Available online at www.sciencedirect.com

SciVerse ScienceDirect

www.compseconline.com/publications/prodinf.htm

Information Security Technical Report

Bridging 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

Keywords:
RBAC
Migration
Model comparison
Role engineering
Role mining

ABSTRACT

In the context of role-based access control (RBAC), 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. We conducted two comparative studies to identify which visualization technique is most suitable to make migration guides available to human users. Based on the results of these comparative studies, we implemented tool support for the derivation and visualization of migration guides. Our software tool is based on the Eclipse Modeling Framework (EMF). Moreover, this paper describes the experimental evaluation of our tool.

© 2013 Elsevier Ltd. All rights reserved.

1. 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., (Coyne and Davis, 2008; Ferraiolo et al., 2007; Sandhu et al., 1996; Strembeck, 2010)). They are equipped with a number of permissions that grant a role the rights to perform specific operations on specific resources (such as files or software applications). Human users or other active entities (subjects) are assigned to roles according to their competencies and responsibilities in the organization.

In particular, RBAC provides a number of advantages for the management of access permissions (see, e.g., (Ferraiolo

☆ This paper is an extended version of the work originally presented at the 7th International Conference on Availability, Reliability and Security, 2012 (Baumgrass and Strembeck, 2012).

* *Corresponding author.*
E-mail address: anne.baumgrass@wu.ac.at (A. Baumgrass).

1363-4127/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.istr.2013.03.003

et al., 2007)). For example, roles serve as an abstraction layer between subjects and permissions, and thoroughly engineered roles tend to change significantly slower than the assignment of subjects to these roles. Moreover, RBAC has matured over the last decades and is now supported by a variety of commercial as well as non-commercial software products. However, from a practical point of view one of the most important reasons for the huge success of RBAC is probably that the technical and organizational advantages of RBAC directly translate into significant economical benefits (see, e.g., (Gallaher et al., 2002; O'Connor and Loomis, 2010)). Yet, in order to realize such benefits, it is important to carefully define an RBAC model that is tailored to the needs of the corresponding organization.

1.1. *Specifying RBAC models*

Role mining approaches apply data mining techniques to derive RBAC models from the software systems of an organization (see, e.g., (Frank et al., 2010; Fuchs and Meier, 2011; Kuhlmann et al., 2003)). For example, role mining is applied to detect patterns in permission-to-subject assignments which are then used to derive roles (see, e.g., (Giblin et al., 2010; Schlegelmilch and Steffens, 2005; Vaidya et al., 2008, 2010; Zhang et al., 2007, 2008)). Such permission-to-subject assignments can be extracted from different sources such as access control lists, capability lists, or lightweight directory access protocol (LDAP) registries. Furthermore, organizational mining techniques can be used to group people into "functional units" based on their execution of similar tasks (see, e.g., (Rembert and Ellis, 2009; Song and van der Aalst, 2008; van der Aalst et al., 2005)). 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., (Baumgrass, 2011; Baumgrass et al., 2012)). In this way, mining techniques document the current state of a system and produce *current-state RBAC models*.

In turn, role engineering is applied to define a tailored RBAC model for a certain organization or information system (see, e.g., (Coyne and Davis, 2008; Ferraiolo et al., 2007; Strembeck, 2010)). 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., (Baumgrass et al., 2011; Mendling et al., 2004; Strembeck, 2005)). However, the migration of a current-state RBAC model to a customized target-state RBAC model is a complex task.

1.2. *Motivation*

In recent years, we see an increasing interest in process-aware information systems (PAIS, see, e.g., (Dumas et al., 2005)) that control the execution of business processes and support the collaboration of users who conduct the different tasks that are included in a business process. Together with this increasing interest in PAIS, a number of access control approaches emerged that explicitly consider the characteristics of task-based systems and address the specification, consistency, and enforcement of task-based access control policies and corresponding constraints (see, e.g., (Ahn and Sandhu, 2000; Irwin et al., 2008; Oh and Park, 2003; Strembeck and Mendling, 2011)). In this context, the assignment of a task to a role essentially assigns the permission to execute instances of this particular task in the context of a corresponding business process instance. In addition, entailment constraints, such as mutual exclusion or binding constraints, are an important means to control the execution of business processes and to enable a correct task allocation at runtime (see, e.g., (Bertino et al., 1999; Strembeck and Mendling, 2010; Tan et al., 2004; Wainer et al., 2003)).

Yet, despite well-elaborated access control approaches in the literature and despite 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. There are a number of possible explanations for this phenomenon, 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 an 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.

However, in recent years 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 municipalities (see, e.g., (Kunz et al., 2010; Strembeck, 2010; Strembeck and Mendling, 2011)), 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. 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 2, we give an overview of our approach. Subsequently, Section 3 discusses the different dimensions in the model comparison context. Next, Section 4 presents the phases that we consider for a comparison of RBAC models. In Section 5, we describe the *migration guide* which results from the comparison of a current-state and a corresponding target-state RBAC model. In Section 6, we present the different options for the visualization and comparison of RBAC models before Section

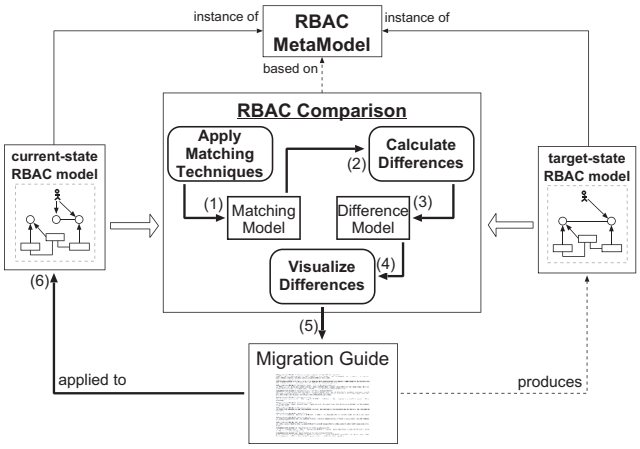


Fig. 1 – Approach overview: derivation of a migration guide.

7 provides two comparative studies to assess the different visualization techniques. Based on the results from these comparative studies, Section 8 describes the implementation of our migration guide software tool, and Section 9 describes the experimental evaluation of this tool.[1] Section 10 discusses related work and Section 11 concludes the paper.

2. Approach synopsis

We apply model comparison techniques (see, e.g., (Altmanninger et al., 2009; Kolovos et al., 2009; Mens, 2002; van den Brand et al., 2011)) 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. Fig. 1 shows the different steps in the comparison process and the artifacts that are produced as a result of each step.

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 form 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 produce the respective target-state RBAC model (see Fig. 1).

[1] Note that in order to compare a current-state RBAC model and a target-state RBAC model, we often have to apply a preprocessing step to ensure that both models are actually comparable. For the purposes of this paper, we focus on the actual comparison of two RBAC models and assume that (potential) preprocessing steps have already been applied. Specifically, we assume that both RBAC models conform to the same metamodel (for details see Sections 3 and 4).

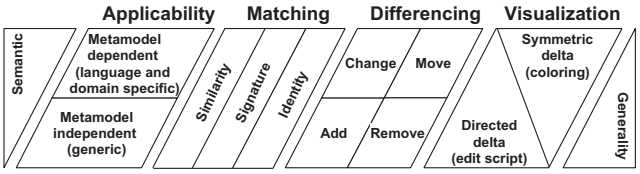


Fig. 2 – Different dimensions of model comparison.

In particular, we apply similarity-based matching techniques (see (Kolovos et al., 2009)) to identify elements with common attributes and relations that are included in the current-state as well as the target-state RBAC models. By applying a customized matching procedure (see Section 4) 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 the 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, or changed in order to produce the target-state RBAC model. In our approach, the difference model is then visualized as migration guide. In particular, 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. Thus, 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 5).

3. Identifying model differences

Model comparison involves the tasks to produce a matching model and then calculate, represent, and visualize model differences (see, e.g., (Kolovos et al., 2009; Mens, 2002)). Fig. 2 gives an overview of the different dimensions that need to be considered in the model comparison context. Below, we will now discuss these dimensions.

3.1. *The applicability dimension*

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 on arbitrary metamodels (see, e.g., (Kolovos, 2009; Kolovos et al., 2009)). In addition, such approaches are often adaptable and can be configured to compare models that are based on the same metamodel (see, e.g., (Brun and Pierantonio, 2008; van den Brand et al., 2010)). 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., (Chen et al., 2000; Cobéna et al., 2002; Ohst et al., 2003; Wang et al., 2003; Xing and Stroulia, 2005)).

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.

3.2. *The matching dimension*

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., (Kolovos, 2009; Kolovos et al., 2009; van den Brand et al., 2010)). In identity-based matching, elements with the same persistent identifier are matched. In signature-based matching, the unique 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. For the similarity-based matching, not all attributes (or relations) of an element may be equally important. Therefore, 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., (Kolovos et al., 2009)).

A *matching model*, such as the simple example shown in Fig. 3a, is the result of a matching algorithm. It consists of equal, similar, or unequal (unmatched) pairs of elements originating from the compared models.

3.3. *The differencing dimension*

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. For example, in Fig. 3a, the unmatched element "Role A" and its relation to subject "S" in the target-state RBAC model can be identified as additional (new) elements that need to be added to the current-state RBAC model. The differences calculated from this matching model are displayed in Fig. 3b.

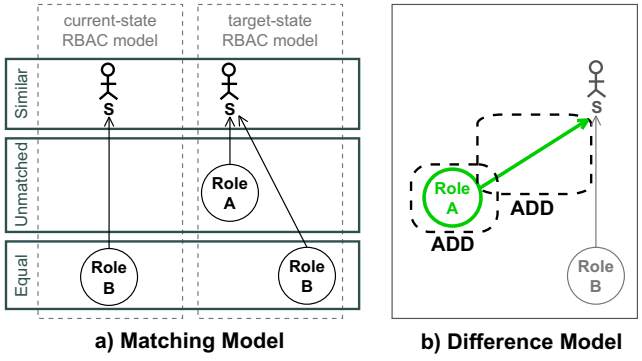


Fig. 3 – Example of a matching between two RBAC models.

3.4. *The visualization dimension*

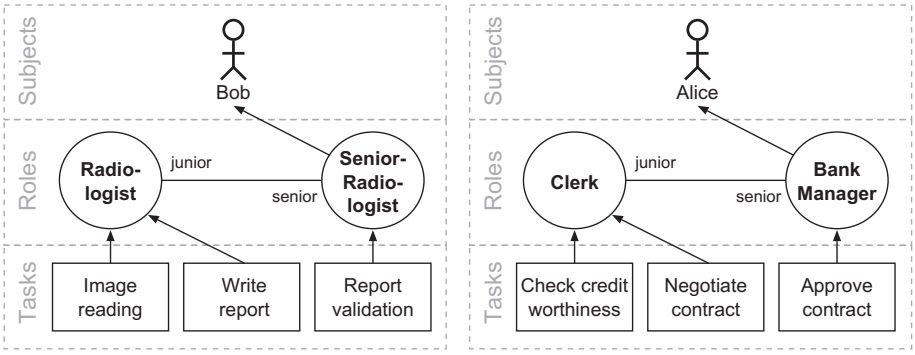
Visualizations of model differences can be subdivided in two types of approaches: "symmetric delta" and "directed delta" (see (Mens, 2002)). *Symmetric delta* displays the differences as a union of two compared models. For example, "coloring" techniques produce a diagram that highlights the equal parts 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).

