# An Approach to Bridge the Gap between Role Mining and Role Engineering via Migration Guides

Anne Baumgrass, Mark Strembeck
Institute for Information Systems and New Media
Vienna University of Economics and Business (WU Vienna), Austria
Email: {firstname.lastname}@wu.ac.at

***Abstract*—Mining approaches, such as role mining or organizational mining, can be applied to derive permissions and roles from a system's configuration or from log files. In this way, mining techniques document the current state of a system and produce *current-state RBAC models*. However, such current-state RBAC models most often follow from structures that have evolved over time and are not the result of a systematic rights management procedure. In contrast, role engineering is applied to define a tailored RBAC model for a particular organization or information system. Thus, role engineering techniques produce a *target-state RBAC model* that is customized for the business processes supported via the respective information system. The migration from a current-state RBAC model to a tailored target-state RBAC model is, however, a complex task. In this paper, we present a systematic approach to migrate current-state RBAC models to target-state RBAC models. In particular, we use model comparison techniques to identify differences between two RBAC models. Based on these differences, we derive migration rules that define which elements and element relations must be changed, added, or removed. A *migration guide* then includes all migration rules that need to be applied to a particular current-state RBAC model to produce the corresponding target-state RBAC model. In addition, we discuss different options for tool support and describe our implementation for the derivation of migration guides which is based on the Eclipse Modeling Framework (EMF).**

## I. Introduction

In role-based access control (RBAC), roles are used to model different job positions and responsibilities within a particular organization or within an information system (see, e.g., [10], [30]). They are equipped with a number of permissions that grant a role the rights to perform specific operations on specific objects. Human users (subjects) are assigned to roles according to their competencies and responsibilities in the organization.

### A. Specifying RBAC Models

Role mining approaches apply data mining techniques to derive RBAC models from the software systems of an organization (see, e.g., [12], [13], [18]). For example, role mining is applied to detect patterns in permission-to-subject assignments which are then used to derive roles. Furthermore, organizational mining techniques can be used to group people into "functional units" based on their execution of similar tasks (see, e.g., [27]). In this context, RBAC artifacts can also be derived from log files that document the execution history of business processes in an information system (see, e.g., [3], [4]). In this way, mining techniques document the current state of a system and produce *current-state RBAC models*.

In contrast, role engineering is applied to define a tailored RBAC model for a certain organization or information system (see, e.g., [9], [10], [30]). In particular, role engineering derives permissions from the descriptions of business processes and scenarios that specify the workflows conducted in a particular organization. Thus, role engineering techniques produce a *target-state RBAC model* that is customized for the business processes which are supported via the respective information system. Similar to mining techniques, certain steps in the role engineering process can be automated (see, e.g., [5], [21], [29]).

### B. Motivation

Despite well-elaborated access control approaches existing in the literature and a widespread use of (collaborative or process-based) software systems in professional organizations, the management of access control policies and constraints for real-world software systems often seems to be more of an "art form" than the result of a systematic rights management procedure. While there are a number of possible explanations for this phenomenon[1], we also see a strong interest of executives to change this situation and to introduce systematic rights management standards in their organizations. From our experiences gained in role engineering projects with companies and area municipalities (see, e.g., [19], [30]), we can say that, compared to the situation we had just a few years ago, there is a significantly increased awareness at the executive level that access control (and information system security in general) is a management topic. However, starting from a situation where we most often have an incomplete documentation of a company's business processes and a system's configuration, the adoption of a systematic rights management procedure is a very complex task.

In this context, mining approaches can be applied to derive *current-state RBAC models* (see Section I-A). However, such current-state RBAC models most often follow from structures

[1]For example, often no organization-wide standards for permission assignment and revocation exist; after an initial assignment of permissions for a certain job position additional permissions are assigned by system administrators in a ad hoc fashion – in this way, long-term employees accumulate rights; business processes and corresponding permission or role assignments are insufficiently documented – sometimes they are not documented at all.

that have evolved over time and are not the result of a systematic rights management approach. In contrast, role engineering is applied to define a tailored RBAC model for a certain organization or information system. Yet, the migration from a current-state RBAC model to a tailored target-state RBAC model is a non-trivial task.

In this paper, we present a systematic approach to migrate current-state RBAC models to target-state RBAC models. In Section II, we first give a high-level overview of our approach. Subsequently, Section III discusses the different dimensions in the model comparison context. Next, Section IV presents the phases that we consider for a comparison of RBAC models. In Section V, we describe the *migration guide* which is a visualization variant for a comparison of RBAC models. In Section VI, we present the different options for tool support of comparison techniques and Section VII provides a discussion concerning the suitability of these approaches for the migration of RBAC models. Section VIII discusses related work and Section IX concludes the paper[2].

## II. Approach Synopsis

We apply model comparison techniques (see, e.g., [1], [16]) to identify differences of current-state RBAC models and target-state RBAC models. Based on these differences, we derive migration rules that define which elements and element relations must be changed, added, or removed. A migration guide then includes all migration rules that need to be applied to a particular current-state RBAC model to produce the corresponding target-state RBAC model. Figure 1 shows the different steps in the comparison process and the artifacts that are produced as a result of each step.

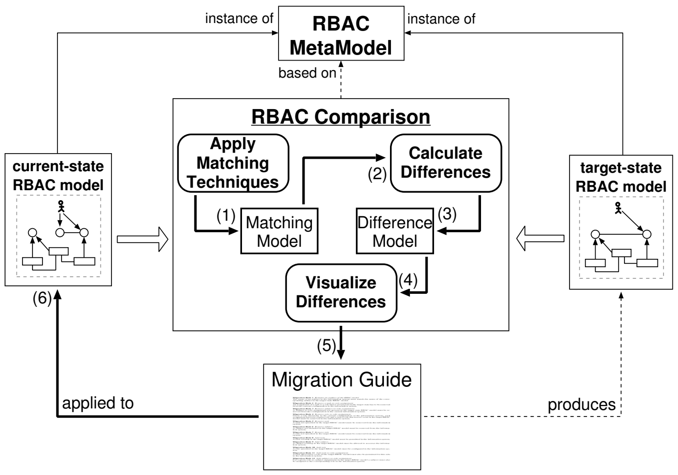


Figure 1. Approach overview: derivation of a migration guide

In the first step, model matching techniques are applied to produce a so-called *matching model* for the respective current-state and target-state RBAC model. Second, this matching model is used as input for a difference calculation. Based on this difference calculation, model differencing techniques are applied to derive a *difference model*. In steps four and five, this difference model is further processed to visualize the differences in a human-readable format and to produce the corresponding *migration guide*. In the sixth and final step, the migration guide is applied to the current-state RBAC model to convert it into the respective target-state RBAC model (see Figure 1).

