Modeling and Planning Collaboration using Organizational Constraints

Michael Igler*, Paulo Moura†, Matthias Faerber*, Michael Zeising*, Stefan Jablonski*

*Chair for Applied Computer Science IV, University of Bayreuth, Germany
{michael.igler, matthias.faerber, michael.zeising, stefan.jablonski}@uni-bayreuth.de

†Dep. of Computer Science, University of Beira Interior, Portugal
Center for Research in Advanced Computing Systems, INESC Porto, Portugal
pmoura@di.ubi.pt

***Abstract*—Process management systems play an important role for today's information systems. They coordinate the work items among employees and ensure the correct execution of processes. In this paper we focus on the organizational perspective of process management systems. This perspective is responsible for assigning people together with their roles within business organizations to process execution. A key issue in integrating the organizational perspective into processes is the strategy for selecting people to execute work steps. This assignment is the basis for collaboration among the people of an organization within a process-based application. We implemented our approach in ESProNa, a Logtalk application running in SWI-Prolog extended with the Thea library providing direct and complete support for OWL2 ontologies. The use of these languages allows the definition of comprehensive organizational constraints. We will cover both, the definition of these constraints in the process model, and their interpretation by the process execution engine. Further we will show how the organizational perspective impacts the order of process execution, i. e. the way of collaboration between the assigned people decisively.**

I. Introduction

It is well accepted that processes are an important means for planning activities in companies [1] [2] [3]. Workflow or process management systems[1] support the definition and execution of processes. They guide employees in their daily work [4] [5] [6] by providing them with tasks they are supposed to perform. Especially the coordination of task execution is an important feature of process management systems as it ensures that all planned steps are executed in the correct order. In this paper we will focus on the question, what aspects of a process affect the execution order of work steps. In particular we will investigate whether and to what extent the organizational perspective of processes has an impact on process execution, i. e. determines how people within a process-based application are collaborating. This so called synchronous form of collaboration is investigated in this paper. We do not consider synchronous collaborative work (e. g. video conferencing, collaborative editors) in this paper.

According to *Perspective Oriented Process Modeling* (POPM) processes can be described by at least five independent perspectives: the functional perspective (work steps), the control flow perspective (control flow), the data-oriented perspective (data flow), the organizational perspective (people) and the operational perspective (tools) [7] [8]. At least three of these perspectives determine the execution order of steps, i. e. the way how assigned people are collaborating. It is obvious that the control flow aspect (as its name indicates) has a strong influence on the order of process execution. Also the data (flow) perspective impacts the execution order: a certain process step cannot be started before mandatory input data is available. Besides this, the organizational perspective also impacts the execution order of process steps decisively. When a process step is ready for execution, i. e. the control flow determines its execution and all necessary input data are available, the organizational perspective must be evaluated: only if one of the persons eligible to execute this process step is available, the process step will be performed. Otherwise the process will not be executed. In this paper we will focus on this latter issue and discuss how this kind of collaboration should be enacted in process management systems.

Many of today's process management systems – especially those which focus on the orchestration of web services – consider only four of the five perspectives mentioned above and almost ignore the organizational component. The BPEL4WS [9] core language for example does not include syntactic elements for specifying human interaction; only the extension BPEL4People considers this aspect. We will also argue in this paper that the role concept on its own is not sufficient to fill out the organizational perspective. However much research work does only focus on the role concept and regards it as sufficient to model the organizational perspective.

Second, we think that the organizational perspective is even more dynamic and unforeseeable than the other two

[1]In this paper we do not distinguish between process and workflow management system. We summarize both under the term "process management system". We assume that process management systems support computer based execution of processes.

COLLABORATECOM 2010, October 9-12, Chicago, USA

DOI 10.4108/icst.collaboratecom.2010.47

perspectives (control flow and data flow). That means, it is much more difficult to predict whether the organizational perspective hinders a process step from execution than it is for one of the other two perspectives. Why do we get to such a perception? The behavior of the control flow perspective can easily be described: when the control flow determines a process A to be executed next, nothing (within the scope of this perspective) hinders this process step to be executed any more. The data perspective works similarly: when previous process steps produced data that form the input for an upcoming process step A then process step A can be executed (from the perspective of data flow). The situation changes when the organizational perspective is considered. Let's assume that process step A is considered to be the next step according to the control flow and the data perspective, i. e. these two perspectives do not prevent process step A from being executed anymore. So, the process management system has determined process step A to be executed next. Process step A is associated with a valid specification of the organizational perspective. We assume that process step A must be executed by a "sales manager". Now, the following "turbulences" might occur:

- Most of the sales managers (or all of them) assigned to this process step are temporarily not available (e.g. due to illness, vacation).
- Most of the sales managers (or all of them) assigned to this process step are permanently not available since they got new obligations (i. e. they are not sales manager any more).
- Still existing sales managers are assigned to other process steps.

None of the three situations above can be called erroneous. Sales managers are always principally available. Nevertheless, the process is hindered from being executed since eligible process executors momentarily are not available. What could be done to resolve such an undesired situation?

- Re-assignment: Another person can be assigned to process step A. This solution is just a provisional solution since it does not solve the problem principally. Nevertheless, it is very powerful since it can always be applied. We acknowledge this solution, but do not pursue it further in this paper, since it is conceptually seen as a modest issue.
- Planning: The problematic situation is predicted and therefore avoided (or at least it is made avoidable). This solution requires a kind of a planning component which calculates process steps that potentially have to be performed. Such a component also computes whether eligible persons for process execution are available. In case it encounters problems, it advices not to continue this line of execution. Alternative execution paths should be pursued or the re-assignment solution should be proposed.
- Re-ordering: Skip this process step and continue with a step that only depends on this one from an organizational view. An example (see Figure 1) clarifies this solution:

