Enterprise Information Systems
Vol. 00, No. 00, February 2009, 1–30

RESEARCH ARTICLE

Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project

Stefanie Rinderle-Ma and Manfred Reichert
Institute of Databases and Information Systems, Ulm University, Ulm, Germany
{stefanie.rinderle, manfred.reichert}@uni-ulm.de
(*Received 00 Month 200x; final version received 00 Month 200x*)

The definition and management of access rules (e.g., to control access to business documents and business functions) is a fundamental task in any enterprise information system (EIS). While there exists considerable work on how to specify and represent access rules, only little research has been spent on access rule changes. Examples include the evolution of organizational models with need for subsequent adaptation of related access rules as well as direct access rule modifications (e.g., to state a previously defined rule more precisely). This paper presents a comprehensive change framework for the controlled evolution of role-based access rules in EIS. First, we consider changes of organizational models and elaborate how they affect existing access rules. Second, we define change operations which enable direct adaptations of access rules. In the latter context, we define the formal semantics of access rule changes based on operator trees. Particularly, this enables their unambiguous application; i.e., we can precisely determine which effects are caused by respective rule changes. This is important, for example, to be able to efficiently and correctly adapt user worklists in process-aware information systems. Altogether this paper contributes to comprehensive life cycle support for access rules in (adaptive) EIS.

Keywords: Enterprise Information System, Change, Access Control, Access Rule Lifecycle

1. Introduction

A fundamental aspect in the design of any enterprise information system (EIS) concerns *access control*, i.e., granting certain rights to specific users (e.g., the right to access a certain business document for a restricted group of users). There exists a multitude of models for defining access control mechanisms; e.g., GRANT / REVOKE statements in Database Management Systems or Role Based Access Control (RBAC) in process-aware information systems (PAIS). Usually such models comprise a set of *access rules* which are defined based on an organizational model capturing organizational entities as well as their relationships. Access rules then control which rights shall be granted to which users. In the context of PAIS, for example, access rules specify which tasks shall be offered as *work items* to which users in their worklists during the execution of a particular process

ISSN: 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/1751757YYxxxxxxxx
http://www.informaworld.com

instance.

1.1. *Problem statement*

Due to changes of organizational structures or evolving security policies, access rules have to be frequently adapted. This, in turn, must be effectively handled by the EIS in order to be able to cope with these organizational or policy changes in a quick, flexible, and secure way. So far, only little research has been spent on the evolution of access rules and the resulting effects on the underlying EIS. In particular, access rules might be subject to the following kinds of changes:

- *Organizational Change:* Access rule adaptations often become necessary when changing an organization and its organizational model respectively. For instance, assume that access control within a PAIS (i.e., the assignment of work items to user worklists) is based on the simple access rules depicted in Fig. 1. Assume further that these rules are based on organizational model OM. To streamline the organization, units `LoanM` and `LoanH` are merged into organizational unit `LoanH'`, and role `Clerk` is deleted from OM resulting in organizational model OM'. Obviously, access rule AR1 is not affected by this organizational change, whereas access rules AR2 and AR3 now refer to entities no longer being present in OM'. If the affected access rules were not adapted this would threaten robustness or security constraints of the PAIS. Worst case, for access rules which cannot be resolved properly, the associated task is offered to unauthorized users (e.g., process administrators, actors of preceding tasks, and so forth).

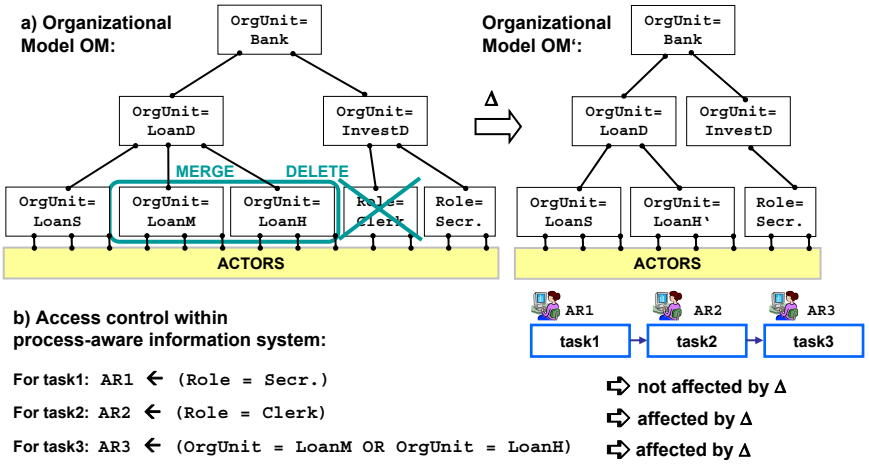


Figure 1. Organizational changes affecting access rules

- *Direct Access Rule Changes:* Generally, it must be also possible to directly adapt access rules within EIS (cf. Fig. 2). This will become necessary, for example, if access rules are not specified precisely enough. Either the access rule covers a too broad range of users (i.e., only a subset of the users qualifying for the access rules actually work on the associated work items) or it is too narrow (e.g., the related task is always delegated to substitutes). In both cases, the specified access rules do not reflect the real situation. As a consequence, task assignment is handled outside the system and thus might be not properly documented. Furthermore, manual task assignment or adaptation can be complex, time-consuming and error-prone.

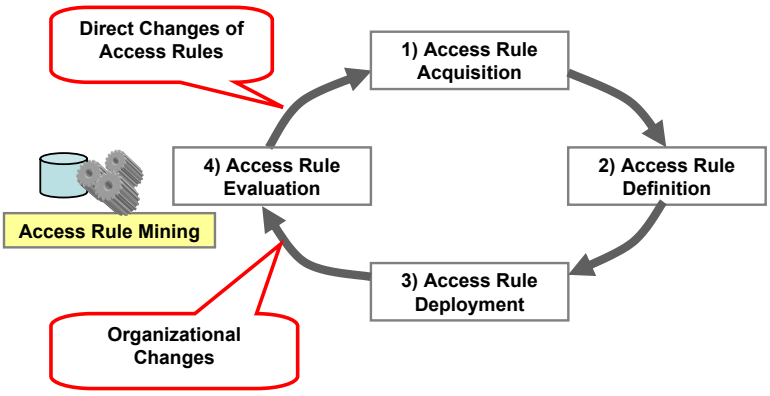


Figure 2. Access rule life cycle

1.2. *Contribution*

In this paper we present our CEOSIS[1] framework for evolving access rules within EIS in a controlled and secure way. We first consider changes of organizational models and evaluate how they affect existing access rules. Then we cope with issues related to direct changes of access rules. As discussed in Section 1.1 both scenarios are highly relevant in practice and should therefore be adequately supported by an EIS.

There are two basic requirements for changes of access rules in EIS. First of all, they must be conducted in a *correct* way; i.e., they must not violate any structural constraints set out by the formalism for access rule specification. To fulfill this requirement we base the definition of access rule changes on an operator tree representation and equip respective operators with formal pre- and post-conditions which ensure their correct application. Second, the *formal semantics* of access rule changes must be specified. Only this guarantees their unambigous application and supports the precise analysis of change effects. To achieve this, we base the semantics of access rule changes on the effects they have on associated *valid actor sets*; i.e., the set of actors who qualify for the particular access rule. This also enables, for example, analysis of access rule change effects on user worklists in PAIS.

Note that this paper significantly extends the work we presented in Rinderle-Ma and Reichert (2008). Besides giving more technical details and additional examples, we complement our results on the change of access rules by a comprehensive discussion on the controlled evolution of organizational models and their effects on access rules. Completely new conceptual results are provided with respect to the support of high-level change operations on access rules. Here, the interesting conclusion is that based on additional knowledge from the associated organizational model the semantics of such high-level operations can be determined without recalculating valid actor sets from scratch.

The remainder of this paper is organized as follows: Section 2 provides fundamentals on organizational models and corresponding access rules. Based on this, Section 3 deals with changes of organizational models and discusses how they might affect related access rules. Following this, we deal with direct access rule changes in EIS. Section 4 provides well-defined change operations for this and Section 5 defines their formal semantics. We extend this work in Section 6 by showing how to exploit organizational knowledge for enabling high-level access rule changes. Section 7 discusses our approach and Section 8 deals with related work. We close with a summary and outlook in Section 9.

[1]CEOSIS: Controlled Evolution of Organizational Structures in Information Systems

2. Organizational models and access rules

In this section we provide information needed for the formal underpinning of our work.

2.1. *Organizational models*

An *organizational meta model* defines entity and relation types based on which concrete organizational structures can be modeled; i.e., the meta model can be seen as the schema which can be instantiated multiple times by concrete *organizational models*. The organizational meta model OMM used in this paper is based on the *Role Based Access Control Model* (RBAC) as described in Ferraiolo *et al.* (2003). Basically, it consists of entity types `OrganizationalUnit`, `Actor`, and `Role` (cf. Fig. 3). Concrete organizational units (e.g., `clinic`) can be hierarchically related to each other based on relation type `is_subordinated`. Similarly, concrete roles can be specialized by introducing corresponding sub roles (relation type `specializes`). Thereby, a sub role inherits all abilities of its superior role, but may have additional ones. Finally, actors may have roles (relation type `has`) and belong to organizational units (relation type `belongs_to`).

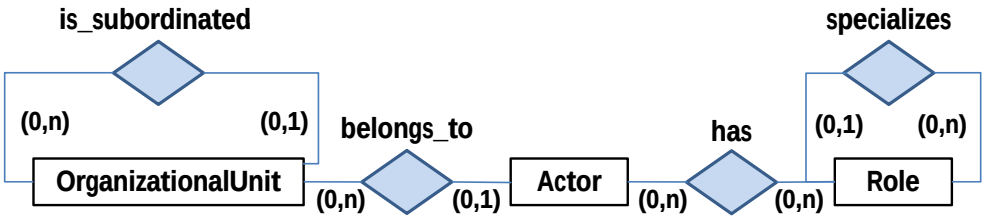


Figure 3. Organizational meta model (in ER notation)

We use this well-established, but rather simple organizational meta model OMM in this paper in order to focus on core issues related to access rule changes. However, subsequent considerations can be transferred to more complex organizational meta models as well; e.g., capturing entities such as abilities or substitution relations (see Rinderle and Reichert (2007) for examples).

Based on OMM, concrete organizational models can be defined; i.e., a concrete organizational model OM constitutes an instance of OMM (cf. Def. 2.1). Consider the example shown in Fig. 4. The depicted organizational model OM comprises the three organizational units (OU) `CallCenter`, `Accounting` and `Marketing`, which are hierarchically subordinated to organizational unit `WebBank`. Role `CAgent`, in turn, is specialized by roles `CAgent_p` and `CAgent_b`. Finally, actors are assigned to roles and belong to one organizational unit (e.g., actor `Black` has role `Secretary` and belongs to organizational unit `Accounting`).