In particular, we apply similarity-based matching techniques (see [16]) to identify elements with common attributes and relations that are included in the current-state as well as the target-state RBAC models. Using a customized matching algorithm (see Section IV) we obtain the matching model which contains the elements of the current-state RBAC model and identifies the elements of the target-state RBAC model that are either equal, similar, or unequal. After a customized difference calculation based on the matching model, we obtain a difference model that highlights the elements of the current-state RBAC model that need to be added, deleted, changed, or moved in order to produce the target-state RBAC model. Based on the difference model we derive the corresponding migration guide. In essence, the *migration guide* is a difference catalog for two specific RBAC models. The *migration rules* included in the migration guide describe which RBAC model elements and element relations have to be changed, added, or removed. Thereby, they describe a sequence of operations that can be applied to migrate the current-state RBAC model to the target-state RBAC model (see Section V).

Note that, although it would be possible, we do *not* automate the migration from a current-state to a target-state RBAC model. This is because we think that the security configuration of a software system is too sensitive for such an automation step and should always be approved by a security engineer. We do, however, provide tool support to produce the migration guide and to enable a semi-automated step-by-step migration that is conducted by a security engineer (see Section VI).

## III. Identifying Model Differences

Model comparison involves the tasks to produce a matching model and then calculate, represent, and visualize model differences (see [16], [22]). Figure 2 gives an overview of the different dimensions that need to be considered in the model comparison context.

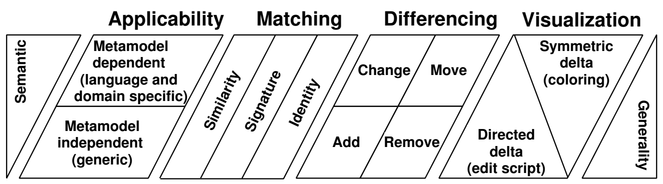

Applicability
Matching
Differencing
Visualization
Semantic
Metamodel dependent (language and domain specific)
Metamodel independent (generic)
Similarity
Signature
Identity
Change
Move
Add
Remove
Symmetric delta (coloring)
Directed delta (edit script)
Generality

Figure 2. Different dimensions of model comparison

In general, one can distinguish between metamodel independent (generic) and metamodel dependent (language and domain-specific) model comparison approaches. *Metamodel independent* approaches are able to compare models based

[2]We provide an extended version of this paper on our Web page. In the extended version we re-inserted the text and examples that we had to cut from the paper due to the page restrictions for the proceedings version.

on arbitrary metamodels (see, e.g., [15], [16]). In addition, such approaches are often adaptable and can be configured to (efficiently) compare models that are based on the same metamodel (see, e.g., [6], [32]). In contrast, *metamodel dependent* comparison approaches provide language-specific and domain-specific matching algorithms. For these approaches, syntactic and semantic information of the respective (textual or graphical) modeling language or modeling domain is considered to calculate differences between two models (see, e.g., [7], [23]).

The application of metamodel dependent comparison approaches in the context of RBAC models has the advantage that negligible changes in RBAC models (such as the order of the elements in an RBAC model) can be excluded from a model comparison via customization. Moreover, metamodel dependent approaches are able to consider the syntax and semantics of specific (modeling) languages to make the comparison more precise. However, tailoring a comparison approach to a certain syntax and corresponding language semantics usually requires a high customization effort.

*Model matching* is conducted to find common elements in two comparable models. A matching is based on unique identities, the signature, or the similarity of the elements in two models (see, e.g., [15], [16], [32]). In identity-based matching, elements with the same persistent identifier are matched. In signature-based matching, the uniquely identifying signatures of model elements are used for a matching. The signature of a certain element can be calculated based on the attributes and relations of this element. For similarity-based matching, a so-called similarity function calculates the similarity between two model elements. Elements are considered to be similar if the result of the calculation is greater than a predefined threshold value. Corresponding algorithms can be customized, for example by assigning weights to attributes or to relations of the elements to express their relevance for the calculation. Furthermore, language-specific matching algorithms are similarity-based approaches that are customized for a particular modeling language (see, e.g., [16]).

A *matching model* is the result of a matching algorithm. It consists of equal, similar, or unequal (unmatched) pairs of elements originating from the compared models. The matching model is the basis for the *difference calculation*. The differences of two models are derived from the similar and unequal elements and result in rules that describe what elements or element relations were (or need to be) changed, added, or removed from one model to the other.

*Visualizations* of model differences can be subdivided into two types of approaches: "symmetric delta" and "directed delta" (see [22]). Symmetric delta displays the differences as a union of two compared models. For example, "coloring" techniques produce a diagram that highlights the equal parts as well as the unequal parts (e.g. by using different colors). Directed delta, also called "edit script", describes a sequence of operations needed to convert the current model into the future model. This means, it describes actions specifying how the current model (e.g. a current-state RBAC model) must be modified to produce the future model (e.g. a target-state RBAC model).

For the purposes of this paper, we assume the most general case that current-state RBAC models and target-state RBAC models are constructed (or derived) independently of each other. For this reason, we cannot apply identity-based or signature-based matching algorithms which use persistent identifiers or the signature of model elements. In our approach, we therefore use metamodel dependent similarity-based matching algorithms. To consider the language-specific semantic and syntax of RBAC models we customized the matching and difference calculation (see also Sections IV-B and IV-C).

## IV. Comparing current-state and target-state RBAC Models

Comparing two models means to find their equivalences, similarities, and differences. Although the general characteristics of two RBAC models, such as the number of roles, subjects, or permissions/tasks, can be used to measure a similarity value between these RBAC models, this type of similarity value is most often unsuitable to reveal their differences. This means, while the number of elements in two models can be identical, the RBAC models may have completely different semantics. Therefore, this paper does not aim to determine the similarity of two models but to reveal the elements that differ between two compared RBAC models. Thus, we examine the properties and associations of each RBAC artifact (subjects, roles, permissions/tasks, and constraints). The subsequent sections describe the phases for the migration of current-state RBAC models to target-state RBAC models.

### *A. Definition of RBAC Models*

A current-state RBAC model can be derived via mining approaches. On the other hand, target-state RBAC models are defined by applying role engineering techniques (see Section I-A). For our approach, we assume that current-state and target-state RBAC models conform to the same RBAC metamodel. We think this is a reasonable assumption because RBAC is a well-understood domain with well-defined model elements (subjects, roles, permissions/tasks, and constraints). Furthermore, we assume that the RBAC models are available (or can be exported) in a machine-readable format that we can use for our model comparison, for example as XML documents. In this paper, we use the process-related RBAC metamodel from [31] and export RBAC models in XML-based formats for model comparison purposes. We decided to use this metamodel because we apply it in our role engineering projects and because a UML extension for this metamodel exists that allows for a straightforward visualization of the different model elements (see [31]). Note, however, that our general approach for the derivation of migration guides does not rely on a specific RBAC metamodel variant but can easily be adapted to other metamodel variants. The metamodel from [31] includes the basic concepts of process-related RBAC models, i.e. subjects, roles, tasks, and (business) processes.

