Managing the Life Cycle of Access Rules in CEOSIS

Stefanie Rinderle-Ma, Manfred Reichert
Institute of Databases and Information Systems, Ulm University, Germany
{stefanie.rinderle, manfred.reichert}@uni-ulm.de

Abstract

The definition and management of access rules (e.g., to control the access to business documents and business functions) is an important task within any enterprise information systems (EIS). Many EIS apply role-based access control (RBAC) mechanisms to specify access rules based on organizational models. However, only little research has been spent on organizational changes even though they often become necessary in practice. Examples comprise the evolution of organizational models with subsequent adaptation of access rules or direct access rule modifications. In this paper, we present a change framework for the controlled evolution of access rules in EIS. Specifically, we define change operations which ensure correct modification of access rules. Finally, we define the formal semantics of access rule changes based on operator trees which enables their unambiguous application; i.e., we can precisely determine which effects are caused by respective adaptations. This is important, for example, to be able to efficiently adapt user worklists in process-aware information systems. Altogether this paper contributes to comprehensive life cycle support for access rules in (adaptive) EIS.

1 Introduction

A fundamental aspect in the design of any enterprise information systems (EIS) concerns *access control*, i.e., granting certain rights to specific users; e.g., the right to access a certain business document for a restriced group of users in an EIS. There is a multitude of models for defining such access control mechanisms; e.g., GRANT / REVOKE statements in DBMS or Role Based Access Control (RBAC) [5] in process-aware information systems (PAIS). Usually such models comprise a set of *access rules* which are defined based on organizational models capturing organizational entities and their relationships. Access rules control which rights shall be granted to which users. In case of PAIS, for example, access rules specify which tasks (i.e., *work items*) shall be offered to which users in their worklists during the execution of a particular process.

Due to changes of the organization 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 organizational changes in a quick, flexible, and reliable (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 kind of changes:

1. *Organizational Change:* Access rule adaptations might become necessary after changes of organizations and organizational models respectively [10, 11]. Assume that access control within a PAIS (i.e., the assignment of work items to users) is reflected by the simple access rules depicted in Fig. 1. Assume 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 the organizational change whereas access rules AR2 and AR3 now refer to entities no longer present in OM'. If the affected access rules were not adapted this could 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 administrator, users working on preceding tasks, etc.).

2. *Direct Changes:* Generally, it must be also possible to adapt access rules directly within EIS (cf. Fig. 2). This often becomes necessary, for example, for access rules 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 assign-

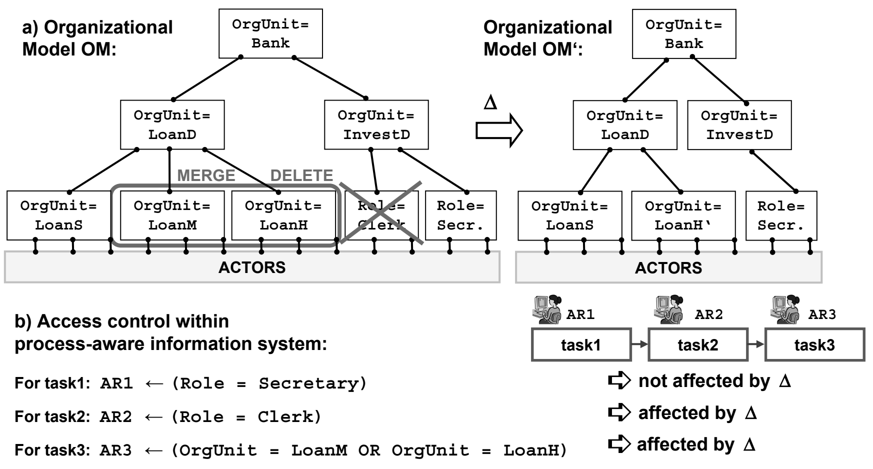


Figure 1. Organizational changes affecting access rules

ment or adaptation can be complex and error-prone.

