can define their union, too: $\hat{N}_{P^0} := \bigcup_{n\in\mathbb{N}} \hat{N}_n := \left(\bigcup_{n\in\mathbb{N}} \hat{P}_n, \bigcup_{n\in\mathbb{N}} \hat{T}_n, \bigcup_{n\in\mathbb{N}} \hat{F}_n\right)$.
– The initial meta-marking $\hat{m}_0$ marks all meta-places off_t and the task places $\hat{p}$ that the organisation is responsible for:

$$\hat{m}_0 := \{\hat{p} \mid p \in P^0\} \cup \{\mathsf{off}_t \mid t \in \mathcal{T}\}$$

Figure 5. shows the initial fragment of the meta-organisation $\hat{N}_{P^0}$ for the set $P^0 = \{\mathsf{task}^{O_1}_{PC[Prod,Cons]}\}$.

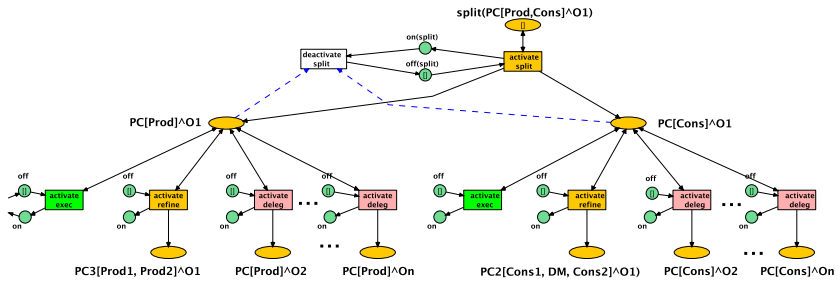

Fig. 5. Fragment of the Meta-Organisation $\hat{N}_{P^0}$ with $P^0 = \{\mathsf{task}^{O_1}_{PC[Prod,Cons]}\}$

The following proposition gives a characterisation of the reachable markings.

Proposition 1. *Assume* $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}$ *for some* $\hat{w} = \hat{t}_1 \cdots \hat{t}_n$.

1. *For all* $t \in \mathcal{T}$ *the places* on_t *and* off_t *are 1-safe, since* $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$.
2. *For each* $\hat{p} \in \hat{P}$ *we have:* $\hat{m}(\hat{p}) = 1_{P^0}(p) + |I^+(\hat{w}, \hat{p})| - |I^-(\hat{w}, \hat{p})|$

where $I^+(\hat{w}, \hat{p}) := \{i \in \{1..n\} \mid \hat{t}_i = \mathsf{activate}_t \wedge t \in {}^\bullet p\}$
$I^-(\hat{w}, \hat{p}) := \{i \in \{1..n\} \mid \hat{t}_i = \mathsf{deactivate}_t \wedge t \in {}^\bullet p\}$.

Induced Organisation The meta-places on_t are used to "link" an organisation and its meta-organisation: We obtain an organisation by selecting those transitions t that are activated by a token on on_t.

Definition 3. *Let $\hat{N}_{P^0}$ be a meta-organisation and $\hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0)$ a reachable marking. The* organisation induced by $\hat{m}$ *is $N(\hat{m}) := (P_{\hat{m}}, T_{\hat{m}}, F_{\hat{m}})$, where $T_{\hat{m}} := \{t \in \mathcal{T} \mid \hat{m}(\mathsf{on}_t) = 1\}$, $P_{\hat{m}} = P^0 \cup {}^\bullet T_{\hat{m}} \cup T_{\hat{m}}{}^\bullet$, and $F_{\hat{m}} = \mathcal{F} \cap (P_{\hat{m}} \cup T_{\hat{m}})^2$.*

In SONAR, we use meta sequences $\hat{w}$ to encode organisations, since each firing sequence $\hat{w} \in \hat{T}_n^*$ starting in $\hat{m}_0$, i.e. $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}$, generates an organisation net in a natural way: $N(\hat{w}) := N(\hat{m})$.