Definition 2.1: Organizational model. *An organizational model is a tuple OM = (Actors, Roles, OrgUnits, has, belongs_to, is_subordinated, specializes), where:*

- *Actors corresponds to the set of actors; i.e., the people performing activities or accessing data objects,*
- *Roles corresponds to the set of organizational roles,*
- *OrgUnits corresponds to the set of organizational units,*
- *has* $\subseteq$ *Roles* $\times$ *Actors captures the relations linking actors to roles,*

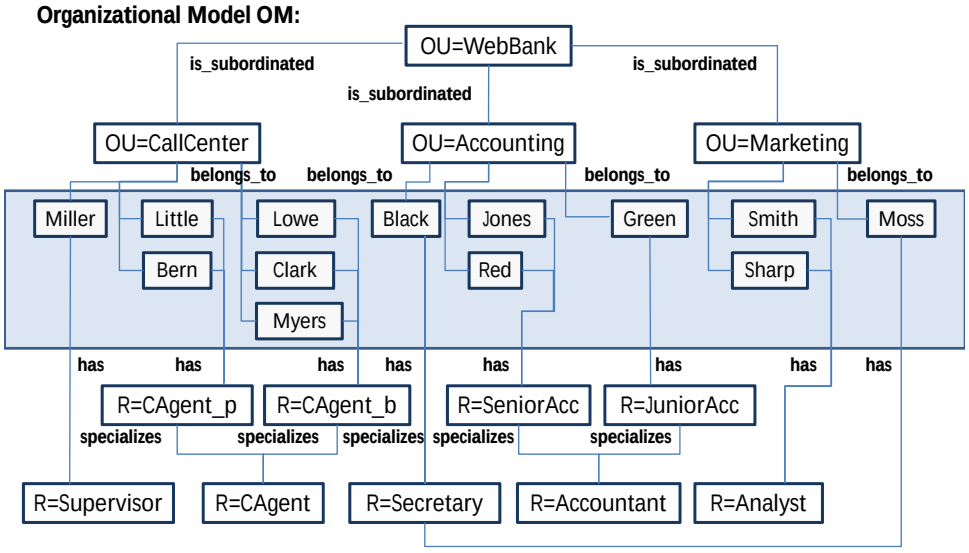


Figure 4. Organizational model for online banking scenario

- *belongs_to* $\subseteq$ *OrgUnits* $\times$ *Actors captures the relations linking actors to organizational units,*
- *is_subordinated* $\subseteq$ *OrgUnits* $\times$ *OrgUnits defines the organizational hierarchy, and*
- *specializes* $\subseteq$ *Roles* $\times$ *Roles defines the role hierarchy.*

Furthermore, two notions on relations are needed in the following:

- *R(x)* := $\{y \in X \mid (x,y) \in R\}$ *for any relation* $R \in \{$*has, belongs_to, specializes, is_subordinated*$\}$
- R^* *is the transitive closure of R*

Consider Fig. 4. An example for generic relation R(x) is given by `has(Secretary) = {Black, Moss}` (with R = '`has`'). The semantics of the two relations `is_subordinated` and `specializes` can be informally defined as follows: All actors belonging to an organizational unit also belong to its superordinated organizational units; e.g., actors `Smith`, `Sharp` and `Moss`, who all belong to organizational unit `Marketing`, also belong to superordinated organizational unit `WebBank` (cf. Fig. 4). Furthermore, if an actor has a particular role she will possess all superior roles as well; e.g., actor `Jones` has role `SeniorAcc` and therefore also posesses role `Accountant`.

2.2. *Access rules*

We provide a notion for access rules and specify their formal semantics. We need this information later in order to be able to reason about access rule changes.

Definition 2.2: Elementary access rule. *Let OM = (Actors, ...) be an organizational model (cf. Def. 2.1). An elementary access rule EAR on OM is defined as follows:*

$EAR \equiv (EAR0 \longleftarrow \tau^*)^1 \mid$
$(EAR1 \longleftarrow Role = r) \mid$
$(EAR2 \longleftarrow OrgUnit = o) \mid$
$(EAR3 \longleftarrow Role+ = r) \mid$
$(EAR4 \longleftarrow OrgUnit+ = o).$

The set of all entities qualifying for an elementary access rule `EAR` *on* `OM` *can be defined as follows:*

$$QualEntities(OM, \text{EAR}) = \begin{cases} r & : \quad EAR = EAR1 \\ o & : \quad EAR = EAR2 \\ specializes^*(r) & : \quad EAR = EAR3 \\ is\_subordinated^*(o) & : \quad EAR = EAR4 \\ \emptyset & : \quad otherwise \end{cases}$$

Formal semantics of elementary access rule EAR is defined over the set of valid actors qualifying for EAR based on OM. We denote this set as VAS(OM, EAR) ⊆ Actors with

- *VAS(OM, EAR0) = ∅*
- *VAS(OM, EAR1) = has(r), i.e., the set of actors having role r.*
- *VAS(OM, EAR2) = belongs_to(o), i.e., the set of actors belonging to unit o.*
- *VAS(OM, EAR3) = has(specializes*(r)); i.e., the set of actors having role r or a more specialized one.*
- *VAS(OM, EAR4) = has(is_subordinated*(o)); i.e., the set of actors belonging to organizational unit o or a subordinated one.*

Fig. 5 gives an example for the use of access rules in a PAIS. It shows a simple business process model representing a direct marketing measurement in an online banking scenario. For each process activity, its associated access rule specifies which actors qualify for carrying out this activity based on organizational model OM (cf. Fig. 4). First of all, a flyer on a new product is sent to private customers by the secretary of the marketing unit. Then, these customers are contacted by a call agent who is specialized on private customers. If a customer shows interest, an appointment for another consulting will be made by the secretary of the accounting unit. This consulting is subsequently done by a senior or junior accountant. Finally, the outcome of the marketing measurement is evaluated by an analyst.

Specifically, Fig. 5 shows two elementary access rules `AR2` and `AR5` and the associated valid actor sets based on organizational model OM. Taking elementary access rules as basis, the general notion of access rule can be formally defined (see below). Here, elementary access rules may be combined by logical operators AND, OR, and NOT. Note that we restrict the complexity of access rules by using negation only in the context of elementary access rules. However, this constitutes no restriction regarding the expressiveness of access rules since any negation contained within an access rule AR can be always pushed to the elementary access rules contained within AR.

Definition 2.3: Access rule. *Let OM be an organizational model. An access rule AR is defined as concatenation of other access rules by using logical operators AND, OR, and NOT. Formally:*

[1] τ^* denotes an empty term.

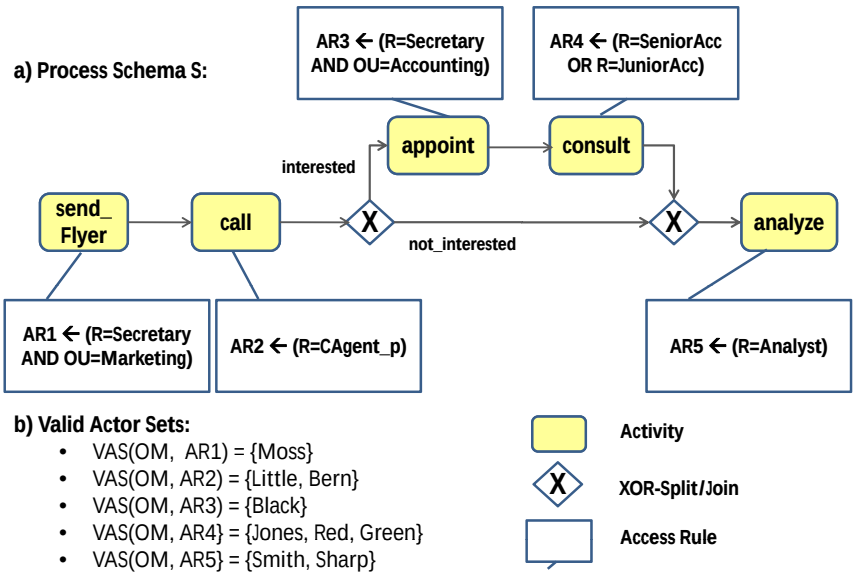


Figure 5. Direct marketing process (in BPMN notation)

$AR \equiv EAR \mid NEAR \mid CAR \mid DAR$
where

- *EAR constitutes an elementary access rule (cf. Def. 2.2),*
- *NEAR ⟵ (NOT (EAR)) where EAR is an elementary access rule,*
- *CAR ⟵ (AR1 AND AR2) where AR1 and AR2 are access rules, and*
- *DAR ⟵ (AR1 OR AR2) where AR1 and AR2 are access rules.*

Formal semantics of elementary access rule EAR has been already given in Def. 2.2, the one of NEAR, CAR and DAR can be defined as follows:

- *VAS(OM,NEAR) = Actors \ VAS(OM,AR) corresponds to the set of actors not qualifying for access rule AR,*
- *VAS(OM,CAR) = VAS(OM,AR1) ∩ VAS(OM,AR2) corresponds to the set of actors qualifying for access rules AR1 and AR2,*
- *VAS(OM,DAR) = VAS(OM,AR1) ∪ VAS(OM,AR2) corresponds to the set of actors qualifying for access rules AR1 or AR2*

$\mathcal{AR}^{OM}$ *denotes the set of all access rules over OM.*

Fig. 5 depicts non-elementary access rules `AR1`, `AR3`, and `AR4`. Regarding `AR1`, for example, valid actor set `VAS(OM, AR1)` corresponds to intersection of `VAS(OM, R=Secretary)` = {`Black,Moss`} and `VAS(OM, OU=Marketing)` = {`Smith,Sharp,Moss`}.

3. Organizational changes and their effects on access rules

In this section we present a framework for modifying organizational models while controlling the effects on access rules at the same time.

3.1. *On changing organizational models*

Assume that our online bank has to streamline its organization due to a financial crisis (cf. Fig. 4 and Fig. 5). Necessary actions are depicted in Fig. 6. Due to the reorganization, there is no longer a distinction between call agents serving private customers and those dealing with business customers; i.e., these two roles are now merged into role `CAgent`. Furthermore, only one secretary for the whole `WebBank` remains; i.e., one secretary is laid off and role `Secretary` is reassigned to organizational unit `WebBank`. The challenge now is to realize these changes for the given organizational model, which is implemented and used by one or more EIS.

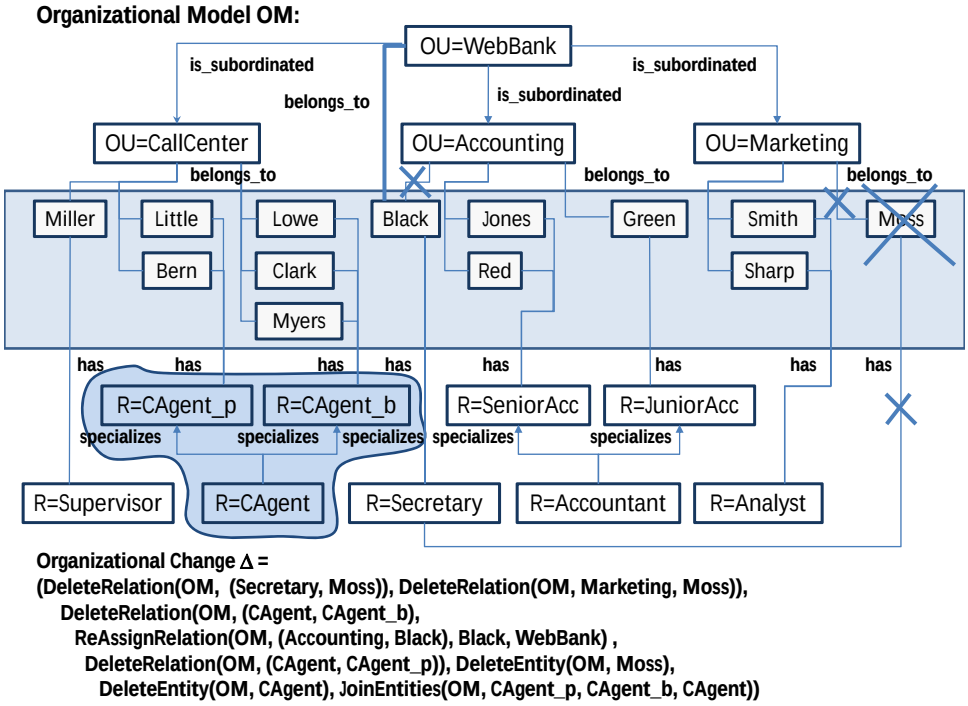


Figure 6. Changes after streamlining the organization

In order to be able to express all relevant kinds of changes of an organizational model OM, CEOSIS provides a complete set of basic change operations[1] with well defined semantics; e.g., for creating or deleting organizational entities as well as the relations between them. For each change operation there are formal pre- and postconditions, which enables preservation of the correctness properties of organizational model OM when applying the operation(s) to it (assuming that OM was a correct model before). In addition to these basic change operations, CEOSIS provides common high-level operations in order to facilitate change definition and to capture more semantics about model changes. As example of such a high-level operation consider the join of two entities (e.g., a fusion of two organizational units as depicted in Fig. 6). Table 1 depicts selected basic as well as high-level operations for changing organizational models in CEOSIS.