In addition, it supports the definition of mutual exclusion and binding constraints for tasks.

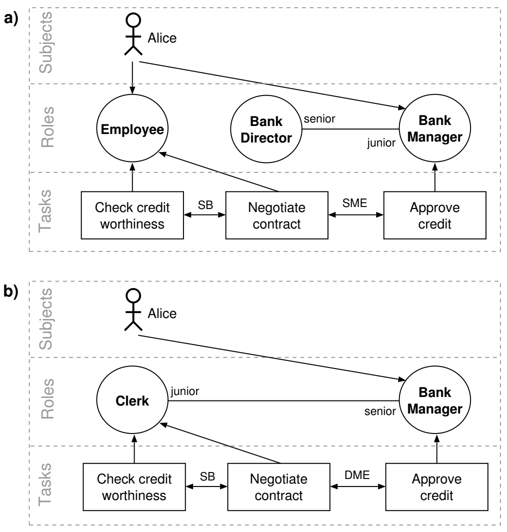


Figure 3. Examples for a) a current-state and b) a target-state RBAC model

Figure 3 shows two simple RBAC models that we use as a running example in the remainder of this paper. In Figure 3a, the current-state RBAC model consists of three roles named “Employee”, “Bank Manager”, and “Bank Director”, the three tasks “Negotiate contract”, “Approve credit”, and “Check credit worthiness”, and a subject named “Alice”. Alice is assigned to the roles “Employee” and “Bank Manager”. The role “Bank Manager” is a junior-role of “Bank Director”. Furthermore, the current-state RBAC model defines a subject-binding (SB) constraint between the tasks “Negotiate contract” and “Check credit worthiness” and a static mutual exclusion (SME) constraint is defined between the tasks “Negotiate contract” and “Approve credit”. In Figure 3b, the target-state model contains a role “Clerk” and its senior-role “Bank Manager”, three tasks named “Approve contract”, “Negotiate contract”, and “Check credit worthiness”, and the subject “Alice” assigned to the “Bank Manager” role. The target-state RBAC model also includes a SB constraint between the tasks “Negotiate contract” and “Check credit worthiness” and a dynamic mutual exclusion (DME) constraint between the tasks “Negotiate contract” and “Approve contract”.

## B. Matching of RBAC Models

When comparing two RBAC models, we essentially distinguish two perspectives. The *content perspective* considers the properties of an artifact. For our purposes, we use the unique identifier and the name of an artifact as properties in the content perspective. The *context perspective* considers the structure of an artifact, i.e. its relations to other elements (e.g. role-to-subject assignments or a mutual exclusion relation between two permissions/tasks). Thus, an RBAC artifact is characterized through its attributes and its relations to other artifacts. For example, the tasks assigned to a role represent a part of the role’s context and the name of the role represents its content. Table I summarizes the RBAC artifacts and corresponding relations in the context perspective that are considered in a matching calculation.

| RBAC Artifact | Relation |
|---|---|
| Subject | Role-to-subject assignment |
| Role | Senior-role relation<br>Junior-role relation<br>Permission/task-to-role assignment |
| Task | Dynamic mutual exclusion (DME) constraint<br>Static mutual exclusion (SME) constraint<br>Role-binding (RB) constraint<br>Subject-binding (SB) constraint |

Table I
ARTIFACTS AND RELATIONS IN THE CONTEXT PERSPECTIVE

In particular, we apply a similarity-based matching to find common elements with common attributes and relations in current-state and target-state RBAC models (see Section III). We customized this matching approach to include RBAC-specific matchings. For example, we assume that the context of RBAC artifacts is more relevant than their names. This is because, changing the context of an artifact may change the semantic meaning of an artifact, while renaming does not necessarily define a new meaning for an artifact. Therefore, we first compute a matching between artifacts with a similar context. For example, the role named “Employee” of the current-state RBAC model and the role named “Clerk” of the target-state RBAC model (see Figure 3) are considered to be similar when their associations to other artifacts are similar. In addition, we can apply a linguistic comparison to identify semantic similarities between artifact names (see, e.g., [25]). The similarity between the name “Clerk” and “Employee” can be identified, for example, via hypernyms, common substrings, string edit distance metrics, user-provided name matchers, a pre-defined taxonomy, an ontology, or dictionary systems (see, e.g., [25]).

At the end of this phase, the matching model contains elements of the current-state RBAC model and the counterparts from the target-state RBAC model that are equal, similar, or unmatched. An excerpt of the matching model for our example from Figure 3 is shown in Figure 4. Equal elements are not considered for further analysis, since they do not need to be adapted to produce the target-state RBAC model. Thus, the similar or unequal elements from the matching model are used to identify model differences in the subsequent phase.

## C. Calculating Differences of RBAC Models

Based on the matching result from the previous phase, we calculate a delta that contains the differences between the RBAC models we compared. These differences include artifacts or artifact relations that need to be added, removed, moved, or changed in the current-state RBAC model to produce the target-state RBAC model (see Figure 2). The

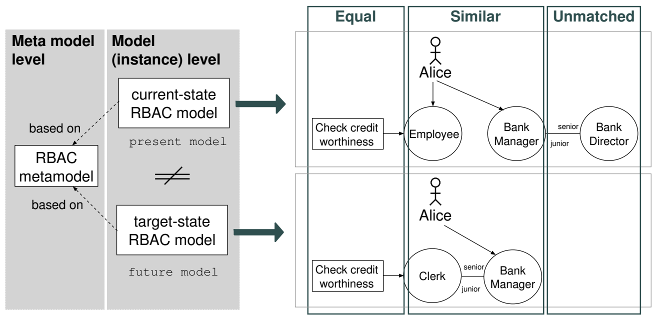


Figure 4. RBAC model matching example

visualization of these differences is conducted in the next phase (see Section IV-D).

Our customization of the matching algorithm to the specific characteristics of RBAC models (e.g. the context of an artifact is more relevant than its name, see Section IV-B), identifies the roles "Employee" and "Clerk" as similar elements (see Figure 4). Therefore, the differencing algorithm proposes to rename the "Employee" role of the current-state RBAC model into "Clerk" instead of removing the "Employee" role and adding a new role named "Clerk" in the current-state RBAC model.