The following shows that construction $N(\cdot)$ is injective:

Lemma 1. *Let N be an organisation net and $\hat{N}_{P^0}$ a meta-organisation. Whenever there is a meta-marking $\hat{m}$ such that $N = N(\hat{m})$ holds, then $\hat{m}$ is uniquely defined.*

Proof. Assume that there are two reachable meta-markings, say $\hat{m}_1$ and $\hat{m}_2$ with $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}_1$ and $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}_2$, that both generate N, i.e. we have: $N = N(\hat{m}_1) = N(\hat{m}_1)$.

We know the invariance $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$ for all reachable markings $\hat{m}$. Assume that $\hat{m}_1$ and $\hat{m}_2$ differ on some $\mathsf{on}_t/\mathsf{off}_t$ pair. Then we have $\hat{m}_1(\mathsf{on}_t) = 1$ and $\hat{m}_2(\mathsf{on}_t) = 0$ (or vice versa) and $t \in T_{N(\hat{m}_1)}$, but $t \notin T_{N(\hat{m}_2)}$ and therefore $N(\hat{m}_1) \neq N(\hat{m}_2)$.

Since $\hat{m}_1$ and $\hat{m}_2$ agree on each $\mathsf{on}_t/\mathsf{off}_t$ pair, we know that for each t we have the same balance in $\hat{w}_1$ and $\hat{w}_2$:

$$|\hat{w}_1|_{\mathsf{activate}_t} - |\hat{w}_1|_{\mathsf{deactivate}_t} = |\hat{w}_2|_{\mathsf{activate}_t} - |\hat{w}_2|_{\mathsf{deactivate}_t}$$

Since this implies $|I^+(\hat{w}_1)| - |I^-(\hat{w}_1)| = |I^+(\hat{w}_2)| - |I^-(\hat{w}_2)|$, we know that $1_{P^0}(p) + |I^+(\hat{w}_1, \hat{p})| - |I^-(\hat{w}_1, \hat{p})| = 1_{P^0}(p) + |I^+(\hat{w}_2, \hat{p})| - |I^-(\hat{w}_2, \hat{p})|$ and by Prop. 1 we have $\hat{m}_1(\hat{p}) = \hat{m}_2(\hat{p})$ for each $\hat{p}$. □

The Meta-Organisation as a High-Level Net In MULAN4SONAR we model the meta organisation as a a high-level Petri net. Figure 6 shows the RENEW-based model for the meta organisation net. The place meta places contains all tokens of the form $\hat{p}$. The place actions contains all tokens of the form on_t. Due to the invariant $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$ the marking of off_t can represented implicitly by looking for absent tokens of the form on_t. The transition add action is enabled whenever there is one token $\hat{p}$ on meta places such that ${}^\bullet t = \{p\}$ (which is specified by the inscription act = [type, task, tasks]) and *no* token of the form on_t on actions. Note, that the arc from actions to add action is an inhibitor arc. Whenever add action fires, it generates the action, i.e. puts the token on_t on the place actions and foreach $p \in t^\bullet$ one token $\hat{p}$ on meta places. Note, that the arc from add action to meta places is a so called flexible arc, which generates a multiset of flexible cardinality.

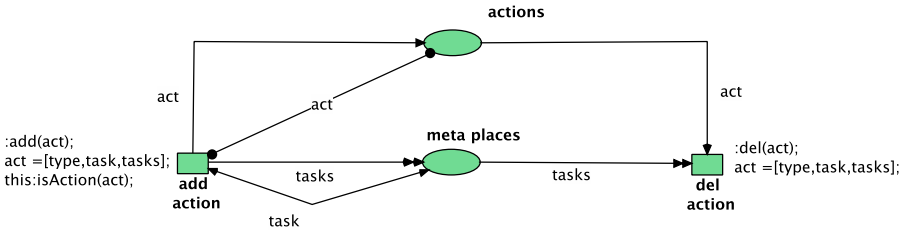

Fig. 6. The High-Level Net Variant for the Meta Organisation Net (Fragment)

Induced Transformations Each meta-transition induces a transformation. In the following we show that for $\hat{m}_1 \xrightarrow{\hat{w}} \hat{m}_2$ the organisation generated from a meta-marking, i.e. $N(\hat{m}_2)$, coincides with the organisation obtained from the induced transformation, i.e. $\tau(w)(N(\hat{m}_1))$.