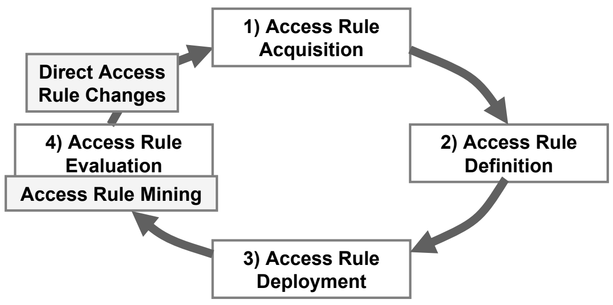

1) Access Rule Acquisition
2) Access Rule Definition
3) Access Rule Deployment
4) Access Rule Evaluation
Access Rule Mining
Direct Access Rule Changes

Figure 2. Access rule life cycle

In [10, 11] we presented an approach for specifying organizational models and for propagating changes of such models to access rules. In particular, such adaptations become necessary to maintain correctness and consistency of access rules; e.g., avoiding the situation where no actor qualifies for a specific access rule anymore. However, our previous work has not considered direct changes of access rules so far; i.e., changes which are independent of whether the underlying organizational model is changed. This will become necessary, if access rules are to be optimized or corrected (e.g., if a work item within a PAIS is manually assigned to a more specialized role as the current one).

One approach for the (semi-) automatic determination of access rule optimizations is *access rule mining* [8, 14] (cf. Fig. 2). Using special mining techniques, it can be detected whether and how users deviate from the pre-modelled access rules within daily business life. As first example assume that a work item within a PAIS is always passed to a subtitute. Another example could be that two users A and B qualify for a particular task (i.e., work item), but the task is always selected by A. Then, the associated access rules should be optimized by applying direct access rule changes.

In this paper we present our CEOSIS[1] framework for evolving access rules in a controlled and secure way. There are two basic requirements for such a framework. First of all, access rule changes must be conducted in a *correct* way; i.e., they must not violate any structural constraints set out by the 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. 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 enables, for example, the analysis of access rule change effects on user worklists in PAIS.

The remainder of this paper is organized as follows: In Sect. 2 we provide background information. Sect. 3 defines change operations for access rules and Sect. 4 provides their formal semantics. In Sect. 5 we discuss related work. We close with a summary and outlook in Sect. 6.

2 Organizational models and access rules

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

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

2.1 Organizational (meta) model

An organizational *meta model* captures all entities and relations an organizational model may consist of (i.e., the meta model can be seen as the schema which can be instantiated by concrete organizational models). The organizational meta model OMM used in this paper is based on the *Role Based Access Control Model* (RBAC) [6]. As depicted in Fig. 3 it consists of entity types `OrganizationalUnit`, `Actor`, and `Role`. Concrete organizational units (e.g., `clinic`) can be hierarchically related to each other by relation `is_subordinated`. Similarly, concrete roles can be specialized by introducing sub roles (relation `specializes`); i.e., the sub role inherits all abilities of the superior role, but may have additional ones. Finally, actors can have roles (relation `has`) and belong to organizational units (`belongs_to`).

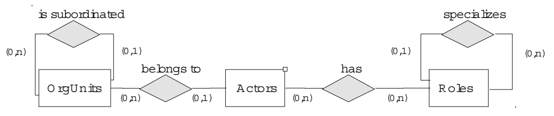


Figure 3. Org. meta model (in ER notation)

In this paper, we use a well-established, but rather simple organizational meta model OMM to focus on core issues related to access rule changes. In principle, OMM and the subsequent considerations can be transferred to more complex organizational meta models as well (e.g., capturing entities such as abilities or substitution relations) [11].

Based on our OMM concrete organizational models can be defined, i.e., an organizational model OM is an instance of OMM (cf. Def. 1). One example is depicted in Fig. 4a. OM captures three organizational units, where `treatment area` and `administration` are hierarchically subordinated to unit `medical clinic`. The most general role used in this model is `staff` which is specialized by roles `internist`, `assistant`, and `secretary`. Finally, actors are assigned to roles and belong to an organizational unit (e.g., actor `Hunter` has role `secretary` and belongs to unit `administration`).

