A Formal Framework for Adaptive Access Control Models

Stefanie Rinderle[1] and Manfred Reichert[2]

[1] Department Databases and Information Systems, University of Ulm, Germany
stefanie.rinderle@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 organizational 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 dangling references or non-resolvable actor assignments). Current approaches do not adequately address this problem, which often leads to security gaps and delayed change implementation.In this paper we introduce a formal framework for the controlled evolution of organizational models and related access rules. Firstly, we introduce a set of operators with well-defined semantics for defining and changing organizational models. Secondly, we show how to define access rules based on such models. In this context we also define a notion of correctness for access rules. Thirdly, we present a formal framework for the (semi-automated) adaptation of access rules when the underlying organizational model is changed by exploiting the semantics of the applied changes. Altogether the presented approach provides an important contribution for realizing adaptive access control frameworks.

1 Introduction

Enterprise information systems 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 respective system services different middleware exists, including workflow management technology, document management systems, and tools for enterprise application integration [1,2,3].

S. Spaccapietra et al. (Eds.): Journal on Data Semantics IX, LNCS 4601, pp. 82–112, 2007.

1.1 Problem Description

Controlled access to its application services as well as to the application objects managed by them (e.g., business processes, documents, resources, or application systems) constitutes an important task for any information system (IS) [4,5,6,7,8]. This results in a large number of *access rules* covering different system aspects and user privileges [9]. Usually, these access rules have to be frequently adapted due to changes of organizational structures [10,11,12]. Such changes become necessary, for instance, when an organizational unit is split into two sub-units, two existing units are joined to a new one, a group of users is reassigned to a new organizational unit, or simply an employee leaves the organization.[1] As a consequence, access rules whose definition refers to organizational entities may have to be modified as well. We denote the ability of an enterprise IS to adapt access rules after organizational model changes as *adaptive access control*.

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, hierarchical relations between organizational units) is kept in an *organizational model*. Based on such a model, access rights and user privileges (e.g., actor assignments in workflow systems or access rules in document-centered applications) can be defined (cf. Fig. 1). Consequently, when organizational changes occur, both the organizational model and related access rules have to be adapted in a correct and consistent manner. The focus of this paper is on the correct handling of the evolution of organizational models and related access rules.

Another problem arises from the fact that the (middleware) components used to build the application services of information systems 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 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. In this paper, however, we abstain from issues related to this heterogeneity problem.

The correct evolution of an organizational model is only one side of the coin when dealing with organizational changes; the other one is to correctly and efficiently adapt the access rules whose definition is based on this organizational model. Note that in large environments hundreds up to thousands of access rules may exist, each of them capturing different privileges of the IS. This, in turn, makes it a hard job for the system administrator to quickly and correctly adapt access rules to model changes. Current approaches do not sufficiently deal with this issue. They neither exploit the semantics of the applied model changes nor do they provide automated support for adaptating access rules and for migrating them to the changed organizational model. In practice, this often leads to problems like non-resolvable actor assignments, unauthorized access

[1] For respective results from one of our case studies in the clinical domain see [10].

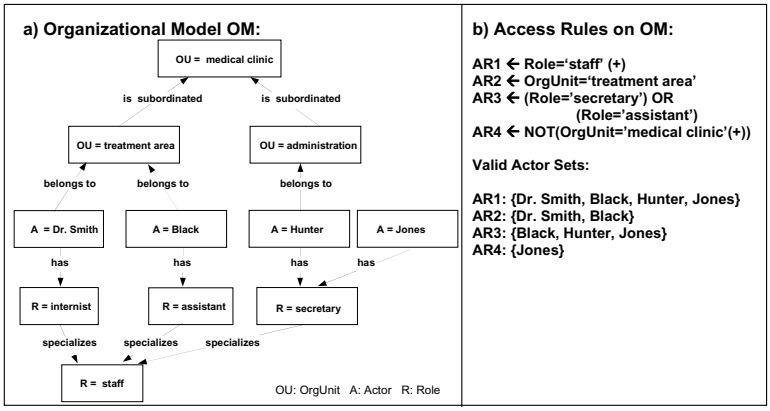

Fig. 1. Organizational Model and Related Access Rules (simplified)

to documents, or inconsistent user worklists. Assume, for example, that two organizational units are joined to make the enterprise more efficient (cf. Fig. 3). If this change is performed in an uncontrolled manner, orphaned (dangling) references may result; i.e., access rules referring to org. entities which are no longer present in the new organizational model. Even more critical might be cases where changes of an organizational model lead to access rules for which no actor qualifies any more. In process–aware information systems [13], for example, such non-resolvable actor assignments lead to tasks which cannot be processed and therefore have to be forwarded to the system administrator. As a consequence, business process execution may be delayed and security gaps may arise.

To deal with these challenges we need an enterprise security service which manages the organizational model as well as its evolution in a consistent and correct manner. Furthermore, model changes have to be efficiently propagated to access rules without causing inconsistencies or security gaps. Finally, we have to consider *passive* access rules, which are checked when a certain privilege is applied (e.g., at the moment a user wants to access a document), as well as *active* access rules used to determine a set of potential users before accessing an object or task (e.g., to create work items for user worklists in workflow systems).

Altogether these tasks are non-trivial. Both organizational models and access rules may have complex structure, and we have to analyze and understand the interdependencies between changes of an organizational model and necessary adaptations of related access rules. This necessitates a framework with precise and formal semantics for reasoning about model and rule changes.

1.2 Contribution

In this paper, we present a formal framework for the controlled evolution of organizational models and related access rules. Firstly, we introduce a meta model and a set of operators with well-defined semantics for defining and changing organizational models. Secondly, we show how to define access rules based on such

models. We provide a precise semantics for access rules and introduce a notion of correctness for them. These are important pre-conditions for reasoning about rule changes. Thirdly, we present a formal framework for the (semi-automated) adaptation of access rules when changing the related organizational model. For selected organizational changes we show how they can be realized in our formal framework, how their effects on access rules look like, and how these access rules can be migrated to the new version of the organizational model. Thereby we make use of the semantics of model changes and we introduce formally sound migration concepts. Altogether the presented approach provides an important contribution for realizing adaptive enterprise access control frameworks.

In [14] we have already introduced first results on adaptive access control (i.e., a criterion for correctness of access rules and exemplary strategies for avoiding dangling references in such rules after model changes). This paper extends this work in several directions: On the one hand, we elaborate these previous results (e.g., by considering more complex access rules and model changes, or by providing more details on architectural issues). On the other hand, as completely new results, the effects of organizational changes on actor sets are evaluated. For example, we deal with the challenging question when actor sets become empty after model changes. The remainder of this paper is organized as follows: Section 2 introduces our framework for defining and changing organizational models. Section 3 shows how to define access rules based on this framework, and Section 4 illustrates how to adapt access rules to model changes. Architectural and implementation issues are sketched in Section 5. Section 6 discusses related work and Section 7 concludes with a summary and an outlook on future work.

2 Framework for Creating and Evolving Organizational Models

In order to be able to analyze changes of organizational models as well as their impact on related access rules we need a formalization of organizational structures; i.e., a formal description of organizational entities and the relations between them. Based on such a formalization it should be possible to specify changes and their operational semantics. For this purpose, first of all, we introduce a meta model for defining organizational structures, which is comparable to the meta models current access control models are based on (e.g., [6,15,16]). In this paper we restrict our considerations to the basic entity types *organizational unit*, *role* and *actor* (cf. Fig. 2), and to the particular relation types existing between respective entities (e.g., actor A_1 belongs to organizational unit O_1, actor A_1 has role R_1, role R_1 specializes role R_0, etc.). In the overall framework, we are currently realizing in the ADEPT2 project [17], we additionally consider entity types like *position*, *group*, and *capability* (see [18] for details). However, in this paper we omit these entity types in order to better focus on core issues related to the evolution of organizational models and related access rules.

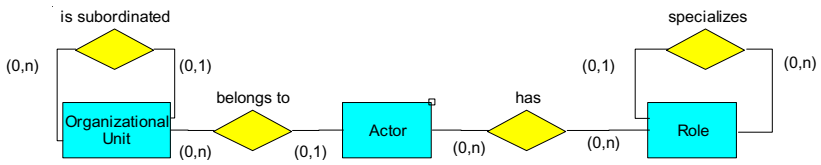

Fig. 2. Organizational Meta Model (in ER Notation)

Regarding the meta model OMM used in this paper (cf. Fig. 2) we specify the set of valid entity types *EntityTypes* and the set of valid relation types *RelationTypes* 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 tuples that can be used to define relations 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., business 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. Furthermore, an actor can play different roles: A physician in a hospital, for example, may possess the two roles *ward doctor* and *radiologist*. Actors with same role are being considered as interchangeable. 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 hospital may have an intensive care unit and an emergency laboratory as subordinated units). Based on this meta model we can define the notion of *organizational model* (cf. Def. 1). For the sake of readability, we do not consider the cardinalities associated with the relation types of our meta model.

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 ⊆ $\mathcal{E}$ and Relations ⊆ $\mathcal{R}_{\mathcal{E}}$ such that

- *all entity identifiers are used in a unique way*
- *there are no cyclic dependencies between roles (relation* `specializes`*) or between organizational units (relation* `is subordinated`*), formally:*
  - $\forall$ *(role, Role)* $\in$ *Entities: (role, Role)* $\notin$ *Spec(OM, (role, Role)) with* $Spec(OM, el) := \bigcup_{el':(el',el,specializes)\in Relations} (\{(el', Role)\} \cup Spec(OM, el'))$
  - $\forall$ *(ou, OrgUnit)* $\in$ *Entities: (ou, OrgUnit)* $\notin$ *Sub(OM, (ou, OrgUnit)) with* $Sub(OM, el) := \bigcup_{el':(el',el,issubordinated)\in Relations} (\{(el', OrgUnit)\} \cup Sub(OM, el'))$

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

As it can be seen from Def. 1 we define a notion of *correctness* imposed on organizational models. It is based on different correctness constraints in order to exclude undesired effects when creating and changing such models. For example, a unique usage of entity identifiers is claimed. Another constraint refers to the exclusion of cyclic dependencies between roles (relation `specializes`) as well as cyclic dependencies between organizational units (relation `belongs to`) due to their unclear semantics. The definition of further correctness constraints depends on the particular application scenario and is omitted in this paper.