4. Comparing RBAC models

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 Sections 4.2 and 4.3).

In general, comparing two models means to find their equivalences, similarities, and differences. Although the basic characteristics of two RBAC models, such as the number of roles, subjects, or permissions/tasks, can be used to measure a similarity value for these RBAC models, this type of similarity value is most often not suitable to reveal specific model differences. This means, while the number of elements in two models can be identical, the RBAC models may have completely different semantics. For example in Fig. 4, the structure of both models is identical, but the context in which both models are used is different. The RBAC model in Fig. 4a belongs to a radiological image reading process in a hospital, whereas the model in Fig. 4b belongs to a credit application process in a bank.

To derive a migration guide, this paper does not aim to determine the similarity of two models but to reveal the elements that differ between two RBAC models. Hence, we examine the properties and associations of each RBAC artifact (subjects, roles, permissions/tasks, and constraints). To support the migration of RBAC models we consider the following phases: definition (Section 4.1), matching (Section 4.2), calculating differences of RBAC models (Section 4.3), as well as the visualization of model differences between RBAC models (Section 4.4). The subsequent sections describe each of these phases in detail.



a) RBAC model for processes to read radiological images in a hospital

b) RBAC model for credit application processes in a bank

Fig. 4 – Two RBAC models with identical structure but different semantics.

4.1. *Definition of RBAC models*

A current-state RBAC model can be derived via mining approaches, while target-state RBAC models are defined by applying role engineering techniques (see Section 1.1). 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 (Strembeck and Mendling, 2011) and use 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 (Strembeck and Mendling, 2011)). 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 (Strembeck and Mendling, 2011) 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 (see, e.g., (Schefer et al., 2011; Strembeck and Mendling, 2010; Tan et al., 2004; Wainer et al., 2003; Warner and Atluri, 2006; Wolter and Schaad, 2007)). Mutually exclusive tasks result from the division of powerful rights or responsibilities to prevent fraud and abuse. In particular, a static mutual exclusion (SME) constraint defines that two mutual exclusive tasks must never be assigned to the same subject. In turn, a dynamic mutual exclusion (DME) constraint defines that two mutual exclusive tasks must never be performed by the same subject in the *same process instance*. In contrast to mutual exclusion constraints, binding constraints define that bound tasks must be executed by the same subject or role. A subject-binding (SB) constraint defines that two bound tasks must be performed by the same individual, while a role-binding (RB) constraint defines that bound tasks must be performed by members of the same role, but not necessarily by the same individual.

Fig. 5 shows two simple RBAC models that we use as a running example in the remainder of this paper. In Fig. 5a, the current-state RBAC model consists of three roles named “Employee”, “Bank Director”, and “Bank Manager”, the three tasks “Check credit worthiness”, “Negotiate credit”, and “Approve contract”, and two subjects named “Alice” and “Bob”. The role “Employee” is assigned to Alice and the role “Bank Manager” to Bob. Furthermore, the role “Bank Manager” is a junior-role of the role “Bank Director”. Moreover, the current-state RBAC model defines a subject-binding (SB) constraint between the tasks “Negotiate credit” and “Check credit worthiness” and a static mutual exclusion (SME) constraint is defined between the tasks “Negotiate credit” and “Approve contract”. In Fig. 5b, the target-state model contains a role “Clerk” and its senior-role “Bank Manager”, four tasks named “Check credit worthiness”, “Negotiate contract”, “Approve contract”, and “Define credit policy”, and the subjects “Alice” and “Claire” owning the “Bank Manager” role. The target-state RBAC model also includes an 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”.

4.2. *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 (`e.g., element_xy`) and the name (`e.g., Bank Manager Role`) 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 by 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 1 summarizes the RBAC artifacts and corresponding relations in the context perspective that are considered in a matching calculation.

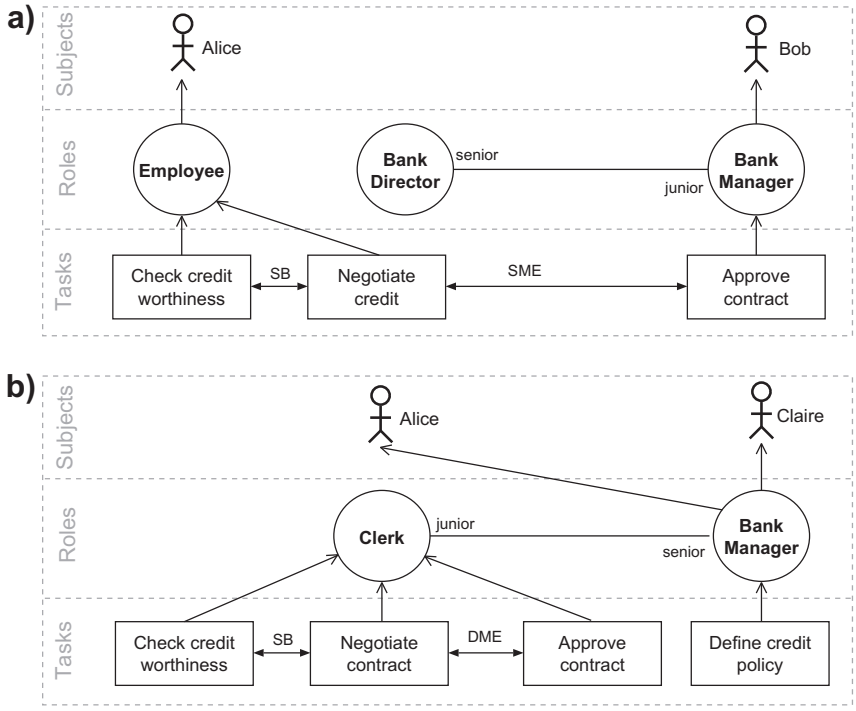


Fig. 5 – Example: (a) current-state and (b) target-state RBAC model.

For a matching of RBAC models, we apply a similarity-based matching to find comparable elements with similar attributes and relations in current-state and target-state RBAC models. We customized the similarity-based matching to include RBAC-specific characteristics. 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 Fig. 5) are considered to be similar if their associations to other artifacts are similar. In addition, we can apply a linguistic comparison to identify semantic similarities between artifact names. 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., (Miller, 1995; Rahm and Bernstein, 2001)).

Table 1 – Artifacts and relations in the context perspective.

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

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. Fig. 6 shows an excerpt of the matching model for our example in Fig. 5. Equal elements are not considered for further analysis, since they do not need to be adapted to produce the target-state RBAC model. Hence, the similar or unequal elements from the matching model are used to identify model differences in the subsequent phase.

4.3. *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. These differences include artifacts or artifact relations that need to be added, removed, or changed in the current-state RBAC model to produce the target-state RBAC model (see Fig. 7). The visualization of these differences is conducted in the next phase (see Section 4.4).

Our customization of the matching procedure to the specific characteristics of RBAC models (e.g. the context of an artifact is more relevant than its name, see Section 4.2), can identify the roles "Employee" and "Clerk" as similar elements (see Fig. 6). 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

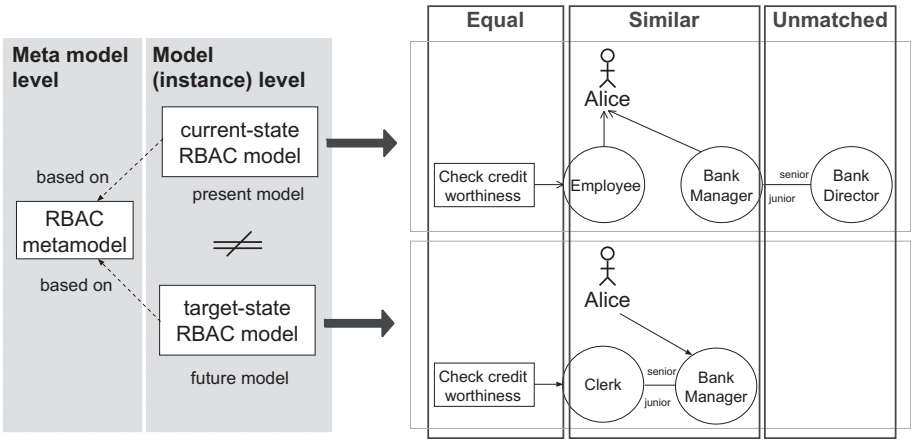


Fig. 6 – RBAC model matching example.

adding a new role named "Clerk" in the current-state RBAC model.

In general, we distinguish nine classes of differences that may result from the comparison of two RBAC models (see Fig. 7 – removals are colored red, additions are colored green, and changes are colored blue):

- *Removals* include removed assignment relations, artifacts, and constraints between tasks (see Fig. 7a–c). For example, Fig. 7a, 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 Fig. 7). 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 Fig. 7d–f). For example, Fig. 7f 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 Fig. 7g–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 5, we define the migration guide based on these nine classes of differences.

4.4. *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 a migration, we have to "visualize" the difference model in a human-readable (and/or machine-readable) form. Edit scripts (see Section 3) 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 (see Section 5). A symmetric delta can be used as another difference model visualization displaying the union of two compared RBAC models with equal and unequal parts highlighted (see Section 3).

5. Migration guides for RBAC models

In our approach, the *migration guide* is a visualization variant of RBAC differences (see Section 4.3) 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 (see also Fig. 8). In other words, the migration guide describes the modifications of 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 4.3, 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.

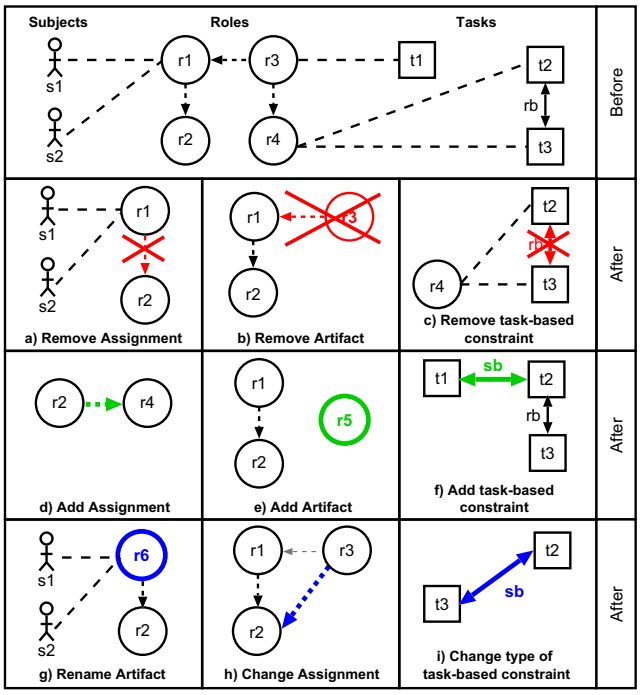


Fig. 7 – Differences for artifacts, assignments, and constraints.

