On the Controlled Evolution of Access Rules in Cooperative Information Systems

Stefanie Rinderle[1,⋆] and Manfred Reichert[2]

[1] Department Databases and Information Systems, University of Ulm, Germany
rinderle@informatik.uni-ulm.de
[2] Information Systems Group, University of Twente, The Netherlands
m.u.reichert@cs.utwente.nl

Abstract. For several reasons enterprises are frequently subject to organizational change. Respective adaptations may concern business processes, but also other components of an enterprise architecture. In particular, changes of organizational structures often become necessary.

The information about organizational entities and their relationships is maintained in organizational models. Therefore the quick and correct adaptation of these models is fundamental to adequately cope with changes. However, model changes alone are not sufficient to guarantee consistency. Since organizational models also provide the basis for defining access rules (e.g., actor assignments in workflow management systems or access rules in document–centered applications) this information has to be adapted accordingly (e.g., to avoid non-resolvable actor assignments). Current approaches do not adequately address this problem, which often leads to security gaps and delayed change adaptations.

In this paper we present a comprehensive approach for the controlled evolution of organizational models in cooperative information systems. First, we introduce a set of operators with well-defined semantics for defining and changing organizational models. Second, we present an advanced approach for the semi-automated adaptation of access rules when the underlying organizational model is changed. This includes a formal part concerning both the evolution of organizational models and the adaptation of related access rules.

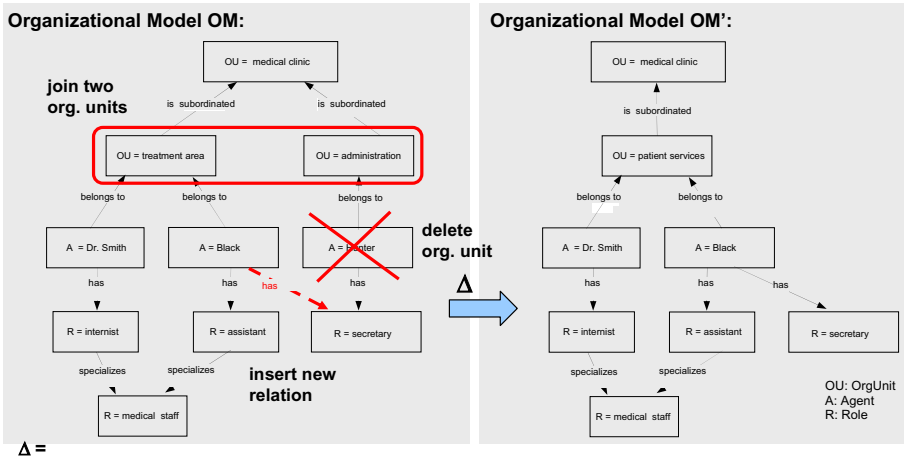
1 Introduction

Enterprise-wide, cooperative information systems (IS) comprise a variety of application and system components. Important tasks to be accomplished include the support of business processes, the management of enterprise documents and the integration of enterprise applications. For the implementation of such services different middleware components exists, like workflow systems, document management systems, and tools for enterprise application integration [1, 2, 3].

The controlled access to its application services as well as to the application objects managed by them (e.g., business processes, business documents,

⋆ This research work was conducted during a postdoc stay at the University of Twente

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 238–255, 2005.



Δ =

```
(JoinEntities(OM, (treatment area, OrgUnit), (administration, OrgUnit), (patient, services, OrgUnit),
  DeleteRelation(OM, ((patient services, OrgUnit), (Hunter, Agent), belongs to)),
    DeleteRelation(OM, ((Hunter, Agent), (secretary, Role), belongs to)),
      DeleteEntity(OM, (Hunter, Agent)), CreateRelation(OM, (Black, Agent), (secretary, Role), has)))
```

Fig. 1. Example of an Organizational Model and a Model Change (simplified)

resources, application systems, etc.) constitutes an important task for any cooperative IS. Usually, this results in a large number of access rules covering different system aspects and user privileges. Furthermore, these rules have to be frequently adapted due to changes of organizational structures [4, 5, 6]. Such organizational changes become necessary, for instance, when an organizational unit is split into two sub-units, two existing units are joined, a group of users is reassigned to a new unit, or simply an employee leaves the organization.[1] As a consequence, the access rules whose definition is based on organizational entities may have to be adapted as well. So far, the controlled evolution of access rules in cooperative IS has not been addressed in sufficient detail, which has led to severe security gaps when organizational changes are introduced.

Typically, information about organizational entities (e.g., organizational units, roles, and users) and the relations between them (e.g., assignment of a user to a role or an organizational unit) is kept in an organizational model. Based on such a model, access rights and user privileges can be defined (e.g., actor assignments in a workflow management system or access rules in document-centered applications). Consequently, when organizational changes occur, both the organizational model and the related access rules have to be adapted in a consistent manner.

Another (practical) problem arises from the fact that the (middleware) components used to build the application services of cooperative IS often maintain their own organizational model and security component; i.e., the information about organizational entities (and their relations) as well as the access rules

[1] For respective results from one of our case studies in the clinical domain see [4].

based on them may be scattered over different system components. On the one hand this has led to functional redundancy, on the other hand (heterogeneous)information about organizational structures is kept redundantly in different security components. The latter very often results in inconsistencies, high costs for system maintainability, and inflexibility when dealing with organizational change.

But even if the organizational model is kept in a central repository, the problem of maintainability remains. The correct and consistent adaptation of the organizational model is only one side of the coin when dealing with organizational changes; the other is to correctly and efficiently adapt the access rules defined on basis of this model. Note that in large environments hundreds up to thousands of access rules may exist, each of them capturing different privileges of the cooperative IS. This, in turn, makes it a hard job for the system administrator to quickly and correctly adapt these rules to changes of the organizational model(s).

Current approaches do not sufficiently deal with this issue. They neither make use of the semantics of the applied model changes nor do they provide any automated support for rule adaptation. In practice, this often leads to problems like non-resolvable actor assignments, unauthorized access to business documents, or inconsistent user worklists. Assume, for example, that two organizational units are joined in order to make the enterprise more efficient (cf. Fig. 1). If this change is performed in an uncontrolled manner, it may result in non-resolvable access rules referring to one of these two units (no longer present in the new organizational model).