At the bottom of Fig. 6 the change operations needed to realize the desired streamlining actions (i.e., adaptations of the given organizational model of our running example)

[1]Completeness means that a given organizational model OM can be applied to an arbitrary other organizational model OM' using the given set of change operations.

Table 1. *Selection of change operations on organizational models (Rinderle and Reichert (2007))*

Let Entities:=Actors ∪ Roles ∪ OrgUnits and Relations:= $\{(e_1, e_2) \mid e_1, e_2 \in Entities \wedge (e_1, e_2) \in \{has, belongs\_to, is\_subordinated, specializes\}\}$	
`CreateEntity:` $\mathcal{OM} \times Identifier \mapsto \mathcal{OM}$ with `CreateEntitiy(OM, eId) = OM'`	
Preconditions:	• eId ∉ Entities
Postconditions:	• Entities' = Entities ∪ {eId} • Relations' = Relations
`DeleteEntity:` $\mathcal{OM} \times \mathcal{E} \mapsto \mathcal{OM}$ with `DeleteEntity(OM, e) = OM'`	
Preconditions:	• e ∈ Entities • ∄ rel = (e1, e2) ∈ Relations with e1 = e ∨ e2 = e
Postconditions:	• Entities' = Entities \ {e} • Relations' = Relations
`CreateRelation:` $\mathcal{OM} \times \mathcal{E} \times \mathcal{E} \mapsto \mathcal{OM}$ with `CreateRelation(OM, e1, e2) = OM')`	
Preconditions:	• e1, e2 ∈ Entities • (e1, e2) ∉ Relations • e2 ∉ $R^*(e1)$ with R ∈ {specializes, is_subordinated}
Postconditions:	• Entities' = Entities • Relations' = Relations ∪ {(e1, e2)}
`DeleteRelation:` $\mathcal{OM} \times \mathcal{R_E} \mapsto \mathcal{OM}$ with `DeleteRelation(OM, relation) = OM'`	
Preconditions:	• relation ∈ Relations
Postconditions:	• Entities' = Entities • Relations' = Relations \ {relation}
`ReAssignRelaton:` $\mathcal{OM} \times \mathcal{R_E} \times \mathcal{E} \times \mathcal{E} \mapsto \mathcal{OM}$ with `ReAssignRelation(OM, r, e, eNew) = OM'`	
Preconditions:	• r = (e1, e2) ∈ Relations • e = e1 ∨ e = e2 (w.l.o.g., we assume e = e1 in the following) • eNew ∈ Entities • e ∈ Actors ⟹ e2 ∉ Actors • e ∈ Roles ⟹ e2 ∉ OrgUnits • e ∈ OrgUnits ⟹ e2 ∉ Roles • e2 ∉ $R^*(e1)$ with R ∈ {specializes, is_subordinated}
Postconditions:	• Relations' = Relations ∪ {(e, eNew)} \ {(e1, e2)}
Postconditions:	• Entities' = Entities
`JoinEntities:` $\mathcal{OM} \times \mathcal{E} \times \mathcal{E} \times Identifiers \mapsto \mathcal{OM}$ with `JoinEntities(OM, e1, e2, nId) = OM'`	
Preconditions:	• e1, e2 ∈ Entities • nId ∉ Entities • e1, e2 ∉ Actors
Postconditions:	• CreateEntity(OM, eNew) := eNew • ∀ (e, e1) ∈ Relations: ReassignRelation(OM, (e, e1), e1, eNew) • ∀ (e, e2) ∈ Relations: ReassignRelation(OM, (e, e2), e2, eNew) • ∀ (e1, e) ∈ Relations: ReassignRelation(OM, (e1, e), e1, eNew) • ∀ (e, e2) ∈ Relations: ReassignRelation(OM, (e, e1), e2, eNew) • DeleteEntity(OM, e1) • DeleteEntity(OM, e2)

are depicted. For instance, there is one operation which reassigns actor `Black` from organizational unit `Accounting` to organizational unit `WebBank`. Merging the organizational roles `CAgent_p` and `CAgent_b`, in turn, is accomplished by first deleting superordinated role `CAgent` and subsequently joining roles `CAgent_p` and `CAgent_b` into new role `CAgent`. (There exist other options to realize this merger which are omitted here.) Fig. 7 depicts the organizational model OM' that results after implementing the desired reorganization.

3.2. *Effects on access rules*

Using the change operations presented in Table 1, we are able to modify organizational models while preserving certain model properties. However, this does not guarantee any control of possible side-effects of such model changes on, for example, access rules. Consider again our process example in Fig. 8. After changing the underlying organizational model, for example, access rule `AR2` cannot be resolved anymore. Reason is that role `CAgent_p`, to which access rule `AR2` refers, is no longer present in OM'. We denote this problem as *dangling reference*. Another problem is present for access rules `AR1` and `AR3`. Even though they do not contain dangling references (and thus are resolvable over OM'), their valid actor sets become empty on OM'. This might raise severe problems when access rules are resolved and the corresponding tasks shall be assigned to the work-

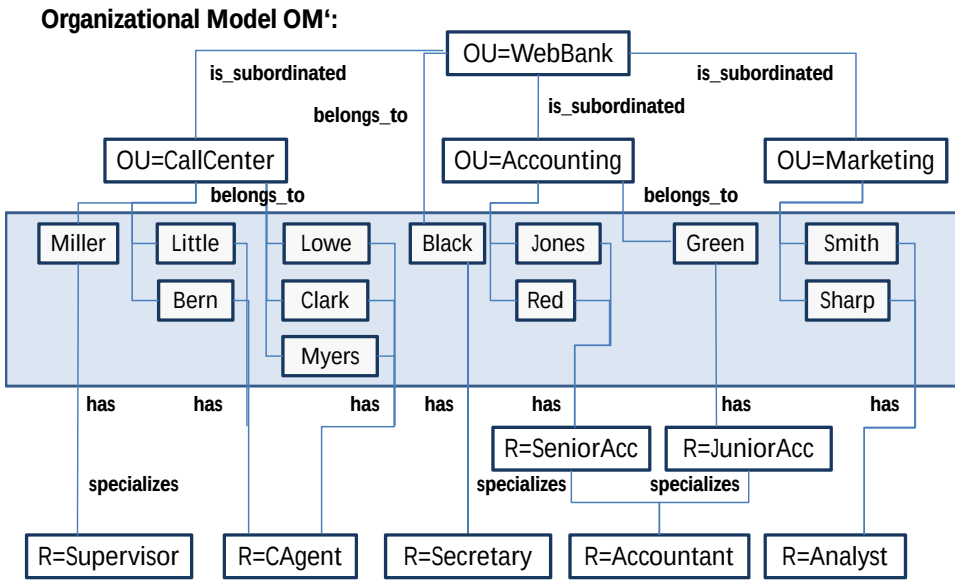


Figure 7. Modified organizational model

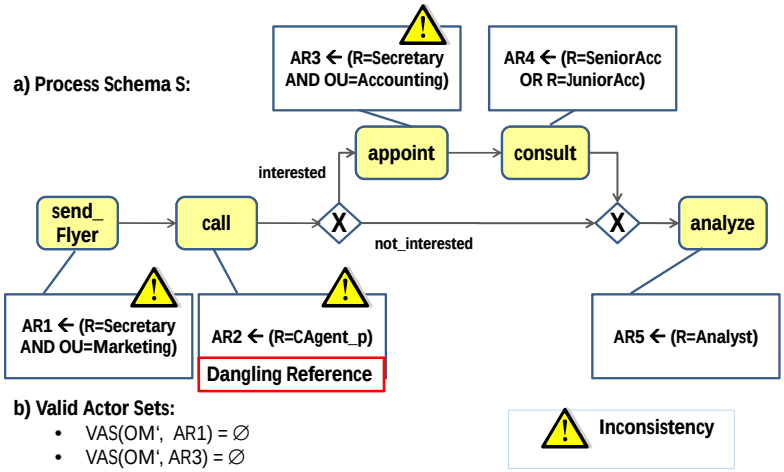


Figure 8. Effects on direct marketing process

lists of qualifying actors.[1] If the valid actor set of an access rule becomes empty for a process activity, either the process will be blocked at this point or the PAIS will assign the corresponding work item to other, potentially non-authorized users (e.g., the system administrator). Particularly in the context of sensitive data such false task assignments constitute a severe security threat.

Consequently, the challenge is to control the side-effects of organizational changes on access rules. Specifically, access rules have to be checked for dangling references and empty valid actor sets. In the following we give formal notions for both problems. Based on these notions, we provide a criterion which allows us to decide whether or not an access rule `AR` is *valid* with respect to a given organizational model OM. We call an access rule valid on OM if the following two conditions hold:

[1]In Process-Aware Information Systems, generally, actors access activities via their worklists.

(1) AR does not contain *dangling references*, i.e., it does not refer to entities not present in OM. Formally:

$$DanglingRef(OM, \mathrm{AR}) = \begin{cases} \mathrm{FALSE} & if\ \forall\ \mathrm{EAR}\ \in\ \mathrm{AR} : QualEntities(OM, \mathrm{EAR}) \neq \emptyset \\ \mathrm{TRUE} & otherwise \end{cases}$$

where the notion EAR $\in$ AR expresses that elementary access rule EAR is contained within 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 component in order to ensure that objects remain accessible or tasks remain doable. Formally:

$$Resolv(OM, \mathrm{AR}) = \begin{cases} \mathrm{TRUE}\ if\ VAS(\mathrm{OM}, \mathrm{AR}) \neq\ \emptyset \\ \mathrm{FALSE}\ otherwise \end{cases}$$

Definition 3.1: 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:*

$$Valid(OM,\ \mathtt{AR}) = \mathtt{TRUE} :\Longleftrightarrow (DanglingRef(OM,\ \mathtt{AR}) = \mathtt{FALSE} \wedge Resolv(OM,\ \mathtt{AR}) = \mathtt{TRUE})$$

3.3. *Adapting access rules after organizational changes*

Determining potential problems in connection with organizational changes (e.g., empty valid actor sets) is a first important step. Another one is to cope with identified problems in an adequate and efficient way. CEOSIS offers a number of sophisticated adaptation policies in this context. In the following, we exemplarily provide one of these policies, which enables adaptations of access rules when applying the high-level change operation JoinEntities (cf. Table 1) to an organizational model. Note that adaptation policies are applied in a semi-automatic manner; i.e., the system only recommends adaptations of the access rules affected by an organizational change, but the final decision has to be made by the user.

Adaptation Policy (Avoiding dangling references). *Let OM = (Entities, Relations) be an organizational model and let* AR *be a valid access rule on OM. Let further* Δ_{op} = JoinEntities(OM, ...) *be a high-level change operation which transforms OM into another organizational 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* Δ_{op} *the corresponding adaptation rule* δ_{AR} *turns out as follows:*

Δ_{op} = JoinEntities(OM, e_1, e_2, *newEnt*) $\Longrightarrow$
δ_{AR}: $\forall$ EAR $\in$ AR *with* EAR *refers to* e_i *(i = 1,2): substitute* e_i *by newEnt*

Consider again access rule AR2 in Fig. 8, which contains a dangling reference to CAgent_p based on OM'. This situation has occured after joining CAgent_p and CAgent_b. Thus, when applying the above adaptation policy, the reference to CAgent_p is substituted by a reference to the newly introduced entity CAgent. Fig. 9 depicts the result.