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 RBAC 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 RBAC 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 RBAC model.

The *migration guide* recommends an ordered sequence of migration rules. In general, this ordered sequence results from the following heuristic: First and second, 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 MR 2). Third, RBAC artifacts that are not included in the target-state model are removed from the current-state model (see MR 3). Forth, artifact's 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). Fifth, 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. Sixth, the (mutual exclusion or binding) constraints in the current-state RBAC model are changed (see MR 6). Seventh, eight, and ninth, missing RBAC artifacts (see MR 7), assignment relations (see MR 8), and constraints (see MR 9) are added to the current-state RBAC model.

6. Options for visualizing RBAC model differences

In the following subsections, we discuss different options for tool support of comparison techniques for RBAC models. Because the comparison of arbitrary graphical models is extremely complex and does not provide an additional benefit

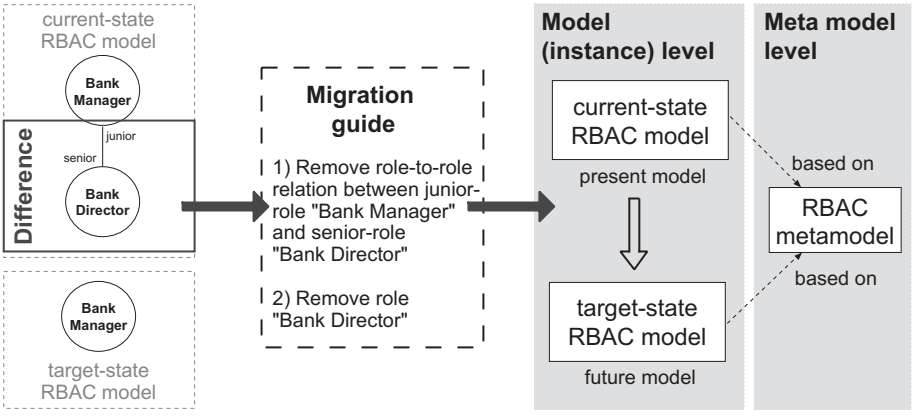


Fig. 8 – RBAC model conversion via migration guides.

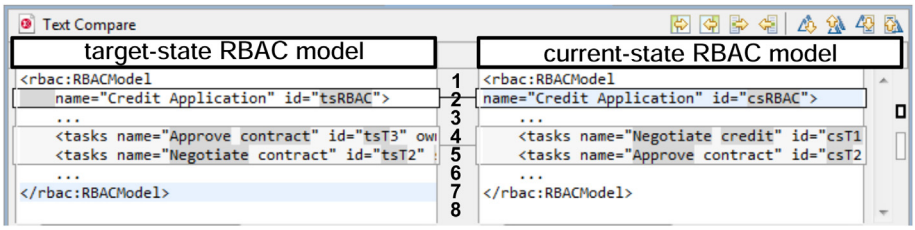


Fig. 9 – Line-based visualization of differences.

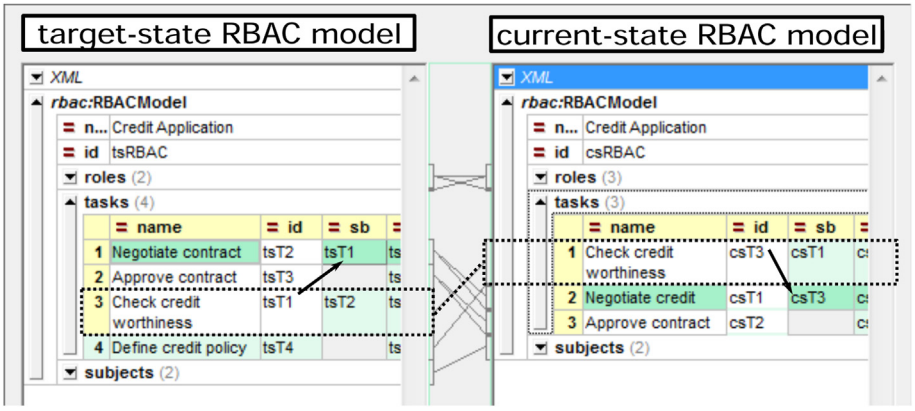


Fig. 10 – Tree-based visualization of differences.

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 symmetric delta or directed delta (see Section 3) to visualize RBAC model differences and to assist the migration of current-state to target-state RBAC models.

6.1. *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 parts of the text. However, the textual representation of the same model may include structural differences that do not change a model's semantics. Examples of structural differences are changed identifiers or a different element order. For example, if an identifier differs for two otherwise similar artifacts, the corresponding line representing this element is visualized as a difference.

Fig. 9 shows an excerpt for the line-based metamodel independent comparison performed with Eclipse[2] for the models depicted in Fig. 5. Apart from the highlighting of different identifiers in both models (e.g. in Line 2), this type of comparison also highlights structural changes in the RBAC model, such as the order of artifacts. Therefore, a line-based comparison may suggest changes that are based on structural differences in the textual model representation but that are not necessary from a semantic point of view. For instance, the comparison in Fig. 9 suggests to rename the task "Negotiate credit" to "Approve contract" because these two tasks are in the same line of the corresponding XML documents (Line 4). Even the introduction of line breaks or tab-stops is shown as difference between the models (Line 2).

6.2. *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 due to the fact that it is able to find similarities between differently ordered artifacts. In the example from Fig. 10, the "Check credit worthiness" task of the target-state RBAC model is the third task in the corresponding XML tree (see left-hand side of Fig. 10) which is compared with the first task of the current-state RBAC model (right-hand side of Fig. 10). However, relations between model artifacts are not included in a tree-based comparison. For example, in Fig. 10, the subject-binding (SB) constraint (indicated via the "sb" attribute 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.

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 Fig. 10 was conducted with Altova DiffDog.[3]

[2] http://eclipse.org/.

[3] http://www.altova.com/diffdog.

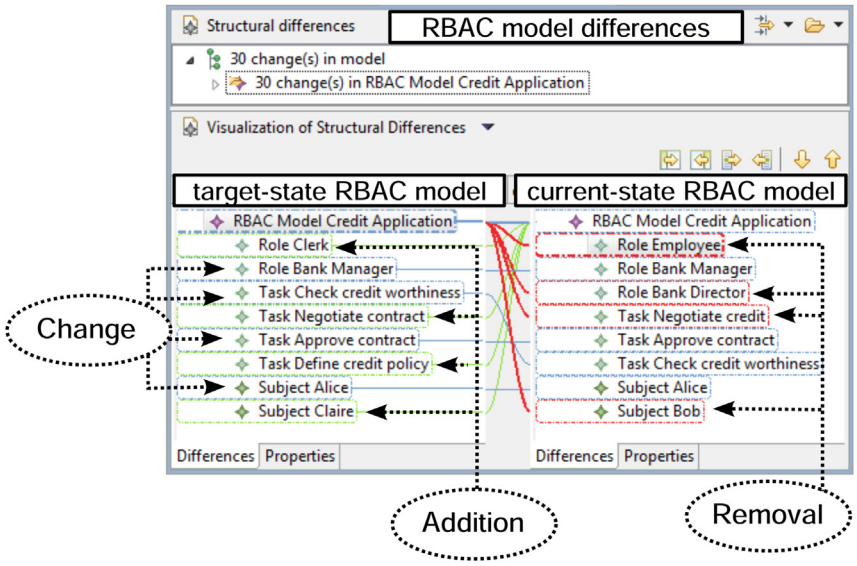


Fig. 11 – Graph-based visualization of differences.

6.3. *Graph-based visualization of RBAC differences*

In addition to the properties of tree-based model comparison, graph-based approaches can consider (indirect) cross-references between different elements in a graph (e.g. expressed via attributes). Software tools such as EMF Compare provide similarity-based matching techniques to support the comparison of any kind of metamodel (see, e.g., (Brun and Pierantonio, 2008)). Fig. 11 shows an example of a difference calculation between the current-state RBAC model and the target-state RBAC model in Fig. 5. This graph-based comparison is based on structural and semantic similarities rather than persistent identifiers (see Section 3).

The difference model depicted in Fig. 11 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 30 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"). However, 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.

The differences shown in the upper compartment of the window from Fig. 11 are expandable and reveal more details about the corresponding modifications. For example, Fig. 12 shows an expanded view of five changes for the task "Check credit worthiness". The first change is a change of the task identifier. In the EMF comparison, this change is specified as `Attribute id: EString in Task Check credit worthiness has changed from 'csT3' to 'tsT1'` (see Fig. 12). Second, a task-to-role assignment relation is added between the role "Clerk" and the "Check credit worthiness" task. Third, the task-to-role assignment relation to role "Employee" is removed. Fourth, a subject-binding has been added to the task "Negotiate contract". Fifth, the subject-binding to task "Negotiate credit" has been removed.

6.4. *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 3). Fig. 13 shows an example for a diagram-based comparison of the example in Fig. 5. The added elements and relations are colored green, the changed elements and relations are colored blue, the elements and relations that have to be removed are colored red, and the elements and relations that are unchanged are colored gray. Examples of

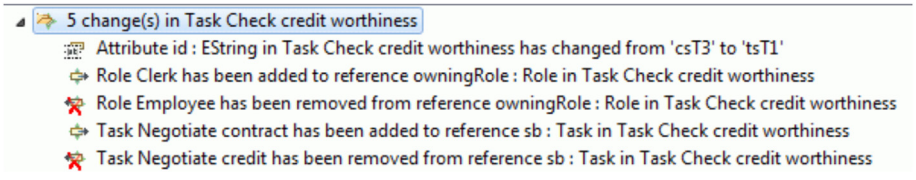


Fig. 12 – Changes of task "check credit worthiness".

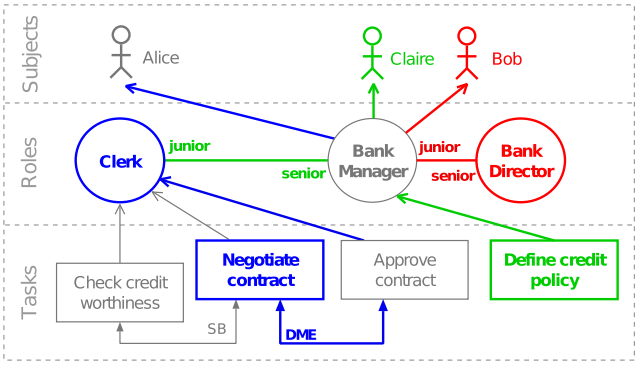


Fig. 13 – Diagram-based visualization of differences.

similar approaches are presented in (Ohst et al., 2003) and (Schipper and Fuhrmann, 2009).