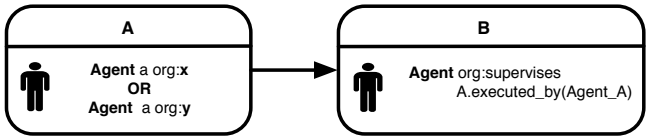

Fig. 1. Exemplary process model for re-ordering

Process step B must be executed by the supervisor of the executor of process step A. There are no more dependencies defined between the two steps. If person x is executing A, then X as supervisor of x has to execute B; if person y has executed A, then Y as supervisor of y has to execute B; etc.
We assume that the organizational perspective is the only remaining unresolved dependency for the process steps A and B. That means the organizational perspective of process step A must be resolved in order to determine the executor of process step B. Under the assumption that the organizational constraint between both process steps could be inverted, we start with the execution of process step B. Afterwards process step A could be executed. In our example this means an inversion of the "is supervisor of" relationship which is the "is supervised by" relationship. Here, the assumption holds that this temporal delay of the execution of process step A and the preferred execution of process step B leads to the situation that executors of process step A become available eventually. So, the overall process can be executed without big delay.
Re-ordering exploits the relationship between (two) process steps. Instead of evaluating such a relationship in the normal forward direction, re-ordering evaluates such a relationship reversely. This is done by applying the inverse relationship to a given relationship.
A prerequisite for the application of the re-ordering technique is that process management systems enable domain experts to define arbitrary relationships between elements of their organizational structure. Such relationships cannot be modeled by roles alone [10]. For the re-ordering technique all the relationships that are reversible are of major interest.

Now we can summarize the goals of this paper. We aim at the provision of a powerful concept to model and enact the organizational perspective of a process management system. Especially, we want to provide a method that supports both

- planning and
- re-ordering.

Both concepts are applied in order to support efficient process execution. A prerequisite of our approach is that process management systems base their organizational perspective on a powerful concept that enables the definition of arbitrary, in many cases reversible relationships between elements of the organizational structure.

Our approach, called ESProNa, provides maximum flexibility with respect to the organizational perspective. Flexibility

here means that the organizational perspective will be exhausted to a maximum in order to support efficient processing without delay. We propose an approach based on a formal logic (first order logic) that allows us to specify powerful organizational constraints. The approach fosters the prediction of undesired situations and allows determining "alternative" process executions. Our approach both comprises a modeling and an execution component.

The paper is organized as follows. The next section discusses related work. Section III introduces our process modeling concept. Section IV presents an overview of our system architecture. Section V clarifies the formal definitions for our research. Section VI shows how the previously mentioned objectives are achieved.

II. Related Work

During the last years several articles were published on the organizational perspective in process management systems. [11] proposes both a life-cycle and a meta-model for organizational issues in process management systems. The workflow life-cycle describes the different phases of a process:

- Goal Specification and Environment analysis: Organizational Analysis
- Process Design: Resource and Assignment Specification
- Process Implementation: Resource Integration
- Process Enactment: Resource Utilization and Maintenance
- Process Monitoring: Resource Monitoring
- Process Evaluation: Resource Evaluation
- Animation, Simulation: Capacity Planning

The concept described in that paper concerns mostly the phases "process design" with the specification and the assignment of resources to a process and "process enactment". The paper also proposes a meta-model for specifying the organizational structure in process management applications. The authors introduce the concept of a "role" which forms the link between a process step and an organizational unit (i. e. the employee). Roles can be divided into two subclasses: "privileges" and "capabilities". A privilege is a property of a resource that is connected with the resource's position in the company; in contrast a capability is a direct property of a resource, independent of a company. We will use a similar meta-model and both interpretations of the "role" to assign people to processes. In [10] a meta-model for specifying the organizational structure is developed. This model is more general than the one described in [11] and allows the specification of domain-specific models. The domain-specific models can be obtained by instantiating an abstract organizational model. In order to query this organization model, a query language is described. The assignment of people to tasks is accomplished by "assignment rules" and "synchronization rules". However, planning and re-assignment are not dealt with by this approach.

BPEL4People [12] and WS-HumanTask [13] are extensions to the BPEL [14] process execution language and provide support for including human interaction into a BPEL process. The language adds so called "people activities" to processes. Once a people activity is reached in a process, a task is entered into the user's work list. Users can select tasks that they want to execute from this list. The identification of the user who has to execute a certain step is made by querying an organization repository. BPEL4People describes some interaction patterns (Four-eye principle, escalation etc.). The aim of the approach is to provide seamless integration between automated and human tasks. However, as BPEL4People uses the same execution order as BPEL, a process is stopped if a certain employee is not available. Again, planning and re-assignment are not dealt with by this approach.

As a summary it can be stated that both [10] and [11] provide a meta-model for the definition of an organizational structure in process management systems. While the one presented in [10] is more generic and can be better adapted to specific domains, both models are sufficient for storing an organizational model. However all approaches do not enforce the organizational perspective during process execution. While this might not be a big issue for the approach [11] [10], it must be an important issue for BPEL.

[15] presents a formal language for the specification and enforcement of authorization constraints in workflow management systems. This formal framework mainly deals with the formalization of logical rules to represent authorization patterns. Furthermore, algorithms are provided to check the consistency of constraints and assign users to tasks. The planning component of [15] is able to evaluate a modified and partially violated workflow. User assignments are generated in order to make a process step valid. Our approach uses logic in a completely different manner. In particular, we are not viewing planning as theorem proving. In our approach we use logic solely to express search control knowledge. We utilize traditional planning representations for actions and states and we generate plans by search. As [15] does not plan on process states but on workflows it differs from our work: our goal is to offer the process executor a kind of navigation system that supports him during the execution phase (Section VI) which can be achieved by planning on states.

[16] shows an approach based on rules and patterns making workflows capable of adapting themselves effectively when exceptional situations occur during process execution. In order to separate this research from our work it is important to clarify the difference between unpredictable exceptions and (possibly) predictable exceptions which can be included during the design phase of a workflow. The first ones are not predictable – neither for the process modeler nor the process executor. The latter ones may (eventually) be predicted by the process modeler and must be included during the definition of the process model. Otherwise the process executor has no chance of reacting on a "deviation" during the execution phase. In our approach we mainly concentrate on the latter approach of modeling flexibility and controlled deviations.