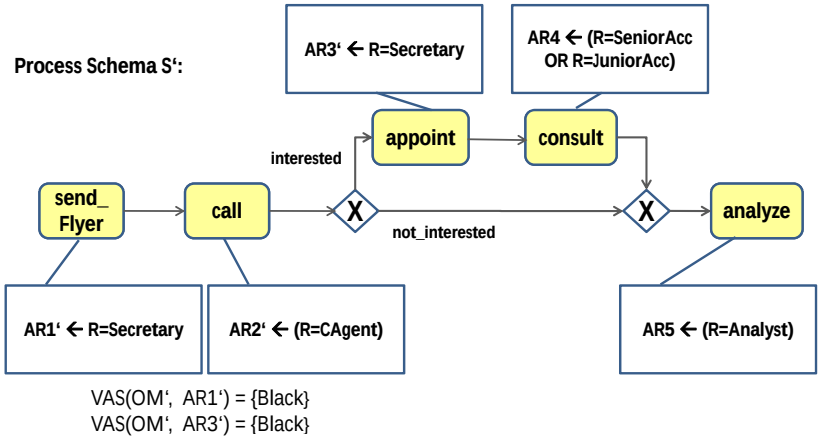


Figure 9. Adapted direct marketing process

Even if the problem of dangling references is satisfactorily solved we still might be confronted with non-resolvable access rules when changing an organizational model. This, in turn, could cause runtime errors or at least runtime delays (e.g., if activities cannot be processed immediately due to the absence of qualifying actors). Worst case severe security problems can result (e.g., in case a manual activity without qualifying actors is offered to the system or process administrator as in some commercial workflow systems).

Let OM be an organizational model which is transformed into another organizational model OM' by change Δ. Furthermore, let `AR` be an access rule on 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'. Fig. 10 depicts possible relations between the valid actor set of `AR` on OM (i.e., `VAS(OM,AR)`) and the valid actor set of `AR` on OM' (i.e., `VAS(OM',AR)`): In Fig. 10a 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. Consequently, `AR` is still resolvable over OM' in this case, and does not require any adaptation of worklists or lists of qualified actors afterwards. Fig. 10b depicts the scenario in which the valid actor set is expanded when migrating `AR` to OM'. In practice this might 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 could be also reduced due to a model change as depicted in Fig. 10c. 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 scenario depicted in Fig. 10c, the valid actor set of `AR` on OM' might become empty; `AR` then would no longer resolvable on OM'.

Another scenario is depicted in Fig. 10d. Here, neither `VAS(OM,AR)` is a subset of `VAS(OM',AR)` nor vice versa. Generally, we can further distinguish between sub-cases d1 and d2. For sub-case d1 there still exist actors contained in both valid actor sets, i.e., intersection of `VAS(OM,AR)` and `VAS(OM',AR)` is non-empty. For such a case, we first 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 in the context of sub-case d2 the privileges associated with `AR` have to be removed for all actors from `VAS(OM,AR)` and be added for those belonging to `VAS(OM',AR)`.

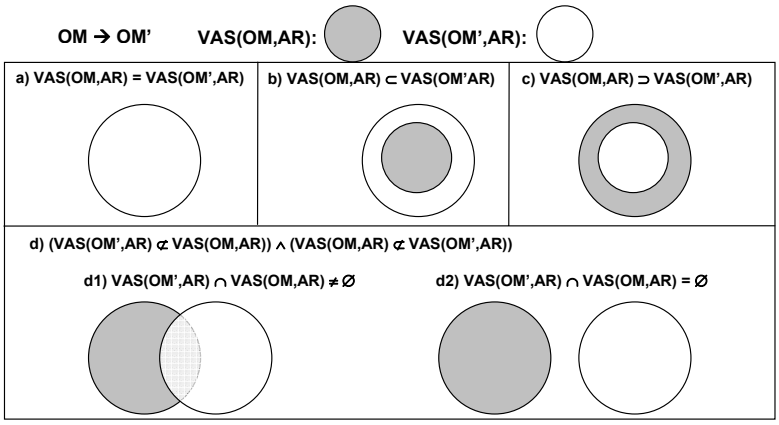


Figure 10. Changing organizational models and migrating access rules

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. For this, Rinderle and Reichert (2007) provided a comprehensive framework for basic as well as high-level change operations on organizational models. Regarding our online banking scenario (cf. Fig. 9), users would be supported by making them aware of possibly empty valid actor sets for `AR1` and `AR3` on the new organizational model. Adequate transformations could be as shown in Fig. 9.

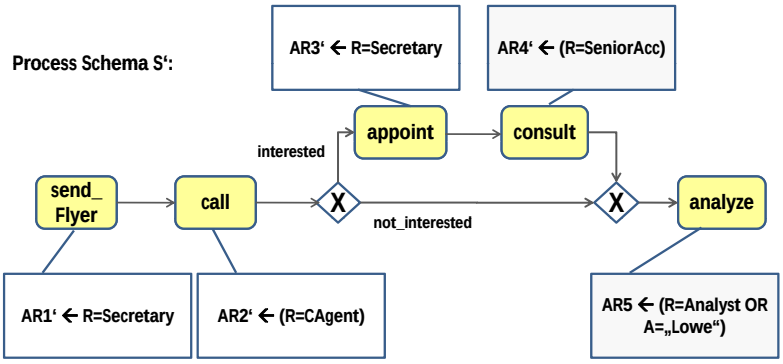
4. Direct access rule changes

The adaptation of access rules in conjunction with changes of organizational models constitutes only one use case. As discussed in Section 1 it must be also possible to directly change access rules. Consider the scenario depicted in Fig. 11. As additional streamlining action, is has been decided to let the consultancy only be done by senior accountants in order to save time. Furthermore, actor `Lowe` is now supposed to assist the analyst since it has turned out that `Lowe` has practical experiences with data mining. Both considerations are reflected in changed access rules `AR4` and `AR5` (cf. Fig. 11).

In order to be able to apply direct access rule changes within the EIS, first of all, we have to define respective operations (e.g., for deleting AND-terms within an access rule). The formal semantics of these change operations is presented in Section 5. It is based on the valid actor sets of the access rules before and after the changes.

4.1. *An operator-tree-based representation for access rules*

In order to precisely define access rule changes, it is necessary to base their definition on a representation other than the intuitive one presented in Definition 2.3. Using the notion given in Definition 2.3 it would be difficult to express at which substructure level of nested access rule structures, a new AND-term shall be added. Thus we have to find a representation which enables convenient access to any substructure level of an access rule. For an access rule `AR`, a suitable representation for this is provided by *operator tree* $\mathcal{OP}_{AR}$ = (O,L). Here O denotes the set of all operator nodes and L denotes the set of all elementary access rules `AR` is built of. $\mathcal{OP}_{AR}$ can be determined similarly to building the operator-tree for a complex expression or SQL-statement. An example of an access



b) Access rule change Δ =
(deleteTerm(OP_{AR4}, (R=JuniorAcc)), addTerm(OP_{AR5}, (A=„Lowe“), OR)

c) Valid Actor Sets:

- VAS(OM‘, AR4‘) = {Jones, Red}
- VAS(OM‘, AR5‘) = {Smith, Sharp, Lowe}

Figure 11. Changing access rules AR4 and AR5

rule with corresponding operator-tree is depicted in Fig. 12. Generally, an operator tree does not necessarily need to be balanced.

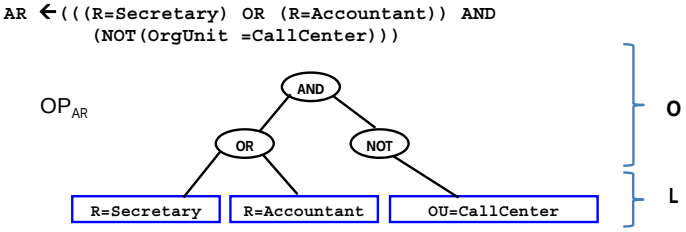


Figure 12. Access rule and operator tree

$\mathcal{OP}_{AR}$ = (O,L) has the following characteristics.

- $\mathcal{OP}_{AR}$ is a binary tree.
- O corresponds to the set of non-leaf nodes (including the tree root) and L corresponds to leaf nodes.
- The tree has to be traversed in Inorder to build the associated access rule term.
- NOT is only used as direct predecessor node of a leaf (remember that NOT can be only used in the context of elementary access rules in CEOSIS, cf. Definition 2.3).

How to build the operator tree for a given access rule is shown in the next section.

4.2. *Basic access rule changes*

We introduce a complete set of basic change operations on access rules[1] and on their operator-tree representation respectively. Before, we define basic notions on operator

[1]Complete means that any access rule AR $\in$ $\mathcal{AR}^{OM}$ can be transformed into any other access rule AR' $\in$ $\mathcal{AR}^{OM}$ by applying a sequence of change operations $<op_1, \ldots, op_n>$.

trees:

Definition 4.1: Functions on op trees. *Let* AR $\in$ $\mathcal{AR}^{OM}$ *be an access rule and let* $\mathcal{OP}_{AR}$ *be its operator tree. Then:*

- *The empty tree* τ *consists of one void node; i.e., it reflects empty access rule EAR0* $\longleftarrow$ τ^*.
- *Let* $\mathcal{OP}^{OM}$ *denote the set of all operator trees for access rules over an organizational model OM. Let further* $\mathcal{N}$ *denote the total set of nodes belonging to any operator tree from* $\mathcal{OP}^{OM}$. *Then:*
  - *pred:* $\mathcal{OP}^{OM} \times \mathcal{N} \mapsto \mathcal{N}$ *with pred(*$\mathcal{OP}_{AR}$*, n) = p determines direct predecessor node p of node n in* $\mathcal{OP}_{AR}$
  - *root:* $\mathcal{OP}^{OM} \mapsto \mathcal{N}$ *determines the root node of operator tree* $\mathcal{OP}_{AR}$.
- *Merge:* $\mathcal{OP}^{OM} \times \mathcal{OP}^{OM} \times \{AND, OR, VOID^1\} \mapsto \mathcal{OP}^{OM}$
  Merge(S,T,op=$[AND|OR|VOID]$*) = S' merges two operator trees S and T using op, where* $T = \mathcal{OP}_{EAR}$ *with EAR being an elementary access rule. Root of S' is op, left child tree is S, right child tree is* $\mathcal{OP}_{EAR}$
- *Substitute:* $\mathcal{OP}^{OM} \times \mathcal{OP}^{OM} \times \mathcal{OP}^{OM} \mapsto \mathcal{OP}^{OM}$
  Substitute(T,S,S') = T' substitutes sub tree S in T by sub tree S' resulting in T' (cf. Fig. 13, Step 1).
- *Optimize:* $\mathcal{OP}_{AR} \mapsto \mathcal{OP}_{AR}$
  Optimize(T) = T' works as follows: If operator tree T contains empty tree τ *then T can be optimized by merging* τ *and its sibling tree S resulting in optimized tree T'. More precisely: Let O be the predecessor of* τ *and S. Then O,* τ*, and S can be merged to S (cf. Fig. 13, Step 2).*

CEOSIS provides a set of basic change operations that allow for adding, deleting, and negating terms within operator trees. We claim that these change operations can only be applied to correct access rules resulting in correct access rules again (i.e., access rules belonging to $\mathcal{AR}^{OM}$). Structural correctness is preserved by formal pre- and postconditions defined for each change operation. In the context of our ADEPT2 process management technology (Rinderle *et al.* (2004a,b)), for example, we additionally ensure compliance with the underlying organizational model OM by forbidding access rule changes which refer to entities not being present in OM.