In order to be able to express all relevant kinds of changes on an organizational model OM our framework provides a complete set of basic change operations; e.g., for creating or deleting organizational entities and the relations between them. For each change operation we define formal pre– and post–conditions, which preserve the correctness properties of OM when applying the operation(s) to this model (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 model changes. Examples for such high-level operations include the join of two entities (e.g., fusion of two organizational units; cf. Fig. 3) or the split of an existing entity into two new entities (e.g., a role; cf. Fig. 3).

Definition 2 (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 which can be transformed into another (correct) organizational model OM' := (Entities', Relations') by applying change (transaction)* $\Delta = op_1, ..., op_n$*. The notion* $\Delta = op_1, ..., op_n$ *describes the sequential application of basic (cf. Tab. 1) or high-level (cf. Tab. 2) change operations* $op_1, ...op_n$ *to OM. This sequence of change operations is encapsulated within change (transaction)* Δ*.*

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

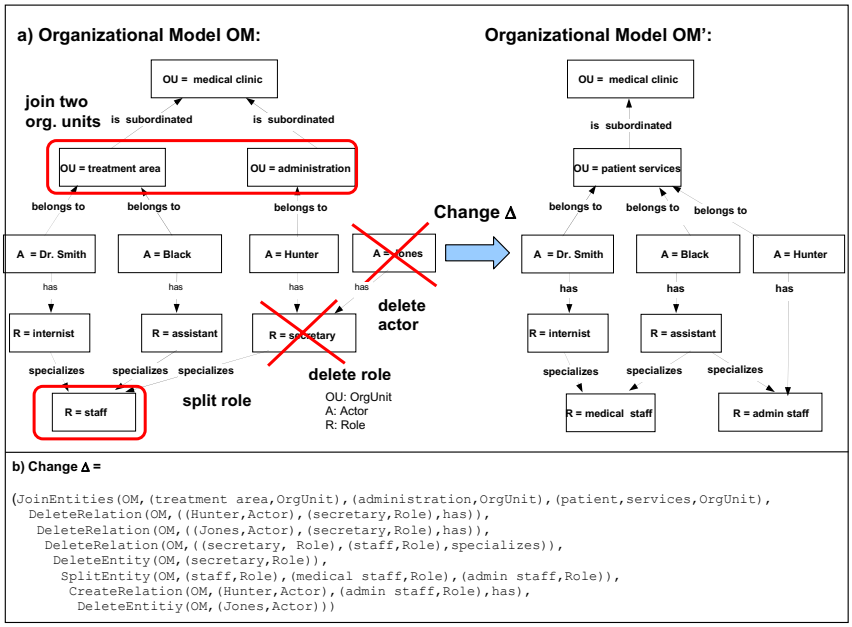

Fig. 3. Structural Change of the Organizational Model OM from Fig. 1

Table 2 contains high–level change operations which can be realized by applying a sequence of basic change operations. The purpose of these high–level operations is to better assist users in defining complex, but common changes. In this paper we consider the operations for reassigning existing relations, for joining two entities (e.g., two organizational units), and for splitting entities (e.g., roles). An example for joining two organizational units `treatment area` and `administration` to the new unit `patient services` is depicted in Fig. 3.

3 Framework for Defining (Correct) Access Rules

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

Let $OM = (Entities, Relations)$ be an organizational model. Based on the entities and relations defined by OM we can specify rules for controlling the access to processes, documents, or other objects. Since the structuring as well as the semantics of these access rules is fundamental for the (semi-) automated derivation of rule adaptations after model changes, we consider this issue in more detail. We distinguish between elementary and complex access rules.

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)}
	• for eType1=eType2 = Role $\wedge$ relType = `specializes`: e1 $\notin$ Spec(OM, e2)$^{\Theta}$
	• for eType1=eType2 = OrgUnit $\wedge$ relType = `is subordinated`: e1 $\notin$ Sub(OM, e2)
`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}

$^{\Theta}$ For a formal definition of *Spec* and *Sub* see Definition 1

An *elementary access rule* (cf. Def. 3) 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 = 'medical clinic'`. For this access rule we denote the organizational unit `medical clinic` as the *qualifying entity*. Furthermore, all actors belonging to this unit own the privileges associated with this rule.

For entities that can be hierarchically organized (i.e., for organizational units and roles in our meta model) we further support the definition of *transitive* elementary access rules. As an example consider the elementary access rule `OrgUnit = medical clinic(+)`. For this transitive rule (indicated by the '`+`') the set of qualifying entities comprises the organizational unit `medical 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 elementary rule.

Similar considerations can be made regarding the 'specializes' relation between entities of type `Role`.

Definition 3 (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` | `EAR2` | `EAR3` *with*

`EAR1` $\longleftarrow$ `(EntityType = el)`, `EAR2` $\longleftarrow$ `(OrgUnit = el(+))`, `EAR3` $\longleftarrow$ `(Role = el(+))`

Table 2. High-Level Change Operations on Organizational Models

`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) ∈ Relations • e = e1 ∨ e = e2 • eNew := (eIdNew, eTypeNew) ∈ Entities • e = e1:=(eId1, eType1) ⟹ eTypeNew = eType1 • e = e2:=(eId2, eType2) ⟹ eTypeNew = eType2 • e = e1:=(eId1, eType1) ⟹ (eNew, e2, relType) ∉ Relations • e = e2:=(eId2, eType2) ⟹ (e1, eNEw, relType) ∉ Relations
Postconditions:	• e = e1 ⟹ Relations' = Relations ∪ {(eNew, e2, relType} \ {(e1, e2, relType} • e = e2 ⟹ Relations' = Relations ∪ {(e1, eNew, relType} \ {(e1, e2, relType} • for e = e1 ∧ eType1 = eType2 = eTypeNew = Role ∧ relType = `specializes`: eNew ∉ pred*(OM, e1) • for e = e2 ∧ eType1 = eType2 = eTypeNew = Role ∧ relType = `specializes`: e2 ∉ pred*(OM, eNew) • for e = e1 ∧ eType1 = eType2 = eTypeNew = OrgUnit ∧ relType = `is subordinated`: eNew ∉ pred*(OM, e1) • for e = e2 ∧ eType1 = eType2 = eTypeNew = OrgUnit ∧ relType = `is subordinated`: e2 ∉ pred*(OM, eNew)
`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) ∈ Entities • (nId, eType) ∉ Entities • eType ≠ `Actor`
Basic Change Operations:	• CreateEntity(OM, (nId, eType)), eNew := (nId, eType) • ∀ (e, e1, relType) ∈ Relations: ReassignRelation(OM, (e, e1, relType), e1, eNew) • ∀ (e, e2, relType) ∈ Relations: ReassignRelation(OM, (e, e2, relType), e2, eNew) • ∀ (e1, e, relType) ∈ Relations: ReassignRelation(OM, (e1, e, relType), e1, eNew) • ∀ (e, e2, relType) ∈ 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 ∈ Entities • (e1Id, eType) := e1, (e2Id, eType) := e2 ∉ Entities • eType ≠ `Actor`
Basic Change Operations:	• CreateEntity(OM, e1) • CreateEntity(OM, e2) • All actors belonging to the splitted org. unit or possessing the role to be splitted have to be assigned to one of the new entities or to both of them • Default behavior$^{\chi}$ for sub-roles: If the entity to split is of type `Role` reassign its sub-roles to both new resulting roles after split. • Default behavior for super-roles: If the entity to split is of type `Role` and has a super-role reassign both resulting roles after split to this super-role. • Default behavior for subordinated org. units: If the entity to split is of type `OrgUnit` reassign its subordinated org. units to exactly one of the new org. units (user decision). • Default behavior for superordinated org. units: If the entity to split is of type `OrgUnit` and has a superordinated org. unit assign both new org. units to this superordinated unit. • 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.
$^{\chi}$ The user may override the default behavior any time.

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 general, the semantics of an access rule (defined on OM) is determined by the set of actors from OM qualifying for this rule (*valid actor set*). Definition 4 presents the valid actor sets for elementary access rules.

Definition 4 (Valid Actor Set for Elementary Access Rules). *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* EAR *be an elementary access rule on OM. Then: Valid actor set VAS(OM,* EAR*) denotes the set of all actors (from OM) who qualify for* EAR*, i.e., who own the privileges associated with rule* EAR*. 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(+)) ⟹

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

In order to enable the definition of more *complex access rules* we allow for the composition of existing rules (cf. Def. 5). For this purpose the following operators can be used: *negation*, *conjunction* and *disjunction*. Def. 5 also sets out a precise semantics for complex access rules based on their valid actor sets.

Definition 5 ((Complex) 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*

– EAR *is an elementary access rule (cf. Def. 3)*

– NEAR ⟵ (NOT (EAR)) *where* EAR *is an elementary access rule*
VAS(OM, NEAR*) = Act(OM) \ VAS(OM,* EAR*)*

- DAR ⟵ (AR1 OR AR2) *with* AR1 *and* AR2 *are access rules*
  VAS(OM, AR*) = VAS(*AR1*) ∪ VAS(*AR2*)*
- CAR ⟵ (AR1 AND AR2) *with* AR1 *and* AR2 *are access rules*
  VAS(OM, AR*) = VAS(*AR1*) ∩ VAS(*AR2*)*

Consider the organizational model OM depicted in Fig. 1a). An example for a complex access rule on OM is the expression AR⟵ (OrgUnit = medical clinic(+) AND Role = assistant) with valid actor set VAS(AR) = {Dr. Smith, Black, Hunter, Jones} ∩ {Black} = {Black}.