DECLARE [17] is a constraint-based system, developed at the University of Eindhoven, that is focused on modeling constraints between processes. DECLARE uses the ConDec

[18] modeling language. Modeled constraints in ConDec are translated to a Linear Temporal Logic (LTL) formula. An automaton is generated for every specific constraint in order to verify it. Furthermore, a second automaton is generated over all constraints. The support for the organizational perspective in DECLARE is, however, limited as hierarchical organizational structures cannot be modeled. A planning component that can be consulted for advice during execution phase is also absent. This is because of the fact that LTL does not allow to express the effect in a state space that results from a state transition. Therefore it is not evident how to express a goal state in LTL nor to construct automata for planning an execution scenario in order to reach a certain goal state [19].

[20] presents a survey for specifying workflows based on three different approaches: Temporal Logic, Event Algebra and Concurrent Transaction Logic (CTR). The main difference between these research papers and our framework is that we support the formalization of organizational facts, such as roles, individuals, supervising behavior, etc., through OWL2 ontologies [21] which can be loaded into our system ESProNa. Through an OWL2 parser [22] we can access these information very efficiently. As already mentioned in the comparison with [17] a Temporal Logic (LTL) prevents the possibility for planning on process models.

EM-BrA^{2}CE (Enterprise Modeling using Business Rules, Agents, Activities, Concepts and Events) is a framework for unifying vocabulary and execution models for declarative process modeling [23]. The vocabulary is described in terms of the Semantics for Business Vocabulary and Rules (SBVR) standard and the execution model is presented as a Colored Petri Net (CP-Net). EM-BrA^{2}CE also follows the same concept we use in this paper to specify a state space transition relation based on rules. Every process must be described in the form of the mentioned Business Vocabulary. In our opinion this concept slows down re-reading of process models by different users or the process modeler itself after some period of time. EM-BrA^{2}CE supports the organizational perspective through "Activity Authorization Constraints" which are assigned to a process. We could not derive from [23] how the underlying organizational informations (see Section IV-A) are stored.

III. Process Modeling

This section illustrates how processes are modeled in ESProNa. In this paper we concentrate on the functional, behavioral, and organizational perspective of the POPM approach (Sections III-A to III-C). In particular, we show how dependencies between processes, which impact their execution order, are modeled.

A. Functional Perspective

A process is represented as a rectangle (Figure 2). In the upper part the name of the process is written. A `PID` (process identification) identifies a process uniquely. The lower part of a process rectangle contains constraints which refer to the perspectives of a POPM process model. The clipboard icon refers to the functional perspective. It determines how often a process can be executed. Two different notations are offered:

- The `min..max` notation expresses the range how often a process must/can be executed; it must be executed at least `min` times and can be executed at most `max` times.
- The `#=` notation specifies the exact number of executions of a process.

Both the `min..max` and the `#=` notation normally refer to constant numbers (positive integers). However, it is also possible to refer to executions of other (related) processes. This means that the number of executions of a process is related to the number of executions of another (related) process. Table I provides an overview on the spectrum of specifications:

Notation	Expression	Meaning
min	INT	Execute minimum as often as Integer value encodes
	0	Optional (does not have to be performed)
	> 0	Mandatory
	#(PID)	At least as often as process PID was / will be executed
max	INT	Execute maximum as often as Integer value encodes
	$*$	Arbitrary times
	> 0	Mandatory
	#(PID)	At most as often as process PID was / will be executed
#=	INT	Execute exactly as often as Integer value encodes
	#(PID)	Exactly as often as process PID was / will be executed
	#(arith(PID,INT))	Arithmetic operation(+, -, /, *) applied on PID with value encoded in INT (prefix notation)

TABLE I
DIFFERENT NOTATIONS OF QUANTIFICATION

B. Behavioral Perspective

The behavioral perspective is identified by a graph icon (similar to the USB icon) in the lower part of a process rectangle. In Figure 3 the process "Acknowledge Surgery Plan" includes a behavioral constraint stating that process `PID 1` must be marked as `done` in order to start the execution of process `PID 2`. `done` forms an execution state of a process; execution states are explained in Section V. Nevertheless, we already can say that `done` refers to the completion of a process execution.

Alternatively, the behavioral perspective can be expressed through a solid arrow connecting two processes. It can be modeled in this way as long as the execution of a subsequent process depends on the completion of a former one. Figure 2 depicts this equivalent notation of a behavioral constraint. Since users are accustomed to this notation, we allow to apply it. Of course, the notation using the graph icon is more powerful since it can reference arbitrary states of a process (beyond `done`).

C. Organizational Perspective

The organizational perspective is represented by a person shape in the lower part of a process rectangle. It has to be defined, which persons are eligible to start, to finish, etc. a process step. Although we shortly discuss process states in Section V we do not detail this aspect in this paper. Here, we always assume that persons that are selected are eligible to perform all operations of a process step.

The key word `Agent` is used to define a set of agents who are eligible to execute a certain process. This key word is followed by a set definition. Sets can be combined by the usual set operators (union, difference, intersection). To determine these sets an organizational structure (including population) must be defined. Figure 4 depicts an example organization. Organizations are described using an ontology; they consist of two basic elements: sets and relationships. In the figure, three sets are defined: Role, Person, and Department. Also not directly shown in the figure but indicated through the dotted rectangles, instances are assigned to these sets. Instances of the same or different sets are related through relationships. All notions are borrowed from certain namespaces (e.g. `org:, clinic:`). The ontology-based definition of organizational structures is most flexible and enables the specification of arbitrary organizations.

Two issues must be mentioned here. First, Figure 4 shows clearly that "Role" is just one out of many concepts needed to define an organizational structure. Second, relationships are optional for the definition of organizations, too. Thus, approaches that are merely based upon the "role concept" are not sufficient to represent arbitrary organizational structures. In Figure 2 the process "Record Radiogram" must be executed by either an MTA (Medical Technical Assistant) or by an "Assistant Doctor". Process step "Analyze Radiogram" must be performed by the "Assistant Medical Director" or by the "Medical Superintendent". We want to mention here that no relationship between the two process steps concerning the executors is given.