Facing these challenges we need a logically centralized component which manages the organizational model and its changes in a consistent and efficient manner. Furthermore, model changes have to be propagated to access rules (used within different system components) in a correct and efficient manner. Finally, we have to consider both, access rules which are checked when a certain privilege is applied (e.g., when a user wants to access a document) and rules which are used to determine a set of authorized users (e.g., the set of actors who may work on a certain workflow activity).

In this paper we present a comprehensive approach for the controlled evolution of organizational models in cooperative IS. This approach complements our previous work on the controlled evolution of process models and process instances in adaptive process management systems [7, 8, 9, 10, 11]. First, we introduce a set of operations with well-defined semantics for defining and changing organizational models. Second, we present an advanced approach for the semi-automated adaptation of access rules when the referred organizational model is modified. For selected organizational changes we show how they can be realized in our formal framework and how their effects on access rules look like. We then try to derive migration strategies for affected access rules. Thereby we make use of the semantics of the applied model changes and we introduce formally sound migration concepts. For this purpose, we introduce a formal meta model for defining and changing organizational models and related ac-

cess rules. Altogether this paper includes a formal part concerning the evolution of organizational models and related access rules as well as a discussion of practical issues.

Section 2 discusses related work. In Section 3 we present a sample meta model for defining organizational models and access rules. Section 4 deals with the evolution of organizational models and the (semi-automated) adaptation of related access rules. Use cases and practical issues are sketched in Section 5. We conclude with a short summary and an outlook on future work.

2 Related Work

The provision of adequate access control mechanism is indispensable for any cooperative IS. In the literature many approaches exist dealing with corresponding issues (e.g., [6, 12, 13, 14]). Most of them use *Role–Based Access Control (RBAC)* models for defining and managing user privileges [15, 16, 12, 17], e.g., for ensuring the controlled access to business documents when using *document management* technology [18] or for resolving the set of actors that qualify for a certain task in a *workflow management system* [19, 20, 21, 13, 14]. Regarding workflow–based applications, in addition, dynamic constraints (e.g., separation of duties) have been considered [20, 21]. So far, however, only few approaches [22, 23] have addressed the problem of organizational change (see below).

Issues related to the modeling of organizational structures have been considered by different groups [5, 13, 24]. Most of them suggest a particular meta model for capturing organizational entities and their relations. Model changes and the adaptation of access rules, however, have not been studied by these approaches in sufficient detail.

In [25] several issues related to changes of process and organizational structures have been discussed. In this work the authors also motivate the need for the controlled change of organizational models. In particular, they discuss different kinds of adaptations that have to be supported (e.g., to extend, reduce, replace, and re-link model elements). However, no concrete solution approach is provided (like, for example, formal change operators with well–defined semantics or mechanisms for adapting access rules after model changes).

In [6, 26] the author identifies eight different categories for structural changes of organizational models. Examples of such change categories include the splitting of organizational units, the creation of new organizational entities, and the re-linkage of a user to a new unit. In principle, all these cases can be captured by our change framework as well. As opposed to [6], however, we have followed a rigorous formal approach in order to be able to derive the effects of organizational changes on related access rules as well. This issue has not been addressed in [6].

Other approaches have introduced role–based access control model for adaptive workflows [14, 23]. In [23] the authors additionally address issues related to the evolution of access rights in workflow management systems. However, no formal considerations are made and only simple cases are managed when compared to our approach.

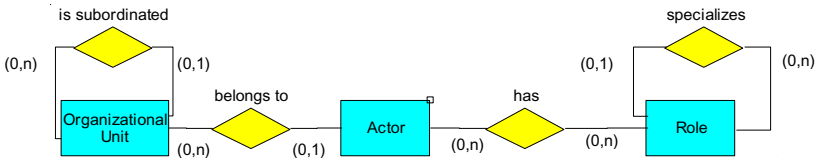


Fig. 2. Organizational Meta Model

3 Organizational Models and Access Rules

In order to be able to reason about organizational changes and their concrete impact on access rules we need a formalization of organizational structures. For this purpose, first of all, we introduce an (organizational) meta model, which is comparable to existing RBAC models (e.g., [15]) and which can be used for describing organizational entities and the relations between them (cf. Fig. 2). Due to lack of space, in this paper we restrict our considerations to the basic entity types *organizational unit*, *role* and *actor*, and to the particular relation types existing between them (e.g., actor A_1 belongs to organizational unit O_1, role R_1 specializes role R_0, etc.). In our complete framework currently implemented in the ADEPT2 project, we also consider entity types like *position*, *group* or *capability* when defining and changing organizational models [24].

Regarding the meta model OMM assumed in this paper (cf. Fig. 2) we specify the set of valid entity types and the set of valid relation types as follows:

- *EntityTypes* := {`OrgUnit, Actor, Role`}
- *RelationTypes* := {`(OrgUnit, OrgUnit, is subordinated), (Role, Role, specializes), (Actor, OrgUnit, belongs to), (Actor, Role, has)`}

We further denote

- $\mathcal{E}$:= $\mathcal{E}_{Id}$:= {(entId, entType) | entId ∈ Id, entType ∈ EntityTypes)} as the set of all entities definable over a set of identifiers *Id* and
- $\mathcal{R}_{\mathcal{E}}$:= {($e_1$, e_2, relType) | e_1 = (eId_1, $eType_1$), e_2 = (eId_2, $eType_2$) ∈ $\mathcal{E}$, ($eType_1$, $eType_2$, relType) ∈ RelationTypes} as the set of all relations definable over $\mathcal{E}$

Actors are users (or resources) who need privileges to work on certain tasks (e.g., workflow activities) or to access certain data objects (e.g., documents). Generally, access rules are not directly linked to actors, but to the more abstract concept of a *role*. Roles group privileges and are assigned to actors based on their capabilities and competences. Generally, an actor can play different roles: A physician in a hospital, for example, may play the two roles *ward doctor* and *radiologist*. Actors possessing the same role are considered as being interchangeable. Furthermore, roles can be hierarchically organized, i.e., a role may have

one or more specialized sub–roles. Thereby a sub–role inherits all privileges of its super–role and may extend this set by additional privileges. Finally, each actor can be assigned to an *organizational unit*. Like roles, organizational units can be hierarchically structured; i.e., a particular unit may have one or more subordinated units (e.g., a medical hospital may have an intensive care unit and an emergency laboratory).

Based on the introduced meta model we can now define the notion of *organizational model*. For the sake of simplicity, in this paper we do not consider the cardinalities associated with the relation types of our meta model (cf. Fig. 2).