For example, Fig. 13 shows that the role "Bank Director" must be removed from the current-state RBAC model (colored in red). For this reason the senior-role relation to the "Bank Manager" role must also be removed. Similarly, the subject "Bob" and the corresponding role-to-subject assignment relation must 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) and the task "Negotiate credit" task has to be renamed into "Negotiate contract". In addition, "Approve contract" is assigned to the "Clerk" role (instead of "Bank Manager"), the role "Bank Manager" assigned to "Alice" (instead of "Clerk"), and the type of the task-based constraint between the two tasks "Negotiate contract" and "Approve contract" is changed from an "SME" (static mutual exclusion) into a "DME" (dynamic mutual exclusion) constraint. Moreover, a new subject "Claire" and the task "Define credit policy" – both assigned to the "Bank Manager" role – have to be added to the current-state RBAC model (colored in green). To conform to the target-state RBAC model, we must also add a role-to-role assignment relation between "Clerk" and "Bank Manager".

Note that the original/previous names of changed artifacts (i.e. the names of the current-state model) are not visible in the diagram-based 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 properly rename them.

Table 2 – Migration guide as an edit script.

MR 2	Remove role-to-subject assignment relation between role *Bank Manager* and subject *Bob*.
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 3	Remove subject *Bob*.
MR 4	Rename role *Employee* to *Clerk*.
MR 4	Rename task *Negotiate credit* to *Negotiate contract*.
MR 5.1	Assign the role *Bank Manager* instead of role *Clerk* to *Alice*.
MR 5.2	Assign the task Approve contract to role *Clerk* instead of *Bank Manager*.
MR 6	Change the type of the mutual exclusion constraint between the tasks *Negotiate contract* and *Approve contract* from an *SME* into a *DME* constraint.
MR 7	Add task *Define credit policy*.
MR 7	Add subject *Claire*.
MR 8	Add a role-to-subject assignment relation between subject *Claire* and role *Bank Manager*.
MR 8	Add a role-to-role assignment relation between senior-role *Bank Manager* and junior-role *Clerk*.
MR 8	Add a task-to-role assignment relation between role *Bank Manager* and task *Define credit policy*.

6.5. *Visualization of RBAC differences as an edit script*

An edit script is an additional option to visualize differences between RBAC models. As discussed in Section 3, an *edit script* describes a sequence of operations needed to convert one model into another model. In other words, the operations included in a particular edit script describe the differences between the respective models. An edit script can be documented in a human-readable as well as in a machine-readable form. If we use edit scripts in the migration of RBAC models, we can also define a custom order for the corresponding edit operations (see also Section 5). In particular, we can associate each operation with its respective migration rule and thus derive an ordered sequence of edit steps to migrate a current-state RBAC model to a target-state RBAC model (see Section 5).

Table 2 shows an example of a corresponding edit script including the migration rules for the RBAC models in Fig. 5. Each row in the table defines exactly one migration rule – the left column references the corresponding generic migration rule (see Section 5) while the right column describes which artifacts need to be adapted to produce the corresponding target-state RBAC model. For example, the first row from Table 2 refers to *Migration Rule* 2 and describes the edit operation "Remove role-to-subject assignment relation between role *Bank Manager* and subject Bob".

7. Comparative studies of the visualization techniques

In this section, we describe two case studies that we conducted to assess the different visualization options. Five individuals participated in each of the case studies (i.e. ten individuals in total). The case studies were carefully designed and conducted in accordance with well-documented guidelines for case study research (see (Benbasat et al., 1987; Cavaye, 2008; Eisenhardt, 1989; Kitchenham et al., 1995; Runeson and Höst, 2009)). In particular, Section 7.1 describes our first case study which was designed to assess the overall suitability of different visualization options for a migration guide. In this case study, each participant applied each of the visualization techniques discussed in Section 6 to migrate five medium-sized current-state models into corresponding target-state models. Our second case study is presented in Section 7.2. It was designed to assess the scalability of the different visualization options. To this end, each of the

participants performed the migration of a large current-state model into a respective target-state model.

In the context of design science research, an assessment of an artifact or a technique is often done by comparison with competing artifacts or techniques (see, e.g., (Hevner et al., 2004)). Thus, for our comparative studies we first identified criteria that should be used to assess the suitability of the different techniques for RBAC model comparison and visualization (see Sections 3 and 6). As a result, we identified the six criteria shown in Table 3. In our studies, these criteria are mainly used to assess the suitability of different visualization options for a migration of RBAC models.

7.1. Case study on the suitability of different visualization options

The first case study included the migration of five medium-sized current-state RBAC models into corresponding target-state RBAC models. Five different individuals participated in the case study. Four of the participants own a Master's degree in Business Informatics, one participant owns a Ph.D. degree in Business Informatics. The case study was designed so that each participant applied each of the five visualization techniques (see Section 6), and each visualization technique was applied to each model. Moreover, to preclude learning effects, the case study was designed so that each participant applied each visualization technique exactly once to exactly one model.

Table 4 shows the artifacts that were included in each of the five models. In addition to the number of roles and tasks, the table includes the number of task-to-role assignment (TRA) and role-to-role assignment (RRA) relations.

After each of the five participants performed his/her migration tasks, we conducted a four-phase evaluation of the results. First, we measured the time that each of the participants needed to perform his/her five migration tasks. Table 5 shows the respective times (measured in minutes) and the average for each visualization technique. In this case study, the migrations conducted with edit scripts and with the diagram-based visualization both required an average of seven minutes, followed by the graph-based visualization with an average of eight minutes. In contrast, the tree-based and line-based visualizations required an average of twenty and twenty-one minutes respectively.

Second, we inspected the models to check if the result of the migration procedure (i.e. the migrated current-state RBAC model) conforms to the respective target-state RBAC model. Table 6 gives an overview for which models and visualization options the migrated model conforms to the target-state RBAC model. A "YES" indicates that the migrated model conforms to the corresponding target-state model, while a "NO" indicates that the migrated model does not conform to the respective target-state model. In particular, only edit scripts always resulted in a correct model and thereby in an accurate migration.

To better understand how the different visualization options affect the migration task, we measure the *accuracy* of each migrated model. The accuracy measure expresses what percentage of the model artifacts and relations in the migrated model conforms to the corresponding target-state model. The accuracy measure is calculated as follows:

$$accuracy = 1 - \frac{\text{number of deviations in the migrated model}}{\text{total number of artifacts and relations in the target} - \text{state model}}$$

Thus, we first count the number of deviations for each model and visualization. In particular, each modification (change, addition, removal) or an omitted modification is considered as deviation if it leads to a missing or an incorrect task, role, task-to-role assignment relation, or role-to-role assignment relation. The number of deviations is divided by the total number of artifacts and relations included in the target-state RBAC model. Table 7 shows the *accuracy* in percentage for each of the migrated models.

Third, we asked each participant to rank the visualization techniques with respect to the criteria from Table 3 except for the scalability criterion.[4] For each of the criteria from Table 3 (except for the scalability criterion) the participants of our first case study ranked the visualization options from first (1.) to fifth (5.) place respectively. The ranking is defined on an ordinal scale that does not include the size or the degree of differences but a relation between the visualization options (see (Siegel, 1957; Stevens, 1946)).

Fourth, we conducted semi-structured interviews (see, e.g., (Myers and Newman, 2007)) with each of the five participants in order to receive a detailed verbal feedback on the pros and cons of the different visualization techniques. The interviews included the following questions:

- How did you conduct the migration using the diagram-based visualization?
- How did you conduct the migration using the tree-based visualization?
- How did you conduct the migration using the line-based visualization?
- How did you conduct the migration using the graph-based visualization?
- How did you conduct the migration using the edit script?
- In which sequence did you use the visualizations and models for a migration?
- Do you think that the sequence in which you used the visualizations influenced your results?
- For which of the visualization techniques are you sure that your results are correct?
- What was the greatest challenge?

[4] Because all of the models from our first case study are similar in size, the scalability criterion was evaluated in a second case study that included the migration of a much more complex RBAC model (the results of the second case study are discussed below).

- Did you require additional information to conduct the migration?
- Do you have any suggestions for improvement?
- Which visualization would you use if you have to conduct another migration?

Below, we summarize the results of the first case study. In particular, we shortly discuss the results of the interviews and the assessment concerning the criteria from Table 3.

The **clarity** criterion is defined as a measure to evaluate if the differences (i.e. the migration steps) between RBAC models require extra knowledge and interpretation to read and understand them. Table 8 shows the ranking with respect to the clarity criterion. All the participants in our studies had specific knowledge on the underlying RBAC metamodel (see (Strembeck and Mendling, 2011)) which is required to understand differences between RBAC models. An edit script represents the migration steps in natural language. Hence, edit scripts do not require knowledge of a specific modeling language or notation. For this reason, the participants ranked edit scripts first for the clarity criterion. In a diagram-based visualization, the user has to differentiate between four coloring schemes – green for additions, red for removals, blue for changes and gray for unchanged elements (of course, other coloring schemes are possible). Therefore, the interpretation of these coloring differences require a (moderate) extra knowledge on how the different colors must be interpreted. A user of a graph-based visualization requires knowledge about the respective tool (i.e. the software tool that generates the graph) and its specific way of visualizing differences (see Fig. 12). The diagram-based and graph-based visualizations are ranked on a shared second place. The line-based and tree-based visualizations were perceived as the most cumbersome visualizations.

If a technique is able to visualize all information that is necessary to conduct a migration of RBAC models we define it as **complete**. Note that this criterion does not evaluate the precision or required knowledge to understand and use the visualization option for a migration. These aspects are captured via the clarity, conciseness, and practicality criteria respectively. Table 9 shows the ranking with respect to the completeness criterion. Except for the diagram-based visualization each visualization option includes all necessary information for a migration. This is because the security expert who conducts the migration must be familiar with the artifacts and relations of the current-state RBAC model, e.g. to identify artifacts that are renamed (see also Sections 3 and 6). Again, in average the edit scripts were ranked first, followed by the graph-based and diagram-based visualization options. Thus, although the diagram-based visualization does not include all migration-relevant information it was preferred over the line-based and tree-based options.

The **conciseness** criterion evaluates if a technique produces precise migration operations. Table 10 shows the ranking with respect to the conciseness criterion. At best a certain technique would provide an ordered list of migration operations that can be performed consecutively to produce the target-state RBAC model. Moreover, it is an advantage if each migration operation results in a consistent RBAC model.

Table 3 – Summary of evaluation criteria for visualization options.

Clarity	Extent to which the differences between RBAC models that were derived with a certain technique are understandable.
Completeness	Extent to which the differences between RBAC models that were derived with a certain technique describe all necessary information for a migration.
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.
Practicality	Extent to which the differences between RBAC models require specific knowledge to use them for the migration.
Scalability	Effort for using a specific technique to migrate RBAC models with increasing complexity.