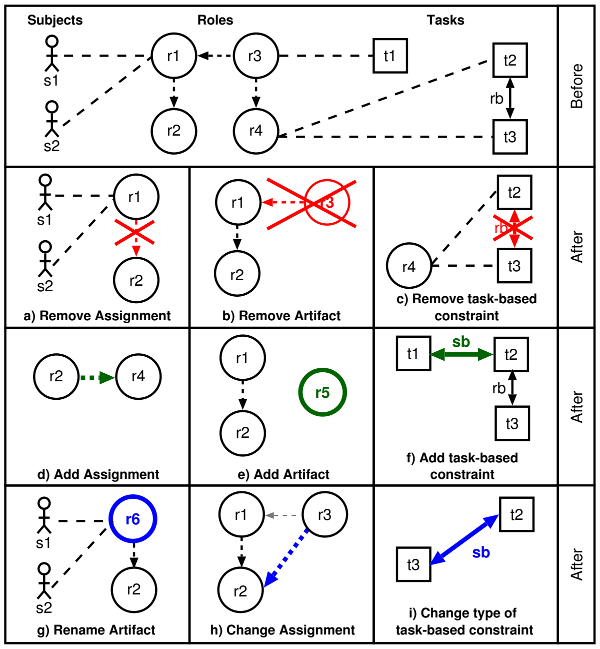


Figure 5. Differences for artifacts, assignments, and constraints

In general, we differentiate between nine classes of differences that may result from the comparison of two RBAC models (see Figure 5 – removals are colored red, additions are colored green, and changes are colored blue). *Removals* include removed assignment relations, artifacts, and constraints between tasks (see Figure 5a-c). For example, Figure 5a, shows a role-to-role assignment between the roles *r1* and *r2* that has been removed from the model (with respect to the model in the "Before" compartment of Figure 5). In case an artifact is removed, the respective relations to other artifacts (assignments or constraints) also have to be removed. *Additions* include the definition of new assignment relations, artifacts, and constraints between tasks (see Figure 5d-f). For example, Figure 5f shows a new subject-binding (SB) constraint between the tasks *t1* and *t2* that has been added to the model. *Changes* include renaming an artifact, changing an assignment relation, or changing the type of a constraint (e.g. changing a static-mutual exclusion (SME) constraint into a dynamic mutual exclusion (DME) constraint or vice versa) (see Figure 5g-i). Note that we consider *moved artifacts* as change of the corresponding assignment relations. In other words, moving an RBAC artifact means a change of one (or more) assignment relation(s). In Section V, we define the migration guide based on these nine classes of differences.

## D. *Visualizing Differences of RBAC Models*

The *difference model* contains information about the artifacts, assignment relations, and constraints that have to be added, removed, or changed in the current-state RBAC model to produce the target-state RBAC model. To actually conduct such a migration, we have to "visualize" the difference model in a human-readable (and/or machine-readable) format. Edit scripts (see Section III) for RBAC models describe a sequence of add, remove, or change operations to convert the current-state RBAC model into the target-state RBAC model. The *migration guide* is one particular visualization of an edit script resulting from the comparison of two RBAC models (Section V describes migration guides in detail). A symmetric delta, also referred to as coloring, is used to display differences in a diagram. This diagram is the union of two compared models with common and differential parts highlighted (see Section III).

## V. Migration Guides for RBAC Models

In our approach, the *migration guide* is a visualization variant of the difference model (see Section IV-C) and contains the corresponding migration rules. Each *migration rule* (MR) describes a particular edit step. The migration guide includes an ordered list of migration rules and thereby describes the ordered sequence of edit steps that need to be applied to a particular current-state RBAC model to produce the corresponding target-state RBAC model. In other words, the migration guide describes the modifications for a current-state RBAC model so that the result conforms to the target-state RBAC model.

Based on the nine difference classes described in Section IV-C, we define the following generic migration rules:

**Remove rules:**

Migration Rule *1: Remove a constraint*
A (mutual exclusion or binding) constraint that is not included in the target-state RBAC model must be removed from the current-state RBAC model.

Migration Rule *2: Remove an assignment relation*
An assignment relation that is not included in the target-state

RBAC model must be removed from the current-state RBAC model.

Migration Rule 3: *Remove an artifact*
An artifact (subject, role, permission/task) that is not included in the target-state RBAC model must be removed from the current-state RBAC model.

**Change rules:**

Migration Rule 4: *Rename an artifact*
The name of an artifact in the current-state RBAC model must match the name of the corresponding artifact in the target-state RBAC model.

Migration Rule 5: *Change an assignment relation*
An assignment relation in the current-state RBAC model must be equal to the comparable assignment relation in the target-state RBAC model.

Migration Rule 5.1: *Change the source of an assignment*
The source of an assignment relation in the current-state RBAC model must be equal to the source of the comparable assignment relation in the target-state RBAC model.

Migration Rule 5.2: *Change the target of an assignment*
The target of an assignment relation in the current-state RBAC model must be equal to the target of the comparable assignment relation in the target-state RBAC model.

Migration Rule 6: *Change the type of a constraint*
The type of a (mutual exclusion or binding) constraint in the current-state RBAC model must be equal to the type of the comparable constraint in the target-state RBAC model.

**Add rules:**

Migration Rule 7: *Add an artifact*
An artifact that is included in the target-state RBAC model but is absent in the current-state RBAC model must be added to the current-state model.

Migration Rule 8: *Add an assignment relation*
An assignment relation that is included in the target-state RBAC model but is absent in the current-state RBAC model must be added to the current-state model.

Migration Rule 9: *Add a constraint*
A (mutual exclusion or binding) constraint that is included in the target-state RBAC model but is absent in the current-state RBAC model must be added to the current-state model.

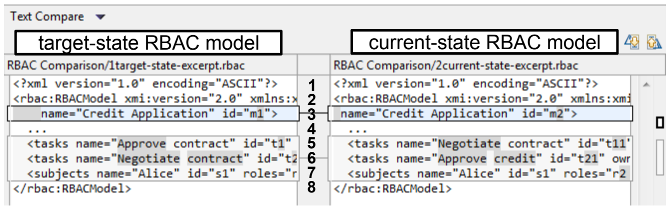
The *migration guide* recommends an ordered sequence of migration rules. In general, this ordered sequence results from the following heuristic: First, constraints and assignment relations which are not included in the target-state RBAC model are removed from the current-state RBAC model (see MR 1 and 2). Afterwards, RBAC artifacts that are not included in the target-state model are removed from the current-state model (see MR 3). Second, artifact attributes are changed in the current-state RBAC model in order to match a comparable artifact from the target-state RBAC model (see MR 4). Third, assignment relations are changed (see MR 5). A change of an assignment relation is a change of its source or target – for example the source of a role-to-subject assignment is the corresponding role and the target is the respective subject. Fourth, the (mutual exclusion or binding) constraints in the current-state RBAC model are changed (see MR 6). Fifth, missing RBAC artifacts (see MR 7), assignment relations (see MR 8), and constraints (see MR 9) are added to the current-state RBAC model.