The *transformation induced by the meta-transition* $\hat{t}$ is $\tau_{\hat{t}}$, which defined as:

$$\tau_{\hat{t}}(N) = \begin{cases} N(\hat{m}'), & \text{if } \exists \hat{m}, \hat{m}' : \hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) \wedge N = N(\hat{m}) \wedge \hat{m} \xrightarrow{\hat{t}} \hat{m}' \\ \text{undef.}, & \text{otherwise} \end{cases}$$

Note, that Lemma 1 guarantees that $\tau_{\hat{t}}$ is well defined.

We extend the induced transformations to sequences of meta-transitions: Define $\tau_{(\hat{t}_1 \cdots \hat{t}_n)} := \tau_{\hat{t}_1}; \ldots; \tau_{\hat{t}_n}$ and $\tau_\epsilon = id$.

We can formalise the correspondence of the induced transformations and atomic transformations by defining the isomorphism $h : (\hat{T} \cup \{\bot\})^* \to ATF^*$ by

$$h(\mathsf{activate}_t) = \mathsf{add}_t, \quad h(\mathsf{deactivate}_t) = \mathsf{del}_t, \quad \text{and} \quad h(\bot) = id.$$

Each meta-sequence $\hat{w}$ induces a corresponding transformation:

Lemma 2. *Let $\hat{m}$ be a reachable marking. Each meta-sequence $\hat{w} = \hat{t}_1 \cdots \hat{t}_n$ induces a transformation $\tau_{\hat{w}}$ which coincides with $h(\hat{w})$ on $N(\hat{m})$:*

$$\hat{m} \xrightarrow{\hat{w}} \hat{m}' \quad \Longrightarrow \quad \tau_{\hat{w}}\ (N(\hat{m})) = h(\hat{w})\ (N(\hat{m}))$$

Proof. The general proposition follows by induction over the length over $\hat{w}$. The case $n = 0$ is clear, since $\tau_{\hat{\epsilon}} = id = h(\epsilon)$. The induction step follows form the definitions of the basic transformations add_t and and del_t: Whenever $\hat{t} = \mathsf{activate}_t$, then $\tau_{\hat{t}}$ is equivalent to add_t and whenever $\hat{t} = \mathsf{deactivate}_t$ then $\tau_{\hat{t}}$ is equivalent to del_t. □

Conversely, each transformation τ induces a corresponding meta-sequence:

Lemma 3. *Let the transformation $\tau = \tau_1; \ldots; \tau_n$ be defined for the organisation N and let $N = N(\hat{m})$ for some $\hat{m}$.*

Then τ induces the meta-sequence $h^{-1}(\tau)$ and $h^{-1}(\tau)$ is enabled in $\hat{m}$, i.e. $\hat{m} \xrightarrow{h^{-1}(\tau)} \hat{m}'$, and the generated organisations are equal: $\tau(N) = N(\hat{m}')$

Proof. Similar to the proof above. □

The relationship of transformations, organisations, and meta-organisation is illustrated by the following diagram:

$$\begin{array}{ccc} \hat{m} & \xrightarrow{\hat{w}} & \hat{m}' \\ \downarrow{\scriptstyle N(\cdot)} & & \downarrow{\scriptstyle N(\cdot)} \\ N = N(\hat{m}) & \xrightarrow{\tau_{\hat{w}}} & N' = N(\hat{m}') \end{array}$$

The product $D(OT) \otimes_{\alpha_{OT}} \hat{N}_{P^0}$ fuses each team-DWFN transition t_D with $\lambda(t_D) = \mathsf{add}_t$ with a copy of the meta-transition $\mathsf{activate}_t$ and each t_D with $\lambda(t_D) = \mathsf{del}_t$ with a copy of $\mathsf{deactivate}_t$.