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:

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

where the notion EAR ∈ AR describes all elementary access rules EAR contained in access rule AR.

(2) AR is resolvable, i.e., the set of valid actors VAS(OM, AR) does not become empty. We consider this second constraint as an important property of any access control module in order to ensure that objects remain accessible or tasks remain doable. Formally:

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

Note that dangling references or non-resolvable access rules might occur when organizational models are changed in an uncontrolled manner (cf. Fig. 4).

Definition 6 (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 ⟺ *(DanglingRef(OM,* AR*) =* False ∧ *Resolv(OM,* AR*) =* True*)*

As an example consider the change scenario depicted in Fig. 4 where organizational model OM is transformed into another organizational model OM' by applying change Δ (for a formal definition of this change see Fig. 3 b)). Access rule AR1 ⟵ Role='staff(+)' defined on OM would contain a dangling reference when migrating this rule to the new organizational model OM'. The same holds for access rules AR2 and AR3. Access rule AR4 ⟵ NOT(OrgUnit='medical clinic'(+)) is resolvable on OM (VAS(OM',AR4) = {Jones}), but no longer resolvable on OM'. These simple examples demonstrate that uncontrolled changes of an organizational model can lead to security gaps or access errors later on if not treated in an adequate way. In the following section we introduce a formalism for adaptive access control rules in order to avoid such problems.

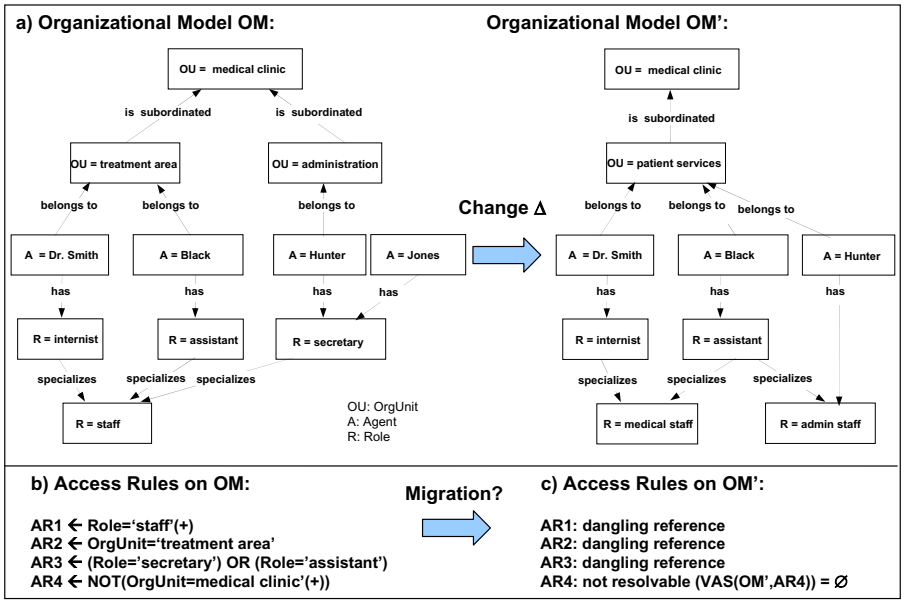

Fig. 4. Changing the Organizational Model OM from Fig. 1 and the Resulting Problem of Migrating Access Rules

4 Impact of Organizational Changes on Access Rules

In this section we introduce our formal framework for realizing adaptive access control models. 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 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. Def. 6). Otherwise, we have to adapt access rules that are no longer valid in order to keep the total set of access rules on the new model version OM' consistent. Due to the potentially large number of access rules to be managed we want to assist users 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 changes of the organizational model if possible. Finding meaningful access rule adaptations is based on exploiting the semantics of the applied change operation(s). With 'semi-automated' we mean that the system shall assist the user in an adequate way, i.e., by explaining the potential conflicts arising after org. model changes (e.g., dangling references) and by making suggestions about potential rule transformations.

In Section 4.1 we provide a general criterion for the correct migration of access rules when changing the organizational model these rules are based on. Section 4.2 deals with the problem of dangling references. In Section 4.3 we

analyze how the valid actor set of an access rule may change when migrating this rule to a modified organizational model.

4.1 Basic Migration Rule

First of all, we provide a general criterion for the correct migration of access rules, which is based on the considerations we made in Section 3:

Axiom 1 (Direct Migration of Access Rules). *Let OM = (Entities, Relations) be a (correct) organizational model and* `AR` *be a valid access rule on OM, i.e., Valid(OM,* `AR`*) =* `True`*. Let further* $\Delta = op_1, ..., op_n$ *be a change (transaction) consisting of a sequence of basic and/or high–level change operations, which transforms OM into another (correct) organizational model OM'. Then:* `AR` *can be directly migrated to OM' if Valid(OM',* `AR`*) =* `True`*.*

As a simple example consider the scenario depicted in Fig. 4a). Assume that access rule `AR5` ⟵ `Role = 'internist'` is defined on OM. When migrating `AR5` to OM' there are no dangling references since entity `internist` is still present in OM'. Further, the actor set of `AR5` remains resolvable over OM' (`VAS(OM',AR5) =` {`Dr. Smith`}). Consequently, `AR5` is a valid access rule on OM' as well (i.e., Valid(OM', `AR`) = `True`) and can therefore be directly migrated to OM' according to Axiom 1.

4.2 Static Aspect – Dangling References

We analyze the problem of dangling references when migrating access rules to a changed organizational model. For the sake of readability, first of all, we consider the application of one single change operation. Following this, we deal with multi-operation changes and their effects on access rules.

Application of Single Change Operations Δ_{op}. We consider a change consisting of one single, basic or high–level change operation $\Delta_{op} := \Delta = op$ applied to an organizational model OM. We analyze the effects of this model change on related access rules, particularly regarding the occurence of dangling references.

As a first important result we can conclude that direct migration of an access rule from OM to OM' (without additional checks) is always possible in connection with change operation `CreateEntity(OM, ...)`(cf. Proposition 1).

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

When creating a new entity and solely adding this entity to OM we can always guarantee that an arbitrary access rule valid on OM will remain valid on the

new model version OM' as well: No dangling references occur and the change is invariant regarding the set of valid actors (of any access rule).

If an access rule AR cannot be directly transferred to the changed org. model OM' there may be two reasons for that. Either there are dangling references (e.g., after deleting an entity from OM to which AR refers) or the set of valid actors becomes empty for AR on OM'. In this section we cope with the first problem. Proposition 2 states for which basic change operations we can guarantee that there will be no dangling references within existing rules after a change.

Proposition 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_{op} be a change operation which transforms OM into another (correct) organizational model OM'. Then: DanglingRef(OM',* AR*) =* False *if* $\Delta_{op} \in$ {CreateEntity(OM,...), CreateRelation(OM,...), DeleteRelation(OM,...), ReAssignRelation(OM,...)}.

The application of all other basic and high–level change operations $\Delta_{op} \in$ {DeleteEntity, JoinEntities, SplitEntity} to an org. model OM may result in dangling references for access rules defined on OM. The challenging question is whether we can adapt respective access rules in a syntactically and semantically correct manner in order to migrate them to the new org. model OM; i.e., no dangling reference must occur after the rule transformation and the derived rule should still be compliant with its original objective.

We have a more detailed look at the two change operations JoinEntities and SplitEntity from Table 2 in order to deal with these questions. When applying one of these high–level change operations to an organizational model, obviously, dangling references within access rules might occur. Adaptation Policy 1 indicates which rule adaptations can be automatically derived in such a case. Particularly, Adaptation Policy 1 makes use of the semantics of these high-level change operations. For example, if two entities $e1$ and $e2$ are joined to a new entity $e3$, resulting dangling references to $e1$ or $e2$ within access rules could be substituted by references to $e3$. At this point it is important to mention that all derived rule adaptations solely constitute suggestions, i.e., users may apply another strategy if more favorable.

Rule Adaptation Policy 1 (Avoiding Dangling References). *Let OM = (Entities, Relations) be a (correct) org. model and let* AR *be a valid access rule on OM. Let further $\Delta_{op} \in$ {JoinEntities(OM, ...), SplitEntity(OM, ...)} be a high-level change operation which transforms OM into another (correct) org. model OM'. Then: When applying adaptation rule δ_{AR} (see below) to* AR *this rule can be transformed into an access rule* AR' *on OM' which does not contain dangling references and which is semantically "close" to* AR*. For respective Δ_{op} the adaptation rule δ_{AR} turns out as follows:*

- Δ_{op} = JoinEntities(OM, e1, e2, newE) $\Longrightarrow \delta_{AR}$:
  ∀ [N]EAR *in* AR *with*
  [N]EAR:= [NOT](EntityType = e1) ∨ [N]EAR:= [NOT](EntityType = e2)
  replace [N]EAR *by* [N]EAR' ≡ [NOT](EntityType = newE) ∧

∀ [N]EAR *in* AR *with*
[N]EAR:=[NOT](EntityType=e1(+)) ∨ [N]EAR:= [NOT](EntityType=e2(+))
replace [N]EAR *by* [N]EAR' ≡ [NOT](EntityType = newE(+))

– Δ_{op} = SplitEntity(OM, e, e1, e2) $\Longrightarrow$ δ_{AR}:
∀ [N]EAR *in* AR *with* [N]EAR:= [NOT](EntityType = e)
replace [N]EAR *by*
[N]EAR ≡ [NOT](EntityType = e1 OR EntityType = e2) ∧
∀ [N]EAR *in* AR *with* [N]EAR:= [NOT](EntityType = e(+))
replace [N]EAR *by*
[N]EAR ≡ [NOT](EntityType = e1(+) OR EntityType = e2(+))

We illustrate these rule adaptations policies by means of examples. Figure 5a shows the join of two organizational units OU1 and OU2 resulting in a new organizational unit OUNew. Access rules AR1 and AR2 on OM refer to one or both of the joined organizational units (cf. Figure 5b). According to Adaptation Policy 1 these access rules could then be adapted by substituting the "old" reference to OU1(+) OR OU2(+) in AR1 by a reference to OUNew(+) and the "old" reference to OU1(+) in AR2 by a reference to OUNew (+) (analogously for AR3).