While the organizational constraints in the example of Figure 2 do not imply any dependencies between the two process steps, the organizational constraints in Figure 3 very well impose a dependency. The specification of `PID 2` in process step "Acknowledge Surgery Plan" references the executor of `PID 1`; thus, an inter-process dependency is defined which significantly determines the execution order of process steps. In principle, the executor of `PID 2` can only be determined when the executor of `PID 1` has been evaluated `executed_by(...)`. We say that the starting point (open variables) for the determination of a set of eligible agents must be provided. In this case, the executor of `PID 1` is that starting point. Then, eligible agents are found by traversing the relationship `supervises`. It is worth to mention here that such type of dependencies between process steps, caused by the organizational perspective, are starting points for our concept "re-ordering" which is discussed in Section VI-B.

At the end of this sub-section we shortly want to assess

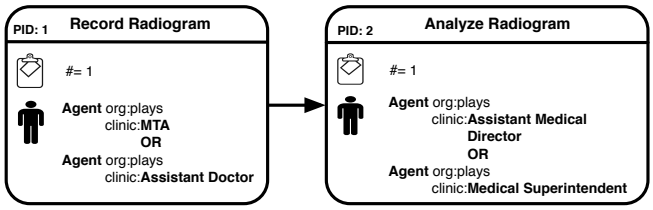

Fig. 2. Clinical example model

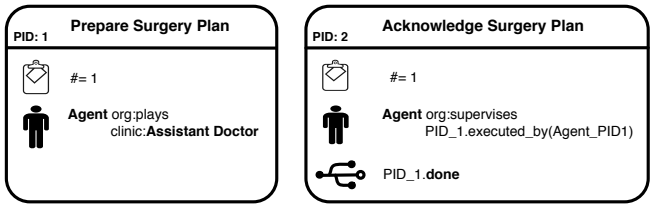

Fig. 3. Clinical example model with organizational dependencies

the capability of our implementation of the organizational perspective. On this account, we chose to design our first use case scenario (Figure 2) similar to Workflow Pattern No. 4 taken from the Workflow Patterns Initiative [24]. The pattern is called "Authorization" and consists of two process steps that have to be executed by two, totally independent sets of agents with specific properties. The evaluation of this pattern [25] shows that the majority of current process management systems cannot facilitate this pattern; only the languages FLOWer [26] and COSA [27] provide support. The same evaluation shows that commonly used languages and industry standards like BPEL or BPMN [28] do not support this pattern at all. The ESProNa approach is able to represent this pattern completely and in a generic way.

The process model from Figure 3 is similar to the Workflow Pattern No. 10 (Organizational Distribution). Within this model a relationship between the persons executing the two process steps is established: the executor of step `PID 2` must be the supervisor of the person who executed process step `PID 1`. The evaluation of this pattern [25] shows again that only two systems provide support for this pattern: COSA [27] and WebSphere MQ Workflow [29]. Again, our approach does support this pattern in a very general way, and therefore its capabilities are proven.

D. Executing Process Models

At the end of this section we want to briefly demonstrate how processes are interpreted for execution. In principle, a process can be executed when its constraints (lower part of the process rectangle) all evaluate to true. That means in detail:

- The functional perspective allows for (another) execution of the process.
- The behavioral perspective evaluates to true.

- The data perspective evaluates to true, i. e. all input data are available (not shown in this paper)
- The organizational perspective evaluates to true.

The discussion of the organizational perspective will be stated precisely. When does it evaluate to true? Two conditions must be fulfilled: First, all (open) variables that are needed for evaluation are set (in the example of Figure 2, the executor of process "Prepare Surgery Plan" must be determined). Second, there are eligible agents available. These two prerequisites are starting points for the two main contributions of this paper. Our aim is to improve process execution, i. e. to prevent process execution from being halted. From the viewpoint of the organizational perspective a process execution delay might be caused by missing variables which are not determined yet. This issue is mitigated through the introduction of the new concept of re-ordering (Section VI-B). A second cause for a delay is that not enough eligible agents are available. Such a situation could be avoided by applying our second innovative concept: planning (Section VI-C). This analysis splendidly shows that and how our contributions enhance the flexibility and efficiency of process execution.

IV. System Architecture

ESProNa implements a three-tier system architecture consisting of a data tier (representing organizational informations), a logic tier (deduction on processes and states) and a presentation tier (front end for process execution). In this paper we will concentrate on the description of the data and logic tier; details on the presentation tier can be found in [30] and [31]. The purpose of the discussion in this section is to show that ESProNa is capable to model any organizational structure and is also capable to implement any perspective of the POPM process model.

A. Data Tier: Representing Organizational Informations

We will first have a look on the data tier where the structure of an organization (with respect to agents, relationships between agents, roles of agents, etc.) is stored. It must be guaranteed that arbitrary relations between organizational entities can be modeled. We decided to implement the data structure on the basis of OWL2 ontologies. This is not a general prerequisite as other kinds of data structures (e. g. a relational database) would also be feasible. Nevertheless, the OWL2 data model, combined with the reasoner (Section IV-B) enables both the representation of any organizational structure and an efficient interpretation of the POPM process model.

In the two use cases (Figure 2, Figure 3) several organizational elements are referenced. We shortly discuss the main concepts which are needed to model the organizational structures needed in the two examples.

First of all, agents (individuals) have to be defined (see [10]). In our organizational structure (Figure 4) a couple of persons (e. g. Charles, John, Jack) are defined. Agents are usually the entities that are executing process steps. Agents belong to different sets. In our example organizational structure, the persons belong to the set "Person".