Definition 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 roles,*
- `OrgUnits` *corresponds to the set of organizational units,*
- `has` $\subseteq$ `Roles` $\times$ `Actors` *corresponds to the relation linking actors to roles,*
- `belongs_to` $\subseteq$ `OrgUnits` $\times$ `Actors` *corresponds to the relation 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, the following notions on relations are needed:

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

Consider Fig. 4. An example for generic relation R(x) given in Def. 1 is `has(assistant) = Black` (with R = 'has'). The semantics of the two relations `is_subordinated` and `specializes` can be defined as follows: All actors belonging to an organizational unit also belong to its superordinated organizational units (e.g., actors `Smith`, `Black`, `Hunter`, and `Dr.Smith` all belong to org. unit `medical clinic`, Fig. 4). If an actor has a particular role she will also possess all superior roles (e.g., actor `Black` has roles `assistant` and `staff`).

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 (Elementary access rule) *Let OM = (*`Actors, ...`*) be an organizational model (cf. Def. 1). An elementary access rule* `EAR` *on OM is defined as follows:*

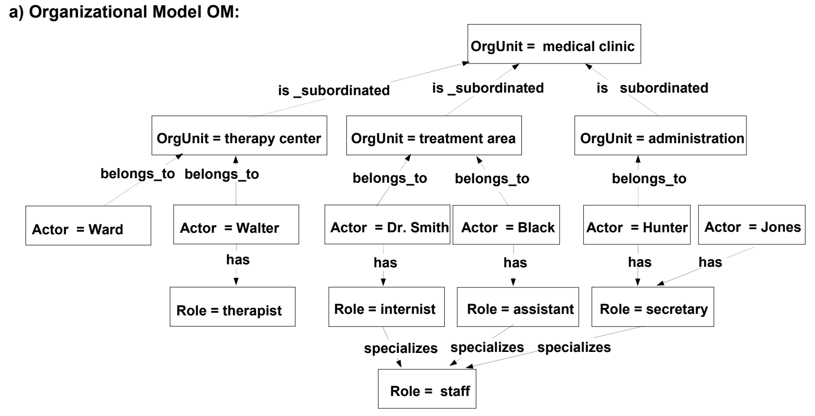
$$
\begin{aligned}
\texttt{EAR} \equiv\ & (\texttt{EAR0} \longleftarrow \tau^*)^{2} \mid \\
& (\texttt{EAR1} \longleftarrow (\texttt{Role = } r)) \mid \\
& (\texttt{EAR2} \longleftarrow (\texttt{OrgUnit = } o)) \mid \\
& (\texttt{EAR3} \longleftarrow (\texttt{Role+ = } r)) \mid \\
& (\texttt{EAR4} \longleftarrow (\texttt{OrgUnit+ = } o)).
\end{aligned}
$$

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)` $\subseteq$ `Actors` *with*

- `VAS(OM, EAR0) =` $\emptyset$
- `VAS(OM, EAR1) = has(r)` *(where* `has(r)` *corresponds to the set of actors with role* `r`*, cf. Def. 1)*
- `VAS(OM, EAR2) = belongs_to(o)` *(i.e., the set of actors belonging to unit* `o`*)*
- `VAS(OM, EAR3) = has(specializes`*`(r))` *(where* `has(specializes`*`(r))` *corresponds to the set of actors having role* `r` *or a more specialized one, cf. Def. 1)*
- `VAS(OM, EAR4) = has(is_subordinated`*`(o))` *(i.e., the set of actors belonging to organizational unit* `o` *or a subordinated one).*

[2] τ^* denotes an empty term.

a) Organizational Model OM:



b) Access Rules on OM:

```
AR1 ← Role+=‘staff’
AR2 ← OrgUnit=‘treatment area’
AR3 ← (Role=’secretary’) OR (Role=’assistant’)
AR4 ← NOT(OrgUnit+=’medical clinic’)
```

c) Valid Actor Sets:

```
VAS(OM,AR1) = {Dr. Smith, Black, Hunter, Jones}
VAS(OM,AR2) = {Dr. Smith, Black}
VAS(OM,AR3) = {Black, Hunter, Jones}
VAS(OM,AR4) = {Jones}
```

Figure 4. Organizational model, access rules, and valid actor sets

Fig. 4b+c show two elementary access rules AR1 and AR2 and the associated valid actor sets based on organizational model OM. Taking Def. 2 the general notion of access rules can be formalized. It is based on elementary access rules which can 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. Any negation contained within an access rule AR can be always pushed to the elementary access rules contained within AR.