## VI. Tool Support to Visualize RBAC Model Differences

In the following subsections, we present different options for tool support of comparison techniques for RBAC models. Because the comparison of arbitrary graphical models is extremely complex and does provide an additional benefit for our purposes, we use a special-purpose XML format as a textual representation of the corresponding RBAC models. The XML documents describing the respective RBAC models are then analyzed and compared. The different tool options use edit scripts or colored diagrams (see Section III) to visualize RBAC model differences and to enable the migration of current-state to target-state RBAC models.

### A. Line-based Visualization of RBAC Differences

In this type of comparison, each model is considered as a piece of text (e.g. via XML-based documents) for which a line-based comparison is conducted. The lines of the text are compared with each other to reveal added, deleted, and changed lines of text. However, the textual representation of the same model may include semantical differences as well as structural differences that do not change model's semantics.


Text Compare
target-state RBAC model
current-state RBAC model
RBAC Comparison/1target-state-excerpt.rbac
RBAC Comparison/2current-state-excerpt.rbac

```
1 <?xml version="1.0" encoding="ASCII"?>
2 <rbac:RBACModel xmi:version="2.0" xmlns:x
3     name="Credit Application" id="m1">
4   ...
5   <tasks name="Approve contract" id="t1"
6   <tasks name="Negotiate contract" id="t2
7   <subjects name="Alice" id="s1" roles="r
8 </rbac:RBACModel>
```

```
1 <?xml version="1.0" encoding="ASCII"?>
2 <rbac:RBACModel xmi:version="2.0" xmlns:xmi
3  name="Credit Application" id="m2">
4   ...
5   <tasks name="Negotiate contract" id="t11'
6   <tasks name="Approve credit" id="t21" owr
7   <subjects name="Alice" id="s1" roles="r2
8 </rbac:RBACModel>
```

Figure 6. Line-based visualization of differences

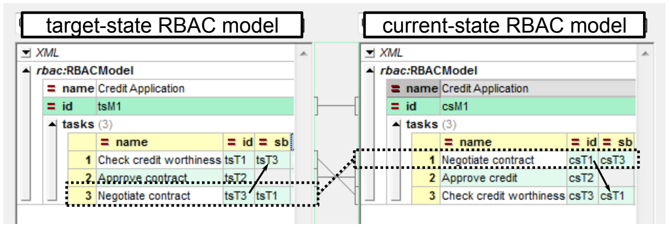
For example, Figure 6 shows a metamodel independent comparison performed with Eclipse[3] for the models visualized in Figure 3. Apart from the highlighting of different identifiers in both models (e.g. in line 3), this type of comparison also highlights structural changes in the RBAC model, such as the order of artifacts. Therefore, such a line-based comparison may suggest changes that are based on structural differences but are not necessary from a semantic point of view. In particular, the comparison from Figure 6 suggests to rename the task "Negotiate contract" to "Approve contract" because these two tasks are in the same line of the corresponding XML documents (line 5). Even the introduction of line breaks or tab-stops are shown as difference between the models (line 3).

### B. Tree-based Visualization of RBAC Differences

Each XML document essentially describes a tree structure. A tree-based comparison of two XML documents is significantly more powerful than a line-based comparison. In

[3]http://eclipse.org/

particular, a tree-based comparison is able to find similarities between differently ordered artifacts. For example, in the target-state RBAC model “Negotiate contract” is the third task in the corresponding XML tree (see left-hand side of Figure 7) which is compared with the first task of the current-state RBAC model (right-hand side of Figure 7). However, relations between model artifacts are not represented in a tree-based comparison. For example, in Figure 7, the subject-binding (SB) constraint (indicated via the “sb” attributes of the respective tasks) between the bound tasks “Negotiate contract” and “Check credit worthiness” is ignored because each of the two compared RBAC models references the tasks via different identifiers.



XML
rbac:RBACModel
name Credit Application
id tsM1
tasks (3)

| | name | id | sb |
|---|---|---|---|
| 1 | Check credit worthiness | tsT1 | tsT3 |
| 2 | Approve contract | tsT2 | |
| 3 | Negotiate contract | tsT3 | tsT1 |

XML
rbac:RBACModel
name Credit Application
id csM1
tasks (3)

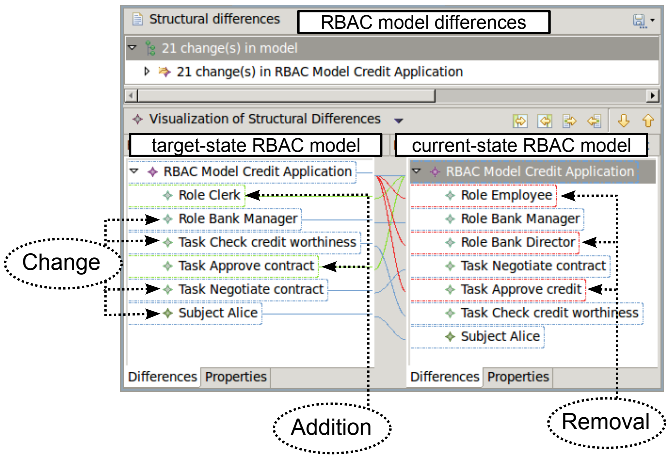
| | name | id | sb |
|---|---|---|---|
| 1 | Negotiate contract | csT1 | csT3 |
| 2 | Approve credit | csT2 | |
| 3 | Check credit worthiness | csT3 | csT1 |

Figure 7. Tree-based visualization of differences

A tree-based comparison allows us to define filters which enable to choose elements and attributes that should be ignored for comparison. For example, we can exclude element identifiers in an RBAC comparison. A number of software tools exist that support a tree-based comparison of XML documents. The comparison shown in Figure 7 was conducted with Altova DiffDog[4].