Note that the described adaptation policies may also affect the valid actor sets of access rules when migrating them to the changed organizational model OM'. For example, for access rule AR2 its valid actor set on OM' becomes bigger: VAS(OM, AR2) = {A1, A2} and VAS(OM',AR2) = {A1, A2, A3}. Generally, changes of the valid actor set are more critical if it becomes smaller or even an empty set. Regarding our example from Figure 5, for instance, this would be exactly the case for access rule AR3 when migrating it to OM' in the described way. We come back to this problem in Proposition 3 (cf. Section 4.3).

Figure 6 shows how access rules can be adapted when applying a split operation (here splitting organizational unit OU2 into two new organizational units OU2_1 and OU2_2). According to Adaptation Policy 1 the given access rule containing a reference to the splitted organizational unit OU2 could be adapted by

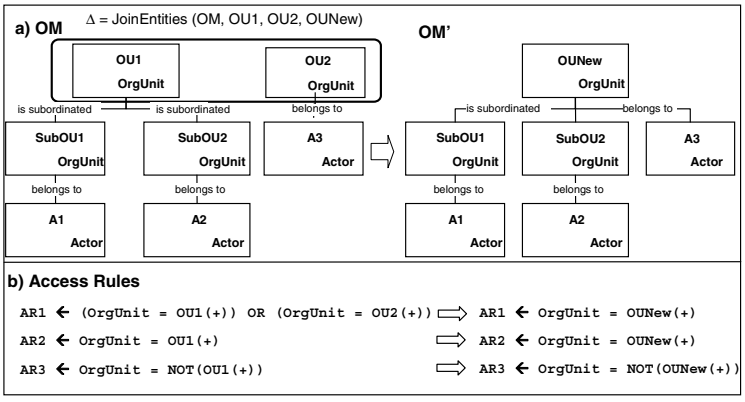

Fig. 5. Automatic Adaptation of Access Rules when Applying a Join Operation

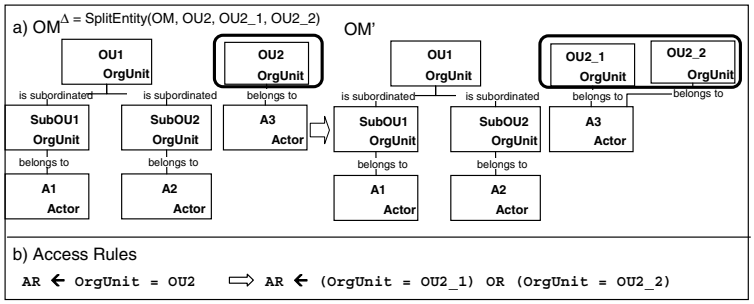

Fig. 6. Automatic Adaptation of Access Rules when Applying a Split Operation

replacing this reference with the expression (`OU2_1 OR OU2_2`). Again it has to be pointed out that this only constitutes a suggestion by the system.

In addition to split and join operations the deletion of entities may lead to dangling references. 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 exist many situations in which automatic rule 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 complex access rules (cf. Def. 5). As an example take access rule (`AR` ⟵ `Role = R1` ∨ `Role = R2`). Assume that role `R2` is deleted from the used org. model. This model change causes a dangling reference in `AR`. A meaningful suggestion for automatically adapting rule `AR` would then be to delete expression (`EAR` ⟵ `Role = R2`) from `AR`. This would result in the simplified rule (`AR` ⟵ `Role = R1`), which does not contain dangling references. Furthermore, we could exploit the semantics of hierarchical relations in order to come up with some adaptation suggestions for affected access rules. Assume, for example, that role r, which is a specialization of another role r_{super}, is deleted. Assume further that there exists an access rule `AR` ⟵ `"... Role = r ..."` affected by this change. Then it could be a reasonable strategy to suggest adapted rule `AR'` ⟵ `"... Role =` r_{super} `..."` instead of `AR` (e.g., if it is not longer necessary to have a more specialized nurse working on a specific ward, patient care can be performed by a regular nurse as well). The same strategy may be applied in the contrary direction if super-role r is deleted and the associated references within affected access rules are adapted to reference more specific role r_{sub}. Justification is that actors having role r_{sub} possess all capabilities assigned to role r and therefore are able to substitute actors having role r. Similar considerations hold for hierarchical relations between organizational units.

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.

Application of Complex Changes $\Delta = op_1, ..., op_n$. We now consider the application of a sequence of change operations to an organizational model OM; i.e., the application of a change (transaction) $\Delta = op_1, ..., op_n$ to OM (resulting in OM') and its effects on related access rules. Again, when considering an access rule `AR` on OM, dangling references within `AR` may result after migration to OM'. As an example consider Fig. 4 where the migration of access rules `AR1`, `AR2`, and `AR3` to OM' results in dangling references when applying change Δ to OM. In order to deal with this problem, we have to analyze the effects of each applied change operation op_i (i = 1, ..., n) on access rules defined on OM. Regarding a particular access rule this analysis is accomplished in the order these operations were applied to OM. For those change operations op_i which cause dangling references and for which there exists an adaptation policy (cf. Adaptation Policy 1) we can adapt the affected access rules accordingly.

For change Δ from Fig. 4 and Fig. 3, respectively, we check for the effects of operations op_1 = `JoinEntities(OM, ...)`, op_2 = `DeleteRelation(OM, ...)`, and so on (for a complete definition of Δ see Fig. 3b). Consider, for example, access rule `AR2` ⟵ `OrgUnit='treatment area'` in Fig. 4b. The application of op_1 = `JoinEntities(OM, ...)` already results in a dangling reference for `AR2`. Therefore `AR2` is modified to (`AR2` ⟵ `OrgUnit = 'patient services'`) by applying Adaptation Policy 1 for the join operation. According to Proposition 2 the following three change operations related to Δ (and being of type `DeleteRelation(OM, ...)`) do not cause dangling references when migrating access rules to OM'. For the applied `DeleteEntity(OM, ...)` operation there may be dangling references, but not for access rule `AR2`. The next `SplitEntity(OM, ...)` operation does not affect `AR2` and the following `CreateRelation(OM, ...)` operation is uncritical regarding dangling references. Finally, the last `DeleteEntity(OM, ...)` operation could cause dangling references, but again not for access rule `AR2`. Altogether, `AR2` can migrate to OM' by adapting it to (`AR2` ⟵ `OrgUnit = 'patient services'`). According to Adaptation Policy 1 we can ensure that `AR2` does not contain dangling references based on OM'.

4.3 Dynamic Aspect – Valid Actor Set

Even if the problem of dangling references is satisfactorily solved we still may be confronted with non–resolvable access rules when changing an org. model. This may cause runtime errors or at least runtime delays (e.g., if activities cannot immediately be worked on since there is no qualifying actor any more). It may also impose security problems (e.g., if then the non-resolvable activity is offered to the system or process administrator as it is the case in several existing systems).

General Considerations. Let OM be an org. model which is transformed into another org. model OM' by change Δ. Furthermore, let `AR` be an access rule on

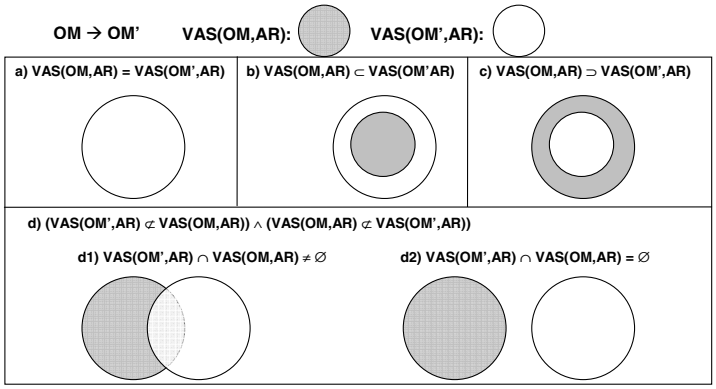

Fig. 7. Changing Organizational Models and Migrating Access Rules

OM. First of all, we illustrate at an abstract level how the valid actor set of an access rule `AR` based on OM may change when migrating this rule to the new model version OM'. Figure 7 depicts possible relations between the valid actor set of `AR` on OM `VAS(OM,AR)` and the valid actor set of `AR` on OM' `VAS(OM',AR)`:

In Fig. 7a the migration of `AR` from OM to OM' does not influence the valid actor set, i.e., the set of valid actors remains the same. In this case, first of all, `AR` is still resolvable over OM' and does not require any adaptation of work lists or lists of qualified actors afterwards.

Figure 7b shows the case where the valid actor set is expanded when migrating `AR` to OM'. In practice this may require, for example, an update of user worklists by additionally inserting the associated work items into the worklists of newly qualified actors from the difference set `VAS(OM', AR)` $\setminus$ `VAS(OM, AR)`. By contrast, the valid actor set may be also reduced due to a model change as depicted in Fig. 7c. Consequently, for all actors no longer qualified for accessing the associated object or task (i.e., `VAS(OM, AR)` $\setminus$ `VAS(OM', AR)`) the associated access privileges have to be adapted accordingly. Note that for the case depicted in Fig. 7c, it is possible that the valid actor set of `AR` on OM' becomes empty, i.e., `AR` may be no longer resolvable on OM'.

The scenario depicted in Figure 7d, where `VAS(OM, AR)` is not a subset of `VAS(OM', AR)` (or vice versa) can be further divided into two sub-cases d1 and d2. For case d1 there are still actors contained in both valid actor sets, i.e., the intersection of `VAS(OM, AR)` and `VAS(OM', AR)` is non–empty. For this case, we firstly have to withdraw the privileges associated with `AR` for all actors contained in `VAS(OM, AR)` $\setminus$ `VAS(OM', AR)`. Second, we have to newly assign these privileges to the actors contained in `VAS(OM', AR)` $\setminus$ `VAS(OM, AR)`. Finally, if `VAS(OM, AR)` and `VAS(OM', AR)` are disjoint as depicted in case d2 the privileges associated with `AR` have to be removed for all actors from `VAS(OM, AR)` and be added for all actors from `VAS(OM', AR)`.