Definition 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:*

`AR` $\equiv$ `EAR | NAR | CAR | DAR`
where

- `EAR` *constitutes an elementary access rule (cf. Def. 2),*
- `NAR` $\longleftarrow$ `(NOT (EAR))` *where* `EAR` *is an elementary access rule,*
- `CAR` $\longleftarrow$ `(AR1 AND AR2)` *where* `AR1` *and* `AR2` *are access rules, and*
- `DAR` $\longleftarrow$ `(AR1 OR AR2)` *where* `AR1` *and* `AR2` *are access rules.*

Formal semantics of `EAR` *has been given in Def. 2, the one of* `NAR`, `CAR` *and* `DAR` *is defined as follows:*

- `VAS(OM,NAR) = Actors \ VAS(OM,AR)` *corresponds to the set of actors not qualifying for access rule* `AR`,
- `VAS(OM,CAR) = VAS(OM,AR1)` $\cap$ `VAS(OM,AR2)` *corresponds to the set of actors qualifying for access rules* `AR1` *and* `AR2`,
- `VAS(OM,DAR) = VAS(OM,AR1)` $\cup$ `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. 4b depicts two non-elementary access rules `AR3` and `AR4`.

3 Access rule changes

To be able to analyze the effects of changes of an organizational model we introduced respective change operations with precise formal semantics in [10, 11]. Similarly, in this section, we define operations for changing access rules directly; e.g., deleting AND-terms. The formal semantics of these change operations is presented in Sect. 4. It is based on the valid actor sets of the access rules before and after the changes.

3.1 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 Def. 3. Using the notion given in Def. 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 allows for the 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). 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 formula or SQL-statement. An example of an access rule with corresponding operator-tree is shown in Fig. 5. Fig. 6 shows another example for an imbalanced operator tree.

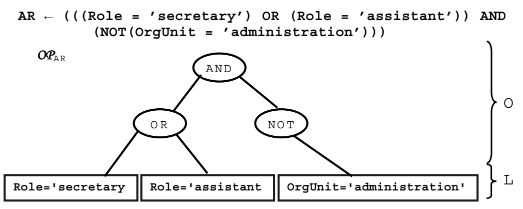

AR ← (((Role = 'secretary') OR (Role = 'assistant')) AND
(NOT(OrgUnit = 'administration')))
OP_AR
AND
OR
NOT
Role='secretary
Role='assistant
OrgUnit='administration'
O
L

Figure 5. 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. Def. 3).

How an operator tree can be built based on a given access rule is shown in the next section.

3.2 Basic access rule change

We introduce a complete set of basic change operations on access rules[3] and their operator-tree representation respectively. Before we define basic notions on operator trees:

Definition 4 (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 $\longleftarrow \tau^*$.
- *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[4]$\} \mapsto \mathcal{OP}^{OM}$
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. 6, 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. 6, Step 2).*

Our set of basic change operations allows 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 $\in \mathcal{AR}^{OM}$). Structural correctness is preserved by formal pre- and postconditions for the change operations. Within our ADEPT process management technology [12, 13], we additionally ensure compliance with the underlying organizational model OM by forbidding access rule changes which refer to entities not being present in OM (e.g., referring to `Role='clerk'` in Fig. 4).

Definition 5 (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,$\mathcal{OP}_{NOT(EAR)}$)*

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

[4]VOID represents the empty operator.

An example for applying the delete operation is depicted in Fig. 6. First the sub tree reflecting elementary access rule `EAR` $\leftarrow$ `Role='secretary'` (cf. Fig. 5) is substituted by empty tree τ. Then the optimization function is run on the resulting tree which eliminates τ by lifting up remaining elementary access rule `EAR'` $\leftarrow$ `Role='assistant'` to the next level within the operator tree.