Definition 1 (Organizational Model). *For the organizational meta model OMM let $\mathcal{E}$ be the set of all entities over a given set of identifiers and let $\mathcal{R}_{\mathcal{E}}$ be the set of all relations over $\mathcal{E}$ (see above). Then: An organizational model OM is defined as a tuple (Entities, Relations) with Entities $\subseteq \mathcal{E}$ and Relations $\subseteq \mathcal{R}_{\mathcal{E}}$.*

The set of all org. models definable on basis of OMM is denoted as $\mathcal{OM}$.

Let $OM = (Entities, Relations)$ be an organizational model. Based on the organizational entities and relations described by OM we can define rules in order to control the access to tasks, services, documents, or other objects. Since the structuring and semantics of the access rules is fundamental for the (semi-) automated derivation of rule adaptations, we consider this issue in more detail. We distinguish between elementary and complex access rules.

An *elementary access rule* consists of a simple expression that qualifies a set of entities from OM (i.e., a subset of *Entities*) for this rule. The elementary access rule `Actor = 'Hunter'`, for example, expresses that exactly one entity, namely the actor with name 'Hunter', qualifies for this rule and therefore owns the privileges associated with it. As a second example consider the elementary access rule `OrgUnit = clinic`. For this access rule we denote the organizational unit `clinic` as the *qualifying entity*. Furthermore, all actors belonging to this unit own the privileges associated with this rule (e.g., getting access to a certain business document). For entities that can be hierarchically organized (i.e., for organizational units and roles in our meta model) we further allow for the definition of *transitive* elementary access rules. As an example consider the elementary access rule `OrgUnit = clinic(+)`. For this rule the set of qualifying entities comprises the organizational unit `clinic` itself and all of its directly or indirectly subordinated units (i.e., the *transitive closure* with respect to the 'is-subordinated' relation). All actors belonging to one of these qualifying units own the privileges associated with this rule. Similar considerations can be made regarding the 'specializes' relation between entities of type `Role`.

Definition 2 (Elementary Access Rule).
Let OM = (Entities, Relations) be an organizational model based on OMM. Then an elementary access rule `EAR` *on OM is defined as follows:*

`EAR` $\equiv$ `(EAR1` $\longleftarrow$ `(EntityType = el))` | `(EAR2` $\longleftarrow$ `(OrgUnit = el(+)))` | `(EAR3` $\longleftarrow$ `(Role = el(+)))`

The set of entities qualifiying for one of the elementary access rules `EAR1`*,* `EAR2` *or* `EAR3` *can be determined as follows:*

– EAR1 ⟵ (EntityType = el)

$$QualEntities(OM, \texttt{EAR1}) = \begin{cases} \{(el, EntityType)\} & : \ (el, EntityType) \in Entities \\ \emptyset & : \ otherwise \end{cases}$$

– EAR2 ⟵ (OrgUnit = el(+))

$$QualEntities(OM, \texttt{EAR2}) = \begin{cases} \{(el, \texttt{OrgUnit})\} \cup Sub(OM, el) & : \ (el, \texttt{OrgUnit}) \in Entities \\ \emptyset & : \ otherwise \end{cases}$$

with

$$Sub(OM,\ el) := \bigcup_{el':(el', el, \texttt{issubordinated}) \in Relations} (\{(el', \texttt{OrgUnit})\} \cup Sub(OM, el'))$$

– EAR3 ⟵ (Role = el(+))

$$QualEntities(OM, \texttt{EAR3}) = \begin{cases} \{(el, \texttt{Role})\} \cup Spec(OM, el) & : \ (el, \texttt{Role}) \in Entities \\ \emptyset & : \ otherwise \end{cases}$$

with

$$Spec(OM,\ el) := \bigcup_{el':(el', el, \texttt{specializes}) \in Relations} (\{(el', \texttt{Role})\} \cup Spec(OM, el'))$$

In order to enable the definition of more complex access rules we allow for the composition of existing rules (cf. Definition 3). For this purpose the following operators can be used: negation, conjunction, and disjunction.

Definition 3 (Access Rule).
Let OM = (Entities, Relations) be an organizational model based on OMM. Then an access rule AR on OM is defined as follows:

AR ≡ EAR | NEAR | CAR | DAR *with*

– NEAR ⟵ (NOT (EAR)) *where* EAR *is an elementary access rule*
– CAR ⟵ (AR1 AND AR2) *with* AR1 *and* AR2 *are access rules*
– DAR ⟵ (AR1 OR AR2) *with* AR1 *and* AR2 *are access rules*

Consider the organizational model OM depicted in Fig. 1. An example for a composed access rule on OM is AR ⟵ (OrgUnit = medical clinic(+) AND Role = assistant). Regarding the first part of this rule (i.e., the elementary access rule EAR1 ⟵ (OrgUnit = medical clinic(+)) we obtain QualEntities(OM, EAR1) = {medical clinic, treatment area} as the set of qualifying entities. For the second elementary rule EAR2 ⟵ (Role = assistant) the set of qualifying entities is QualEntities(OM, EAR2) = {assistant}.

We can use Definition 2 and Definition 3 in order to determine the set of actors qualifying for access rule AR (cf. Definition 4). Corresponding actors then own the privileges associated with this rule.

Definition 4 (Valid Actor Set). *Let OM = (Entities, Relations) be an organizational model. Let Act(OM) := {(a,* Actor*)| (a,* Actor*) ∈ Entities} be the set of all actors defined by OM, and let* AR *be an access rule on OM. Then: VAS(OM,* AR*) denotes the set of all actors (from OM) who qualify for* AR *(i.e., who own the privileges associated with rule* AR*). Formally:*

– AR ⟵ (EntityType = el) ⟹

$$VAS(OM, \texttt{AR}) = \begin{cases} \{(el, \texttt{Actor}) | (el, \texttt{Actor}) \in Act(OM)\} & if\ EntityType = \texttt{Actor} \\ \{(a, \texttt{Actor}) | (a, \texttt{Actor}) \in Act(OM) \wedge \\ \quad \exists (a, el, \texttt{belongsto}) \in Relations)\} & if\ EntityType = \texttt{OrgUnit} \\ \{(a, \texttt{Actor}) | (a, \texttt{Actor}) \in Act(OM) \wedge \\ \quad \exists (a, el, \texttt{has}) \in Relations)\} & if\ EntityType = \texttt{Role} \end{cases}$$