Definition 4.2: Basic change operations on access rules. *Let* $AR \in \mathcal{AR}^{OM}$ *be an access rule with operator tree* $\mathcal{OP}_{AR}$. *Let further EAR be an elementary access rule with operator tree* $\mathcal{OP}_{EAR}$. *Assume that AR is transformed into another access rule* $AR' \in \mathcal{AR}^{OM}$ *(represented by* $\mathcal{OP}_{AR'}$*) by applying change op* $\in \{addTerm, deleteTerm, negateTerm\}$ *with*

- *addTerm(*$\mathcal{OP}_{AR}$*, S,* $[AND|OR|VOID]$*,EAR)* $= \mathcal{OP}_{AR'}$
  Precond.: S is sub-tree of $\mathcal{OP}_{AR}$
  Postcond.: $\mathcal{OP}_{AR'}$ *= Optimize(Substitute(*$\mathcal{OP}_{AR}$*,S,Merge(S,*$\mathcal{OP}_{EAR}$*)))*
- *deleteTerm(*$\mathcal{OP}_{AR}$*,S)* $= \mathcal{OP}_{AR'}$
  Precond.: S is sub-tree of $\mathcal{OP}_{AR}$ *and S does not contain the root node (the root node is deleted by tree merging if a direct sub tree of the root node is deleted)*
  Postcond.: $\mathcal{OP}_{AR'}$ *= Optimize(Substitute(*$\mathcal{OP}_{AR}$*,S,*τ*))*
- *negateTerm(*$\mathcal{OP}_{AR}$*,S)* $= \mathcal{OP}_{AR'}$
  Precond.: S is leaf node; i.e., $S = \mathcal{OP}_{EAR}$ *and pred(*$\mathcal{OP}_{AR}$*,EAR)* $\neq NOT$ *(the second condition is necessary to achieve operator trees where NOT is only used in connection with leaf nodes according to the definition of access rules.).*
  Postcond.: $\mathcal{OP}_{AR'}$ *= Substitute(*$\mathcal{OP}_{AR}$*,S,*$\mathcal{OP}_{NOT(EAR)}$*)*

Figure 13 depicts an example for applying the delete operation. First, the sub tree reflecting elementary access rule EAR $\leftarrow$ R=Secretary (cf. Figure 12) is substituted by empty tree τ. Then the optimization function is run on the resulting tree which eliminates

[1]VOID represents the empty operator.

τ by lifting up remaining elementary access rule `EAR'` $\leftarrow$ `R=SeniorAcc` to the next level within the operator tree.

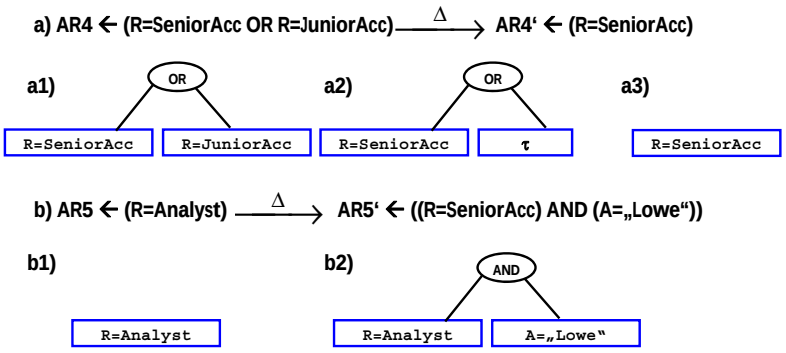


Figure 13. Delete operation with subsequent optimization

Generally, a sequence of basic change operations $<op_1, \ldots, op_n>$ can be used to built up the operator tree for an access rule starting with the empty tree. Consider access rule `AR` as given in Fig. 12. Starting from empty tree τ, the following basic change operations build $\mathcal{OP}_{AR}$:

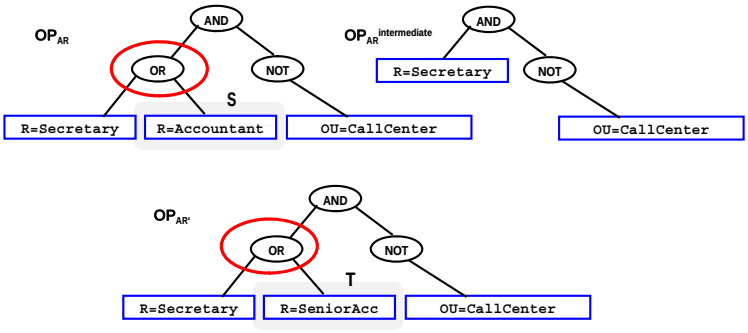
op_1: $\mathcal{OP}_1$ = addTerm(τ, τ, VOID, R=Secretary)
op_2: $\mathcal{OP}_2$ = addTerm($\mathcal{OP}_1$, $\mathcal{OP}_1$, OR, R=Accountant)
op_3: $\mathcal{OP}_3$ = addTerm($\mathcal{OP}_2$, $\mathcal{OP}_2$, AND, OU=CallCenter)
op_4: $\mathcal{OP}_{AR}$ = negateTerm($\mathcal{OP}_3$, $\mathcal{OP}_{OU=CallCenter}$)

Formal semantics of these change operations is presented in Section 5.

4.3. *High-level access rule changes*

For more sophisticated user support CEOSIS additionally provides *high-level change operations*. Their definition is based on the elementary change operations introduced in Definition 4.2. As example consider high-level change `substituteAccessRules(`$\mathcal{OP}_{AR}$`, S ,T)`. Here sub tree S of operator tree $\mathcal{OP}_{AR}$ is substituted by another sub tree T. At access rule level this means to substitute a part of the access rule by another access rule. In Fig. 14, for example, sub tree S (representing the elementary access rule `R=Accountant`) is substituted by sub tree T (representing the elementary access rule `R=SeniorAcc`).

Basically, the substitution of access rules can be realized by applying operation `deleteTerm(`$\mathcal{OP}_{AR}$`,S)` first, followed by a sequence of addTerm operations. They build up T within $\mathcal{OP}_{AR}$ by inserting the elementary access rules out of which T consists (cf. Fig. 14). The pre- and postconditions of high-level change operations can then be derived by aggregating the pre- and postconditions of the constituting basic change operations. One important benefit regarding the ease-of-use of high-level change operations is that all "details" of applying the basic change operations are encapsulated within the high-level operation. For example, if we want to be able to add the new terms in the right way afterwards, it will become necessary to "memorize" some operator nodes before deleting the terms to be substituted. In Fig. 14, for example, it has to be memorized that S was connected to $\mathcal{OP}_{\text{R=Secretary}}$ by an OR-operator. Without memorizing this information explicitly, it will be lost after deleting S as it can bee seen from $\mathcal{OP}_{\text{AR}}^{intermediate}$.



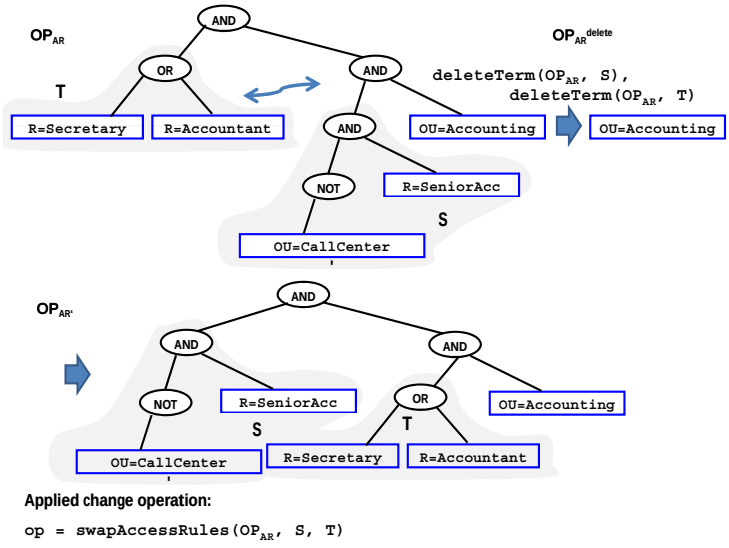
Applied change operation:

op = substituteAccessRule(OP_{AR}, R=Accountant, R=SeniorAcc) =
(deleteTerm(OP_{AR}, S),
addTerm($OP_{AR}^{intermediate}$, $OP_{R=Secretary}$, OR, T))
with S = $OP_{R=Accountant}$ and T = $OP_{R=SeniorAcc}$

Figure 14. Example for substituting access rules

Another high-level change operation is swapAccessRules($\mathcal{OP}_{AR}$, S ,T) which swaps sub trees S and T within $\mathcal{OP}$. An example is depicted in Figure 15. Again swapAccessRules($\mathcal{OP}_{AR}$, S ,T) can be expressed by basic change operations; i.e.:

swapAccessRules($\mathcal{OP}_{AR}$, S ,T) =
(deleteTerm($\mathcal{OP}_{AR}$, S), deleteTerm($\mathcal{OP}_{AR'}$, T),
addTerm($\mathcal{OP}_{AR}^{delete}$, $\mathcal{OP}_{AR}^{delete}$, AND, T),
addTerm($\mathcal{OP}_{AR}^{intermediate}$, $\mathcal{OP}_{AR}^{intermediate}$, AND, S))



Applied change operation:

op = swapAccessRules(OP_{AR}, S, T)

Figure 15. Example for swapping operator trees

Again it has to be memorized which operator nodes were used to fix the sub trees to be swapped within the operator tree (in Fig. 15, for example, S and T are fixed by an

AND-operator). The semantics of high-level change operations is discussed in Section 6.

5. Semantics of access rule changes

Formal semantics of access rule changes can be expressed based on the effects the changes have on valid actor sets (cf. Section 2.2). In particular, we are interested in statements such as "valid actor set of access rule `AR` is reduced, expanded, or not affected by the change". Note that for the following considerations on change semantics we assume direct access rule changes; i.e., the underlying organizational model has not been modified.

Definition 5.1: Reduction / expansion of actor sets. *Let $AR \in \mathcal{AR}^{OM}$ be an access rule (over organizational model OM) and let Δ_{AR} be a change which transforms AR into another access rule AR' $\in \mathcal{AR}^{OM}$. Then: the effect of Δ_{AR} on AR is called*

- *reduction iff VAS(OM,AR') $\subset$ VAS(OM,AR)*
- *expansion iff VAS(OM,AR') $\supset$ VAS(OM,AR)*
- *zero_effect iff VAS(OM,AR') = VAS (OM,AR)*

5.1. *Root level and substructure level changes*

For elementary access rules the analysis of change effects is easy to accomplish. For example, let `EAR` $\longleftarrow$ `(R=Secretary)` and `EAR'` $\longleftarrow$ `(R=Accountant)` be two elementary access rules with operator trees $\mathcal{OP}_{EAR}$ and $\mathcal{OP}_{EAR'}$ respectively. Let further change *op = addTerm($\mathcal{OP}_{EAR}$, $\mathcal{OP}_{EAR}$, AND,* `EAR'`*)* = `AR` transform `EAR` into `AR`. Then the effect on the actor set of `EAR` is a *reduction*, more precisely, the actor sets of `AR` can be determined as intersection of the actor sets of `EAR` and `EAR'`.

Things will become more complicated if the access rules to be changed are more complex. Consider, for example, access rule `AR` in Fig. 12 and elementary rule `EAR'` $\longleftarrow$ `(OU=Accounting)`. If we apply change operation op^a = `addTerm(`$\mathcal{OP}_{AR}$`,`$\mathcal{OP}_{AR}$`,AND,EAR')` (cf. Figure 16b) the effect can be determined as intersection of the actor sets of `AR` and `EAR'` again. However, when applying change operation op^b = `addTerm(`$\mathcal{OP}_{AR}$`,` $\mathcal{OP}_{NOT(OrgUnit='administration')}$`, AND, EAR')` (cf. Fig. 16c), effects on the valid actors sets of `AR` cannot be determined straightforward. Note that op^a operates at the *root level* of `AR` (the formal meaning is described in the following), whereas op^b operates at a *substructure level* of `AR`. The notions of root level and substructure level changes of access rules are presented in Definition 5.2.