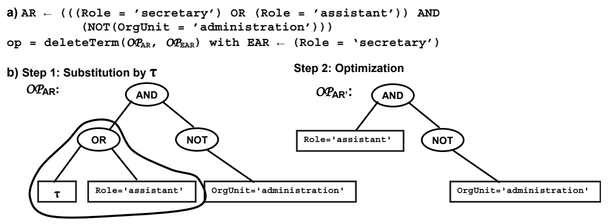

a) AR ← (((Role = 'secretary') OR (Role = 'assistant')) AND (NOT(OrgUnit = 'administration')))
op = deleteTerm(OP_AR, OP_EAR) with EAR ← (Role = 'secretary')
b) Step 1: Substitution by τ
OP_AR:
AND
OR
NOT
τ
Role='assistant'
OrgUnit='administration'
Step 2: Optimization
OP_AR':
AND
Role='assistant'
NOT
OrgUnit='administration'

Figure 6. 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. 5. Then the following basic change operations build $\mathcal{OP}_{AR}$ starting from empty tree τ:

op_1: $\mathcal{OP}_1$ = addTerm(τ, τ, VOID, Role='secretary')
op_2: $\mathcal{OP}_2$ = addTerm($\mathcal{OP}_1$, $\mathcal{OP}_1$, OR, Role='assistant')
op_3: $\mathcal{OP}_3$ = addTerm($\mathcal{OP}_2$, $\mathcal{OP}_2$, AND, OrgUnit='administrator')
op_4: $\mathcal{OP}_{AR}$ = negateTerm($\mathcal{OP}_3$, $\mathcal{OP}_{OrgUnit='assistant'}$)

Formal semantics of these change operations is presented in Sect. 4.

3.3 High-level access rule changes

For better user support we have defined some *high-level change operations* based on the basic change operations introduced in Def. 5. As an example consider *substituteAccessRules($\mathcal{OP}_{AR}$,S,T)* where 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. A substitution of access rules can be accomplished 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 T consists of. The pre- and postconditions of high-level change operations can be derived by aggregating the pre- and postconditions of the underlying basic change operations. Due to lack of space we omit further details here.

4 Semantics of access rule changes

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

Definition 6 (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)`

4.1 Root level and substructure level changes

For elementary access rules the analysis of change effects is easy to accomplish. For example, let `EAR` $\longleftarrow$ `(Role = 'doctor')` and `EAR'` $\longleftarrow$ `(Role = 'therapist')` 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 set 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. 5 and elementary rule `EAR'` $\longleftarrow$ `(Role = 'therapist')`. If we apply change operation op^a = *addTerm($\mathcal{OP}_{AR}$,$\mathcal{OP}_{AR}$,AND,*`EAR'`*)* (cf. Fig. 7a) the effect can be determined as intersection of the actor set 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. 7b), 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 Def. 7.

Definition 7 (Root vs. substructure level changes) *Let* $\mathcal{OP}_{AR}$ *be the operator tree representation of access rule*

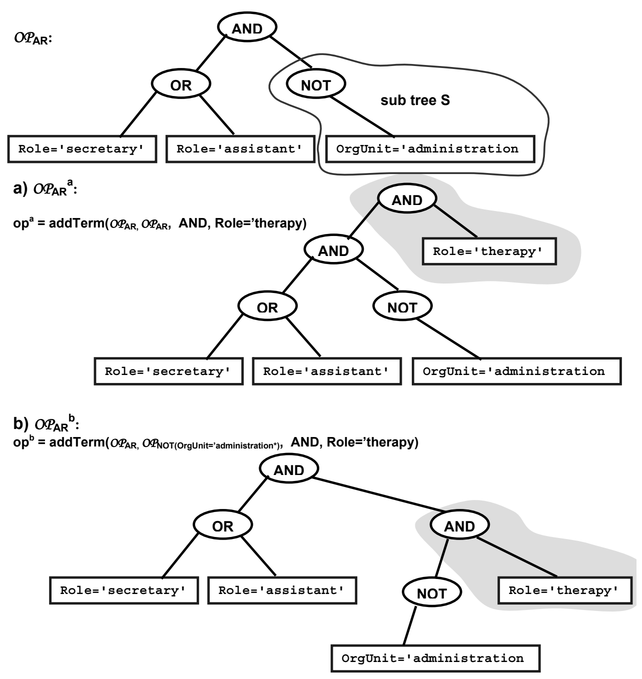


Figure 7. Access rule changes

$\texttt{AR} \in \mathcal{AR}^{OM}$ and let $\texttt{op}$ *be a basic change operation which transforms* $\texttt{AR}$ *into another access rule $\texttt{AR}' \in \mathcal{AR}^{OM}$. Then we denote* $\texttt{op}$ *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*