For example, removing a role before removing corresponding role-to-role assignment relations can result in an inconsistent RBAC model. Furthermore, we denote a technique as being more precise if it identifies a single migration operation for a certain change in the model. This means, a technique is less precise if more than one migration operation is used to describes a single change (e.g. removing an artifact and adding a new artifact instead of renaming an existing artifact). For the conciseness criterion, the tree-based and the line-based visualizations are ranked last. Depending on the models and the respective software tool, the tree-based visualization may result in a comparatively low number of differences. However, these differences often include more than one change and therefore result in more than one migration operation. Moreover, the tree-based visualization sometimes subsumes differences that are unrelated from a semantic point of view. This property, again, results in an imprecise list of migration operations. The line-based visualization neither includes an ordered sequence of changes nor has it the lowest amount of differences and resulting migration operations. From the diagram-based visualization it is not obvious in which order the changes have to be applied to preserve a consistent RBAC model after each migration step. In the same way, the graph-based visualizations does not describe the order of changes for a migration. The graph-based visualization does, however, contain a lower amount of change operations. The edit script visualization was perceived as the most concise option (see Table 10).

The **expressiveness** criterion assesses to which extend a technique is able to consider the syntax and semantics of RBAC models. Table 11 shows the ranking with respect to the expressiveness criterion. For this criterion, line-based model comparison techniques do not consider the structural or semantic similarities at all. Tree-based approaches are more flexible because they can consider the order of different artifacts. However, they limited since relations between different model artifacts are ignored. Thus, tree-based comparison techniques do not consider all structural or semantic similarities that are required for the comparison of RBAC models. Therefore, line-based and tree-based comparison

Table 4 – Number elements for the models from our first case study.

	Roles	Tasks	TRA	RRA
Current-state RBAC model				
Model 1	4	8	11	0
Model 2	5	11	14	0
Model 3	1	5	5	0
Model 4	4	15	16	0
Model 5	4	15	16	0
Target-state RBAC model				
Model 1	4	8	11	0
Model 2	7	12	14	3
Model 3	2	5	5	1
Model 4	6	15	19	2
Model 5	6	15	17	2

Table 5 – Time for migration (in minutes).

	Line	Tree	Graph	Diagram	Edit script
Participant 1	25	15	10	5	10
Participant 2	25	17	5	5	1
Participant 3	9	28	20	5	8
Participant 4	20	19	2	8	2
Participant 5	25	20	5	10	7
Average	**21**	**20**	**8**	**7**	**7**

techniques are ranked in the last place for the expressiveness criterion.[5] In general, a language-specific graph-based approach is more appropriate than a line- or tree-based comparison because it builds a difference model considering (indirect) relations between model elements. An RBAC-specific edit script (e.g. resulting from a customized graph-based approach, see Section 8) and a customized diagram-based visualization enable to consider the language-specific syntax and semantics of RBAC models in a comparison. In the case study, the participants ranked edit scripts ahead of graph-based and diagram-based visualizations (see Table 11).

The **practicality** criterion is defined as the amount of knowledge that is required to use the visualized differences for a migration. Table 12 shows the ranking with respect to the practicality criterion. Edit scripts document an ordered sequence of steps that can be used to directly migrate a current-state to a target-state RBAC model. Thus, they do not require extra (technique-specific) knowledge to conduct a migration. Each participant of the case study perceived edit scripts as the best option. The diagram-based and graph-based visualization are ranked in a shared second place. For the diagram-based visualization, the user conducting the migration must be familiar with the modeling language that is used to visualize the RBAC models. In a similar way, the user also needs specific knowledge about the respective tool to interpret the results of a graph-based visualization. The line-based and tree-based visualizations are ranked in the shared last place. For the line-based visualization we require knowledge of the respective document format and structure to interpret the differences between RBAC models and to use them for a migration. The same type of knowledge is required for the tree-based visualization. In addition, we might need specific knowledge about the respective software tool and its way of visualizing the differences to use them for a migration.

[5] Although the tree-based visualization is able to consider the order of different artifacts, some participants ranked the line-based visualization ahead of the tree-based visualization. In the semi-structured interviews, the participants explained that they were more familiar with the textual XML representation of RBAC models and are therefore able to identify the syntax and semantics of RBAC in plain XML documents better than via the respective tree-based visualization.

Table 6 – Conformance of the migrated model with the target-state model.

	Line	Tree	Graph	Diagram	Edit script
Model 1	YES	YES	YES	NO	YES
Model 2	NO	NO	NO	YES	YES
Model 3	YES	NO	NO	YES	YES
Model 4	NO	NO	YES	YES	YES
Model 5	NO	NO	YES	NO	YES

7.2. *Case study on the scalability of different visualization options*

In order to assess the scalability criterion (see Table 3), our second case study included the migration of a large current-state RBAC model into a corresponding target-state model. The current-state RBAC model includes 25 roles, 97 tasks, 112 task-to-role assignment relations. The respective target-state RBAC model includes 19 roles, 99 tasks, 9 role-to-role assignment relations, and 112 task-to-role assignment relations. Again, five different individuals participated in this study. Each of the participants owns a Master's degree in Computer Science or Business Informatics. To preclude learning effects, the participants from the second study were different from the participants in the first study (i.e. none of the participants attended both case studies). The case study was designed so that each participant applied one of the five visualization techniques (see Section 6).

After the case study, we performed a two-phase evaluation of the results. First we conducted semi-structured interviews with each of the five participants. Because the case study included the migration of a large RBAC model, each migration was significantly more time consuming than the migration of the medium-size models from the first case study (see discussion below). Moreover, each of the participants in the second study applied exactly one of the visualization techniques. Therefore, the questions for the semi-structured interviews are slightly different from the questions from the first study:

- How did you conduct the migration?
- Are you convinced that your results are correct?
- What was the greatest challenge?
- Did you require additional information?
- Do you have any suggestions for improvement?
- Would you use the visualization again for another migration?

Table 7 – Accuracy of the migrated models.

	Line	Tree	Graph	Diagram	Edit script
Model 1	100%	100%	100%	82.61%	100%
Model 2	80.56%	52.78%	97.22%	100%	100%
Model 3	100%	92.31%	92.31%	100%	100%
Model 4	76.19%	92.86%	100%	100%	100%
Model 5	52.50%	82.50%	100%	97.50%	100%

Table 8 – Ranking with respect to the clarity criterion.

	Line	Tree	Graph	Diagram	Edit script
Participant 1	5	4	3	2	1
Participant 2	5	4	3	2	1
Participant 3	3	4	2	5	1
Participant 4	5	4	1	1	1
Participant 5	4	5	3	2	1
Average	**4.4**	**4.2**	**2.4**	**2.4**	**1**

Ranking: 1 (best) – 5 (weakest).

Table 9 – Ranking with respect to the completeness criterion.

	Line	Tree	Graph	Diagram	Edit script
Participant 1	5	4	3	2	1
Participant 2	5	4	2	3	1
Participant 3	1	2	3	5	3
Participant 4	4	4	1	1	1
Participant 5	1	5	2	2	2
Average	**3.2**	**3.8**	**2.2**	**2.6**	**1.6**

Ranking: 1 (best) – 5 (weakest).

Table 10 – Ranking with respect to the conciseness criterion.

	Line	Tree	Graph	Diagram	Edit script
Participant 1	5	4	2	3	1
Participant 2	5	4	2	3	1
Participant 3	3	3	2	5	1
Participant 4	4	4	1	1	1
Participant 5	1	5	2	2	2
Average	**3.6**	**4**	**1.8**	**2.8**	**1.2**

Ranking: 1 (best) – 5 (weakest).

During the case study, one of the authors was present to observe the experiment and offer guidance if necessary. Assistance was especially required to conduct the migration with the line-based, tree-based, and diagram-based visualizations. However, in spite of our assistance, the participants conducting the migration with the line-based and the tree-based visualization were not able to finish and aborted the migration after two hours. Table 13 shows the respective times and the resulting accuracy for the different visualization techniques.

Afterwards, in the semi-structured interview, these two participants said that on the one hand these visualizations contain more information than they needed (i.e. not only the changed artifacts and artifact relations) and on the other hand they would require different types of tool support to conduct a migration on the basis of line-based and tree-based visualizations. For example, one participant suggested to implement a script to sort and filter (parts of) the visualization information.

The **scalability** criterion is used to assess how a certain technique scales for the migration of RBAC models with an increasing complexity. The experiences in our studies show that the effort for using line-based and tree-based visualizations to migrate RBAC models was very high. Therefore, we can confirm the findings reported in the literature (see, e.g., (Altmanninger et al., 2009; Wenzel, 2008)) that line-based model comparison techniques are hardly human-readable for large and complex real-world models. In addition, we can report the same finding for tree-based visualizations. In general, a diagram-based comparison can be used to visualize all differences and respective modifications of the current-state RBAC model. However, considering the size and complexity of real-world models, all changes shown in a single diagram (including unchanged artifacts) make a diagram-based visualization complex and cluttered. Moreover, existing research indicates that the size and complexity of a model has a significant impact on model comprehensibility (see, e.g., (Bowen et al., 2009; Moody, 2009; Reijers and Mendling, 2011; Sweller, 1988)). The graph-based visualization and edit scripts are most suitable with respect to the scalability criterion because they do only depend on the number of differences between two models and are not affected by the number of artifacts an RBAC model includes. Moreover, in the second case study only the graph-based visualization and edit scripts resulted in a correct (i.e. 100% accuracy) target-state model. However, edit scripts are significantly more time efficient (see Table 13).

In summary, our comparative studies showed that edit scripts are the most suitable visualization option for migration guides. In particular, edit scripts can consider the language-specific syntax and semantics of RBAC models. In contrast to line-based, tree-based, graph-based, and diagram-based visualizations, an edit script documents a concise sequence of migration operations which is human-readable and does not require knowledge of a specific modeling language or document format for its understanding. For these reasons, we chose edit scripts as the base technique for the implementation of our software tool (see Section 8).

8. Implementation of the migration guide software tool

With respect to the discussions in Sections 6 and 7, edit scripts are a flexible and very expressive means for the definition of migration rules. Therefore, we chose to use edit scripts as base technique for the implementation of our migration guide.

In general, each difference calculation (line-based, tree-based, graph-based, or diagram-based) can be used to derive a migration guide. However, the graph-based visualization of differences is especially well-suited for our purposes because it does not consider the order of artifacts but in turn the associations among artifacts in a model. For this reason, we use

a graph-based difference calculation to derive a difference model and then use automated model transformations to generate a corresponding edit script from the difference model. In particular, the graph-based difference calculation available in the Eclipse Modeling Framework (EMF) (Steinberg et al., 2008) (via EMF Compare (Brun and Pierantonio, 2008)) fits our requirements to implement the migration guide. This is because EMF Compare is an open-source project[6] which provides a variety of components designed for extensibility. Fig. 14 gives an overview of the different steps that are supported by our tool implementation. Moreover, Fig. 14 also indicates which of the EMF Compare components we extended (black) and which were merely reused (gray).

In order to enable the loading and editing of RBAC models in Eclipse, we followed the process of plug-in development with EMF (see, e.g., (Griffin, 2002)). First, we built an Ecore metamodel describing the syntax and the semantics of our RBAC models. Ecore models can be serialized using the XML Metadata Interchange (XMI) standard (MOF 2.0/XMI Mapping Specification, 2007). For our purposes, the Ecore metamodel for RBAC is based on the generic metamodel for process-related RBAC models from (Strembeck and Mendling, 2011). Based on this metamodel we built an editor and a corresponding Eclipse plugin. In this way, we are able to load RBAC models into Eclipse, manipulate them in a customized editor, and use them for the derivation of the migration guide. Moreover, we added an extension to append our newly defined RBAC model file type to the EMF Compare framework.