## C. Graph-based Visualization of RBAC Differences

In addition to the properties of tree-based model comparison, graph-based approaches can also consider (indirect) cross-references between different elements in a graph (e.g. expressed via attributes). Software tools such as EMF Compare support the comparison and merging of any kind of metamodel based on similarity techniques [6]. EMF Compare[5] is a part of the Eclipse Modeling Framework (EMF) [28] and can be used to calculate and visualize model differences that consider (indirect) relations between model elements. For an EMF comparison we first construct an Ecore metamodel to describe the syntax and semantics of our RBAC models. Ecore models are serialized using the XML Metadata Interchange (XMI) standard [24]. Thus, we are able to load XML-based files as models in Eclipse. To represent RBAC models in EMF we defined an Ecore metamodel based on the formal generic metamodel for process-related RBAC models from [31]. Based on this metamodel a model comparison reveals structural and semantic differences between two RBAC models. An example result of a difference calculation between the current-state RBAC model and the target-state RBAC model from Figure 3 is shown in Figure 8. This comparison is based on structural and semantic similarities rather than persistent identifiers (see Section III).



21 change(s) in model
21 change(s) in RBAC Model Credit Application

RBAC Model Credit Application
Role Clerk
Role Bank Manager
Task Check credit worthiness
Task Approve contract
Task Negotiate contract
Subject Alice

RBAC Model Credit Application
Role Employee
Role Bank Manager
Role Bank Director
Task Negotiate contract
Task Approve credit
Task Check credit worthiness
Subject Alice

Figure 8. Graph-based visualization of differences

The difference model depicted in Figure 8 shows how the current-state RBAC model must be adapted to produce the target-state RBAC model. The upper compartment of the window shows the difference/change count. In this example, the comparison revealed 21 differences between the models. The lower compartment of the window is divided in two separate panes. On the left-hand side it shows the artifacts of the target-state RBAC model and on the right-hand side the artifacts of the current-state RBAC model. In these panes deleted artifacts are surrounded by a red frame (e.g., the role *Employee*), changed artifacts are surrounded by a blue frame (e.g., the role *Bank Manager*) and additional artifacts are surrounded by a green frame (e.g., the role *Clerk*). This graph-based comparison disregards a structural similarity between artifacts if their names differ. Therefore, instead of simply renaming the role, this comparison suggests to remove “Employee” and to add a new role “Clerk” to the current-state RBAC model.

## D. Diagram-based Visualization of RBAC Differences

A diagram-based comparison between two models is a metamodel dependent comparison producing a symmetric delta. It visualizes differences via coloring of different diagram elements (see Section III). Figure 9 shows an example for a diagram-based comparison of the example from Figure 3. The added elements and relations are colored green, the changed elements and relations are colored blue, and elements and relations that have to be removed are colored red. Examples of similar approaches are presented in [23] and [26].

In particular, Figure 9 shows that the role “Bank Director” must be removed from the current-state RBAC model. For this reason, also the senior-role relation to the role “Bank Manager” has to be removed. Similarly, the role-to-subject relation from “Alice” to the role “Clerk” has to be removed. To build the target-state RBAC model, the role “Employee” of the current-state RBAC model has to be renamed into “Clerk” (colored in blue). Note that the original/previous

[4]http://www.altova.com/diffdog
[5]http://wiki.eclipse.org/EMF_Compare

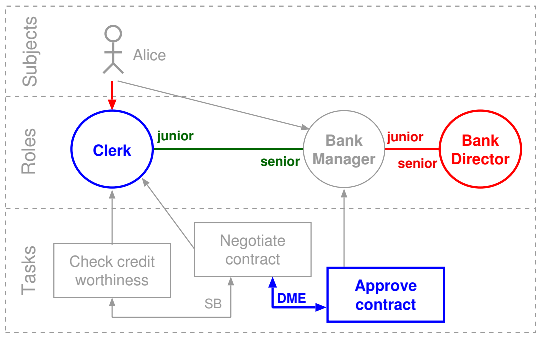


Figure 9. Diagram-based visualization of differences

names of changed artifacts are not visible in this type of visualization. Therefore, the person building the target-state RBAC model has to know which name the artifacts have in the current-state RBAC model in order to rename them. In addition, the task "Approve credit" task has to be renamed into "Approve contract" and the type of the task-based constraint between the two tasks is changed from a "SME" (static mutual exclusion) into a "DME" (dynamic mutual exclusion) constraint. Moreover, a role-to-role assignment relation has to be added to the current-state RBAC model to conform to the target-state RBAC model (colored in green).

### *E. Migration Guide as Visualization of RBAC Differences*

Based on the experiences we gained from comparing the different visualization options and the different options for tool support (see Sections VI-A to VI-D), we implemented a customized tool for the derivation of migration guides. Our implementation is based on EMF Compare (see Section VI-C). We customized the implementation of the matching and differencing algorithms in EMF Compare that enable us to apply a more specific similarity-based matching for RBAC models. For this reason, we extended the generic matching and difference engine (see [6]). In particular, we adapted different operations and procedures for the processing of RBAC-specific syntax and semantics. For example, a customization of the differencing algorithm enables us to ignore artifact identifiers in a comparison. The identifier is important to reference an artifact in a model. However, it is irrelevant if we compare two RBAC models. Therefore, we specify that this attribute is ignored for computing the difference between two RBAC models.

To identify renamed artifacts, we consider the context of an artifact rather than the artifact's name (see Section IV-B). For example, our matching algorithm suggests that the role "Employee" in the current-state RBAC model is similar to the role "Clerk" in the target-state RBAC model. The resulting change of this matching is the renaming of the role "Employee" into "Clerk". In contrast to the graph-based comparison described in Section VI-C, we obtain a more precise difference model. In the example, only 12 instead of 21 differences are identified (see Figure 8)[6].

With EMF Compare the difference model can be exported as XML document. Subsequently, we can automatically derive the migration guide from this XML document. To actually build a migration guide, we derive the sequence of actions from the XML-based difference model calculated with our customized comparison.

| | |
|---|---|
| MR 2 | Remove role-to-subject assignment relation between role *Employee* and subject *Alice*. |
| MR 2 | Remove role-to-role assignment relation between senior-role *Bank Director* and junior-role *Bank Manager*. |
| MR 3 | Remove role *Bank Director*. |
| MR 4 | Rename role *Employee* to *Clerk*. |
| MR 4 | Rename task *Approve credit* to *Approve contract*. |
| MR 6 | Change the type of the mutual exclusion constraint between the tasks *Negotiate contract* and *Approve contract* from a *SME* into a *DME* constraint. |
| MR 8 | Add a role-to-role assignment relation from senior-role *Bank Manager* to junior-role *Clerk*. |

Table II
MIGRATION GUIDE AS A SEQUENCE OF MIGRATION RULES

Table II shows a human-readable version of the migration guide derived from the XML-based difference model of the RBAC models from Figure 3. The left column references the corresponding generic migration rule (see Section V) while the right column describes which artifacts need to be adapted to produce this very target-state RBAC model.