Besides agents, so called non-agents are part of an organizational structure (see [10]). In our example, there are a couple of non-agents defined: "XRayDepartment", "AssistantDoctor", etc. Non-agents are grouping and characterizing agents; non-agents cannot usually execute process steps. Only the agents that are related to them can finally perform process steps. In Figure 4 we see that some non-agents are of type role (e.g. "AssistantDoctor", "MTA"); other non-agents are of type "Department" (e.g. "CardiologyDepartment"). Here, it becomes obvious that roles are not sufficient to model organizational structure. Concepts that – for example – model the structure of an organization are also required.

The third concept of our organizational modeling approach are relationships. Relationships can connect agents and non-agents arbitrarily. They may represent various facts of a real-world organization, e. g.:

- An agent belongs to a certain department (i. e. non-agent).
- An agent plays a certain role (i. e. non-agent).
- An agent is "the supervisor" of another agents.

This listing is not complete but just shows some important practical modeling scenarios. Refer to [10] for more details on the modeling of organizational structures. We just want to summarize here, that our approach is capable of modeling all the organizational elements that are defined in [10] which is to the best of our knowledge one of the most powerful approaches in that area.

To offer this powerful modeling capability is one of the key features of ESProNa. The challenge now is to set up the logical tier in such a way that these powerful concepts can be evaluated.

B. Logic Tier: Deduction on Processes and States

The basic design rationale of ESProNa is the separation of process model, process state and reasoner. The process model, which represents the constraints on all processes, is loaded into ESProNa. Further is the organizational model (Figure 4) (the organizational constraints of the loaded process model refer to this data structure) loaded as an OWL2 instance separated from the process model. Through this strict separation of process model and organizational model the following advantages can be achieved:

- The Logic Tier is independent of the relationships defined in the organizational model. In case new relationships between agents or non-agents are to be modeled, these additional informations can be added to the organizational model and automatically will be included and evaluated during execution phase.
- Only the process model needs to be updated in case additional information are to be referenced inside the organizational perspectives of the processes. This leads to a very modular design of our different components resulting in a very flexible and easily adaptable system.

The logic tier contains two main components: The reasoner module (RM), which is responsible for computing the next executable steps in the process model and a session administration module (SAM), which manages the different instances

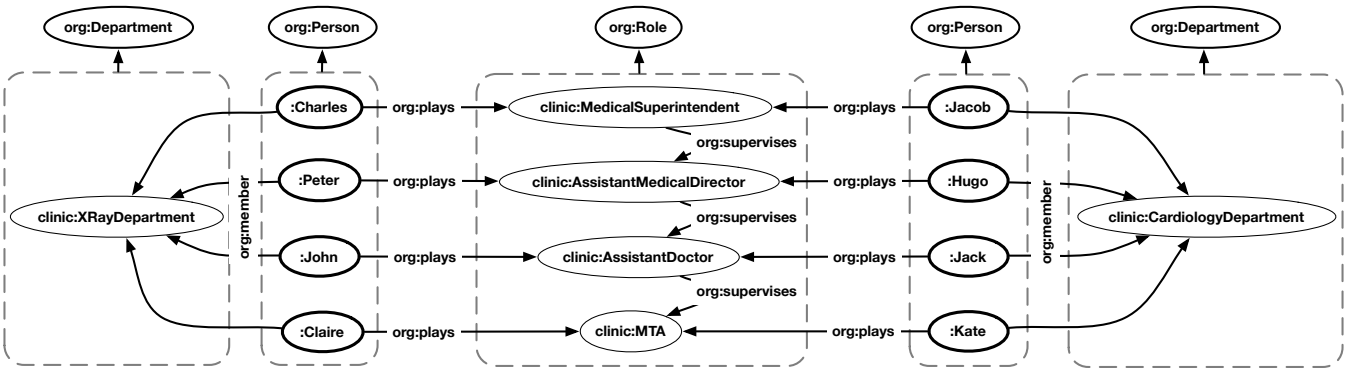

Fig. 4. Extract from a clinical organization representing among others the transitive supervising behavior

of a process (through process states) and communicates with the data and presentation tier. Process execution starts with the loading of the process model from the data tier and the initialization of the processes (the process state is set up) in the SAM. For every loaded process model, a so called process state is managed, which represents all available information about all process instances at a certain time. The reasoner evaluates this state and computes the next executable process steps by evaluating all constraints that are defined for a process step. Since these constraints are classified by the perspectives of a process, the RM evaluates for each perspective whether the process is executable from the viewpoint of the corresponding perspective. For each process that comes into consideration, the conjunction of all evaluations (of the perspectives) is calculated. The state of a process to be executable may change during execution: in the process model shown in Figure 3 the process step "Acknowledge Surgery Plan" cannot be executed in the initial state (this state represents the situation when no process has been executed yet). The behavioral perspective (visualized by the graph icon) of process `PID 2` constraints that it is required that process step `PID 1` is completed (marked as `done`). Therefore the evaluation of process step `PID 2` fails in the initial state (only `PID 1` is executable in the initial state). In a state where process step `PID 1` is marked as `done` (process step `PID 1` has been executed) the evaluation of process step `PID 2` returns "true", because the behavioral perspective evaluates to "true" in this state.

The interesting issue here is that the RM could be extended by further perspectives included into a process model. As already mentioned, each perspective is evaluated individually. The evaluation of the organizational perspective is done by a separated OWL2 reasoner [22]. Therefore arbitrary organizational structures can be evaluated by the RM.

V. Formal Definitions

In this section we will generalize our approach to specify organizational constraints and establish a formal definition which we developed prior to the implementation of the system. It serves as a formal foundation in order to be able to proof the correctness of our reasoning system RM. We implemented the reasoner in Logtalk [32], an object-oriented logic programming language. As already mentioned in Section IV-B the state of each process is stored in a global state object. It contains the `PID` of a process, its execution history (what actions have been performed on it and who performed it) and a boolean variable indicating whether the process is completed or not. Formally this state object can be recursively defined as followed:

- A global state is a list of tuples of the form: `(PID, HL, SC)` (for every process such a triple is stored inside the global state)
- `PID` represents the global and unique process identifier
- The history list `HL` is set up of compound terms of the form `A-P`:
- `A` represents the action that was performed on the process
- `P` is the person who performed that action `A`
- The status code `SC` is a boolean variable indicating whether a process is completed or not

The execution state of a process results from the combined states of its process steps. A set of actions (start, finish and abort) are defined that change the state of a process. For example, when a process is started, this information together with the person who is performing the action is stored in the corresponding part of a state object. State objects are changed by transitions; a transition occurs when a user starts, finishes or aborts a process. The allowed transitions with their succeeding states of a certain situation are derived from a given state object.