Next, we customized the implementation of the matching and differencing algorithms in EMF Compare to allow for a tailored similarity-based matching of RBAC models (see also Fig. 14). For this purpose, we extended the generic matching and difference engine (see (Brun and Pierantonio, 2008)) and adapted the respective procedures to enable the processing of RBAC-specific syntax and semantics. Per default, EMF Compare compares the names of two artifacts (of the same type) to decide if these artifacts match. With our adaptations for EMF Compare, we are able to compare the string representations of two given RBAC artifacts and return a value between 0 and 1. For the migration guide, we customized the generic match engine so that it considers two artifacts of the same type (except subjects) as similar if the return value of the comparison procedure is greater then 0.7.[7] In this way, our customized comparison procedure favors the context of an artifact over the artifact's name to identify renamed artifacts (see Section 4.2). 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 (see Fig. 15). 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 6.3, our customized comparison procedure results in a more precise difference model. In the example, our difference calculation identifies 24 instead of 30 differences (see Fig. 15). Note that this reduction in the number of differences means that we have more precise descriptions of the differences. Hence, we typically have fewer change operations because we can, for example, simply rename an artifact instead of deleting it and creating a new artifact (see also Section 7).

For the RBAC-specific differencing (see Fig. 14), we extended the EMF Compare attribute checker to ignore artifact identifiers in a comparison. The identifier is important to reference an artifact in a model. However, it is irrelevant for the comparison of two RBAC models. Therefore, we specify that this attribute is ignored for computing the difference between two RBAC models (see also Section 5). In a similar way, we ignore upper and lower case letters in a comparison of artifact names.

In a final step, we customized the export functionality of EMF Compare. In particular, we use the RBAC difference model to derive the migration guide (see Section 2 and Fig. 16). All differences that can be represented in such a model are specified via the EMF Compare difference metamodel.[8] For example, we use the `UpdateAttribute` element to derive a *Migration Rule* 4 (see Section 5). A corresponding example of the XML representation for this element is shown in Listing 1.

```
1 <subDiffElements xsi:type=''diff:UpdateAttribute''>
2  <attribute href=''http:///org/eclipse/examples/rbac.ecore
       #//Base/name''/>
3  <leftElement href=''models/target-state.rbac#tsT2''/>
4  <rightElement href=''models/current-state.rbac#csT1''/>
5 </subDiffElements>
```

Listing 1 – Excerpt of an XML-based difference model.

Listing 1 shows a change of an artifact's attribute as an example for a model difference. In the XML-based difference model such a change is documented via the `diff:UpdateAttribute` (Line 1 of Listing 1). For the RBAC comparison this means that the value of an attribute has changed, i.e. a property of an artifact in the current-state RBAC model has to be adapted to produce the target-state RBAC model. The attribute that needs to be changed is shown in Line 2 (`name`). The direction of the change is defined via the corresponding `leftElement` (Line 3) and `rightElement` (Line 4). For our RBAC comparison, the `leftElement` identifies the artifact from the target-state RBAC model and the `rightElement` identifies the corresponding artifact in the current-state RBAC model that has to be changed. The attribute that is to be changed is referenced via its identifier (Lines 3 and 4). With respect to our example in Fig. 5, this change means that the task "Negotiate credit" in the current-state has to be renamed into "Negotiate contract" to conform to the target-state RBAC model.

In addition, we can derive new assignment relations and constraints (i.e. assignment relations and constraints that must be added to the current-state RBAC model) from the `ReferenceChangeLeftTarget`. In contrast, assignment relation and constraints that have to be removed are identified via the `ReferenceChangeRightTarget` element. Subject, roles,

[6] http://wiki.eclipse.org/EMF_Compare.

[7] Note that this value is not hard-coded and can be modified if necessary. However, a threshold of 0.7 has produced the best results in our experiments.

[8] The complete metamodel is available at http://help.eclipse.org/helios/topic/org.eclipse.emf.compare.doc/tutorials/Using_Compare_Services.html.

Table 11 – Ranking with respect to the expressiveness criterion.

	Line	Tree	Graph	Diagram	Edit script
Participant 1	5	4	2	3	1
Participant 2	5	4	3	1	2
Participant 3	3	4	1	5	1
Participant 4	4	5	2	1	2
Participant 5	2	3	4	1	4
Average	**3.8**	**4**	**2.4**	**2.2**	**2**

Ranking: 1 (best) – 5 (weakest).

and permissions/tasks that must be added to the current-state model or removed from the current-state model are derived from the `ModelElementChangeRightTarget` and `ModelElementChangeLeftTarget` elements respectively. In case a removed constraint (Migration Rule 1, see Section 5) is related to a corresponding added constraint (Migration Rule 9, see Section 5) we merge both operations to define a single change operation for the respective constraint (Migration Rule 6, see Section 5). In a similar way, we identify a changed assignment relation (Migration Rule 5, see Section 5) by checking the source and target nodes of removed and added assignment relations.

In order to visualize and store the migration guide we defined a customized export function. In particular, we implemented a respective extension (see, e.g., (des Rivières and Wiegand, 2004)) for the "export extension point" of EMF Compare (see Fig. 14). Listing 2 shows an excerpt of the corresponding configuration file. In particular, the configuration file refers to the respective extension point (`org.eclipse.emf.compare.ui.export`). Furthermore, it shows the action that is performed (`org.eclipse.emf.compare.examples.export.rbac.action.RBACExportAction`) for an export of RBAC files (i.e. files with the rbac file extension).

```
1 <extension point=''org.eclipse.emf.compare.ui.export''>
2   <action fileExtension=''rbac'' id=''RBACExportAction''
3     class=''org.eclipse.emf.compare.examples.export.rbac.
          action.RBACExportAction''>
4   </action>
5 </extension>
```

Listing 2 – Extending export functionality of EMF compare.

In summary, an eclipse application that includes our plugin and the corresponding extensions enables the comparison of RBAC models and supports the export of the resulting migration guide. Furthermore, we use the customized export format to visualize the migration guide as an "edit script" (see also Sections 3 and 6, as well as Fig. 14). Fig. 16 shows a screenshot of our Migration Guide plugin for EMF Compare. In particular, the figure highlights the three main steps in the comparison of two RBAC models. The first step is the comparison of two RBAC files. In the second step we export the migration guide in a machine-readable format. Finally, the third step visualizes the migration guide as an ordered sequence of migration rules. In essence, our customizations and the conversion of the difference model results in the human-readable version of the migration guide.

9. Experimental evaluation of the software tool

In Section 7, we discussed our comparative studies of the different visualization options. Based on the results from these studies, we chose edit scripts as the base technique for the implementation of the migration guide software tool. However, because "evaluation includes the integration of the artifact within the technical infrastructure of the business environment" (Hevner et al., 2004), we also conducted an experimental evaluation of our migration guide software tool.

A common result of the experiences we gained from our role engineering projects (see Section 1.2) is the demand for systematic and tool-supported procedures for the derivation of role engineering artifacts and for the migration of current-state to target-state RBAC models. Therefore, we first developed an approach to derive current-state RBAC artifacts from process execution histories (see (Baumgrass, 2011; Baumgrass et al., 2012)) and an approach to derive target-state RBAC artifacts from process and scenario models (see (Baumgrass et al., 2011)). In this paper, these approaches serve as a starting point and input for the generation of the migration guide.

In essence, we conducted a case study for the experimental evaluation of the migration guide software tool (see, e.g., (Eisenhardt, 1989; Runeson and Höst, 2009)). The case study is based on complex real-life business process that defines how researchers can receive their postdoctoral lecture qualification (called "habilitation" or "venia docendi") at WU Vienna – (Senat der Wirtschaftsuniversität Wien, 2012) provides a textual description of the respective processes (including procedural and legal details). For our case study, we first translated the textual into graphical BPMN models (OMG, 2010). The resulting BPMN process models include 10 different higher-level entities and 9 associated entities.[9] Furthermore, the process models include 69 tasks, 3 embedded subprocesses, 4 collapsed subprocesses performed, as well as 38 message flows between the different entities.

After defining the BPMN models, we applied the approach from (Baumgrass et al., 2011) to automatically derive target-state RBAC artifacts from these models. The resulting target-state RBAC model is composed of 19 roles, 99 tasks, 9 role-to-role assignment relations, and 112 task-to-role assignment relations. Furthermore, we generated process execution histories via the CPN Tools process simulation feature (see (de Medeiros and Günther, 2005; Jensen et al., 2007)) to derive current-state RBAC artifacts (as described in (Baumgrass, 2011; Baumgrass et al., 2012)). The respective process execution histories were used to derive a current-state RBAC model

[9] Higher-level entities: *Habilitation applicant*, *Senate*, *Rector's Council*, *Equal Opportunities Working Group*, *Habilitation Committee*, *Chair of involved Department*, *Chair of not involved Department*, *Research Associates of the WU*, *Full Professors of WU*, and *Experts*. Associated entities: *Member of Senate*, *Chair of Senate*, *Full Professors of Senate*, *Chair of Habilitation Committee*, *Senior of Habilitation Committee*, *Members of Habilitation Committee*, *Curia of Full Professors*, and *FPA*, *Member of RA and CAP*, *Full Professors and habilitated Associates of WU*. "FPA" is an abbreviation for "Full Professors Association, "RA" for "Research Associates", and "CAP" for "Committee for Academic Programs".

Table 12 – Ranking with respect to the practicality criterion.

	Line	Tree	Graph	Diagram	Edit script
Participant 1	5	4	3	2	1
Participant 2	5	4	3	2	1
Participant 3	3	4	2	4	1
Participant 4	4	4	2	3	1
Participant 5	4	5	3	2	1
Average	**4.2**	**4.2**	**2.6**	**2.6**	**1**

Ranking: 1 (best) – 5 (weakest).

Table 13 – Time and accuracy for the migration of a large RBAC model.

	Line	Tree	Graph	Diagram	Edit script
Time	>2 h	>2 h	1 h	1 h	40 min
Accuracy	aborted	aborted	100%	93.31%	100%

containing 25 roles, 97 tasks, 112 task-to-role assignment relations as well as 122 subjects with 399 role-to-subject assignment relations. The current-state and the target-state RBAC models are then fed into the migration guide tool (see Section 8). For the sake of reproducibility, we removed the subjects from the current-state RBAC model and replaced the automatically computed role names with sequential numbers.