## VII. DISCUSSION

Different comparison techniques can be used to visualize differences between RBAC models. In Section VI, we discussed a line-based, a tree-based, a graph-based, and a diagram-based visualization of differences as well as the migration guide. Each of the techniques was applied to the two models from Figure 3.

In this section, we discuss the visualization options with respect to their suitability for a migration from current-state to target-state RBAC models. For this purpose, we identified criteria to compare the visualization techniques (see Table III). In particular, the discussion in this section results from the practical application of the above mentioned model comparison techniques.

| | |
|---|---|
| Clarity | Extent to which the differences between RBAC models derived via a certain technique are understandable. |
| Conciseness | Extent to which a technique produces precise descriptions of the differences between RBAC models. |
| Expressiveness | Extent to which a technique considers the syntax and semantics of RBAC models. |
| Extensibilitxy | Extent to which a technique and its parameters can be customized for an RBAC comparison. |

Table III
SUMMARY OF CRITERIA FOR VISUALIZATION OPTIONS

[6]Note that this reduction in the number of differences means that we have more precise descriptions of the differences.

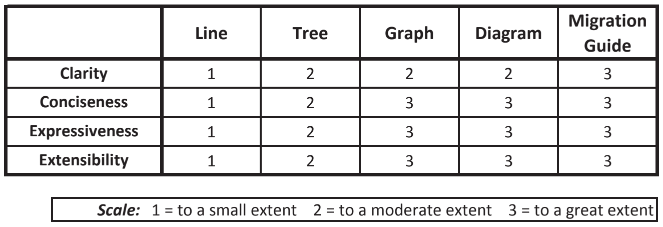
Line-based model comparison techniques are hardly human-readable for large and complex real-world models (see, e.g., [1], [33]). Furthermore, this technique does not consider structural or semantic similarities in the comparison of RBAC models. Tree-based approaches are also limited since relations between the model artifacts are ignored. Thus, tree-based comparison techniques also do not consider all structural or semantic similarities that are required for the comparison of RBAC models.

The ability to consider certain language-specific matchings and difference calculations makes the graph-based approach more appropriate than a line- or tree-based comparison. However, the user comparing the models needs specific knowledge about the respective tool (here EMF Compare) to interpret the results. Furthermore, specific knowledge on the underlying metamodel is required.

In principle, the diagram-based comparison can be used to visualize the differences and respective modifications of the current-state RBAC model. However, considering the size and complexity of real-world models, showing all changes in a single diagram (including unchanged artifacts) makes this type of visualization complex and cluttered. In addition, the security expert responsible for conducting the migration must be familiar with the modeling language that is used to visualize the RBAC models, and he has to be familiar with the current-state RBAC model, e.g. to identify artifacts that are renamed. Furthermore, from the diagram-based visualization it is not obvious in which order the changes have to be applied.

A migration guide documents an ordered sequence of steps that can be applied to migrate the current-state to the target-state RBAC model. It can be represented in natural language, hence, no knowledge of a specific modeling language is required for its understanding. In comparison with coloring approaches, the migration guide does not contain any unchanged elements or relations of the compared models. With regard to the compactness, the migration guide produces a human-readable difference model including only the artifacts involved in the changes. In other words, we automate the complex task to interpret the difference model and provide the respective security engineer with a structured sequence of migration rules that is easy to understand.

From our experiences, Figure 10 summarizes our results and shows the extent to which the visualization techniques from Section VI can fulfill the criteria from Table III.



| | Line | Tree | Graph | Diagram | Migration Guide |
|---|---|---|---|---|---|
| Clarity | 1 | 2 | 2 | 2 | 3 |
| Conciseness | 1 | 2 | 3 | 3 | 3 |
| Expressiveness | 1 | 2 | 3 | 3 | 3 |
| Extensibility | 1 | 2 | 3 | 3 | 3 |

*Scale:* 1 = to a small extent 2 = to a moderate extent 3 = to a great extent

Figure 10. Suitability of visualization techniques

In summary, the migration guide is customized to consider the language-specific syntax and semantics of RBAC models. In contrast to line-, tree-, graph, and diagram-based visualizations, a migration guide documents a concise sequence of steps which is human-readable and does not require knowledge of a specific modeling language for its understanding.

## VIII. Related Work

A number of approaches for policy analysis exist that focus on policy verification, conflict detection, and similarity detection. In [11], a research prototype called Margrave is presented to verify, analyze, and compare access-control policies defined via the eXtensible Access Control Markup Language (XACML). The change impact analysis of Margrave enables a comparison to reveal changes between two XACML policies. A semantic differencing between versions of policies is provided. However, the comparison focuses on policies defined in XACML. In a similar approach, Kolovski et al. [17] provide a formalization of XACML for a verification and a change impact analysis.

The RoleUpdater [14] is a tool that uses model checking techniques to provide suggestions how to update an access control system. In particular, one has to formulate update requests that are fed into the RoleUpdater. The RoleUpdater then checks if a security definition is already supported or it gives an example how the definition can be added to the system.

Evaluating the similarity of access control policies can be seen as preliminary step for policy analysis and comparison. In [20], Lin et al. present a policy similarity measure for XACML-based policies. The approach can detect the most similar policies from a given policy-set and then use the result of the similarity check as a starting point for policy merging. In [2], Backes et al. present an approach for the comparison of privacy policies.

Furthermore, the huge body of work on model comparison is directly related to the approach presented in this paper. EMF Compare [6] provides an approach for a differences calculation and representation of a model comparison. It supports different kinds of granularity and allows to customize the matching and difference calculation. Moreover, the Epsilon Comparison Language (ECL) (see [15]) and the ATL Transformation Language (ATL) (see, e.g., [8]) can be configured to compare models based on the same metamodel and provide difference models as a result. In this context, the representation and visualization of difference models is of high importance. A number of existing approaches aims to improve the readability and interpretation of difference models that are defined via edit scripts, via coloring, or through a combination of both (see, e.g., [23], [26], [33]). The approach presented in this paper, complements existing approaches. It enables the comparison of RBAC models and produces a migration guide that allows for a systematic migration between a current-state and a target-state RBAC model.

## IX. Conclusion

In this paper, we presented a systematic approach, including tool support, to migrate current-state RBAC models to target-state RBAC models. First, we use customized model comparison techniques to compare two RBAC models. Based on this comparison, we calculate the differences between these RBAC models and automatically derive corresponding migration rules. These rules are aggregated in a *migration guide*. This migration guide recommends a sequence of edit operations for a stepwise migration of the respective current-state model to the target-state RBAC model. Note that, although it would be possible, we do *not* automate the migration/transformation between a current-state and a target-state RBAC model. This is because we think that the security configuration of a software system is too sensitive for such an automation step and should always be approved by a security engineer. We do, however, provide tool support to produce the migration guide and to enable a semi-automated stepwise migration that is conducted by a security engineer.