The root level change depicted in Fig. 7a) shows that operator tree $\mathcal{OP}_{AR}$ grows by adding a new root node, whereas the substructure level change (cf. Fig. 7b) is conducted by substituting sub tree S = $\mathcal{OP}_{NOT(OrgUnit='administration')}$ by sub tree S' = $\mathcal{OP}_{(NOT(OrgUnit='administration'))AND(Role='therapist')}$. As it can be seen new root AND has been added to S.

4.2 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. 9); 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, the question is if, for example, 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 1 (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 $\texttt{AR}_S$ and T to access rule $\texttt{AR}_T$*
- 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 as follows (see Fig. 8):*

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) $\cap$ VAS(OM, EAR)
  $\Longrightarrow$ **Reduction**
- *op: addTerm($\mathcal{OP}_{AR}$, $\mathcal{OP}_{AR}$, OR, EAR) = $\mathcal{OP}_{AR'}$:*
  VAS(OM,AR') = VAS(OM,AR) $\cup$ VAS(OM,EAR)
  $\Longrightarrow$ **Expansion**

op: deleteTerm($\mathcal{OP}_{AR}$,S) = $\mathcal{OP}_{AR'}$
- *root($\mathcal{OP}_{AR}$) = AND:*
  VAS(OM,AR) = VAS(OM,AR_T) $\cap$ VAS(OM,AR_S)
  $\subseteq$ VAS(OM,AR_T) = VAS(OM,AR')
  $\Longrightarrow$ **Expansion**
- *root($\mathcal{OP}_{AR}$) = OR:*
  VAS(OM,AR) = VAS(OM,T) $\cup$ VAS(OM,S)
  $\supseteq$ VAS(OM,T) = VAS(OM,AR')
  $\Longrightarrow$ **Reduction**

op: negateTerm($\mathcal{OP}_{EAR}$, $\mathcal{OP}_{EAR}$) = $\mathcal{OP}_{AR'}$:
VAS(OM,AR') = Actors $\setminus$ VAS(OM,EAR)
$\Longrightarrow$ *effect cannot be determined in terms of expansion, reduction, or zero_effect*

Fig. 8 illustrates the different cases covered by Prop. 1.

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 Prop. 1 for root level changes and are reflected by the *resulting sub tree* (cf. Def. 2). According to Prop. 1, it is not possible to derive the effects of negation in terms of *reduction*, *expansion*, or *zero_effect*. We know that VAS(OM,AR) and

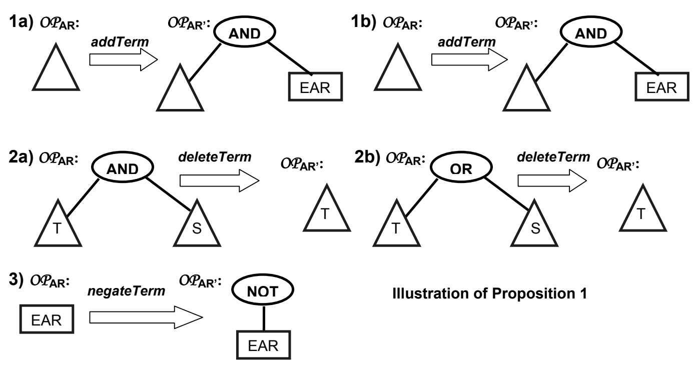


Figure 8. Effects of root level changes

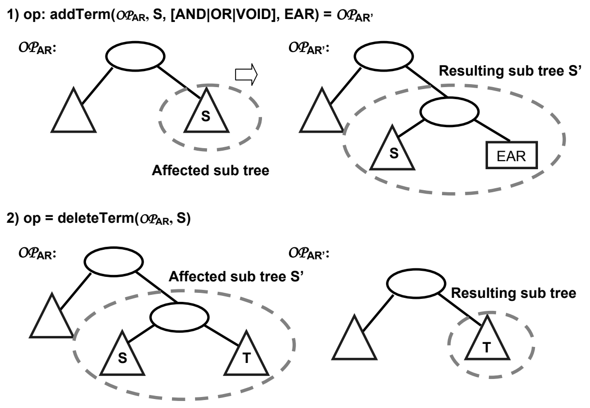


Figure 9. Affected and resulting sub trees

VAS(OM,AR') are *disjoint* which can be used to describe the semantics of the negation operation. To describe 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 this paper, we focus on operations *addTerm* and *deleteTerm* and leave *negateTerm* to future work.