5 Analysis of Organisation Transformations

In SONAR, policies are used to describe those properties that have to remain invariant during reorganisation processes of the model.

The set of atomic propositions is $AP = \{\mathbf{P}[t], \mathbf{O}[t], \mathbf{F}[t] \mid t \in \mathcal{T}\}$, where $\mathbf{P}[t]$ ($\mathbf{O}[t]$, $\mathbf{F}[t]$) means that it is *p*ermitted (*o*bligated, *f*orbidden) to perform the basic operation t.

A *policy* Φ is a propositional logic formula with AP as the set of atomic propositions. The set of all t occurring within the formula is denoted $\mathcal{T}_\Phi$.

Usually, it is not possible to permit and forbid a at the same time. Analogously, if there is an obligation to do t then t is usually permitted and not forbidden. Usually these constraints are encoded inside modal logic and the deontic qualifiers are modelled as modalities. For simplicity reasons, we use propositional logics and add a constraint for the truth assignment function instead.

Definition 4. *An* assignment *of a policy Φ is a mapping $v : AP \to \{0, 1\}$ with the property: $v(\boldsymbol{F}[t]) = 1 \implies v(\boldsymbol{P}[t]) = 0$ and $v(\boldsymbol{O}[t]) = 1 \implies v(\boldsymbol{P}[t]) = 1$ for all $t \in \mathcal{T}$. The set of all assignments is $ASSIGN$.*

We are interested in the fact, whether a organisation is a model for a policy.

Definition 5. *An organisation N is a* model *for a policy Φ (denoted $N \models \Phi$) whenever we have:*

$$\forall v \in ASSIGN : v(\Phi) = 1 \implies \big(\forall t \in \mathcal{T}_\Phi : \begin{array}{rcl} v(\boldsymbol{O}[t]) = 1 & \implies & t \in T \\ \wedge\, v(\boldsymbol{P}[t]) = 0 & \implies & t \notin T \\ \wedge\, v(\boldsymbol{F}[t]) = 1 & \implies & t \notin T \big) \end{array}$$

When we use a marked meta organisation $(\hat{N}_{P^0}, \hat{m})$ instead of N, then $t \in T$ in the definition above has to be replaced by $\hat{m}(\mathsf{on}_t) = 1$.

Analysis Each organisation transformation has to preserve the organisational policy, i.e. whenever $N \models \Phi$ holds and τ is a transformation, then $\tau(N) \models \Phi$ holds, too. Define the set of meta-markings that satisfy a policy Φ as:

$$SAT(\Phi) := \{\hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) \mid N(\hat{m}) \models \Phi\}$$

Definition 6. *The meta-organisation* $\hat{N}$ enforces *the policy* Φ *if* $SAT(\Phi) = RS(\hat{N}_{P^0}, \hat{m}_0)$.

But in almost all cases this property does not hold, and this is not always problematic, since it is not necessary that each $N(\hat{m})$ satisfy Φ after each step of the team plan. It is only necessary that each $N(\hat{m})$ satisfy Φ at the *end* of the execution of the team plan.

We could formulate this property as follows: Assume that we have an initial model $N = N(\hat{m}_0)$ that satisfies the policy Φ and the meta-marking $\hat{m}$ is reachable, i.e. $N_1 = N(\hat{m})$ might be the result of a transformation. Whenever N_1 does not satisfy the policy, the question arises whether it is possible to reach a meta-marking $\hat{m}'$ such that $N_1 = N(\hat{m}')$ satisfies Φ:

$$\forall \hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) : \exists \hat{m}' \in RS(\hat{N}_{P^0}, \hat{m}) : \hat{m}' \in SAT(\Phi)$$

Whenever this is not the case, then we known that there exist some transformation which should be suppressed in all team-plans, because we can never repair the situation. If the property holds, we know that each transformation can be extended to one that satisfies the policy again.