With our approach, we try to help bridge the gap between role mining techniques and role engineering. Role mining techniques are well-suited to reveal the current configuration of an access control system (current-state RBAC model), while role engineering is focused on defining a tailored (desired) access control configuration (target-state RBAC model). However, the migration from a current-state to a target-state RBAC model is a very complex task, and neither role mining nor role engineering support such a migration. Our migration guides support this type of migration in a systematic way.

## References

[1] K. Altmanninger, M. Seidl, and M. Wimmer. A survey on model versioning approaches. *International Journal of Web Information Systems*, 5(3), 2009.

[2] M. Backes, G. Karjoth, W. Bagga, and M. Schunter. Efficient Comparison of Enterprise Privacy Policies. In *Proc. of the 2004 ACM Symposium on Applied Computing (SAC)*. ACM, 2004.

[3] A. Baumgrass. Deriving Current-State RBAC Models from Event Logs. In *International Workshop on Security Aspects of Process-aware Information Systems (SAPAIS), Proc. of the 6th International Conference on Availability, Reliability and Security (ARES)*. IEEE Computer Society, 2011.

[4] A. Baumgrass, S. Schefer-Wenzl, and M. Strembeck. Deriving Process-Related RBAC Models from Process Execution Histories. In *IEEE International Workshop on Security Aspects of Process and Services Engineering (SAPSE), Proc. of the 2012 IEEE 36th International Conference on Computer Software and Applications Workshops (COMPSACW)*, July 2012.

[5] A. Baumgrass, M. Strembeck, and S. Rinderle-Ma. Deriving Role Engineering Artifacts from Business Processes and Scenario Models. In *Proc. of the 16th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2011.

[6] C. Brun and A. Pierantonio. Model Differences in the Eclipse Modelling Framework. *UPGRADE, The European Journal for the Informatics Professional*, IX(2), April 2008.

[7] Y. Chen, F. Douglis, H. Huang, and K. Vo. TopBlend: An efficient implementation of HtmlDiff in Java. In *Proc. of the World Conference on the WWW and Internet (Web-Net)*, 2000.

[8] A. Cicchetti, D. Di Ruscio, and A. Pierantonio. A Metamodel Independent Approach to Difference Representation. *Journal of Object Technology*, 6(9), 2007.

[9] E. Coyne and J. Davis. *Role Engineering for Enterprise Security Management*. Artech House, 2008.

[10] D. Ferraiolo, D. Kuhn, and R. Chandramouli. *Role-Based Access Control, Second Edition*. Artech House, 2007.

[11] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and Change-Impact Analysis of Access-Control Policies. In *Proc. of the 27th International Conference on Software Engineering (ICSE)*. ACM, 2005.

[12] M. Frank, J. M. Buhmann, and D. Basin. On the definition of role mining. In *Proc. of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2010.

[13] L. Fuchs and S. Meier. The Role Mining Process Model. In *Proc. of the 6th International Conference on Availability, Reliability and Security (ARES)*. IEEE Computer Security, 2011.

[14] J. Hu, Y. Zhang, and R. Li. Towards Automatic Update of Access Control Policy. In *Proc. of the 24th International Conference on Large Installation System Administration (LISA)*. USENIX Association, 2010.

[15] D. Kolovos. Establishing Correspondences between Models with the Epsilon Comparison Language. In *Model Driven Architecture - Foundations and Applications (ECMDA-FA), Lecture Notes in Computer Science (LNCS), Vol. 5562, Springer Verlag*, 2009.

[16] D. S. Kolovos, D. D. Ruscio, A. Pierantonio, and R. F. Paige. Different models for model matching: An analysis of approaches to support model differencing. In *ICSE Workshop on Comparison and Versioning of Software Models*. IEEE Computer Society, 2009.

[17] V. Kolovski, J. Hendler, and B. Parsia. Analyzing Web Access Control Policies. In *Proc. of the 16th International Conference on World Wide Web (WWW)*. ACM, 2007.

[18] M. Kuhlmann, D. Shohat, and G. Schimpf. Role Mining - Revealing Business Roles for Security Administration using Data Mining Technology. In *Proc. of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2003.

[19] S. Kunz, S. Evdokimov, B. Fabian, B. Stieger, and M. Strembeck. Role-Based Access Control for Information Federations in the Industrial Service Sector. In *Proc. of the 18th European Conference on Information Systems (ECIS)*, June 2010.

[20] D. Lin, P. Rao, E. Bertino, and J. Lobo. An Approach to Evaluate Policy Similarity. In *Proc. of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2007.

[21] J. Mendling, M. Strembeck, G. Stermsek, and G. Neumann. An Approach to Extract RBAC Models from BPEL4WS Processes. In *Proc. of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, June 2004.

[22] T. Mens. A State-of-the-Art Survey on Software Merging. *IEEE Transactions on Software Engineering*, 28, 2002.

[23] D. Ohst, M. Welle, and U. Kelter. Differences between versions of UML diagrams. In *Proc. of the 9th European Software Engineering and the 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2003.

[24] MOF 2.0 / XMI Mapping Specification. available at: http://www.omg.org/technology/documents/formal/xmi.htm, December 2007. Version 2.1.1, formal/2007-12-01, The Object Management Group.

[25] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10, December 2001.

[26] A. Schipper, H. Fuhrmann, and R. v. Hanxleden. Visual Comparison of Graphical Models. In *Proc. of the 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE Computer Society, 2009.

[27] M. Song and W. van der Aalst. Towards comprehensive support for organizational mining. *Decision Support Systems*, 46(1), 2008.

[28] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional, 2nd edition, 2008.

[29] M. Strembeck. A Role Engineering Tool for Role-Based Access Control. In *Proc. of the 3rd Symposium on Requirements Engineering for Information Security (SREIS)*, August 2005.

[30] M. Strembeck. Scenario-Driven Role Engineering. *IEEE Security & Privacy*, 8(1), January/February 2010.

[31] M. Strembeck and J. Mendling. Modeling process-related RBAC models with extended UML activity models. *Information and Software Technology*, 53(5), 2011.

[32] M. van den Brand, Z. Protić, and T. Verhoeff. Fine-Grained Metamodel-Assisted Model Comparison. In *Proc. of the 1st International Workshop on Model Comparison in Practice (IWMCP)*. ACM, 2010.

[33] S. Wenzel. Scalable Visualization of Model Differences. In *Proc. of the 2008 International Workshop on Comparison and Versioning of Software Models (CVSM)*. ACM, 2008.