Definition 5.2: Root vs. substructure level changes. *Let $\mathcal{OP}_{AR}$ be the operator tree representation of access rule $AR \in \mathcal{AR}^{OM}$ and let* `op` *be a basic change operation which transforms AR into another access rule AR' $\in \mathcal{AR}^{OM}$. Then we denote* `op` *as*

- *root level change if either a new root is added to operator tree $\mathcal{OP}_{AR}$ or the root of $\mathcal{OP}_{AR}$ is deleted (e.g., by tree merging after applying the delete operation). This holds for*
  - *op = addTerm($\mathcal{OP}_{AR}$,$\mathcal{OP}_{AR}$,[AND|OR|VOID],EAR)*
  - *op = deleteTerm($\mathcal{OP}_{AR}$,S) with pred(root(S)) = root($\mathcal{OP}_{AR}$)*
  - *op = negateTerm($\mathcal{OP}_{EAR}$, $\mathcal{OP}_{EAR}$) with EAR being an elementary access rule*
- *substructure level change otherwise*

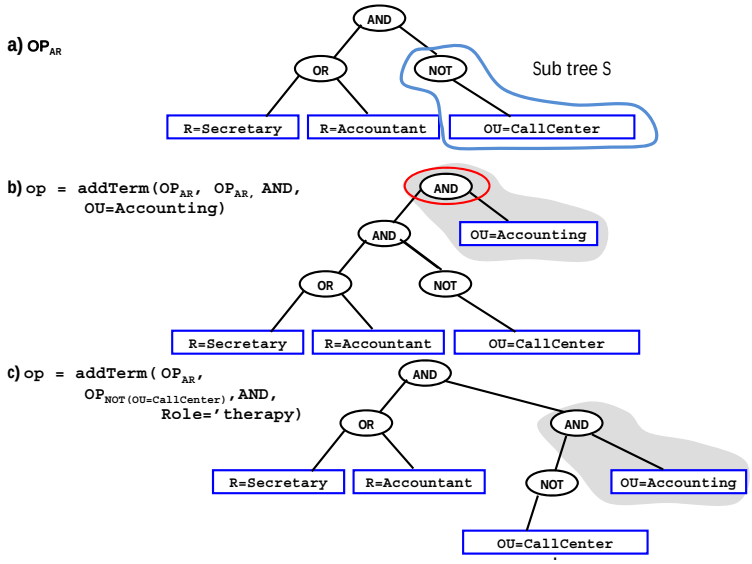


Figure 16. Access rule changes

The root level change depicted in Fig. 16b shows that operator tree $\mathcal{OP}_{AR}$ grows when adding a new root node, whereas the substructure level change (cf. Fig. 16c) is conducted by substituting sub tree S = $\mathcal{OP}_{NOT(OrgUnit='administration')}$ by the sub tree S' = $\mathcal{OP}_{(NOT(OrgUnit='administration'))AND(Role='therapist')}$. As can be seen new root AND has been added to S.

5.2. *Basic access rule changes*

The basic idea of our approach for determining the effects of access rule changes on valid actor sets (in terms of *reduction*, *expansion*, or *zero_effect*) is as follows: First, it must be checked how root level changes affect valid actor sets of respective access rules. Second, for substructure level changes the following observation can be made: A substructure level change is a root change regarding the *affected sub tree* (cf. Fig. 18); i.e., we can determine effect $e \in$ {*reduction, expansion, or zero_effect*} on the affected sub tree. Effect e can then be "propagated" upwards to the root of the new operator tree. Then, a fundamental question is, for example, whether a reduction on the affected sub tree remains a reduction when being propagated to the root level.

Thus, for determining the effects of basic access rule changes on valid actor sets a first step is to present the effects of root level changes on operator trees.

Proposition 5.3: Effects of root level changes. *Consider the following elements:*

- $\mathcal{OP}_{AR}$*: Operator tree of access rule AR over organizational model OM with S and T being sub trees of* $\mathcal{OP}_{AR}$ *with pred(*$\mathcal{OP}_{AR}$*,root(S)) = pred(*$\mathcal{OP}_{AR}$*,root(T)) = root(*$\mathcal{OP}_{AR}$*). S corresponds to access rule* AR_S *and T to access rule* AR_T
- *EAR: Elementary access rule with operator tree* $\mathcal{OP}_{EAR}$
- *op: root level change which transforms* `AR` *into another access rule AR' with operator tree* $\mathcal{OP}_{AR'}$

Then: The effect (i.e., formal semantics) of operation `op` *on* $\mathcal{OP}_{AR}$ *can be determined according to the following table (see Fig. 17 for a graphical illustration):*

op: addTerm($\mathcal{OP}_{AR}$,$\mathcal{OP}_{AR}$,[AND|OR],EAR) = $\mathcal{OP}_{AR'}$
• *op: addTerm($\mathcal{OP}_{AR}$, $\mathcal{OP}_{AR}$, AND, EAR) = $\mathcal{OP}_{AR'}$: VAS(OM,AR') = VAS(OM,AR) ∩ VAS(OM, EAR)*
⟹ **Reduction**
• *op: addTerm($\mathcal{OP}_{AR}$, $\mathcal{OP}_{AR}$, OR, EAR) = $\mathcal{OP}_{AR'}$: VAS(OM,AR') = VAS(OM,AR) ∪ VAS(OM,EAR)*
⟹ **Expansion**
op: deleteTerm($\mathcal{OP}_{AR}$,S) = $\mathcal{OP}_{AR'}$
• *root($\mathcal{OP}_{AR}$) = AND: VAS(OM,AR) = VAS(OM,AR_T) ∩ VAS(OM,AR_S) ⊆ VAS(OM,AR_T) = VAS(OM,AR')*
⟹ **Expansion**
• *root($\mathcal{OP}_{AR}$) = OR: VAS(OM,AR) = VAS(OM,T) ∪ VAS(OM,S) ⊇ VAS(OM,T) = VAS(OM,AR')*
⟹ **Reduction**
op: negateTerm($\mathcal{OP}_{EAR}$, $\mathcal{OP}_{EAR}$) = $\mathcal{OP}_{AR'}$: VAS(OM,AR') = Actors \ VAS(OM,EAR)
⟹ *effect cannot be determined in terms of expansion, reduction, or zero_effect*

Figure 17 illustrates the different cases covered by Proposition 5.3.

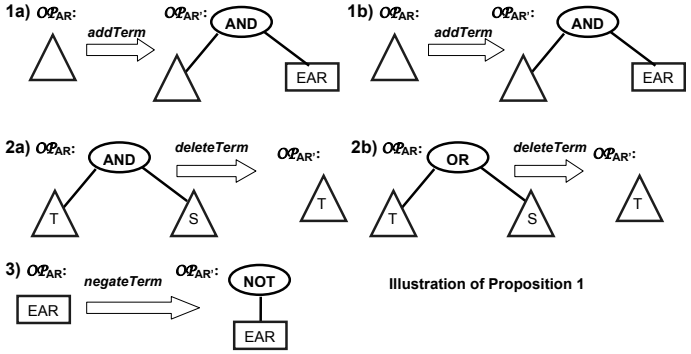


Figure 17. Effects of root level changes

So far, we have only considered root level changes. For substructure level changes, we first determine the (minimal) sub tree which is affected by the respective change (*affected sub tree*). The effects of the substructure change on the affected sub tree can be determined using Proposition 5.3 for root level changes and are reflected by the *resulting sub tree* (cf. Proposition 5.4). According to Proposition 5.3, it is not possible to derive the effects of negation in terms of *reduction, expansion*, or *zero_effect*. When applying a negation operation, however, at least we know that `VAS(OM,AR)` and `VAS(OM,AR')` are *disjoint*, which can be used to describe the semantics of the negation operation. To provide the semantics of substructure level changes, the effects on the affected sub tree are propagated to the root. In this paper, we show how this can be done for effects *reduction, expansion*, and *zero_effect*. Obviously, for other effects (such as *disjoint*) other techniques have to be applied in order to analyze the effects of substructure level changes. Thus, in the following, we focus on operations *addTerm* and *deleteTerm* and leave *negateTerm* to future publications.

Proposition 5.4: Effects of substructure level changes. *Let $\mathcal{OP}_{AR}$ be the operator tree of access rule* `AR` *over organizational model OM. Let further* `op` *be a change operation which transforms* `AR` *into another access rule* `AR'` *with operator tree $\mathcal{OP}_{AR'}$. Then: The affected and resulting sub trees of op on $\mathcal{OP}_{AR}$ can be determined as follows (cf. Fig. 18):*

(1) op: addTerm($\mathcal{OP}_{AR}$,S,[AND|OR|VOID],EAR) = $\mathcal{OP}_{AR'}$ $\Longrightarrow$
- *affected sub tree of op on $\mathcal{OP}_{AR}$ is S*
- *resulting sub tree of op on $\mathcal{OP}_{AR}$ is S' with root(S') = pred($\mathcal{OP}_{AR'}$,root(S)) having sub trees S and $\mathcal{OP}_{EAR}$*

(2) op: deleteTerm($\mathcal{OP}_{AR}$, S) = $\mathcal{OP}_{AR'}$ $\Longrightarrow$
- *affected sub tree of op on $\mathcal{OP}_{AR}$ is S' with root(S') = pred($\mathcal{OP}_{AR}$,root(S))*
- *resulting sub tree of op on $\mathcal{OP}_{AR}$ is T with T being a sibling tree of S based on S' in $\mathcal{OP}_{AR}$*

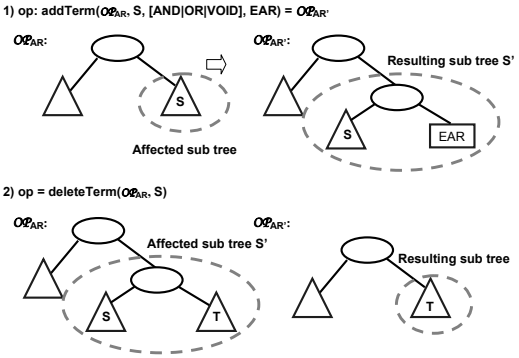


Figure 18. Affected and resulting sub trees

To determine overall effect of a substructure level change on access rule `AR`, the effect on the affected sub tree has to be *pushed* towards the root node of the operator tree of `AR'`. Pushing means that we climb up the tree over the different operators and check the impact on the effects. We start with pushing the effect over the predecessor node of the root of the affected sub tree (*one step push*) and extend this to a *multi step push* towards the root afterwards. To specify the one step push we introduce notion *embracing tree* of the affected sub tree. An example for an embracing tree is shown in Fig. 19.

Definition 5.5: Embracing tree. *Let $\mathcal{OP}_{AR}$ be an operator tree and let S be a sub tree of $\mathcal{OP}_{AR}$. Then we denote T as the embracing tree of S in $\mathcal{OP}_{AR}$ iff*
- *T is sub tree of $\mathcal{OP}_{AR}$ or T = $\mathcal{OP}_{AR}$*
- *root(T) = pred($\mathcal{OP}_{AR}$,root(S))*

Based on Definition 5.5, the *one step push* can be formalized.

Proposition 5.6: One step push. *Let $\mathcal{OP}_{AR}$ be the operator tree of access rule* `AR` *over organizational model OM. Let further* `op` *be a change operating at substructure level with affected sub tree S. Assume that the effect of* `op` *on S corresponds to e $\in$ {Reduction, Expansion, Zero_effect}. Then: One step push of e towards root($\mathcal{OP}_{AR}$) means to lift up e over root(T) where T denotes the embracing tree of S'; i.e., we analyze how e is affected by lifting it over the next operator node on the way to the root. The effect of the one step push remains e; i.e., e is not affected by a one step push.*

For example, if the effect on the affected sub tree is a reduction, intuitively the effect remains a reduction if we "climb over" an OR node. The effects of a one step push over an AND node is illustrated by Fig. 19. Valid actor set of affected sub tree S is reduced according to Proposition 5.3; i.e., `VAS(OM,S')` $\subseteq$ `VAS(OM,S)`. Lifting this effect over the root node of the embracing sub tree T', which is an AND node, keeps the effect of reduction.