Next, we used our tool to derive the corresponding migration guide (see Section 8). This migration guide includes 84 migration rules[10]. Fig. 17 shows a matrix for the derived task-to-role assignment relations of the current-state and the target-state RBAC models. In Fig. 17, directly assigned tasks are colored in black and inherited tasks are colored in gray. Furthermore, the figure shows the matching of roles from the current-state RBAC model with roles of the target-state RBAC model. Changed roles or tasks are framed either in green (for added artifacts) or red (for removed artifacts). For example, the green vertical frame on the right-hand side of Fig. 17 highlights the two roles *Research Associate of WU* and *Habilitation applicant* that are added in the target-state RBAC model. In particular, the addition of these roles results from merging roles with identical tasks that were included in the current-state RBAC model. Furthermore, the target-state RBAC model includes a role-hierarchy. For this reason, the context of the roles *Research Associate of WU* and *Habilitation applicant* automatically differs slightly from the corresponding roles in the current-state RBAC model.

Subsequently, we used the migration guide to modify the current-state RBAC model. The migration was conducted stepwise until all migration rules were applied. Because the removal of a role or a task from an RBAC model automatically includes the removal of all associations of the respective artifacts, the migration could be conducted on the basis of 64 (instead of 84) migration rules because 20 out of 22 task-to-role assignment changes were associated with the removal of a related role.

[10] In particular, the 84 migration rules include 22 removed task-to-role assignment relations (*Migration Rule 2* from Section 5), 8 removed roles and 1 removed task (*Migration Rule 3* from Section 5), 17 renamed roles (*Migration Rule 4* from Section 5), 2 added roles and 3 added tasks (*Migration Rule 7* from Section 5), 9 added role-hierarchies and 22 added task-to-role assignment relations (*Migration Rule 8* from Section 5).

10. Related work

Table 14 shows an overview of related work on RBAC model migration.[11] With respect to the different dimensions discussed above, we use a √ if a related approach provides similar or comparable support for a certain concept, and a ○ if a related approach provides at least partial support for a particular concept.

A number of approaches for policy analysis focus on policy verification, conflict detection, and similarity detection. Most of these approaches are either based on model checking (see, e.g., (Fisler et al., 2005)), the Boolean satisfiability problem (see, e.g., (Kolovski et al., 2007)), or graph transformation (see, e.g., (Koch et al., 2001)). In general, these approaches mainly focus on the policy similarity analysis but do not discuss the direct impact or usage for an RBAC model migration.

In (Fisler et al., 2005), 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. Kolovski et al. (Kolovski et al., 2007) provide a formalization of XACML for a verification and a change impact analysis, while Mazzoleni et al. (Mazzoleni et al., 2008) describe the policy similarity process and policy integration algorithms for XACML. From these three approaches, Margrave is most similar to our approach (see Table 14). Margrave enables the tool-supported matching and differencing of RBAC models defined via XACML and is able to visualize the differences. However, Margrave aims to identify if the combination of policies causes violations of access control properties defined in the policies rather than supporting a migration from one RBAC model to another RBAC model.

In (Fuchs, 2009), Fuchs and Müller present a catalog of use cases describing potential changes in the access control system of an organization, e.g. if an employee is assigned to a new position. The approach is implemented in the checkROLE tool and enables a role manager to identify discrepancies between a valid security definition and the present situation within an organization. Thus, checkROLE provides a tool-supported matching and differencing of access control models. In checkROLE, the access control policies are defined in busiROLE (Fuchs and Preis, 2008). Though similar, the checkROLE approach differs from our migration guide approach. In particular, the role managers have to interpret the changes for a migration of RBAC models. Although the differences are presented in a human-readable format which

[11] Further approaches which are in some aspects related but have a different focus are not included in the table. Nevertheless, they are also discussed in this section.

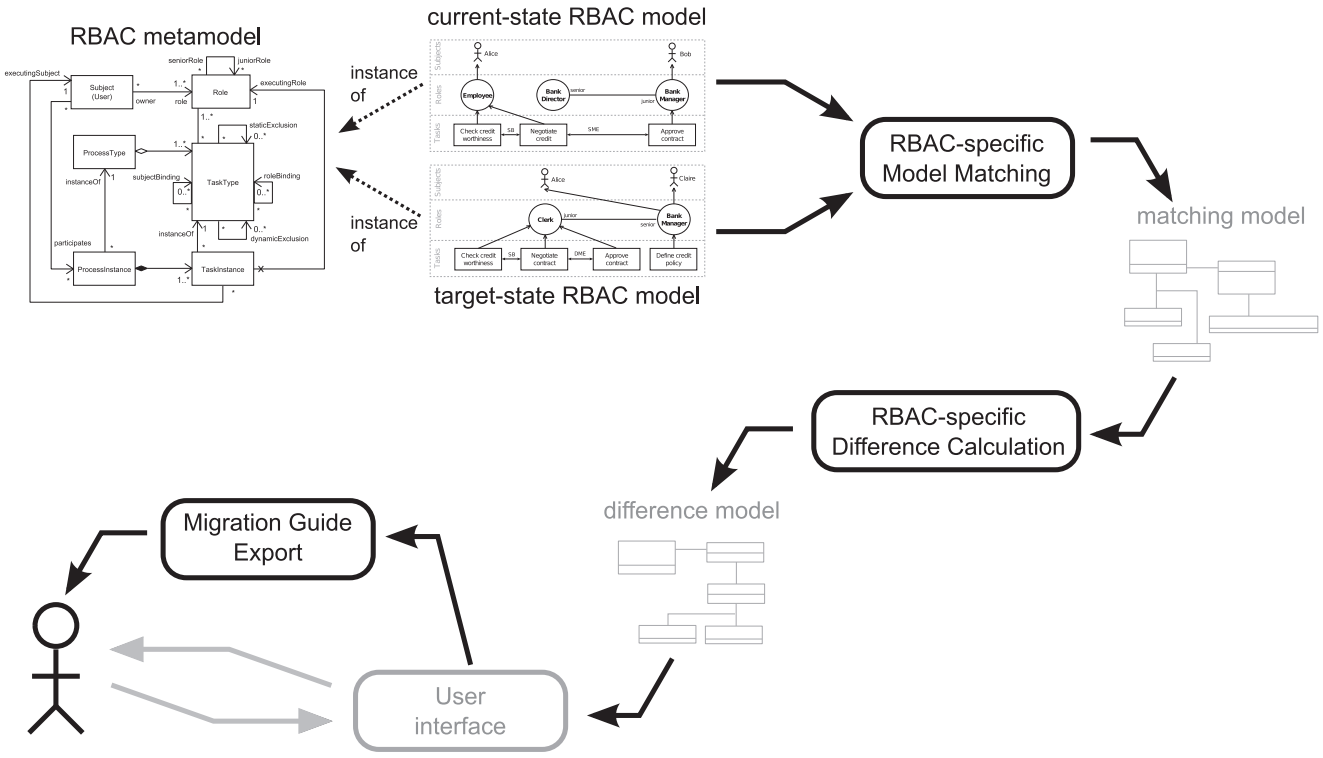


Fig. 14 – Tasks supported by our EMF compare extension (RBAC-specific loading, matching, differencing and migration).

is similar to our approach, checkROLE does not propose specific operations that can be used to migrate RBAC models.

The RoleUpdater (Hu et al., 2010a, 2010b) 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 representing specific properties of an RBAC model that are fed into the RoleUpdater. The RoleUpdater then checks if a security definition (i.e. a property of an RBAC model) is already supported. If it is not yet supported it gives an example how the definition can be added to the system. Different from our approach, the security expert must formalize the target-state RBAC model via RoleUpdater requests, check whether the requests are satisfied, and may then use the generated update operations to modify the current RBAC model. Thus, RoleUpdater is request-driven – it focuses on model checking and is able to present updates in the form of assign and revoke actions, whereas our approach uses model comparison techniques and visualizes all differences between two RBAC models in order to support the migration to a target-state RBAC model.

Evaluating the similarity of access control policies can be seen as a preliminary step for policy analysis and comparison. In (Lin et al., 2007), 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 uses the result of the similarity check as a starting point for policy merging. Moreover, Lin et al. (Lin et al., 2010) developed EXAM (Environment for XACML policy Analysis and Management) as an environment for a variety of policy analysis. The core component of EXAM is the policy similarity analyzer for XACML policies. For a comparison, policies are formalized as MTBDDs (Multi-Terminal Binary Decision Diagram) and combined to run policy analysis queries. Similar to Margrave, policy analysis queries in EXAM are used to check if certain properties hold in a given policy-set rather than providing specific operations for a migration of RBAC models (see Table 14). EXAM uses the comparison for merging access control policies and to decide which resources can be shared between different parties/users. In contrast, our approach applies model comparison techniques to reveal differences between different versions of an RBAC model.

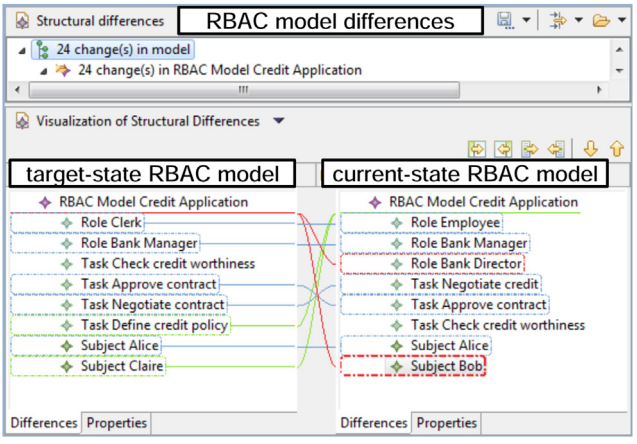


Fig. 15 – Customized EMF comparison between two RBAC models.

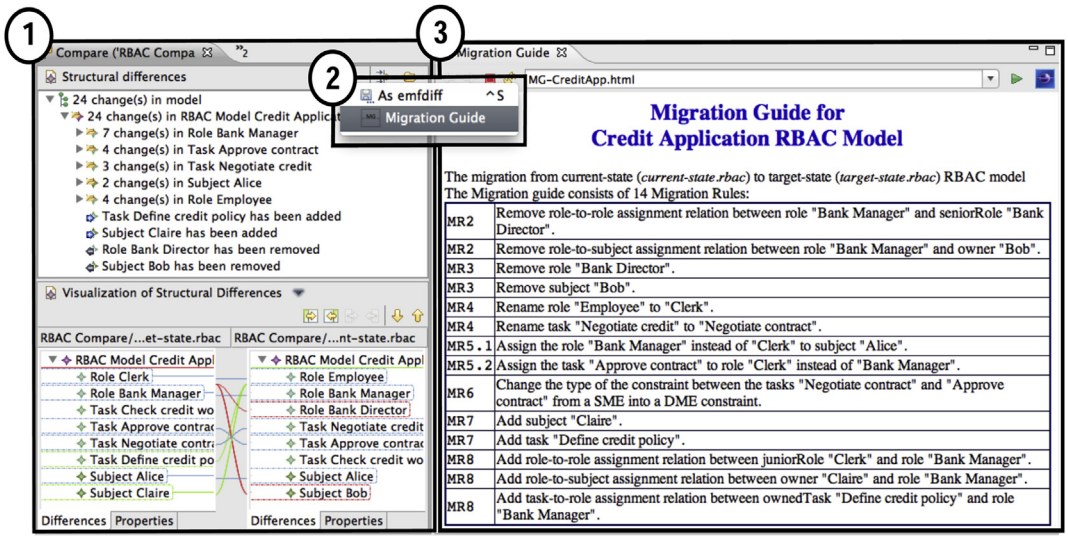


Fig. 16 – RBAC comparison in EMF compare including the resulting migration guide.