We have developed a planning system that computes all possible ways a process model may be executed potentially. All theoretically possible transitions form a graph that contains two special kinds of nodes: an `INITIAL`-state and a `COMPLETE`-state. The initial state (no process is executed yet) is stored in a so called "initial state object". The final state (all processes are marked as "done") is also computed. Together with state transitions it is possible to calculate the complete automaton that represents all possible execution scenarios of a given process model. Formally, our state transition system is a tuple $\Sigma = (S, P, A, \gamma)$ where

- $S = \{s_1, s_2, ..., s_n\}$ is a finite or recursively enumerable set of states

- $P = \{startable, abortable, finishable\}$ is a finite set of preconditions
- $A = \{start, abort, finish\}$ is a finite set of actions
- $\gamma : S \times P \times A \rightarrow S$ is a state-transition function

A state-transition system can be represented by a directed graph whose nodes are states of S. If $s \in \gamma(s, p_i, a_i)$ with $p \in P_n$ and $a \in A_n$, then the graph contains an arc from s to s' that is labeled with u. Each state-transition can have one or more successive states s' and therefore can be non-deterministic. The state s_{init} is the initial state with no incoming edges; the state s_{goal} is the final state with no outgoing edges. s_{init} refers to the state of the process model when no process has been executed yet (no action applied); s_{goal} refers to a situation where all processes have been executed already (all processes are marked as "done"). An action a, that is performed on a process, changes the state in a way that the precondition for a second process evaluates to true and therefore an action a' can be applied to it. For the organizational perspective we need to further specify the set of preconditions and actions:

- $AG = \{agent_1, agent_2, ..., agent_n\}$ is a finite or recursively enumerable set of agents
- $C = \{candidate_1, candidate_2, ..., candidate_n\}$ is a finite or recursively enumerable set of candidates; $C \subseteq AG$
- $\tau : AG \rightarrow C$ is a relation that maps agents to candidates

The relation $m : AG \rightarrow C$ holds a mapping of certain elements of AG to C. The set of agents can be found in Figure 4 inside the dashed boxes (second from left and right in each case). The candidates are retrieved through an evaluation of the constraints (asserted in the process model) on these agents. Listing 1 illustrates such an evaluation which retrieves all "Assistant Doctors" who are candidates and build the set C. Therefore every element p_i of preconditions P is a tuple (s, c) where $s \in S$ and $c \subseteq C$. C, represents the set of candidates who can perform the process step.

```
1 'clinic:XRayDepartment'::'org:member'(M),
2 M::'org:plays'('clinic:AssistantDoctor').
```

Listing 1. Retrieving all Assistant Doctors of the X-Ray department

Every element a_i of actions A is a triple (s, c, s') where $s, s' \in S$ and $c \subseteq C$. s represents the state that is considered for the evaluation of γ and τ. The calculated state s' represents the state when action a_i has been performed. The state transition γ, which is of non-deterministic kind, calculates for a given state $s \in S$ one or more succeeding states s'. If no succeeding states can be calculated then $s = s_{goal}$. The second component u of the state transition γ must be valid (true), so the transition can be applied. First, the precondition p_i is evaluated against all processes. During this evaluation all perspectives of a certain process are validated. If none of the perspectives is violated, then the precondition of the process is valid and therefore the desired action can be performed on the process.

VI. Process Execution

A. Normal Execution

The process model we use in this subsection is the one presented in Figure 3. We want to check which process can be started in the initial state. For this purpose the execution prerequisites for every process have to be checked. In this case we evaluate all perspectives of each process. In process `PID 1` the functional constraint defines that the process can be executed exactly once. The second constraint (organizational perspective) restricts the persons who can perform the process to the ones playing the role of an "Assistant Doctor". The persons who are candidates for this task are (according to Figure 4) `John` from the X-Ray Department and `Jack` from the Cardiology department. As no behavioral constraint is given in process "Prepare Surgery Plan" (neither through an ingoing arrow nor through an explicit constraint inside the process) the prerequisite check evaluates that the process can actually be started by the persons `John` and `Jack`. In case of process `PID 2` the evaluation of the organizational constraint would lead to a negative result, as the prerequisites prescribe that process `PID 1` has to be executed before process `PID 2` (the system must know the executor of `PID 1` to determine the executor of `PID 2` who is the supervisor of the former one). Therefore process "Acknowledge Surgery Plan" cannot be marked as `startable` in the initial state.

We now assume that process `PID 1` has been executed by `John`. This state is marked as ❷ in Figure 5 which represents the automaton for the process model of Figure 3. All possible execution scenarios are included in this graph. It is a virtual state space (all possible execution scenarios are visualized), since it is computed by ESProNa but is not stored explicitly. One path (bold arrows) from the initial state ❶ of the tree to one leaf state ❸ is marked. It represents a possible execution scenario of the process model where `John` starts the process "Prepare Surgery Plan" and `Peter`, as his supervisor, finishes it.

In situation (state) ❷ the system computes that process `PID 2` now becomes `startable`. The evaluation of the behavioral constraint in that situation returns "true" because process `PID 1` is completed. The interesting part is to determine, who (which person from the organization) can perform the process. This information can be retrieved from the organizational model in the data tier (Figure 4). Therefore we first have to retrieve the person who started process `PID 1`. We assumed that this was `John`. Hence a query against the data (this query is stated by our system internally during evaluation) would retrieve that `Peter` and `Charles` are candidates, because they keep a supervisor role with respect to `John`.