– AR ⟵ (EntityType = el(+)) ⟹

$$VAS(OM, \texttt{AR}) = \begin{cases} \{(a, \texttt{Actor}) | (a, \texttt{Actor}) \in Act(OM) \wedge \\ \quad \exists el' \in QualEntities(OM, \texttt{AR}) : \\ \quad\quad \exists (a, el', \texttt{belongsto}) \in Relations)\} & if EntityType = \texttt{OrgUnit} \\ \{(a, \texttt{Actor}) | (a, \texttt{Actor}) \in Act(OM) \wedge \\ \quad \exists el' \in QualEntities(OM, \texttt{AR}) : \\ \quad\quad \exists (a, el', \texttt{has}) \in Relations)\} & if EntityType = \texttt{Role} \end{cases}$$

- AR $\longleftarrow$ *(*NOT (AR1)*) with* AR1 *is* EAR $\Longrightarrow$
  VAS(OM, AR*)* $= Act(OM) \setminus$ *VAS(OM,* AR1*)*
- AR $\longleftarrow$ (AR1 AND AR2) *with* AR1 *and* AR2 *are access rules* $\Longrightarrow$
  VAS(OM, AR*)* $=$ *VAS(*AR1*)* $\cap$ *VAS(*AR2*)*
- AR $\longleftarrow$ (AR1 OR AR2) *with* AR1 *and* AR2 *are access rules* $\Longrightarrow$
  VAS(OM, AR*)* $=$ *VAS(*AR1*)* $\cup$ *VAS(*AR2*)*

Finally, we provide a criterion which allows us to decide when an access rule AR is *valid* with respect to a given organizational model OM. We call an access rule valid if the following two conditions hold:
(1) AR does not contain *dangling references*, i.e., it does not refer to entities which are not present in OM. Formally:

$$DanglingRef(OM, \texttt{AR}) = \begin{cases} \texttt{False} \; if \forall \; \texttt{EAR} \; in \; \texttt{AR} : QualEntities(OM, \texttt{EAR}) \neq \emptyset \\ \texttt{True} \quad otherwise \end{cases}$$

(2) AR is resolvable, i.e., the set of valid actors VAS(OM, AR) does not become empty. Formally:

$$Resolv(OM, \texttt{AR}) = \begin{cases} \texttt{True} \quad if VAS(OM, \texttt{AR}) \neq \emptyset \\ \texttt{False} \; otherwise \end{cases}$$

Note that dangling references and/or non-resolvable access rules might occur when changing organizational models in an uncontrolled manner.

Criterion 1 (Valid Access Rule) *Let OM = (Entities, Relations) be an organizational model and let* AR *be an access rule on OM. Then* AR *is valid regarding OM if and only if there are no dangling references within the elementary access rules contained in* AR *and* AR *is resolvable over the set Entities, formally:*
Valid(OM, AR*)* $=$ True $\Longleftrightarrow$ *(DanglingRef(OM,* AR*)* $=$ False $\wedge$ *Resolv(OM,* AR*)* $=$ True*)*

4 Organizational Evolution and Effects on Access Rules

How do changes of an organizational model OM affect the access rules based on it? In order to find a precise and satisfactory answer to this question, first of all, we must be able to formally define a model change Δ and its semantics (i.e., its effects on OM). Based on this information it should be possible to determine which access rules (on OM) are affected by the model change, how the effects of Δ on these rules look like, and which rule adaptations become necessary.

In the following we assume the scenario depicted in Fig. 3: A (correct) organizational model OM is transformed into another (correct) model OM' by applying change $\Delta = op_1, ..., op_n$ to it. The challenge then is to adapt valid

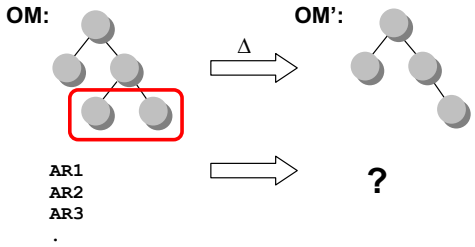


Fig. 3. Changing Organizational Models and Migrating Access Rules

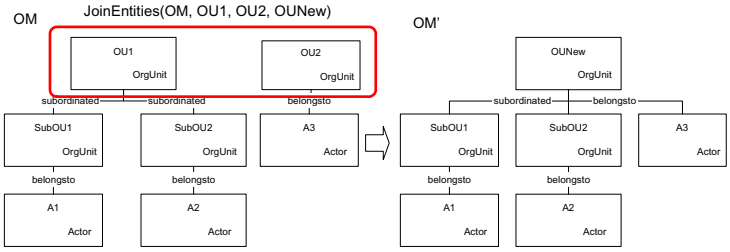


Fig. 4. Joining Organizational Units

access rules on OM in a way that they remain valid on OM' as well; i.e., to migrate these access rules from OM to OM', but without causing errors or inconsistencies (e.g., dangling references).

In order to be able to express all kind of changes on an organizational model OM, we provide a complete set of basic change operations to the user (e.g., for creating / deleting entities and relations). For each change operation we define formal pre– and post–conditions, which preserve the correctness of OM when applying the operation(s) to it (assuming that OM has been a correct model before). In addition to these basic change operations we provide frequently used, high–level operations in order to facilitate change definition and to capture more semantics about changes. Examples include operations for joining two entities (e.g., organizational units; cf. Fig. 4) to a new one or for splitting an existing entity (e.g., a role) into two new entities. Formally:

Definition 5 (Change Framework for Organizational Models). *Let $\mathcal{E}$ be the set of all entities over a set of identifiers and let $\mathcal{R}_{\mathcal{E}}$ be the set of all relations over $\mathcal{E}$. Let further OM = (Entities, Relations) be a (correct) organizational model. Table 1 defines basic change operations Δ, which transform OM into another (correct) organizational model OM':=(Entities', Relations'). In Table 2, in addition, two high–level change operations are given, whose definition is based on the basic change operations from Table 1.*