Proposition 2 (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 (see Fig. 9):*

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 $\mathcal{OP}_{AR}$*

To determine the overall effect of a substructure level change on the whole 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. 10.

Definition 8 (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 Def. 8, the *one step push* can be formalized.

Proposition 3 (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. 10. The valid actor set of affected sub tree S is reduced according to Prop. 1; 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.

Finally, it has to be analyzed how a *multiple step push* towards the root affects the effects of substructure level changes. As stated in Prop. 3, 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

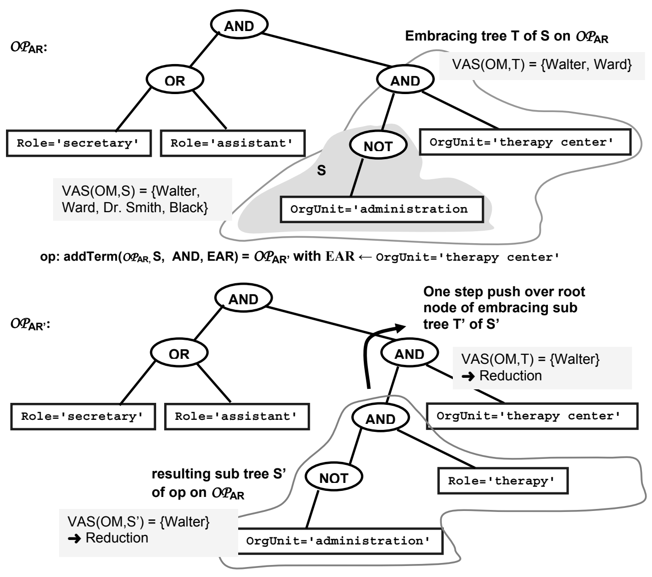

$\mathcal{OP}_{AR}$:
AND
OR
AND
Embracing tree T of S on $\mathcal{OP}_{AR}$
VAS(OM,T) = {Walter, Ward}
Role='secretary'
Role='assistant'
NOT
OrgUnit='therapy center'
S
VAS(OM,S) = {Walter, Ward, Dr. Smith, Black}
OrgUnit='administration
op: addTerm($\mathcal{OP}_{AR}$, S, AND, EAR) = $\mathcal{OP}_{AR'}$ with EAR ← OrgUnit='therapy center'
$\mathcal{OP}_{AR'}$:
AND
One step push over root node of embracing sub tree T' of S'
OR
AND
VAS(OM,T) = {Walter} ➜ Reduction
Role='secretary'
Role='assistant'
AND
OrgUnit='therapy center'
resulting sub tree S' of op on $\mathcal{OP}_{AR}$
NOT
Role='therapy'
VAS(OM,S') = {Walter} ➜ Reduction
OrgUnit='administration'

Figure 10. Effects of one step push (example)

determined quickly. This is important, for example, in the context of adapting user worklist in PAIS as we will discuss in Sect. 4.4.

Proposition 4 (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.*

4.3 High-level access rule changes

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., if the effect of op_i is a reduction and the effect of op_j is an *expansion* ($i \neq j$). In this case, valid actor sets before and after the high-level change have to be re-calculated and compared. Generally, re-calculation could be used for determining the effects of basic change operations as well. However, as we will show in Sect. 4.4, 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.

Several optimizations exist 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 R1 by role R2 resulting in AR' ← Role='R2'. Then, if we knew 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 will be 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 an *expansion*. Same considerations hold for the hierarchial relations between organizational units. We omit formal definitions here.