When understood as a question for the meta-organisation the answer is trivially "yes", since each transformation in $\hat{N}$ can be undone ad $\hat{N}$ is revertible. But of course it is undesired to reach a policy satisfying marking again, by undoing all transformations. Therefore we exclude some (or all) $\mathsf{deactivate}_t$ from the analysis: For $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ and $\hat{A} \subseteq \mathcal{P} \cup \mathcal{T}$ we define the subnet $(\hat{N} - \hat{A}) := (\hat{P} \cap \hat{B}, \hat{T} \cap \hat{B}, \hat{F} \cap \hat{B}^2)$, where $\hat{B} = (\mathcal{P} \cup \mathcal{T}) \setminus \hat{A}$.

Definition 7. *The meta-organisation* $\hat{N}$ *is* stable *w.r.t. the policy* Φ *if we have for* $\hat{A} = \{\mathsf{deactivate}_t \mid t \in \mathcal{T}\}$*:*

$$\forall \hat{m} \in RS(\hat{N} - \hat{A}, \hat{m}_0) : \exists \hat{m}' \in RS(\hat{N} - \hat{A}, \hat{m}) : \hat{m}' \in SAT(\Phi)$$

Proposition 2. *Assume, that* $\mathcal{A}$ *and* $\mathcal{D}$ *are finite sets.*

Given a meta-organisation $\hat{N}$*, it can be checked using standard model checking techniques, whether the policy is enforced and whether the policy is stable.*

Proof. Note, that whenever $\mathcal{A}$ and $\mathcal{D}$ are finite sets, then $\hat{N}$ is finite, too, and by Prop. 1 its state space is finite. Enforcement is a safety property and this can be checked doing an exhaustive state space exploration.

Stability is a kind of liveness property and can be checked by computing the strongly connected components (SCC) of the state space and checking whether each terminal SCC contains an $\hat{m}$ that satisfies Φ. □

6 Conclusion

In this paper we studied the organisation-oriented perspective of multi-agent systems and their transformations, which we named org-work as opposed to team-work.

We defined the dynamics of organisation models in the formalism of meta nets. This explicit representation of the history of the model transformation (understood by the meta-marking $\hat{m}$) allows us a deeper insight in the possible transformation that can arise as the byproduct of the teamwork.

Since we use the same formalism for team- and org-work, we can mix both models and obtain an integrated view on the system. The negotiation protocol directly benefits from this: During the negotiation, we do construct a partial plan not only for the team $D(OT)$, but for the synchronous product of $D(OT)$ and the meta-organisation $\hat{N}_{P^0}$. This guarantees that the transformation generated by a team plan π is *always* applicable to the actual organisation, which is a non-trivial property to ensure by negotiation.

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. Malsch, T.: Naming the unnamable: Socionics or the sociological turn of/to distributed artificial intelligence. Autonomous agents and multi-agent systems **4** (2001) 155–186
4. Köhler, M., Moldt, D., Rölke, H., Valk, R.: Linking micro and macro description of scalable social systems using reference nets. In Fischer, K., Florian, M., Malsch, T., eds.: Socionics: Sociability of Complex Social Systems. Volume 3413 of LNAI, (2005) 51–67
5. 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. **5460** (2009) 98–114
6. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae **79** (2007) 415 – 430
7. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer-Verlag (2006)
8. Aalst, W.v.d.: Verification of workflow nets. In Azeme, P., Balbo, G., eds.: Application and theory of Petri nets. Volume 1248 of LNCS (1997) 407–426
9. Kummer, O. et al.: An extensible editor and simulation engine for Petri nets: Renew. In Cortadella, J., Reisig, W., eds.: ATPN 2004. Volume 3099 of LNCS (2004) 484 – 493
10. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In Colom, J.M., Koutny, M., eds.: ATPN 2001. Volume 2075 of LNCS (2001) 224–241
11. 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.: MATES'08. Volume 5244 of LNAI (2008) 37–48