Knowing which of this cases applies in a given change scenario is helpful in order to conduct the necessary adaptations of qualified actor lists or work lists when migrating an access rule to the changed organizational model. In the

following, first of all, we study the effects on valid actor sets of both elementary and complex access rules when a single change operation Δ_{op} is applied to OM. This is followed by a discussion of complex changes where a sequence of operations $op_1, ..., op_n$ is applied within one change transaction Δ to OM.

Impact of Org. Model Changes on Actor Sets When Applying Single Change Operations. Let Δ_{op} be a single change operation which transforms org. model OM into org. model OM'. Let further `AR` be a valid access rule on OM. According to Proposition 2 the application of a change operation $\Delta_{op} \in$ {`CreateRelation(...)` `DeleteRelation(...)` `ReAssignRelation(...)`} does not lead to dangling references in `AR` afterwards. However, Δ_{op} may affect the valid actor set of `AR` when migrating this access rule to OM', i.e. `VAS(OM', AR)` $\neq$ `VAS(OM, AR)` (cf. Fig. 7). Assume, for example, that in the org. model from Fig. 1 the relation indicating that actor `Black` belongs to `treatment area` (i.e., relation `((Black, Actor), (treatment area, OrgUnit), belongs to)`) is reassigned to `(Black, Actor), (administration, OrgUnit), belongs to)`. Then the valid actor set for access rule `(AR2` $\longleftarrow$ `OrgUnit='treatment area')` is then reduced from {`Dr. Smith, Black`} to {`Dr. Smith`}.

We first analyze the effects of Δ_{op} on the valid actor sets of elementary access rules `EAR` and negated elementary access rules `NEAR`. For the sake of readability we do not consider all scenarios from Fig. 7, but focus on the most "critical" cases; i.e., changes of the oganizational model due to which the valid actor set of `[N]EAR` is reduced (or even becomes empty) when migrating this access rule from OM to OM'. These cases are summarized in Table 3. The first column of this table shows the change operation (and its parameters) and the third column the (negated) elementary access rule(s) to be considered. Note that we may examine more than one access rule for a given change operation. Further, the effects of a change operation on the valid actor set of an access rule is depicted in the second column. As can be seen, in most cases the actor set will reduced when applying the change operation and migrating the access rule to the new organizational model. Regarding operation `ReassignRelation` we also give examples where new actors may be also added to the valid actor set.

The following figures illustrate some interesting situations from Table 3. Firstly we consider the creation of a new relation between two entities of the organizational model as depicted in Fig. 8. In this example, for both negated access rules `NEAR1` and `NEAR2` their valid actor set based on OM will be reduced when migrating the rule to OM'. In particular, due to change Δ_2 for `NEAR2` the valid actor set becomes empty afterwards. Analogously, the application of change operation `DeleteRelation(OM, ...)` may lead to reduced actor sets. As an example consider the change scenario from Fig. 9. When deleting the relation `((a2, Actor), (r1, Role), has)` from OM, for instance, for access rules `EAR` $\longleftarrow$ `(Role = 'r1')` or `EAR` $\longleftarrow$ `(ROLE='r5'(+)` the valid actor set will be reduced afterwards. The same applies to access rule `EAR` $\longleftarrow$ `(ROLE=r3(+))` after deleting relation `(((r1, Role), (r2, Role), specializes)`.

As discussed in Section 4.2, when applying change operations `joinEntities` or `splitEntity`, dangling references within certain access rules may emerge

Table 3. Reduction of Valid Actor Set After Application of Change Operation Δ

Assume in the following that organizational model OM is transformed into organizational model OM' by applying change operation Δ. Let further AR be an access rule defined on the basis of OM.

Change Operation Δ	VAS(OM',[N]EAR) = VAS(OM, [N]EAR) \ δ ($\cup$ ϵ)	$\forall$ [N]EAR $\in$ AR $\Longrightarrow$
CreateRelation(OM, (a,Actor),(r,Role),has)	δ = {(a, Actor)}	NEAR ⟵ NOT(Role=r1[(+)]) with (r, Role) ∈ QualEntities(OM,Role=r1[(+)])
CreateRelation(OM, (a,Actor),(o,OrgUnit),belongsTo)	δ = {(a, Actor)}	NEAR ⟵ NOT(OrgUnit=o1[(+)]) with (o, OrgUnit) ∈ QualEntities(OM,OrgUnit=o1[(+)])
CreateRelation(OM, (r1,Role),(r2,Role),specializes)	δ = VAS(OM,EAR') with EAR' ⟵ Role=r1	NEAR ⟵ NOT(Role=r3[(+)]) with (r1,Role), (r2,Role) ∈ QualEntities(OM',Role=r3[(+)])
CreateRelation(OM, (o1,OrgUnit),(o2,OrgUnit), is subordinated)	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=o1	NEAR ⟵ NOT(OrgUnit=o3[(+)]) with(o1,OrgUnit), (o2,OrgUnit) ∈ QualEntities(OM',OrgUnit=o3[(+)])
DeleteRelation(OM, (a Actor),(r,Role),has)	δ = {(a, Actor)}	EAR ⟵ Role = r1[(+)] with (r, Role) ∈ QualEntities(OM,Role=r1[(+)])
DeleteRelation(OM,(a,Actor), (o,OrgUnit),belongsTo)	δ = {(a, Actor)}	EAR ⟵ OrgUnit=o1[(+)] with (o, OrgUnit) ∈ QualEntities(OM,OrgUnit=o1[(+)])
DeleteRelation(OM, (r1,Role),(r2,Role),specializes)	δ = VAS(OM,EAR') with EAR' ⟵ Role=r1	EAR ⟵ Role=r3[(+)] with (r1,Role), (r2,Role) ∈ QualEntities(OM',Role=r3[(+)])
DeleteRelation(OM, (o1,OrgUnit),(o2,OrgUnit), is subordinated)	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=o1	EAR ⟵ OrgUnit=o3[(+)] with (o1,OrgUnit), (o2,OrgUnit) ∈ QualEntities(OM',OrgUnit=o3[(+)])
ReassignRelation(OM, ((a,Actor),(r,Role),has), (r,Role),(rN,Role))	δ = {(a,Actor)}	EAR ⟵ Role=r1[(+)] with (r,Role) ∈ QualEntities(OM,Role=r1[(+)])
	δ = {(a,Actor)}	NEAR ⟵ NOT(Role=r1[(+)]) with (rN, Role) ∈ QualEntities(OM,Role=r1[(+)])
ReassignRelation(OM, ((a,Actor),(o,OrgUnit),belongsTo), (o,OrgUnit),(oN,OrgUnit))	δ = {(a,Actor)}	EAR ⟵ OrgUnit=o1[(+)] with (o,OrgUnit) ∈ QualEntities(OM,OrgUnit=o1[(+)])
	δ = {(a,Actor)}	NEAR ⟵ NOT(Role=o1[(+)]) with (oN,OrgUnit) ∈ QualEntities(OM,OrgUnit=o1[(+)])
ReassignRelation(OM, ((r1,Role),(r2,Role),specializes), (r1,Role),(rN,Role))	δ = VAS(OM,EAR') with EAR' ⟵ Role=r1(+)	EAR ⟵ Role=r'[(+)] with (r2, Role) ∈ QualEntities(OM,Role=r'[(+)])
	δ = VAS(OM,EAR') with EAR' ⟵ Role=r1(+)	NEAR ⟵ Role=r'[(+)] with (r2, Role) ∈ QualEntities(OM,Role=r'[(+)]):
	δ = VAS(OM,EAR') with EAR' ⟵ Role=r1(+)	NEAR ⟵ Role=r'[(+)] with (rN, Role) ∈ QualEntities(OM,Role=r'[(+)])
ReassignRelation(OM, ((r1,Role),(r2,Role),specializes), (r2,Role),(rN,Role))	δ = VAS(OM,EAR')) with EAR' ⟵ Role=r1(+) ϵ = VAS(OM,EAR'') with EAR'' ⟵ Role = rN(+)	EAR ⟵ Role=r'[(+)] with (r2, Role) ∈ QualEntities(OM,Role=r'[(+)])
	ϵ = VAS(OM,NEAR')) with NEAR' ⟵ NOT(Role=r1(+)) δ = VAS(OM,NEAR'') with NEAR'' ⟵ NOT(Role = rN(+))	NEAR ⟵ NOT(Role=r'[(+)]) with (r2, Role) ∈ QualEntities(OM,Role=r'[(+)])
ReassignRelation(OM, ((o1,OrgUnit),(o2,OrgUnit),is subordinated), (o1,OrgUnit),(oN,OrgUnit))	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=o1(+)	EAR ⟵ OrgUnit=o'[(+)] with (o2,OrgUnit) ∈ QualEntities(OM,OrgUnit=o'[(+)])
	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=o1(+)	NEAR ⟵ NOT(OrgUnit=o'[(+)]) with (o2,OrgUnit) ∈ QualEntities(OM,OrgUnit=o'[(+)])
	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=OrgUnit=o1(+)	NEAR ⟵ NOT(OrgUnit=o'[(+)]) with (oN,OrgUnit) ∈ QualEntities(OM,OrgUnit=o'[(+)])
ReassignRelation(OM, ((o1,OrgUnit),(o2,OrgUnit),is subordinated), (o2,OrgUnit),(oN,OrgUnit))	δ = VAS(OM,EAR') with EAR' ⟵ OrgUnit=o1(+) ϵ = VAS(OM,EAR'') with EAR'' ⟵ OrgUnit = oN(+)	EAR ⟵ OrgUnit=o'[(+)] with (o2,OrgUnit) ∈ QualEntities(OM,OrgUnit=o'[(+)])
	ϵ = VAS(OM,NEAR')) with NEAR' ⟵ NOT(OrgUnit=o1(+)) δ = VAS(OM,NEAR'')) with NEAR'' ⟵ NOT(OrgUnit = oN(+))	NEAR ⟵ OrgUnit=o'[(+)] with (o2, OrgUnit) ∈ QualEntities(OM,OrgUnit=o'[(+)])