Table 1. *Basic Change Operations on Organizational Models*

`CreateEntity:` $\mathcal{OM} \times Identifier \times EntityType \mapsto \mathcal{OM}$ with `CreateEntitiy(OM, eId, entType) = OM'`	
Preconditions:	• (eId, entType) $\notin$ Entities
Postconditions:	• Entities' = Entities $\cup$ {(eId, entType)} • Relations' = Relations
`DeleteEntity:` $\mathcal{OM} \times \mathcal{E} \mapsto \mathcal{OM}$ with `DeleteEntity(OM, e) = OM'`	
Preconditions:	• e $\in$ Entities • $\nexists$ rel = (e1, e2, relType) $\in$ Relations with e1 = e $\vee$ e2 = e
Postconditions:	• Entities' = Entities $\setminus$ {e} • Relations' = Relations
`CreateRelation:` $\mathcal{OM} \times \mathcal{E} \times \mathcal{E} \times RelType \mapsto \mathcal{OM}$ with `CreateRelation(OM, e1, e2, relType) = OM')`	
Preconditions:	• e1 := (eId1, eType1), e2 := (eId2, eType2) $\in$ Entities • (e1, e2, relType) $\in \mathcal{R}$ • (e1, e2, relType) $\notin$ Relations
Postconditions:	• Entities' = Entities • Relations' = Relations $\cup$ {(e1, e2, relType)}
`DeleteRelation:` $\mathcal{OM} \times \mathcal{R}_{\mathcal{E}} \mapsto \mathcal{OM}$ with `DeleteRelation(OM, relation) = OM'`	
Preconditions:	• relation $\in$ Relations
Postconditions:	• Entities' = Entities • Relations' = Relations $\setminus$ {relation}
`ReAssignRelaton:` $\mathcal{OM} \times \mathcal{R}_{\mathcal{E}} \times \mathcal{E} \times \mathcal{E} \mapsto \mathcal{OM}$ with `ReAssignRelation(OM, r, e, eNew) = OM'`	
Preconditions:	• r = (e1, e2, relType) $\in$ Relations • e = e1 $\vee$ e = e2 • eNew := (eIdNew, eTypeNew) $\in$ Entities • e = e1:=(eId1, eType1) $\Longrightarrow$ eTypeNew = eType1 • e = e2:=(eId2, eType2) $\Longrightarrow$ eTypeNew = eType2 • e = e1:=(eId1, eType1) $\Longrightarrow$ (eNew, e2, relType) $\notin$ Relations • e = e2:=(eId2, eType2) $\Longrightarrow$ (e1, eNEw, relType) $\notin$ Relations
Postconditions:	• e = e1 $\Longrightarrow$ Relations' = Relations $\cup$ {(eNew, e2, relType} $\setminus$ {(e1, e2, relType} • e = e2 $\Longrightarrow$ Relations' = Relations $\cup$ {(e1, eNew, relType} $\setminus$ {(e1, e2, relType}

A new relation (of type $relType$) between two entities $e1$ and $e2$ of an organizational model to $OM = (Entities, Relations)$, for example, can be created by applying the basic operation `CreateRelation(OM, e1, e2, relType)` to OM. The pre–conditions associated with this operation then ensure that both entities $e1$ and $e2$ are actually present in OM and that the relation $(e1, e2, relType)$ is a valid relation not yet present in OM. The post–condition of this operation, in turn, describes the effects resulting from the application of the operation to OM (in our example, relation $(e1, e2, relType)$ is added to the set $Relations$, whereas the set $Entities$ remains unchanged).

When transforming an organizational model OM into another model OM' one must be able to decide which access rules defined on OM can be *directly migrated* to OM', i.e., which rules can be immediately re–linked to the new model version without need for adaptation. Intuitively, this is the case for access rules which are also valid on OM' (cf. Theorem 1).

Theorem 1 (Direct Migration of Access Rules). *Let OM = (Entities, Relations) be a (correct) organizational model. Let further* `AR` *be a valid access rule based on OM (i.e., Valid(OM,* `AR`*) =* `True`*) and let* Δ *be a (basic or high–level) change operation which transforms OM into another (correct) organizational model OM'. Then* `AR` *can be directly migrated to OM' if Valid(OM',* `AR`*) =* `True`*.*

Table 2. *High–Level Change Operations on Organizational Models*

`JoinEntities`: $\mathcal{OM} \times \mathcal{E} \times \mathcal{E} \times Identifiers \mapsto \mathcal{OM}$ with `JoinEntities(OM, e1, e2, nId) = OM'`	
Preconditions:	• e1= (eId1, eType), e2 = (eId2, eType) $\in$ Entities • (nId, eType) $\notin$ Entities • eType $\neq$ `Actor`
Basic Change Operations:	• CreateEntity(OM, (nId, eType)), eNew := (nId, eType) • $\forall$ (e, e1, relType) $\in$ Relations: ReassignRelation(OM, (e, e1, relType), e1, eNew) • $\forall$ (e, e2, relType) $\in$ Relations: ReassignRelation(OM, (e, e2, relType), e2, eNew) • $\forall$ (e1, e, relType) $\in$ Relations: ReassignRelation(OM, (e1, e, relType), e1, eNew) • $\forall$ (e, e2, relType) $\in$ Relations: ReassignRelation(OM, (e, e1, relType), e2, eNew) • DeleteEntity(OM, e1) • DeleteEntity(OM, e2)
`SplitEntity`: $\mathcal{OM} \times \mathcal{E} \times \mathcal{E} \times \mathcal{E} \mapsto \mathcal{OM}$ with `SplitEntity(OM, eOld, e1, e2) = OM'`	
Preconditions:	• (eIdOld, eType) := eOld $\in$ Entities • (e1Id, eType) := e1, (e2Id, eType) := e2 $\notin$ Entities • eType $\neq$ `Actor`
Basic Change Operations:	• CreateEntity(OM, e1) • CreateEntity(OM, e2) • Reassigment of Relations $\longrightarrow$ manually done by users • DeleteEntity(OM, eOld)

The post conditions of the high–level changes result from the aggregation of the post conditions of the applied basic change operations.

Table 3. *Preliminary Considerations – Adaptation of Access Rules*

Let Δ be a change operation (cf. Def. 5) which transforms a (correct) org. model OM into another (correct) org. model OM'.