B. Re-ordering of Process Models

In this section we want to resume the supervisor example presented in the introduction. The process model in Figure 6 is similar to the one in Figure 3, except that the behavioral restriction between the two process steps is removed: We omit the constraint that process `PID 1` "must" be performed before process `PID 2`.

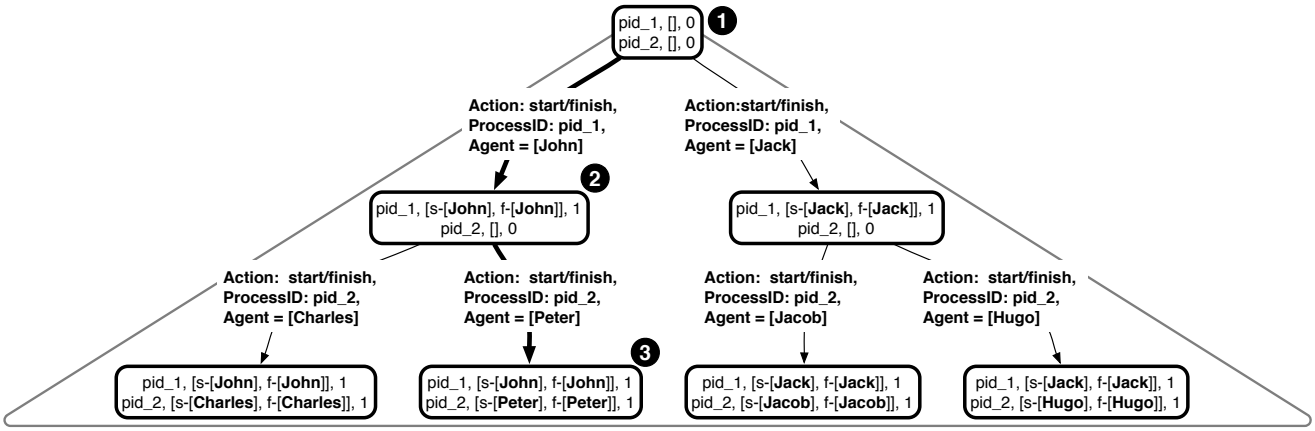

Fig. 5. Illustration of possible execution scenarios of the clinical process model from Figure 3

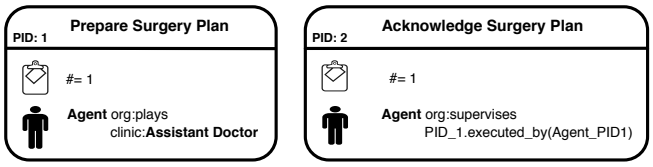

Fig. 6. Modified clinical example model from Figure 3

It is problematic to execute `PID 2` since its execution requires to know who executed `PID 1` since the supervisor of that executor should execute `PID 2`. ESProNa fosters two different execution strategies in this case which enhances execution flexibility enormously. The two strategies to choose from are called Forward and Backward Chaining.

Forward Chaining: This is the standard behavior and described in the previous section. Since the organizational perspective requires the execution of `PID 1` (in order to determine the supervisor), `PID 2` cannot be executed at first with this concept.

Backward Chaining: In this scenario process step `PID 2` is performed first although this seems to conflict with the former observations. However, ESProNa exploits the organizational graph structure of Figure 4 and follows the relationships given in that organizational structure backwards.

The concept of Backward Chaining is realized in the following way: ESProNa first retrieves all supervisors, in our situation `Peter` and `Charles` from the X-Ray department and `Hugo` and `Jacob` from the Cardiology department. Then ESProNa allows them to start `PID 2`. We assume that `Peter` is actually starting process `PID 2`. He acknowledges the surgery plan and finishes the process. Now, ESProNa backward chains the organizational structure of Figure 4. Therefore, it checks the potential executors of process step "Prepare Surgery Plan" regarding that `Peter` executed `PID 2`. A query to the data tier will return that `John` is an "Assistant Doctor" and is supervised by `Peter`. Consequently `John` is eligible to execute `PID 1`.

The latter example shows the flexibility ESProNa provides. It is achieved by modeling the organizational structure in a formal way (Figure 4) and utilizing that structure as much as possible with logic operators. According to our knowledge there is no other implementation of the organizational perspective of process management systems that allows this flexibility. It copes with the "non-predictability" of this perspective since a much broader spectrum of execution orders becomes available through exploiting the logical structure of organizational constraints. This feature helps to bridge temporary bottlenecks in the organizational perspective as it was discussed in Section I.

C. Planning of Process Models

As already mentioned in Section I, the concept of planning on process models is used to predict problematic situations in order to avoid them. As an example, `Peter` as one of the supervisors, is on vacation. Normally the "Assistant Doctors" (in `XRayDepartment`) would consult `Peter` for the acknowledgement of the preparation of a surgery plan. But as `Peter` is not available, the execution of process step `PID 2` would be delayed until he is back from vacation. Our planning module would predict such bottlenecks and guide the user in order to avoid them. Whenever a process step is finished, the planning module calculates potential agents required to perform upcoming process steps. It would encounter that through the absence of Peter (this information must be kept in the data tier) the execution of the process would be hindered. The planning module would inform a system administrator about this bottleneck. The system administrator then could - if it is possible from an application point of view - re-define the organizational constraint for process step `PID 2`. For example, not the direct supervisor but the supervisor of her supervisor should execute process step `PID 2`. So, the execution of the process is not delayed.

As discussed before, the planning module calculates all potential agents that must perform upcoming processes. Not just bottlenecks are detected by this, but also overload situations

could be encountered. In such a situation the planning component would calculate (and recommend) that former process steps should or should not be performed by specific people in order to avoid such a overload. For example, we assume that agents `A1` or `A2` have to execute a certain process step. The next process step has to be executed by the supervisor of the former executor. Let's assume that `S3` is the supervisor of `A1` and `A2`; furthermore, `A2` also has `S4` as supervisor. Let's further assume that `S3` is completely overloaded while `S4` is not occupied at all. Our planning component would detect this situation and would recommend that `A2` instead of `A1` should execute the process step in order to avoid that `S3` is the only agent to execute the next process step. In [33] we detail the implementation underlying our process modeling and planning components.