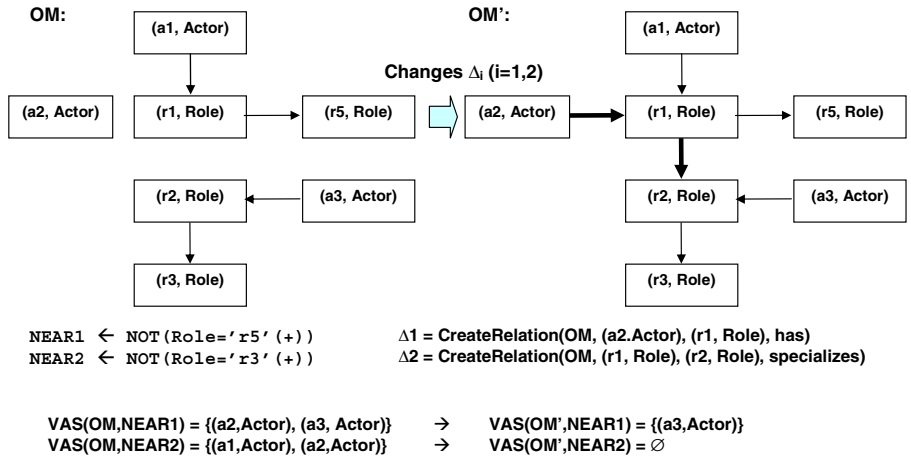

Fig. 8. Reduction of Valid Actors Sets After Creating Relations

(cf. Figures 5 and 6). Therefore we have provided rules (cf. Adaptation Policy 1) which enable the system to suggest automatic adaptations of the affected access rules to the user. However, after applying these adaptation rules the actor sets of the adapted and migrated access rules may be affected as well. In detail: For operation `joinEntities` it has to be checked how the actor sets of affected access rules have changed after the applicastion of adaptation rule δ (cf. Adaptation Policy 1). In particular, for negated access rules, the actor set may become smaller (and therefore even empty) afterwards. For operation `splitEntity` the application of adaptation rule δ does not affect the actor sets of access rules. Therefore no checks become necessary afterwards.

Proposition 3 (Actor Set After Joining Entities and Adaptation of Access Rules). *Let a (correct) organizational model OM be transformed into another (correct) organizational model OM' by applying change operation* $\Delta =$ `JoinEntities(OM, ...)`. *Let further* `AR` *be an access rule defined on OM which is transformed into access rule AR' (by applying adaptation rules* δ_{AR}*; cf. Adaptation Policy 1) and then migrated to OM'. Then:*

- $\forall$ EAR $\longleftarrow$ (OrgUnit=o1) $\vee$ (OrgUnit=o2)) $\wedge$ $\forall$ NEAR $\longleftarrow$ NOT((OrgUnit=o1) $\vee$ (OrgUnit=o2)) $\Longrightarrow$
  VAS(OM',[N] EAR) = VAS(OM,[N] EAR)
- $\forall$ EAR $\longleftarrow$ OrgUnit=o1 $\Longrightarrow$
  VAS(OM',EAR) = VAS(OM,EAR) $\cup$ VAS(OM,EAR') with EAR' $\longleftarrow$ OrgUnit=o2
- $\forall$ NEAR $\longleftarrow$ NOT(OrgUnit=o1) $\Longrightarrow$
  VAS(OM',NEAR) = VAS(OM,NEAR) $\setminus$ VAS(OM,EAR') with EAR' $\longleftarrow$ OrgUnit=o2
- $\forall$ EAR $\longleftarrow$ OrgUnit=o2 $\Longrightarrow$
  VAS(OM',EAR) = VAS(OM,EAR) $\cup$ VAS(OM,EAR') with EAR' $\longleftarrow$ OrgUnit=o1
- $\forall$ NEAR $\longleftarrow$ NOT(OrgUnit=o2) $\Longrightarrow$
  VAS(OM',NEAR) = VAS(OM,NEAR) $\setminus$ VAS(OM,EAR') with EAR' $\longleftarrow$ OrgUnit=o1

Consider, for example, access rule `AR3` as depicted in Fig. 3. Before migration the valid actor set of this rule turns out as `VAS(OM,AR3) =` {`A3`}. According to the adaptation policies provided by Rule Adaptation Policy 1, `AR3` is adapted to

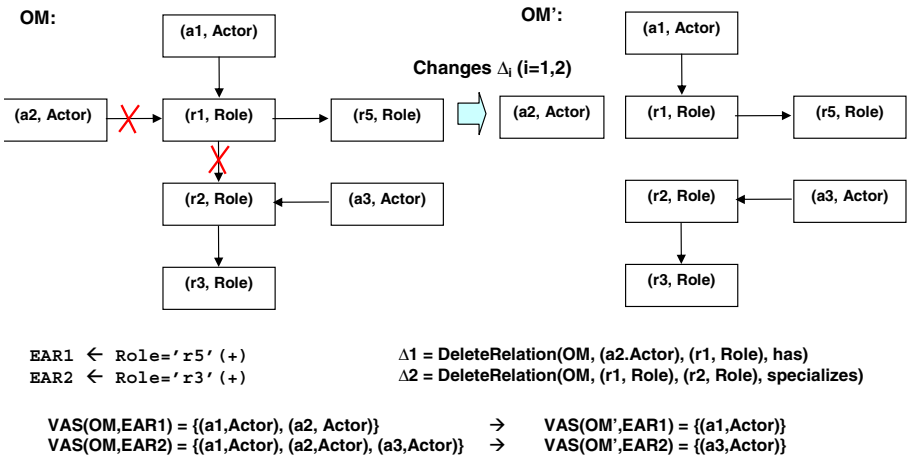

Fig. 9. Reduction of Valid Actors Sets After Deleting Relations

AR3 ⟵ NOT(OUNew). This adaptation affects the valid actor set as follows: VAS(OM',AR3) = VAS(OM,AR3) \ VAS(OM,EAR') with EAR' ⟵ (OrgUnit=OU2) which results in VAS(OM',AR3) = {A3} \ {A3} = ∅ (cf. Proposition 3). Such critical effects should be at least reported to the user.

Impact of Org. Model Changes on Actor Sets When Applying Complex Change Operations. Finally, we sketch potential effects on the valid actor set of access rules when migrating them from an org. model OM to a changed org. model OM' after application of a (complex) change $\Delta = op_1, ..., op_n$. When considering Fig. 7, it becomes clear that the application of each change operation op_i (i = 1, ..., n) may result in a change of the valid actor set. In particular, some of the changes may add actors to the valid actor set whereas others remove elements from it. Therefore a general statement on how the actor set changes is difficult. Similar to the considerations about dangling references resulting after the application of a complex (i.e., multi-operation) change, the effects of each change operation op_i (in the order of their application) on the valid actor set can be determined. Doing so finally results in the new actor set of an access rule based on OM. However, we have to consider the efforts for this approach. In order to decrease the computing time for the resulting actor set, the analysis and adaptation for dangling references and the determination of the actor set changes can be done in one go. We will address the issue of possible optimization methods in future work.

5 Architectural and Implementation Issues

In this section we sketch architectural and implementation issues which arise when realizing the described framework. Section 5.1 summarizes the architecture

of the implemented adaptive enterprise security service. In Section 5.2 we discuss its concrete usage in the context of process-aware information systems.

5.1 Overview of the Enterprise Security Service

As proof-of-concept we realized an advanced *enterprise security service* (ESS) which implements the described framework. We have chosen a service–oriented design in order to support the reuse of the security component in different context and by different system components (e.g., workflow systems or document management tools). Fig. 10 depicts the overall architecture of the ESS (simplified illustration). The developed ESS comprises both tools and programming interfaces for creating, evolving and managing organizational models as well as the access rules based on them.

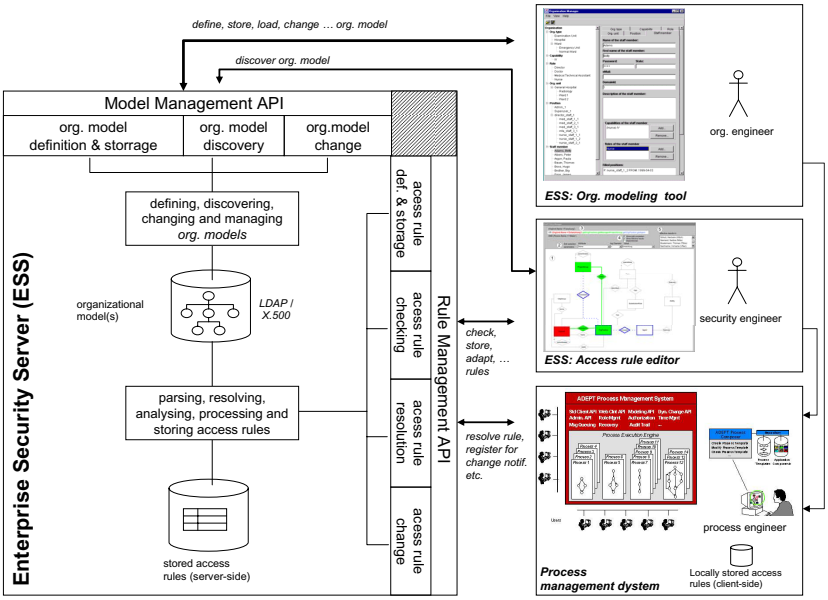

Fig. 10. (Simplified) architecture of the adaptive Enterprise Security Service (ESS)

For creating and adapting org. models the ESS offers a standard editor (cf. Fig. 10). Among other things this tool utilizes the change operations presented in this paper. All changes are logged and are traceable. Different org. models in different versions can be maintained. For editing elementary as well as complex access rules another tool is provided (cf. Fig. 10). Rules can be only released if they are syntactically and semantically correct, which requires cross-checks with the related org. model. The rule editor is realized as plug-in and can therefore be easily integrated in different client applications (for further details see [18]). Implementation is based on Java and SVG (Scalable Vector Graphics).