Δ	**Necessary Adaptations**
`CreateEntity(OM, e, eType)`	I, none
`CreateRelation(OM, e1, e2, relType)`	II, VAS may become bigger for EAR or smaller for NEAR (= $\emptyset$); no dangling references within AR
`DeleteRelation(OM, rel)`	II, VAS may become smaller for EAR (= $\emptyset$) or smaller for NEAR ; no dangling references within AR
`ReassignRelation(OM, rel, e, eNew)`	II, VAS may become smaller (= $\emptyset$) or bigger; no dangling references within AR
`DeleteEntity(OM, e)`	IV, VAS may become smaller (= $\emptyset$) for EAR or bigger for NEAR; dangling references within AR possible
`joinEntities(OM, e1, e2, eNew)`	IV, actor set changes, dangling references possible
`splitEntity(OM, e, e1, e2)`	IV, distribution of actor set is user dependent, dangling references possible

Regarding basic change operations the direct migration of access rules from OM to OM' is only possible in connection with the operation `CreateEntity(OM, ...)`, i.e., when adding new entities to OM (cf. Lemma 1).

Lemma 1 (Direct Migration of Access Rules). *Let OM be a (correct) organizational model and let* `AR` *be a valid access rule on OM (i.e., Valid(OM,* `AR`*)* = `True`*). Let further* Δ *be a change operation which transforms OM into another (correct) organizational model OM'. Then* `AR` *can be directly migrated (re-linked) to OM', i.e., Valid(OM',* `AR` *)* = `True` *if* Δ = `CreateEntity(OM, ...)`.

When creating a new entity, we can always guarantee that there will be no dangling references within existing access rules and that this change is invariant regarding the set of valid actors (cf. Table 3). If an access rule `AR` cannot be directly transferred to the changed organizational model there may be two reasons for that. Either there are dangling references (e.g., due to the fact that an entity to which `AR` refers has been deleted from OM) or the set of valid actors becomes empty for `AR` on OM. The following lemma states for which basic change operations we can guarantee that there will be no dangling references within existing rules after a change (cf. Table 3).

Lemma 2 (No Dangling References). *Let OM be a (correct) organizational model and let* `AR` *be a valid access rule on OM (i.e., Valid(OM,* `AR`*)* = `True`*). Let further* Δ *be a change operation which transforms OM into another (correct) organizational model OM'. Then:*
DanglingRef(OM', `AR`*)* = `False` *if*
$\Delta \in$ {`CreateEntity(OM,...), CreateRelation(OM,...), DeleteRelation(OM,...), ReAssignRelation(OM,...)`}.

For all other basic and high–level change operations, i.e., for changes $\Delta \in$ {`DeleteEntity(OM,...), JoinEntities(OM,...), SplitEntity(OM,...)`}, their application to an organizational model *OM* may result in dangling references within the set of existing access rules (cf. Table 3).

In order to keep the total set of access rules consistent and the respective security component correctly running after applying model changes, we have to adapt those access rules that are no longer valid. Due to the potentially high number of rules that may exist in the system we want to assist the user as much as possible in accomplishing this task. In particular, we aim at the (semi–) automated migration and transformation of access rules in order to adapt them to model changes (if possible and meaningful). With 'semi-automated' we mean that the system shall assist the user in an adequate way, e.g., by exploiting the semantics of the applied change operation(s) and by making suggestions about potential rule transformations.

Theorem 2 indicates which rule adaptations can be automatically derived and suggested to the user in connection with the two high–level change operations presented before (cf. Table 2). In particular, Theorem 2 summarizes adaptation policies that can be applied to access rules containing dangling references. Doing so, our approach makes use of the semantics of the applied changes operations. Note that the derived policies only constitute suggestions, i.e., users may apply another strategy if more favorable.

Theorem 2 (Adaptation of Access Rules).
Let OM = (Entities, Relations) be a (correct) organizational model and let AR *be a valid access rule on OM. Let further Δ be a change operation which transforms OM into another (correct) model OM'. Then:* AR *can be transformed into a valid access rule* AR' *on OM' by applying adaptation rule δ_{AR} (see below) if $\Delta \in$* {JoinEntities(OM, ...), SplitEntity(OM, ...)}. *For respective Δ adaptation rule δ_{AR} turns out as follows:*

- Δ = JoinEntities(OM, e1, e2, newE) $\Longrightarrow \delta_{AR}$:
  $\forall$ EAR *in* AR *with* EAR:= (EntityType = e1) $\vee$ EAR:= (EntityType = e2)
  replace EAR *by* EAR':= (EntityType = newE)) $\wedge$
  $\forall$ EAR *in* AR *with* EAR:= (EntityType = e1(+)) $\vee$ EAR:= (EntityType = e2(+))
  replace EAR *by* EAR':= (EntityType = newE(+)))
- Δ = SplitEntity(OM, e, e1, e2) $\Longrightarrow \delta_{AR}$:
  $\forall$ EAR *in* AR *with* EAR:= (EntityType = e)
  replace EAR *by* EAR:= ((EntityType = e1) OR (EntityType = e2)) $\wedge$
  $\forall$ EAR *in* AR *with* EAR:= (EntityType = e(+))
  replace EAR *by* EAR:= ((EntityType = e1(+)) OR (EntityType = e2(+)))

As an example consider the change scenario depicted in Fig. 1. Take access rule AR_{OM}:= ((OrgUnit = treatment area) AND (Role = assistant)) and change Δ. For the first change operation joinEntities(PM, (treatment area, OrgUnit), (administration, OrgUnit), (patient services, OrgUnit)) the transformation described by Theorem 2 is applied and the access rule is transformed into $AR'_{OM'}$ = ((OrgUnit = patient services) AND (Role = assistant)). According to Theorem 2 deleting the two relations does not have any effect on the access rule. The deletion of entity (Hunter, Actor) is uncritical since the set of valid actors for AR' on OM' is non-empty.

Note that for join, split, and delete operations access rule transformations do not always become necessary. If an access rule does not refer to any entity joined, deleted, or splitted, the rule can stay unaltered after the respective model transformation. Finally, in addition to the described rule transformations in our current implementation we apply a number of other rule optimizations when migrating rules to a new version of the organizational model. The treatment of these optimizations, however, is outside the scope of this paper.

Our approach also deals with the challenging question of how to adapt access rules when entities are deleted from OM. As already mentioned this might lead to dangling references depending on the nature of the respective access rules. In certain cases no automatic strategy for adapting a particular access rule can be provided; the system then only reports the problem to the user and asks him for an adequate solution strategy. However, there are also many cases where automatic adaptations become possible, and thus users can be assisted in transforming rules in a way such that they become valid on the new model version OM' as well. In particular, this possibility exists in connection with the migration of composed access rules. As an example take access rule (AR $\longleftarrow$ Role = R1 $\vee$ Role = R2). If role R2 is deleted from the underlying organizational model this causes a dangling reference in AR. However, a suggestion for an automatic

adaptation would be to delete `EAR` ⟵ `Role = R2` from `AR` which results in the following rule: `AR` ⟵ `Role = R1`. This access rule does note contain dangling references and it remains resolvable on OM'.