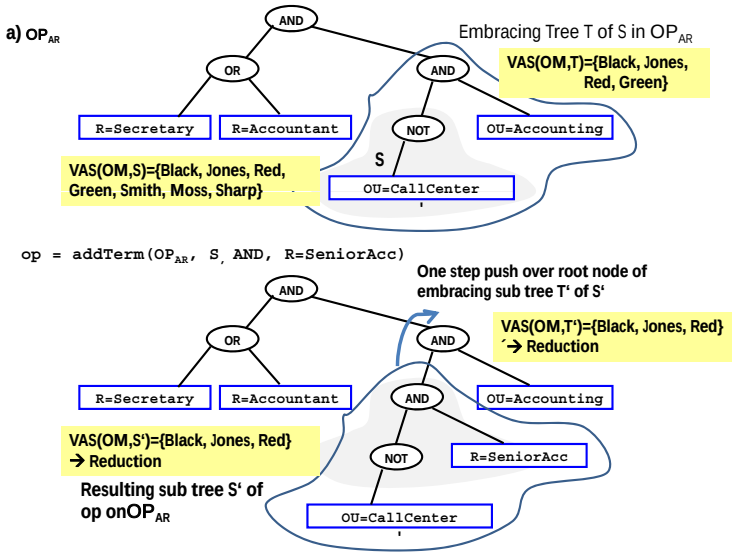


Figure 19. Effects of one step push (example)

Finally, it has to be analyzed how a *multiple step push* towards the root affects the effects of substructure level changes. As stated in Proposition 5.6, a one step push does not affect them. A multi step push can be seen as a one step push which is applied several times. Each time the initial effect of the substructure level change remains the same. Thus, overall, the multi step push does not affect the effect of the substructure level change. This means that the semantics of a substructure level change can be determined as easily as for a root level change. Thus, for any complex access rule and any basic change operation, the effect can be determined quickly. This is essential, for example, when adapting user worklists in PAIS as discussed in Section 7.

Proposition 5.7: Multi step push *Let $\mathcal{OP}_{AR}$ be the operator tree of access rule* AR *over organizational model OM. Let further op be a substructure level change operation with affected sub tree S and resulting sub tree S'. Assume that the effect of op on S is e* $\in$ {*Reduction, Expansion,Zero_effect*}*. Then: Multi step pushing e towards root($\mathcal{OP}_{AR}$) means to lift up e over all nodes on the path to root($\mathcal{OP}_{AR}$) starting from root(T) where T denotes the embracing tree of S'. The effect of a multi step push towards the root remains e; i.e., e is not affected by the multi step push.*

6. Exploiting organizational knowledge for high-level access rule changes

As discussed in Section 4.3, a high-level access rule change Δ can be understood as an ordered sequence of basic access rule changes $op_1, \ldots, op_n$. Thus, it can be tried to aggregate the effects of $op_1, \ldots, op_n$ in order to determine semantics of Δ. However, such aggregation might be impossible; e.g., in case the effect of op_i is a reduction and the effect of op_j is an *expansion* ($i \neq j$). In this case, the actor sets being valid before and after the high-level change have to be re-calculated and explicitly compared. Generally, re-calculation could be used for determining the effects of basic change operations as well. However, as we discuss in Section 7, in many applications it is beneficiary to have a "quick check" on the effects of access rule changes. For example, if

we know that a change has effect *expansion* or *zero_effect*, we can delay the adaptation of user worklists in PAIS until the system is offline. Contrary, it can be expensive to always recalculate the new valid actor sets immediately. However, since high-level change operations consist of the consecutive application of basic change operations, their effects cannot be determined as easy as for basic operations. As example consider change `substituteAccessRules(`$\mathcal{OP}_{\mathcal{AR}}$`, S, T) = deleteTerm(`$\mathcal{OP}_{\mathcal{AR}}$`, S), addTerm(`$\mathcal{OP}_{AR}^{intermediate}$`, `$\mathcal{OP}_{R=Secretary}$`, OR, T)` as depicted in Fig. 14. According to Proposition 5.3, for the `deleteTerm` operation we obtain a reduction effect, whereas for the `addTerm` operations an expansion results. Thus the overall effect cannot be determined.

However, for this scenario, one can think about some optimizations regarding high-level access rule changes. For high-level change `substituteAccessRules(`$\mathcal{OP}_{AR}$`,S,T)`, for example, additional information from the underlying organizational model can be used to determine the effects on the valid actor sets of the changed access rules. Assume, for example, that for access rule `AR` ← `Role='R1'`, we substitute role `R1` by role `R2` resulting in `AR'` ← `Role='R2'`. Then, if we know from the underlying organizational model that `R2` is a sub role of `R1`, it can be concluded that the effect on the valid actor set of `AR` is either *zero_effect* or *reduction*. Reason is that the same set of actors or less actors are assigned to a sub role when compared to the superior one. Vice versa, if a role is substituted by a superior one within an access rule, the effect on the valid actor set will be *zero_effect* or *expansion*. One big advantage of determing the effects of access rule changes as described above is the following: If access rules are changed, the effects on the valid actor sets have to be propagated to user worklists at some point in time. This point in time can be chosen depending on the particular change effect. If, for example, the valid actor set is reduced, this poses a potential security threat on the system. Either work items might be offered to users who are no longer qualified or, if the valid actor set becomes empty, no actor will be qualified anymore. Hence, user worklists should be adapted *immediately*. Contrary, if the valid actor set is expanded, the only consequence might be that work items are not offered to all qualified users. Since this poses no security threat on the PAIS, the propagation of the access rule change to user worklists may be *delayed*; e.g., done offline when no user is working on the PAIS. Finally, in case of zero_effect no action is required at all. Same considerations hold for the hierarchial relations between organizational units. Proposition 6.1 summarizes these considerations:

Proposition 6.1: Valid Actor Set Relations for Specialization and Subordination *Let OM = (Actors, Roles, OrgUnits, has, belongs_to, is_subordinated, specializes) be an organizational model. Let further r1, r2* ∈ *Roles be two roles and o1, o2* ∈ *OrgUnits be to organizational units. Then:*

$$r1 \in specializes^*(r2) \Longrightarrow VAS(OM, R{=}r1) \subseteq VAS(OM, R{=}r2)$$
$$o1 \in is\_subordinated^*(o2) \Longrightarrow VAS(OM, OU{=}o1) \subseteq VAS(OM, OU{=}o2)$$

For the scenario depicted in Fig. 14, for example, we know from OM that `SeniorAcc` is a sub role of role `Accountant`. Thus, according to Propostion 6.1 `VAS(OM, R=SeniorAcc)` ⊆ `VAS(OM, R=Accountant)` holds. Consequently, the overall effect of the `substituteAccessRule(...)` operation can be determined as reduction (or zero_effect). Obviously that holds true since:
`VAS(OM',AR)` = {`Black,Jones,Red,Green`} ⊇ `VAS(OM',AR')` = {`Black,Jones,Red`}.

7. Discussion

In order to elaborate the requirements for a generic component enabling full life cycle support in respect to organizational entities as well as access rules we conducted an in-depth analysis of organizational structures and their evolution in several case studies. Corresponding results have been reported in (Konyen (1996), Konyen *et al.* (1996a,b), Reichert *et al.* (2004), Wiedemuth-Catrinescu (2002)). Besides an in-depth analysis of healthcare processes and relevant access control requirements during these case studies, we identified typical organizational patterns in the medical environment (including organizational relations and complex substitution as well as delegation rules) as well as frequently occurring organizational changes. Examples of characteristic changes we identified over a period of two years include, for example, the relinking of organizational units and actors within the organizational chart, the definition of completely new organizational units, the merging of existing departments, or the outsourcing of organizational units. The latter took place, for example, when a certain medical lab services were outsourced to an external lab. All these observations from our case study have confirmed the high practical need for better supporting the evolution of organizational models at the IT level. In addition, we made similar observations in the context of our projects in other industry including the automotive domain and the financial sector. Finally, from discussions with an IT service center from the University Hospital in Ulm we know that it takes about 10% of the IT staff's working time to implement access control policies and to cope with evolving organizational structures at the IT level. Reason is that most of the access control mechanisms in existing systems are hard-coded within the application programs. Thus, adaptations of organizational structures result in high efforts for code adaptations or – even more – inconsistencies and security holes.

Altogether our case studies have proven that any approach enabling the quick and correct adaptation of organizational models, including the control of their effects on running information systems, is crucial in practice. Our findings are supported by other studies. One example is the study on the dynamics of rules in an organization conducted by Stanford University (see March *et al.* (2000)).

In summary, contributions and benefits of the CEOSIS approach are as follows:

- *Explicit Representation of Organizational Models.* In CEOSIS the representation of organizational structures is separated from the code of the application systems. This enables quick and correct adaptation of organizational models. Furthermore, due to the explicit definition of organizational models, the communication with domain experts and operating departments is facilitated.
- *Model Quality and Consistency.* The central management of organizational structures leads to a decrease of model inconsistencies and thus reduced maintenance costs.
- *Expressive Organizational Meta Model.* The meta model chosen in CEOSIS enables coverage of organizational structures (according to the analyzed use cases) being relevant in practice. In order to capture additional kinds of organizational entities, we have extended the organizational meta model as presented in this paper in several respects. Berroth (2005) and Reichert *et al.* (2004) describe these extensions in detail.
- *Explicit Management of Access Rules.* In CEOSIS organizational models as well as access rules can be explicitely modeled and maintained. Particularly, the references between access rules and the underlying organizational model can be specified and thus monitored and controlled by the information system. This means that potential conflicts in the course of organizational changes (e.g., orphaned or dangling references)

can be resolved based on an adequate user assistance as well as intelligent adaptation strategies.

- *Exploiting the Semantics of Organizational Changes:* In order to find intelligent adaptation strategies to overcome possible conflicts in connection with organizational changes, the CEOSIS approach exploits the maximum available information. Particulary, by analyzing the semantics of the applied changes (at organizational model as well as access rule level) provides the basis for adequate adaptation strategies. This leads to a new quality of determining and handling the effects of organizational changes. As a consequence, conflict detection and resolution is significantly quicker and more effective in CEOSIS than manual adaptation of hard-coded organizational structures within application software.
- *Support of Access Rule Changes:* To our best knowledge, CEOSIS is the first formally rigorous approach to not only support organizational changes, but also direct adaptations of access rules as well. The latter often become necessary in practice as consequence of imprecise analysis or mistakes when modeling the access rules (e.g., a specific role is defined to narrow and thus does not reflect the real situation). The CEOSIS approach enables the specification of such access rule changes and the precise analysis of their effects.

In summary, CEOSIS enables the representation of organizational structures as well as organizational changes that occur frequently in practice. This is based on a formal underpinning. Specifically, organizational changes including modification of organizational models and access rules are formally defined. Furthermore their formal semantics is provided. In addition, the set of offered change operations is complete[1]; i.e., any organizational model OM can be transformed to any other organizational model OM' by applying a set of change operations to OM as offered by CEOSIS. As proof sketch we assume that by applying respective delete operations OM can be first transformed into the empty organizational model OM_{empty}, which can then be transformed into OM' by applying create entity/relation operations afterwards. The same holds for access rule operations. However, note that CEOSIS additionally supports several high-level change operations in order to enable changes as a high level of abstraction as well; i.e., the offered set of change operations is not minimal.

The modeling of organizational entities and relations as well as of access rules has been implemented within the AristaFlow System process management system, which supports an even more expressive organizational meta model as introduced in this paper (see Reichert *et al.* (2009), Berroth (2005)). Currently, the described functionality of organizational evolution is prototypically realized.

8. Related work

Issues related to change management and life cycle management have been addressed in many disciplines including software engineering (see Kramer and Magee (1990)), rule maintenance (see March *et al.* (2000)), and business process management (see Rinderle *et al.* (2004b)). Similarities to the CEOSIS approach can be observed, for example, in the context of business rule maintenance (see Herbst and Myrach (1996), Herbst (1997)). Business rules typically reference application data (e.g., the age of a patient in a medical setting) and thus, if the rule or the referenced data is changed, orphaned references might

[1]but not minimal due to the definition of high-level change operations based on basic change operations

result. Existing solutions apply repository technology to deal with such problems (see Herbst and Myrach (1996)). However, in addition to maintaining the references of rules, CEOSIS exploits the semantics of the applied change operations on organizational structures in order to proactively propose strategies to deal with such orphaned references. Furthermore, when evolving organizational structures other fundamental problems such as empty actor sets of non-resolvable worklists can occur which require new strategies to handle them properly.