As mentioned the ESS offers powerful programming interfaces. The model management API provides the basis for defining and storing new org. models, for adapting existing ones to environmental changes, and for discovering information about org. models. Based on this interface, adapted client components for editing, displaying and analyzing org. models can be realized. Furthermore, the ESS offers a complete interface for the management of access rules. This interface allows to define, check and store new access rules (based on a referred organizational model) and to maintain these rules by the ESS.

In general, we do not require clients of the ESS to store and maintain their access rules within the ESS; this is only an optional feature. After having defined an access rule, it may be also maintained by the client system itself (e.g., a workflow system). In this case, the rule is represented as "query string" following the syntax of our rule definition language. This string can be resolved at runtime by sending it to the ESS (e.g., when an activity in a workflow becomes activated), which then parses and processes the access rule string, finally returning the set of actors who qualify for it. For access rules already stored in the ESS these steps can be partially omitted, resulting in higher system performance and better response rates. In this case the client simply invokes a generic procedure with the respective context information via the ESS interface.

The realized ESS extends the features of existing access control components by offering more advanced change facilities. Adaptations of the org. model are based on the operational framework described in Section 2. The ability to concomitantly adapt access rules is of particular importance. It uses the rule adaptation framework introduced in Section 4. Further, we offer different migration and adaptation policies for access rules depending on whether they are directly maintained by the ESS or by the client application. Access rules stored within the ESS can be immediately processed in order to decide whether there is a need for adaptation and - if yes - how it should look like. Further the ESS determines for which rules actor sets have changed. Based on this information users or clients can be notified in order to accept the suggested rule adaptations or to accommodate them.

Rules outside ESS control cannot be immediately migrated. This, in turn, might lead to non-resolvable access rules which require lazy migration techniques and advanced exception handling mechanisms. By using the provided API, however, clients can register for change notification events. When a model change occurs the ESS notifies registered clients, which then can check the validity of their access rules against the newly released version of the org. model. Finally, if a non-resolvable access rule is sent to the ESS, an exception is thrown providing the client with the information about necessary rule adaptations. Due to lack of space we omit further details and a more precise presentation of the interaction patterns between clients (buildtime and runtime) and the ESS. Current implementation of the ESS is based on Java and relational database management technology. Integration with LDAP (Lightweight Directory Access Protocol) services [19] is one important requirement for the future.

Clients can be components of information systems or supporting technology (e.g., workflow systems). Different clients may share one org. model (e.g., to achieve consistency across multiple systems) or may maintain their own model if favorable. In any case the information about org. entities and access rules can be separated from the business logic implemented by the client programs. Note that this provides the basis for the controlled evolution of org. models and related access rules, and also constitutes a significant improvement when compared to the proprietary, heterogeneous security components we can find in current information systems. We strongly believe that a component like ESS is very useful for dealing with org. change in a secure and intelligent manner, and for providing better maintainability and traceability in this context.

5.2 Managing Actor Assignments and Worklists in Process-Aware Information Systems

To illustrate our results we apply them to important elements of process-aware information systems (PAIS) - activity actor assignments, user worklists, and their adaptation due to org. changes. More precisely we sketch how changes of an org. model have to be handled within a workflow system and how the different system components interact with each other to cope with model changes. Usually, a PAIS maintains different process templates each of them representing a particular business process. Each of these process templates captures different aspects of a business process like process activities, control and data flow between activities, and actor assignments. The latter are of particular interest in the context of the present work. They represent the access rules needed by the PAIS to decide which users may work on instances of the respective activity. As an example, consider the two actor assignment rules `R1: (OrgUnit = OU_2 and Role = Role_2)` and `R2: Role = Role_1` as depicted in Fig. 11. When an activity instance becomes activated at runtime the PAIS determines all actors qualifying for this activity, creates corresponding work items, and adds them to the worklists of these users.

At buildtime the PAIS must support the definition of actor assignments based on an org. model and their correlation with process activities. For this the PAIS either can utilize the standard modeling tools offered by the ESS or realize own buildtime clients based on ESS interfaces. Within the ADEPT project, for example, we have utilized the tools and plug-ins mentioned above. Access rules can be assigned to activities or to other privileges relevant for the PAIS. Both org. models and access rules are stored within the ESS.

Consider the scenario depicted in Fig. 11. When a change occurs within the organization, an authorized user can adapt the org. model accordingly. In the example from Fig. 11, for instance, the two org. units `OU_2` and `OU_3` are joined and the "has-role" relation between `Actor_1` and `Role_1` is deleted. This results in a new version of the respective model, which then triggers the adaptation and migration of related access rules. In the given example, for instance, at the process template level the two actor assignment rules `R1: (OrgUnit = OU_2 and Role = Role_2)` and `R2: Role = Role_1` may have to be adapted. If these rules are directly maintained by the ESS, this service analyses them for necessary

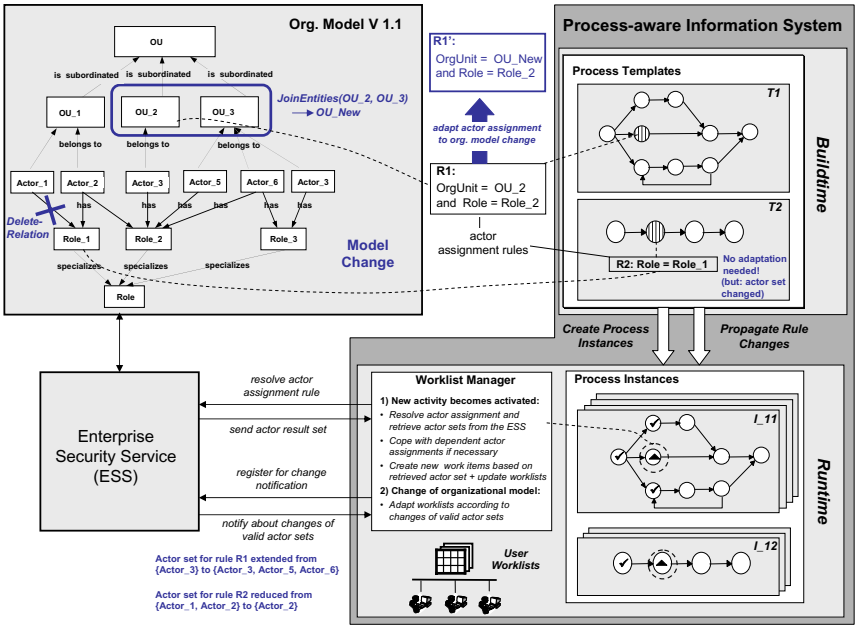

Fig. 11. Adapting actor assignment rules and worklists in process-aware IS

adaptations. As a result the ESS suggests the process engineer to adapt rule `R1` to rule `R1'` (cf. Fig. 11), but to remain rule `R2` unchanged.

As discussed in Section 4 the adaptation of access rules (actor assignments) is only one side of the coin. We also have to analyze the effects of the performed model and rule changes to valid actor sets. This is of particular importance for PAIS in order to avoid outdated or inconsistent worklists. As an example take the adaptation applied for rule `R1` in Fig. 11. Obviously, this extends the valid actor set of this rule from {`Actor_3`} to {`Actor_3, Actor_5, Actor_6`}. For currently activated activity instances based on this rule this should imply the creation of new work items for `Actor_5` and `Actor_6`. As another example consider rule `R2`. Though this rule must not be adapted due to the model change (see above) its valid actor set is reduced from {`Actor_1, Actor_2`} to {`Actor_2`}. Therefore, respective work items currently assigned to `Actor_1` on basis of `R2` should be removed from the worklist of this actor. In our approach, the worklist manager of the PAIS accomplishes such on-the-fly worklist updates based on the interfaces offered by the ESS. Due to lack of space we omit further details. However, we are aware of the fact that the efficient update of user worklists is a big challenge in the given context, particularly when thinking of scenarios with ten thousands of work items.

6 Related Work

The provision of an adequate access control framework is indispensable for any IS. In the literature numerous approaches have been presented dealing with

challenging issues related to access control (e.g., [12,20,21,22]). Most of these approaches apply *Role–Based Access Control (RBAC)* models for defining and managing user privileges [6,23,20,24], e.g., to control the access to business documents and database objects, or to resolve the set of actors that qualify for a newly activated task in a *workflow system* [25,4,8,21,26,22].

Usually, corresponding models provide core RBAC features as well as role hierarchies. Regarding workflow–based applications, in addition, dynamic constraints (e.g., separation of duties) were extensively investigated in the past [4,8,27,28]. Practical issues related to RBAC (e.g., NIST's proposed RBAC standard, integration of RBAC with enterprise IT infrastructures, RBAC in commercial products) are summarized in [24].

In the workflow literature several proposals have been made aiming at adaptive process management (e.g., [29,30,31,32,33,34,35,36,37]). The ADEPT technology, for example, enables controlled changes at the process type as well as the process instance level (for details see [38,39,40]). Thereby, correctness and consistency constraints of a workflow are preserved when dynamically changing its structure, its state, or its attributes during runtime. In [22] an extension to RBAC is proposed in order to accomplish such process changes is a safe way; i.e. to restrict changes to selected user groups or processes if required. Though all these approaches stress the need for adaptive information systems and define different notions of correctness (for an overview see [30]), so far, focus has been on process changes (control and data flow).