5 Practical Issues

To illustrate our results we apply them to an important use case of cooperative information systems – the adaptation of actor assignments in workflow management systems. More precisely we sketch how organizational changes are handled in the ADEPT2 process management system (PMS) [27] and how the different system components interact with each other to cope with model changes. Besides dynamic adaptations of organizational models ADEPT2 provides sophisticated support for process schema evolution [28] and process instance changes [7]. These kinds of process changes have been subject of previous publications on ADEPT and are outside the scope of this paper.

In the ADEPT2 buildtime environment, organizational models can be created and modified by the use of a graphical editor. This tool, whose visualization component is based on SVG (Scalable Vector Graphics), supports users in graphically defining organizational models. Model changes can be accomplished with the same tool and are based on the operations presented in this paper. All changes are logged by a respective system component. Besides this organization modeling component another tool exists, which allows users to define elementary and complex access rules based on the current version of the organizational model. Only such rules can be expressed which are syntactically and semantically correct. Furthermore, this rule editor is realized as plug-in which can be used within different client applications (in our case, for example, the workflow editor makes use of this component for defining actor assignments).

Consider the scenario depicted in Fig. 5. When an organizational change occurs authorized users can adapt the organizational model accordingly. In this case a new version of the organizational model is created which, in turn, triggers the migration and adaptation of related access rules. Rules which can be directly accessed by the change manager are immediately checked and migrated to the new model version. Rules which are kept outside the ADEPT2 system (e.g., access rules within a document management systems), however, require lazy migration techniques and more advanced mechanisms as well. In ADEPT2, any change of the organizational model immediately triggers the migration and adaptation of the access rules maintained within the ADEPT2 system. These rules include, for example, actor assignments for process activities (i.e., execution rights for working on these activities [24]) as well as privileges for changing process models and/or process instances [14].

In our scenario from Fig. 5, for example, a change of the depicted organizational model may require the adaptation of actor assignments within a process model. ADEPT2 indicates necessary adaptations to the process engineer who then can perform the respective changes at the process model level. Thereby we make use of the conceptual framework presented in this paper.

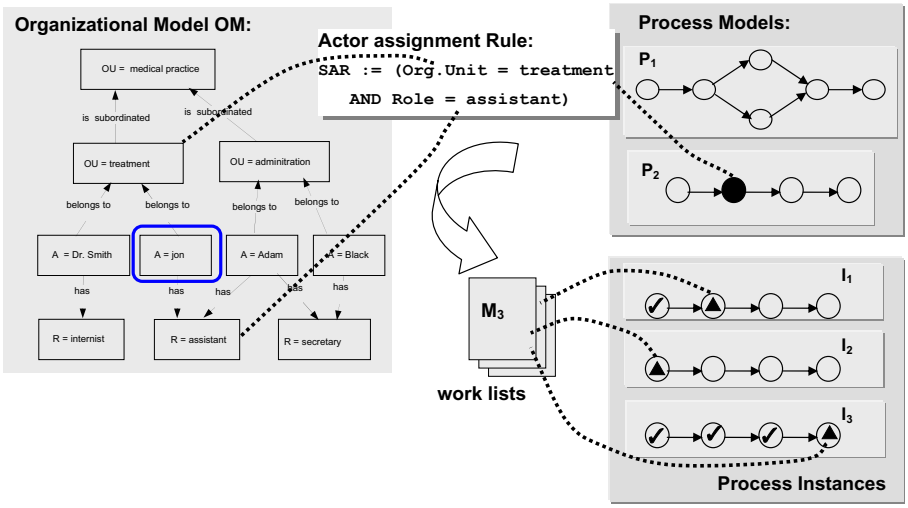


Fig. 5. Change Scenario in Process Management Systems

The needed rule adaptations (i.e., adaptations of actor assignments) are first carried out at the process model level. For long-running processes it might also become necessary to propagate these model changes (i.e., the adaptations of activity actor assignments) to already running process instances. This is only possible for process instances that are compliant with the current process model change. For example, when activities with modified actor assignment have not yet been activated such a change propagation is possible in ADEPT2. Finally, the described model and instance changes may also require the update of user worklists. This is one of the biggest challenges when thinking of realistic scenarios with ten thousands up to millions of work items. Worklist adaptations, however, are outside the scope of this paper. An overview of the scenarios supported in connection with changes of organizational models is given in Table 4.

Fig. 6 gives an overview of the different system components of ADEPT2 and also indicates the complexity arising from the design and implementation of adaptive process management technology. In the scenario described above the following components are involved: Editor, ChangeMgr, WorklistMgr, OrgModelMgr, AccessControlMgr, and LogMgr. Due to lack of space we omit a description of the architecture of this system and refer the interested reader to [27].

Table 4. *Possible Scenarios when Changing the Organizational Model*

	Valid Actor Set	
Access Rule	*unchanged*	*changed*
not directly affected	I	II (adapt worklists)
directly affected	III	IV

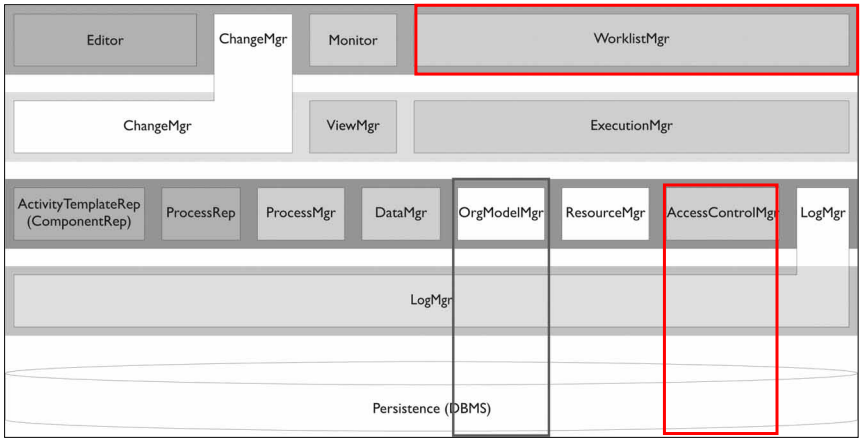


Fig. 6. ADEPT2 system architecture (abstract view)

6 Summary and Outlook

Both the controlled evolution of organizational models and the correct adaptation of access rules will be key ingredients of next generation enterprise security systems, ultimately resulting in highly adaptive access control models. Together with our complementary work on process evolution and dynamic process changes [7, 9, 29, 28] the presented concepts contribute to a platform enabling the realization of highly flexible and adaptive, cooperative information systems.