VII. Conclusion

In this paper we introduced a novel concept for realizing the organizational perspective in process management systems. Organizational constraints are interpreted as constraints that have an impact on the execution order of process steps. We first outlined that arbitrary organizational structures can be defined within our approach. We then described how these organizational structures can be exploited to improve the execution of processes by offering the two concepts "re-ordering" and "planning". The implementation (ESProNa) makes use of the Logtalk object-oriented logic programming language to provide a robust and easily maintainable system.

References

[1] H. Smith and P. Fingar, *Business Process Management (BPM): The Third Wave*. Meghan-Kiffer Press.

[2] J. Cardoso and W. Van Der Aalst, *Handbook of Research on Business Process Modeling*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2009.

[3] M. Weske, *Business Process Management: Concepts, Languages, Architectures*, 1st ed. Springer, November 2007.

[4] S. Jablonski and C. Bußler, "Workflow-management: Modeling concepts, architecture and implementation," International Thomson Computer Press, 1996.

[5] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and Parallel Databases*, vol. 3, no. 2, pp. 119–153, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF01277643

[6] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam, "Adaptive process management with adept2," in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1113–1114.

[7] S. Jablonski and C. Bussler, *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press, September 1996.

[8] S. Jablonski, "Functional and behavioral aspects of process modeling in workflow management systems," in *CON'94: Proceedings of the Ninth Austrian-informatics conference on Workflow management: challenges, paradigms and products*. Munich, Germany, Germany: R. Oldenbourg Verlag GmbH, 1994, pp. 113–133.

[9] OASIS. (2010, Mar) Web services business process execution language version 2.0. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[10] C. Bußler, *Organisationsverwaltung in Workflow Management Systemen*. Dt. Univ.-Verl., Jan 1998.

[11] M. Zur Muehlen, "Organizational management in workflow applications – issues and perspectives," *Inf. Technol. and Management*, vol. 5, no. 3-4, pp. 271–291, 2004.

[12] M. Ford, A. Endpoints, and C. Keller, "Ws-bpel extension for people (bpel4people), version 1.0," 2007.

[13] A. Agrawal, "Web Services Human Task (WS-HumanTask), Version 1.0," 2007.

[14] OASIS Standard. Web service business process execution language version 2.0. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[15] E. Bertino, E. Ferrari, and V. Atluri, "The specification and enforcement of authorization constraints in workflow management systems," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 65–104, 1999.

[16] F. Casati, S. Castano, M. Fugini, I. Mirbel, and B. Pernici, "Using patterns to design rules in workflows," *IEEE Transactions on Software Engineering*, vol. 26, pp. 760–785, 2000.

[17] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support." *Computer Science — R&D*, vol. 23, no. 2, pp. 99–113, 2009.

[18] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "DECLARE: Full support for loosely-structured processes," in *EDOC'07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*. Washington, DC, USA: IEEE Computer Society, 2007, p. 287.

[19] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artif. Intell.*, vol. 116, no. 1-2, pp. 123–191, 2000.

[20] S. Mukherjee, H. Davulcu, M. Kifer, P. Senkul, and G. Yang, "Logic based approaches to workflow modeling and verification." in *Logics for Emerging Applications of Databases*, J. Chomicki, R. van der Meyden, and G. Saake, Eds. Springer, 2003, pp. 167–202. [Online]. Available: http://dblp.uni-trier.de/db/conf/dagstuhl/lead2003.html#MukherjeeDKSY03

[21] C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith, "Owl 2 web ontology language structural specification and functional-style syntax," June 2009. [Online]. Available: http://www.w3.org/TR/2009/CR-owl2-syntax-20090611

[22] V. Vassiliadis, J. Wielemaker, and C. Mungall, "Processing OWL2 Ontologies using Thea: An Application of Logic Programming," in *OWLED*, ser. CEUR Workshop Proceedings, R. Hoekstra and P. F. Patel-Schneider, Eds., vol. 529. CEUR-WS.org, 2009.

[23] S. Goedertier, R. Haesen, and J. Vanthienen, "EM-BrA^{2}CE v0.1: A vocabulary and execution model for declarative business process modeling," K.U.Leuven, FETEW Research Report KBI-0728, 2007.

[24] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[25] W. Aalst, A. H. M. ter Hofstede, and N. Russell. Workflow pattern. [Online]. Available: http://www.workflowpatterns.com

[26] P. Athena. Flower. [Online]. Available: http://www.pallas-athena.com

[27] B.-S. GmbH. The cosa bpm suite. [Online]. Available: http://www.cosa.nl

[28] S. White. (2004, May) Business Process Modeling Notation (BPMN) — Version 1.0. [Online]. Available: http://www.bpmn.org/Documents/BPMN_V1-0_May_3_2004.pdf

[29] IBM. Websphere mq workflow. [Online]. Available: http://www-01.ibm.com/software/integration/wmqwf

[30] M. Faerber, S. Jablonski, and S. Meerkamm, "The ProcessNavigator — Flexible process execution for product development projects," *International Conference on Engineering Design, ICED'09*, 2009.

[31] M. Faerber, F. Jochaud, F. Stöber, S. Jablonski, and H. Meerkamm, "Knowledge oriented Process Design for DfX," *10th International Design Conference*, 2008.

[32] P. Moura, "Logtalk — Design of an Object-Oriented Logic Programming Language," Ph.D. dissertation, Department of Computer Science, University of Beira Interior, Portugal, Sep. 2003.

[33] M. Igler, P. Moura, and S. Jablonski, "ESProNa: Constraint-Based Declarative Business Process Modeling," 2010, Third International Workshop on Dynamic and Declarative Business Processes, DDBP'10.