There are only few approaches [12,41,42,43] which address the problem of organizational change. In [12,41,42] eight categories of structural changes on organizational models are identified. Examples include the splitting of organizational units, the creation of new organizational entities, and the reassignment of an actor to a new organizational unit. In principle, all these cases can be captured by our change framework as well. As opposed to [12], however, we additionally follow a rigorous formal approach in order to be able to derive the effects of organizational changes on related access rules as well. Corresponding issues are factored out in [12]. The approach introduced in [43] deals with the evolution of access rules in workflow systems. However, only very simple scenarios are described without any formal foundation. Furthermore, the compact definition of access rules is aggravated by the lack of adequate abstraction mechanisms (e.g., hierarchical structures).

Issues related to the modeling of organizational structures have been considered by different groups [11,21,18]. Most of them suggest a particular meta model for capturing org. entities and the relationships between them. Model changes and the adaptation of access rules, however, have not been studied by these approaches in sufficient detail. Particularly, no formal considerations exist and no proof-of-concept prototypes have been provided.

In [44] important issues related to changes of processes and org. structures are 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 by respective components (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).

7 Summary and Outlook

The integrated and controlled evolution of organizational models as well as access rules will be key ingredients of next generation enterprise security services, ultimately resulting in adaptive and highly flexible access control frameworks. Together with our previous work on business process evolution and dynamic process change [38,45,40,39] the presented concepts contribute to a powerful platform enabling the realization of flexible and adaptive information systems.

In this paper, we have designed a comprehensive framework for defining and changing organizational models, for specifying access rules in a consistent manner, and for correctly adapting these access rules when model changes occur. We have discussed important challenges and requirements in this context as well as limitations of current approaches. Based on this we have introduced a comprehensive framework for the evolution of organizational models and the adaptation of related access rules. The very important aspect of our work is its formal foundation. We have provided precise definitions and formal propositions which are fundamental for the correct handling of model changes, for reasoning about the effects of such changes on access rules, and for adapting access rule if necessary. The treatment of both elementary and composed access rules as well as the consideration of runtime issues (e.g., effects of model changes on rule actor sets) add to the completeness of our approach. Finally, we have discussed important architectural issues and sketched a proof-of-concept implementation demonstrating the feasibility of the presented concepts.

The implemented security service has been coupled with the ADEPT2 process management system in order to enable the (dynamic) adaptation of actor assignments, user worklists, etc. when changes of the organizational model happen. For the sake of readability, in this paper we have restricted our consideration to a rather simple role-based access control model which applies basic entities (org. unit, role, and actor) and the relations between them (incl. role hierarchies). However, the enterprise security service realized by us within the ADEPT2 project is based on a more expressive meta model (incl. organizational entities like position, capability or project group).

There are many other challenging issues that can be linked to the evolution of org. models and related access rules. Firstly, we should consider semantical constraints as well. Uncontrolled changes of an org. model, for example, may violate semantical constraints like *separation of duty (SoD)* or *mutual exclusion* [27,28,46,47]. Among other things, this may result in security gaps. Secondly, we believe that changes of the org. model must be closely linked to other components of an IS. For example, actor assignments in workflow-based applications may have to be adapted on-the-fly in order to cope with org. 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. Thirdly, it is interesting to investigate how access rules can be improved, for example, based on previous adaptations (*access rule life cycle management*). First work in this field on the mining of access rules from workflow log data has been published in [48]. Finally, changes may not only concern the process model or the org. model but other components of the information systems as well. As an example take resource models or data models, which may be also subject of change. Thew more we extract the specification of these different aspects from application code the better will be the basis for setting up flexible adaptation mechanisms.

References

1. Aalst, v.d.W., van Hee, K.: Workflow Management. MIT Press, Cambridge (2002)
2. Sutton, M.: Document Management for the Enterprise: Principles, Techniques and Applications. John Wiley, Chichester (1996)
3. Linthicum, D.: Enterpise Application Integration. Addison-Wesley, Reading (1999)
4. Bertino, E., Ferrari, E., Alturi, V.: The specification and enforcement of authorization constraints in wfms. ACM Trans. on Inf. and Sys. Sec. 2, 65–104 (1999)
5. Sandhu, Smarati,: Authentication, access control and audit. ACM Computings Surveys 28, 241–243 (1996)
6. Ferraiolo, D., Kuhn, D., Chandramouli, R.: Role–Based Access Control. Artech House (2003)
7. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29, 38–47 (1996)
8. Wainer, J., Barthelmess, P., Kumar, A.: W–RBAC – a workflow security model incorporating controlled overriding of constraints. International Journal of Collaborative Information Systems 12, 455–485 (2003)
9. El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Saurel, C., Deswarte, Y., Miege, A., Trouessin, G.: Organization-based access control. In: Proc. Proc. 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2003)
10. Konyen, I.: Organizational structures and business processes in hospitals. Master's thesis, University of Ulm, Computer Science Faculty (in German) (1996)
11. Jablonski, S., Schlundt, M., Wedekind, H.: A generic component for the computer–based use of organizational models (in german). Informatik Forschung und Entwicklung 16, 23–34 (2001)
12. 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), pp. 375–387 (2001)
13. Dumas, M., ter Hofstede, A.W.A (eds.): Process Aware Information Systems. Wiley Publishing, Chichester (2005)
14. Rinderle, S., Reichert, M.: On the controlled evolution of access rules in cooperative information systems. In: CoopIS'05, pp. 238–255 (2005)
15. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST standard for role-based acces control. ACM ToISS 4, 224–274 (2001)

16. Tolone, W., Ahn, G., Pai, T.: Access control in collaborative systems. ACM Computings Surveys 37, 29–41 (2005)
17. 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, pp. 1113–1114 (2005)
18. Berroth, M.: Design of a component for organizational models. Master's thesis, University of Ulm, Computer Science Faculty (in German) (2005)
19. Howes, T., Smith, M., Good, G.: Understanding and Deploying LDAP Directory Services. New Riders (2001)
20. Bertino, E.: Data security. DKE 25, 199–216 (1998)
21. zur Muehlen, M.: Resource modeling in workflow applications. In: Proc. of the 1999 Workflow Management Conference (Muenster), pp. 137–153 (1999)
22. 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)
23. NIST: Proposed Standard for Role-Based Access Control (2004), `http://csrc.nist.gov/rbac/rbacSTDACM.pdf`
24. Ferraiolo, D.F., Kuhn, D.R.: Role based access control. In: 15th National Computer Security Conference (1992)
25. Botha, R.A., Eloff, J.: A framework for access control in workflow systems. Information Management and Computer Security 9, 126–133 (2001)
26. Pfeiffer, V.: A framework for evaluating access control concepts in workflow management systems. Master's thesis, University of Ulm, Computer Science Faculty (in German) (2005)
27. Giuri, L., Iglio, P.: A formal model for role-based access control with constraints. In: Proc. Computer Security Foundations Workshop, pp. 136–145 (1996)
28. Kuhn, D.: Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In: Proc. 2nd ACM Workshop on Role-based Access Control, pp. 23–30. ACM Press, New York (1997)
29. Aalst, v.d.W.: Exterminating the dynamic change bug: A concrete approach to support worfklow change. Information Systems Frontiers 3, 297–317 (2001)
30. 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, 9–34 (2004)
31. Agostini, A., De Michelis, G.: Improving flexibility of workflow management systems. In: Proc. Int'l Conf. on Business Process Management (BPM'00), pp. 218–234 (2000)
32. Joeris, G., Herzog, O.: Managing evolving workflow specifications. In: Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'98), New York City, pp. 310–321 (1998)
33. Weske, M.: Workflow management systems: Formal foundation, conceptual design, implementation aspects. University of Münster, Germany, Habilitation Thesis (2000)
34. Sadiq, S., Marjanovic, O., Orlowska, M.: Managing change and time in dynamic workflow processes. IJCIS 9, 93–116 (2000)
35. Fent, A., Reiter, H., Freitag, B.: Design for change: Evolving workflow specifications in ULTRAflow. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 516–534. Springer, Heidelberg (2002)
36. Kochut, K., Arnold, J., Sheth, A., Miller, J., Kraemer, E., Arpinar, B., Cardoso, J.: IntelliGEN: A distributed workflow system for discovering protein-protein interactions. Distributed and Parallel Databases 13, 43–72 (2003)

37. Edmond, D., ter Hofstede, A.: A reflective infrastructure for workflow adaptability. Data and Knowledge Engineering 34, 271–304 (2000)
38. Reichert, M., Dadam, P.: ADEPT$_{flex}$ - supporting dynamic changes of workflows without losing control. JIIS 10, 93–129 (1998)
39. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases 16, 91–116 (2004)
40. Rinderle, S., Weber, B., Reichert, M., Wild, W.: Integrating process learning and process evolution - a semantics based approach. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, Springer, Heidelberg (2005)
41. 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)
42. Domingos, D., Rito–Silva, A., Veiga, P.: Authorization and access control in adaptive workflows. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 23–28. Springer, Heidelberg (2003)
43. Aalst, v.d.W., Jablonski, S.: Dealing with workflow change: Identification of issues an solutions. ESORICS 2003 15, 267–276 (2000)
44. Klarmann, J.: Using conceptual graphs for organization modeling in workflow management systems. In: Proc. Conf. Professionelles Wissensmanagement (WM'01), pp. 19–23 (2001)
45. Rinderle, S., Reichert, M., Dadam, P.: Disjoint and overlapping process changes: Challenges, solutions, applications. In: Meersman, R., Tari, Z. (eds.) On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. LNCS, vol. 3290, pp. 101–120. Springer, Heidelberg (2004)
46. Simon, R., Zurko, M.: Separation of duty in role based environments. In: Proc. Computer Security Foundations Workshop X (1997)
47. Botha, R., Eloff, J.: Separation of duties for access control enforcement in workflow environments. IBM Systems Journal 40(3) (2001)
48. Ly, T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: Castellanos, M., Weijters, T. (eds.) First International Workshop on Business Process Intelligence (BPI'05), Nancy, France, pp. 177–190 (2005)