In this paper we have focussed on the common support of changes on organizational models and on the necessary adaptations of related access rules. We have discussed important challenges and requirements for the evolution of organizational models as well as the limitations of current approaches. The very important aspect of our work is its formal foundation. We have given precise definitions and formal theorems which are fundamental for the correct handling of model changes and adaptations of corresponding access rules. The treatment of both elementary and composed access rules adds to the overall completeness of our approach. Finally, in our ADEPT2 project a powerful proof-of-concept prototype has been implemented, which demonstrates the feasibility of the presented concepts. This prototype even uses a more expressive meta model to describe organizational structures in a compact and user-friendly way. Furthermore it considers the dynamic adaptation of actor assignments in process management systems and necessary worklist updates as well.

There are many other challenging issues related to changes of organizational models and of related access rules. First, we believe that respective changes must be closely linked with other components of cooperative information systems. For example, actor assignments in workflow–based applications may have to be adapted on–the–fly in order to cope with organizational changes. This, in turn, may require change propagation to hundreds up to thousands of in-progress

process instances as well as to related user worklists. Doing this in a correct and efficient manner is a non-trivial problem that will be investigated by us in more detail in future. Finally, changes may not only concern the process model or the organizational model but other components of the cooperative information systems as well. As an example take resource models or data models, which may be also subject of change.

References

1. v.d. Aalst, W., van Hee, K.: Workflow Management. MIT Press (2002)
2. Sutton, M.: Document Management for the Enterprise: Principles, Techniques and Applications. John Wiley (1996)
3. Linthicum, D.: Enterpise Application Integration. Addison-Wesley (1999)
4. Konyen, I.: Organizational structures and business processes in hospitals. Master's thesis, University of Ulm, Computer Science Faculty (1996) (in German).
5. Jablonski, S., Schlundt, M., Wedekind, H.: A generic component for the computer–based use of organizational models (in german). Informatik Forschung und Entwicklung **16** (2001) 23–34
6. Klarmann, J.: A comprehensive support for changes in organizational models of workflow management systems. In: Proc. 4th Int'l Conf. on Inf Systems Modeling (ISM'01). (2001) 375–387
7. Reichert, M., Dadam, P.: ADEPT$_{flex}$ - supporting dynamic changes of workflows without losing control. JIIS **10** (1998) 93–129
8. Rinderle, S., Reichert, M., Dadam, P.: On dealing with structural conflicts between process type and instance changes. In Desel, J., Pernici, B., Weske, M., eds.: Proc. 2nd Int'l Conf. on Business Process Management (BPM'04). LNCS 3080, Potsdam, Germany (2004) 274–289
9. Rinderle, S., Reichert, M., Dadam, P.: Disjoint and overlapping process changes: Challenges, solutions, applications. In: Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'04). LNCS 3290, Agia Napa, Cyprus (2004) 101–120
10. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – a survey. Data and Knowledge Engineering, Special Issue on Advances in Business Process Management **50** (2004) 9–34
11. Reichert, M., Rinderle, S., Dadam, P.: On the common support of workflow type and instance changes under correctness constraints. In: Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'03). LNCS 2888, Catania, Italy (2003) 407–425
12. Bertino, E.: Data security. DKE **25** (1998) 199–216
13. zur Muehlen, M.: Resource modeling in workflow applications. In: Proc. of the 1999 Workflow Management Conference (Muenster). (1999) 137–153
14. Weber, B., Reichert, M., Wild, W., Rinderle, S.: Balancing flexibility and security in adaptive process management systems. In: Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'05), Agia Napa, Cyprus (2005)
15. Ferraiolo, D., Kuhn, D., Chandramouli, R.: Role–Based Access Control. Artech House (2003)
16. NIST: Proposed Standard for Role-Based Access Control. http://csrc.nist.gov/rbac/rbacSTDACM.pdf (2004)
17. Ferraiolo, D., Kuhn, D.: Role based access control. In: 15th National Computer Security Conference. (1992)

18. Sutton, M.: Document Management for the Enterprise – Principles, Techniques, and Applications. Wiley Computer Publ., New York (1996)
19. Botha, R., Eloff, J.: A framework for access control in workflow systems. Information Management and Computer Security. **9** (2001) 126–133
20. Bertino, E., Ferrari, E., Alturi, V.: The specification and enforcement of authorization constraints in wfms. ACM Trans. on Inf. and Sys. Sec. **2** (1999) 65–104
21. Wainer, J., Barthelmess, P., Kumar, A.: W–RBAC – a workflow security model incorporating controlled overriding of constraints. International Journal of Collaborative Information Systems **12** (2003) 455–485
22. Klarmann, J.: A comprehensive support for changes in organizational models of workflow management systems. In: Proc. Int'l Conf. on Information Systems Modeling (ISM'01), Hradec nad Moravici, Czech Republic (2001)
23. Domingos, D., Rito-Silva, A., Veiga, P.: Authorization and access control in adaptive workflows. In: Proc. Europ. Symposium on Research in Computer Science (ESORICS'03), Gjovik, Norway (2003) 23–28
24. Berroth, M.: Design of a component for organizational models. Master's thesis, University of Ulm, Computer Science Faculty (2005) (in German).
25. v.d. Aalst, W., Jablonski, S.: Dealing with workflow change: Identification of issues an solutions. Int'l Journal of Comp. Systems, Science and Engineering **15** (2000) 267–276
26. Klarmann, J.: Using conceptual graphs for organization modeling in workflow management systems. In: Proc. Conf. Professionelles Wissensmanagement (WM'01). (2001) 19–23
27. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with adept2. In: Proc. 21st Int'l Conf. on Data Engineering (ICDE'05), Tokyo (2005) 1113–1114
28. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases **16** (2004) 91–116
29. Rinderle, S., Weber, B., Reichert, M., Wild, W.: Integrating process learning and process evolution - a semantics based approach. In: 3rd Int'l Conf. on Business Process Management (BPM'05), Nancy, France (2005)