4.4 Discussion

Using our CEOSIS approach, any basic substructure level change can be treated as root level change since the effect remains the same when being pushed towards the root. Thus, the effect of any access rule change can be precisely determined in terms of *reduction*, *expansion*, or *zero_effect* of valid actor sets. One big advantage in the context of access control and user worklist management in PAIS 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. Thus, for direct access rule changes, the approach presented in this paper supports a quick check on the effects such that adequate action can be taken; e.g., a delayed propagation. For access rule changes triggered by organizational modifications we have precisely determined the effects on the valid actor sets in [11]. For the third possibility, i.e., the interplay between organizational modifications and direct access rule changes, interesting effects on each other might occur, on which we will report in future papers.

5 Related work

In literature many approaches have been presented dealing with challenging issues related to access control (e.g., [7, 1, 18, 17]). Most of these approaches apply *Role-Based Access Control (RBAC)* models for defining and managing user privileges [6, 9, 1, 5]; 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 PAIS [3, 2, 16, 18, 17]. Practical issues related to RBAC (e.g., NIST's proposed RBAC standard, integration of RBAC with EIS infrastructures, RBAC in commercial products) are summarized in [5].

There are only few approaches [15, 7, 4] which address the problem of organizational change. In [7] eight categories of structural changes on organizational models are identified which can be captured by our change framework as well. 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. The approach introduced in [4] 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). In [15] important issues related to the controlled change of organizational models are discussed. 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).

6 Summary and outlook

In this paper, we introduced an approach for managing the life cycle of access rules. In our previous work, the impact of organizational changes on access rules have already been elaborated. Here, we focused on direct access rule changes (e.g., due to optimizations or access rule mining). To be able to directly change access rules, a complete set of change operations was presented. Furthermore, we precisely defined the formal semantics of these change operations in order to avoid any ambiguity when applying these operations. 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

[1] E. Bertino. Data security. *Data & Knowl. Eng.*, 25(1–2):199–216, March 1998.

[2] E. Bertino, E. Ferrari, and V. Alturi. The specification and enforcement of authorization constraints in WFMS. *ACM Trans. on Inf. and Sys. Sec.*, 2(1):65–104, 1999.

[3] R. Botha and J. Eloff. A framework for access control in workflow systems. *Information Management and Computer Security.*, 9(3):126–133, 2001.

[4] D. Domingos, A. Rito-Silva, and P. Veiga. Authorization and access control in adaptive workflows. In *Proc. ESORICS'03*, pages 23–28, 2003.

[5] D. Ferraiolo and D. Kuhn. Role based access control. In *15th National Computer Security Conference*, 1992.

[6] D. Ferraiolo, D. Kuhn, and R. Chandramouli. *Role–Based Access Control*. Artech House, 2003.

[7] J. Klarmann. A comprehensive support for changes in organizational models of workflow management systems. In *Proc. ISM'01*, pages 375–387, 2001.

[8] T. Ly, S. Rinderle, P. Dadam, and M. Reichert. Mining staff assignment rules from event-based data. In *Int'l Workshop BPI'05*, pages 177–190, 2005.

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

[10] S. Rinderle and M. Reichert. On the controlled evolution of access rules in cooperative information systems. In *CoopIS'05*, pages 238–255, 2005.

[11] S. Rinderle and M. Reichert. A formal framework for adaptive access control models. *Int'l Journal of Data Semantics*, IX(9):82–112, 2007.

[12] S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems – a survey. *Data and Knowl. Engineering*, 50(1):9–34, 2004.

[13] S. Rinderle, M. Reichert, and P. Dadam. Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004.

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

[15] W. v.d. Aalst and S. Jablonski. Dealing with workflow change: Identification of issues an solutions. *Int'l Journal of Comp. Systems, Science and Eng.*, 15(5):267–276, 2000.

[16] J. Wainer, P. Barthelmess, and A. Kumar. W–RBAC – a workflow security model incorporating controlled overriding of constraints. *International Journal of Collaborative Information Systems*, 12(4):455–485, 2003.

[17] B. Weber, M. Reichert, W. Wild, and S. Rinderle. Balancing flexibility and security in adaptive process management systems. In *CoopIS'05*, pages 59–76, 2005.

[18] M. zur Muehlen. Resource modeling in workflow applications. In *1999 Workflow Management Conf.*, pages 137–153, 1999.