The provision of an adequate access control framework is indispensable for any EIS. In the literature numerous approaches have been presented dealing with challenging issues related to access control (e.g., Klarmann (2001a), Bertino (1998), zur Muehlen (2004), Weber *et al.* (2005)). These approaches range from traditional privilege management techniques such as *Access Control Lists* (ACLs) and *Capability Lists (CPs)* (Bishop (2002), Miller *et al.* (2003)) to *Role-Based Access Control (RBAC)*. ACLs and CLs constitute a certain partitioning of the complete *privilege matrix*. In an EIS, for each subject S (e.g., a user) and each object O (e.g., a document) the privilege matrix maintains an entry on the privileges of S on O. ACLs maintain a list of authorized users for each object. CLs, in turn, manage a set of capabilities for each user for accessing certain objects. ACLs and CLs are well suited for privilege management in more static systems, i.e., systems for which users and objects remain quite stable over time. However, as many examples show, the set of users as well as the set of objects are frequently subject to change in EIS (Ferraiolo and Kuhn (1992)). ACLs and CLs are not well suited for such flexible systems due to the enormous reorganization effort after changes.

Thus, many approaches use RBAC for defining and managing user access privileges (Bertino (1998), Ferraiolo and Kuhn (1992), Ferraiolo *et al.* (2003), NIST (2004)) in EIS; 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* (Botha and Eloff (2001), Bertino *et al.* (1999), Wainer *et al.* (2003), zur Muehlen (2004), Pfeiffer (2005), Weber *et al.* (2005)).

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 (Bertino *et al.* (1999), Wainer *et al.* (2003), Kuhn (1997)). 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 Ferraiolo and Kuhn (1992).

In the workflow literature several proposals have been made aiming at adaptive process management (e.g., van der Aalst (2001), Rinderle *et al.* (2004a), Joeris and Herzog (1998), Weske (1999), Sadiq *et al.* (2000), Fent *et al.* (2002), Kochut *et al.* (2003), Edmond and ter Hofstede (2000)). The ADEPT technology, for example, enables controlled changes at the process type as well as the process instance level (for details see Reichert and Dadam (1998), Rinderle *et al.* (2004b)). Thereby, correctness and consistency constraints of a workflow are preserved when dynamically changing its structure, its state, or its attributes during runtime. In Weber *et al.* (2005) 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 notions of correctness (for an overview see Rinderle *et al.* (2004a)), so far, focus has been on process changes. Hence the CEOSIS approach for the evolution of organizational structures is complementary to these approaches and therefore constitutes an important step towards a comprehensive support for adaptive information systems.

There are only few approaches (Klarmann (2001b,a), Domingos *et al.* (2003)) which address the problem of organizational change and its handling in EIS. In Klarmann (2001b,a) 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 Klarmann (2001a), however, we additionally follow a rigorou 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 Klarmann (2001a). In Rinderle and Reichert (2005, 2007) we have ourselves introduced a first approach for evolving organizational models and for propagating corresponding changes to access rules. However, this previous work did not consider direct changes of access rules; i.e., changes which are independent of whether or not the underlying organizational model has been adapted.

The approach introduced in Domingos *et al.* (2003) 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 (Jablonski *et al.* (2001), zur Muehlen (2004), Berroth (2005)). Most of them suggest a particular meta model for capturing organizational 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.

Ly *et al.* (2005) introduced *access rule mining* as first approach for the (semi-)automatic discovery of access rule optimizations (cf. Fig. 2). This work has been extended by Rinderle-Ma and van der Aalst (2007). Using special mining techniques, it can be detected whether and – if yes – how users deviate from pre-specified access rules within daily business life; e.g., in case a task within a PAIS is always passed to a substitute. As another example consider a scenario where two users A and B qualify for a particular task, but always user A selects the corresponding work item. In both cases, the associated access rules should be optimized by directly adapting them.

In van der Aalst and Jablonski (2000) important issues related to changes of processes and organizational 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).

9. Summary and outlook

In this paper, we have introduced an approach for managing the life cycle of access rules. First, we have shown how changes of an organizational model may affect related access rules. Second, we have presented issues emerging in the context of direct access rule changes (e.g., due to optimizations or access rule mining). In order to be able to directly change access rules, a complete set of change operations has been presented. Furthermore, we have precisely defined the formal semantics of these change operations in order to avoid any ambiguity when applying them. Finally, the correct definition of

change operations required the introduction of a tree-based representation of access rules with associated tree operations.

In future work, we will elaborate the effects of access rule changes (direct or due to organizational changes) on user worklists in process-aware information systems. For this we plan to build up cost models to measure the efficiency of different adaptation strategies. Furthermore, we will dig deeper into the area of access rule mining.

References

Berroth, M., Design of a Component for Organizational Models. Master's thesis, University of Ulm, Computer Science Faculty (in German), 2005. .

Bertino, E., 1998. Data Security. *Data & Knowledge Engineering*, 25 (1–2), 199–216.

Bertino, E., Ferrari, E., and Alturi, V., 1999. The Specification and Enforcement of Authorization Constraints in WFMS. *ACM Transactions on Information and System Security*, 2 (1), 65–104.

Bishop, M., 2002. *Computer Security*. Addison-Wesley Professional.

Botha, R. and Eloff, J., 2001. A Framework for Access Control in Workflow Systems. *Information Management and Computer Security*, 9 (3), 126–133.

Domingos, D., Rito-Silva, A., and Veiga, P., 2003. Authorization and Access Control in Adaptive Workflows. *In*: *Europ. Symposium on Research in Computer Science (ESORICS'03)*, Gjovik, Norway, 23–28.

Edmond, D. and ter Hofstede, A., 2000. A Reflective Infrastructure for Workflow Adaptability. *Data and Knowledge Engineering*, 34 (3), 271–304.

Fent, A., Reiter, H., and Freitag, B., 2002. Design for Change: Evolving Workflow Specifications in ULTRAflow. *In*: *Int'l Conference on Advanced Information Systems Engineering*, May., 516–534.

Ferraiolo, D.F., Chandramouli, R., and Kuhn, D.R., 2003. *Role-Based Access Control*. Artech House, Incorporated.

Ferraiolo, D. and Kuhn, D., 1992. Role Based Access Control. *In*: *National Computer Security Conference*.

Herbst, H., 1997. *Business Rule-oriented Conceptual Modeling*. Springer.

Herbst, H. and Myrach, T., 1996. A Repository System for Business Rules. *In*: *IFIP TC2-Working Conf on Data Semantics*.

Jablonski, S., Schlundt, M., and Wedekind, H., 2001. A generic component for the computer–based use of organizational models (in German). *Informatik Forschung und Entwicklung*, 16, 23–34.

Joeris, G. and Herzog, O., 1998. Managing Evolving Workflow Specifications. *In*: *Int'l Conference on Cooperative Information Systems*, August., New York City, 310–321.

Klarmann, J., 2001a. A Comprehensive Support for Changes in Organizational Models of Workflow Management Systems. *In*: *Int'l Conference on Inf Systems Modeling*, 375–387.

Klarmann, J., 2001b. Using Conceptual Graphs for Organization Modeling in Workflow Management Systems. *In*: *Conference Professionelles Wissensmanagement*, 19–23.

Kochut, K., *et al.*, 2003. IntelliGEN: A Distributed Workflow System for Discovering Protein-Protein Interactions. *Distributed and Parallel Databases*, 13 (1), 43–72.

Konyen, I., Organizational structures and business processes in hospitals. Master's thesis, University of Ulm, Computer Science Faculty (in German), 1996. .

Konyen, I., *et al.*, Process Design for an In-patient Chemotherapy. , 1996a. , Internal

Computer Science Report DBIS-5, Ulm University (in German).

Konyen, I., *et al.*, A Process Design for the Area of Minimal-Invasive Surgery. , 1996b. , Internal Computer Science Report DBIS-14, Ulm University (in German).

Kramer, J. and Magee, J., 1990. The Evolving Philosophers Problem: Dynamic Change Management. *IEEE Transactions on Software Engineering*, 16 (11), 1293–1306.

Kuhn, D., 1997. 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*, 23–30.

Ly, L., *et al.*, 2005. Mining Staff Assignment Rules from Event-Based Data. *In*: *Proc. BPM'05 Workshops, Workshop on Business Process Intellingence*, 177–190.

March, J., *et al.*, 2000. *The dynamics of rules*. Stanford University Press.

Miller, M., Yee, K., and Shapiro, J., Capability Myths Demolished. , 2003. , Technical report, Combex, Inc.

NIST, 2004. *Proposed Standard for Role-Based Access Control*. http://csrc.nist.gov/rbac/rbacSTDACM.pdf.

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. .

Reichert, M. and Dadam, P., 1998. ADEPT$_{flex}$ - Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10 (2), 93–129.

Reichert, M., *et al.*, 2009. Architecural Principles and Components of Adaptive Process Management Technology.. *In*: *PRIMIUM - Process Innovation for Enterprise Software.*, 81–97.

Reichert, M., Wiedemuth-Catrinescu, U., and Rinderle, S., 2004. Evolution of Access Control in Information Systems. *In*: *Conf Electronical Business Processes* (in German), Klagenfurt, 100–114.

Rinderle, S. and Reichert, M., 2005. On the Controlled Evolution of Access Rules in Cooperative Information Systems. *In*: *Cooperative Information Systems*, LNCS 3760, 238–255.

Rinderle, S. and Reichert, M., 2007. A Formal Framework for Adaptive Access Control Models. *Journal on Data Semantics*, (IX), 82–112.

Rinderle, S., Reichert, M., and Dadam, P., 2004a. Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey. *Data and Knowledge Engineering*, 50 (1), 9–34.

Rinderle, S., Reichert, M., and Dadam, P., 2004b. Flexible Support Of Team Processes By Adaptive Workflow Systems. *Distributed and Parallel Databases*, 16 (1), 91–116.

Rinderle-Ma, S. and Reichert, M., 2008. Managing the Life Cycle of Access Rules in CEOSIS. *In*: *12th IEEE International Enterprise Computing Conference*, 257–266.

Rinderle-Ma, S. and van der Aalst, W., Life-cycle support for staff assignment rules in information systems. , 2007. , Technical report WP-213, Beta Research School for Operations Management and Logistics, TU Eindhoven.

Sadiq, S., Marjanovic, O., and Orlowska, M., 2000. Managing Change and Time in Dynamic Workflow Processes. *Int'l Journal on Cooperative Information Systems*, 9 (1&2), 93–116.

van der Aalst, W., 2001. Exterminating the Dynamic Change Bug: A Concrete Approach to Support Worfklow Change. *Information Systems Frontiers*, 3 (3), 297–317.

van der Aalst, W. and Jablonski, S., 2000. Dealing with Workflow Change: Identification of Issues an Solutions. *Int'l Journal of Comp. Systems, Science and Engineering*, 15

(5), 267–276.

Wainer, J., Barthelmess, P., and Kumar, A., 2003. W–RBAC – A Workflow Security Model Incorporating Controlled Overriding of Constraints. *International Journal of Collaborative Information Systems*, 12 (4), 455–485.

Weber, B., *et al.*, 2005. Balancing Flexibility and Security in Adaptive Process Management Systems. *In*: *Int'l Conference on Cooperative Information Systems*, November., Agia Napa, Cyprus.

Weske, M., 1999. Adaptive Workflows based on Flexible Assignment of Workflow Schemes and Workflow Instances. *In*: *Proc. GI-Workshop Enterprise-wide and Cross-enterprise Workflow-Management (Informatik'99)*, October., Paderborn, 42–48.

Wiedemuth-Catrinescu, U., Evolution of Organizational Models in Workflow Management Systems. Master's thesis, University of Ulm, Computer Science Faculty (in German), 2002. .

zur Muehlen, M., 2004. Organizational Management in Workflow Applications – Issues and Perspectives. *Inf. Tech. and Management*, 5 (3-4), 271–291.