In (Backes et al., 2004), Backes et al. present an approach for the comparison of privacy policies. This comparison is conducted to check whether one policy refines another. However, it does not support the migration of a policy-set. Furthermore, Koch et al. (Koch et al., 2001) formalize modifications for the evolution, integration, and transition of access control policies. They use graph transformation techniques to represent policy changes and to enable the integration of discretionary access control policies or lattice-based access control policies. Thus, the approach is mainly used to specify changes and does not provide a dedicated migration support.

In addition, the huge body of work on model comparison is directly related to the approach presented in this paper. For example, EMF Compare (Brun and Pierantonio, 2008) provides an approach for a differences calculation. It allows to customize the matching and difference calculation. In our approach, we extended EMF Compare to provide a migration guide software tool for RBAC models (see Section 8).

Moreover, the Epsilon Comparison Language (ECL) (see, e.g., (Kolovos, 2009; Williams et al., 2011)) enables a language-specific rule-based matching for models (based on arbitrary metamodels). Thus, ECL could be used to specify the matching of RBAC models but it does not support the corresponding difference calculation. Yet, ECL can be integrated as matching module in the EMF Compare Framework. Therefore, the combination of ECL with EMF Compare provides partial support for the differencing and visualization of RBAC models. The graph-based visualization of EMF Compare can, in principle, be used to support the migration of RBAC models (see Section 6.3).

In (Cicchetti et al., 2007), Cicchetti et al. present a metamodel independent approach for the representation of model differences. Their approach is implemented via the Atlas Transformation Language (ATL). In particular, Cicchetti et al. provide a generic approach to express the modifications that have to be performed on the initial version of a given model (e.g., a current-state RBAC model) in order to produce another model (e.g., a target-state RBAC model). Thus, the difference model representation via ATL can be seen as an alternative to the EMF Compare difference metamodel that we used for the implementation of our migration guide (see Section 8). However, in comparison to our approach, Cicchetti et al. focus on the tool-supported representation of model differences and do not consider a customization of the matching or the difference calculation for a language-specific model comparison.

In the context of model comparison, 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, coloring schemes, or through a combination of both (see, e.g., (Kim et al., 2009; Ohst et al., 2003; Schipper and Fuhrmann, 2009; Wenzel, 2008)). For instance, Ohst et al. (Ohst et al., 2003) discuss the detection of differences between two versions of software design documents (e.g., UML class diagrams). However, they focus on the suitability and the improvement of coloring techniques to visualize structural changes. Kim and Notkin (Kim et al., 2009) present a similar technique that infers program differences as logic rules.

In (Schipper and Fuhrmann, 2009), Schipper et al. discuss different alternatives to display differences between graphical models. To evaluate different coloring techniques, they provide a prototypical implementation in KIELER.[12] KIELER is based on Eclipse and uses the default comparison algorithms of EMF Compare. Another diagram-based visualization is proposed by Wenzel (Wenzel, 2008). He proposes a scalable presentation of model differences based on polymetric views for a better comprehension of changes in large and complex models. Our comparative studies (see Section 7) showed that edit scripts are a suitable means to define migration rules. However, a customized visual comparison of RBAC model diagrams (based on

[12] http://www.informatik.uni-kiel.de/rtsys/kieler/.

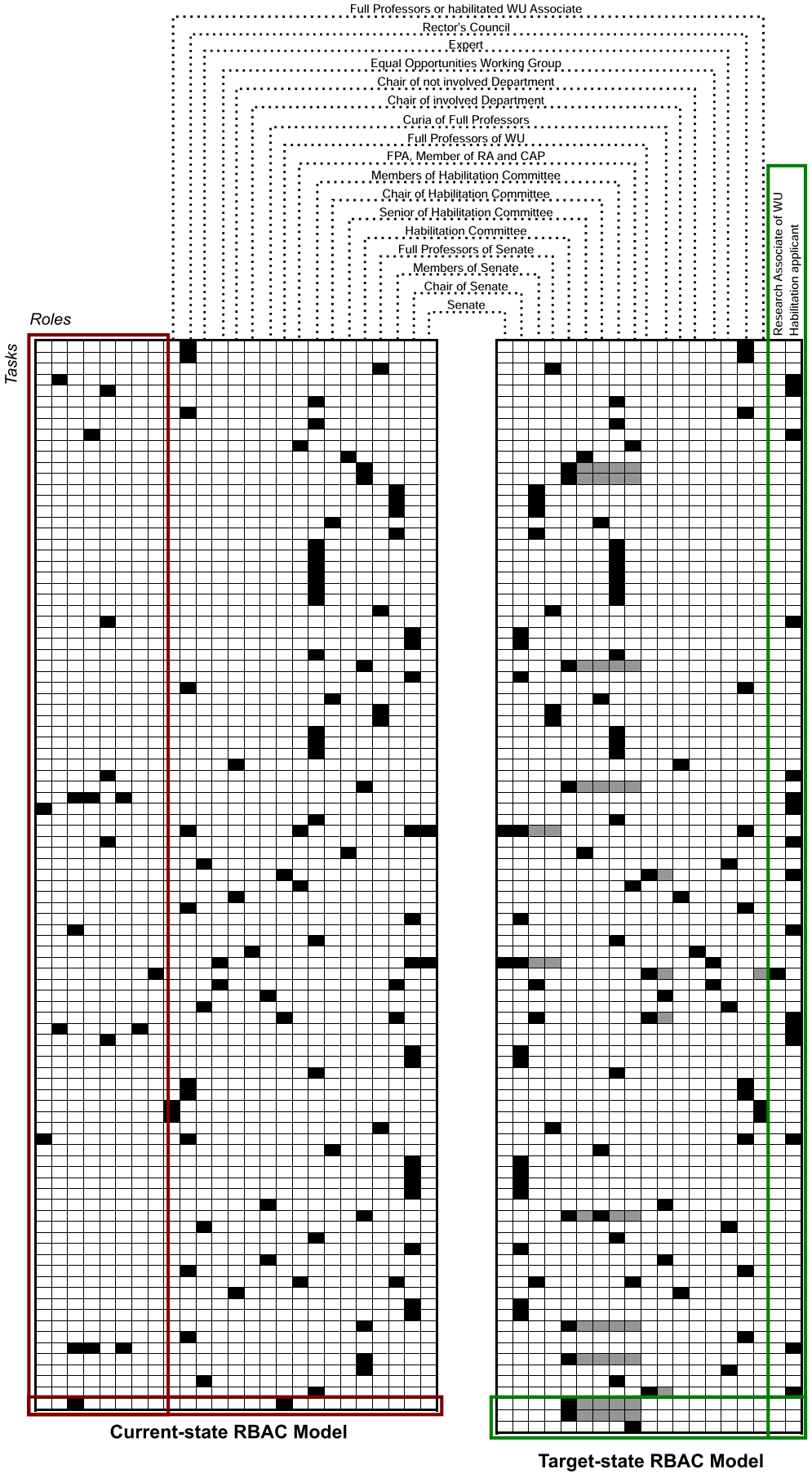


Fig. 17 – Matching of current-state and target-state RBAC models (directly assigned tasks are colored in black and inherited tasks are colored in gray).

Table 14 – RBAC model migration: related work.

	RBAC model matching	RBAC model differencing	Visualization	RBAC migration support	Tool support
Approaches for RBAC model comparison					
Verification and change impact analysis of AC policies (Fisler et al., 2005)	√	√	√		√
Autom. periodic role checks (Fuchs, 2009)	○	○	○	○	√
Role updating for assignments (Hu et al., 2010a, 2010b)			○	○	○
EXAM env. for analysis of AC policies (Lin et al., 2010)	√	√	√		√
Generic model comparison approaches					
Est. correspondences between models with ECL (Kolovos, 2009)	√	○	○	○	√
Metamodel independent approach to diff. representation (Cicchetti et al., 2007)	○	○	√	○	√
Visual comp. of graphical models (Schipper and Fuhrmann, 2009)	○	○	√	○	√
Scalable visualization of model diff. (Wenzel, 2008)	○	○	√	○	√
Migration Guide (our approach)	√	√	√	√	√

polymetric views) could be used to complement the respective (textual) edit scripts.

In summary, the approach presented in this paper complements many of the existing approaches. It enables the comparison of RBAC models and produces a migration guide that allows for a systematic migration between two RBAC models.

11. Conclusion

In this paper, we presented the *migration guide* as a systematic approach to migrate current-state RBAC models to target-state RBAC models. A migration guide recommends a sequence of edit operations for a stepwise migration of the respective current-state RBAC model to the target-state RBAC model. Note that, although it would be possible, we do *not* automate the migration/transformation from 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.

For the implementation of our software tool we conducted two comparative studies to evaluate different visualization options. In particular, we compared line-based, tree-based, graph-based, and diagram-based approaches as well as so called edit scripts. In the comparative studies, edit scripts where identified as the visualization technique that is most suitable to support a stepwise migration of RBAC models. For this reason, we chose edit scripts as the base technique for the migration guide software tool. Note, however, that our general approach for the derivation of migration guides does not depend on a particular visualization technique.

Our software tool includes customized model comparison techniques for RBAC models. Based on the respective comparison functions, we calculate the differences between RBAC models and automatically derive corresponding migration rules. To show the feasibility and application of our approach, we also conducted an experimental evaluation of our software tool.

With the migration guide 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.

In our future work, we will continue to evaluate the use of linguistic approaches to further customize the matching of RBAC artifacts. A linguistic comparison of artifact attributes (e.g. the artifact name) is applied to identify semantic similarity between elements. For instance, to identify similarities between a role with the name *Client* and a role with the name Customer.

Acknowledgments

The authors would like to thank Karin Kloibhofer for modeling the processes in CPN tools and Lukas Haan for deriving and preparing the current-state and the target-state RBAC models for the experimental evaluation of the migration guide software tool.

REFERENCES

Ahn GJ, Sandhu R. Role-based authorization constraints specification. ACM Transactions on Information and System Security (TISSEC) 2000;3(4).

Altmanninger K, Seidl M, Wimmer M. A survey on model versioning approaches. International Journal of Web Information Systems 2009;5(3).
Backes M, Karjoth G, Bagga W, Schunter M. Efficient comparison of enterprise privacy policies. In: Proc. of the 2004 ACM Symposium on applied computing (SAC); 2004.
Baumgrass A. 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.
Baumgrass A, Schefer-Wenzl S, Strembeck M. Deriving process-related RBAC models from process execution histories. In: IEEE International workshop on security aspects of process and services engineering (SAPSE). In: Proc. of the 35th Annual IEEE International computer software and applications conference (COMPSAC); 2012.
Baumgrass A, Strembeck M. An approach to bridge the gap between role mining and role engineering via migration guides. In: Proc. of the 7th International conference on availability, reliability and security (ARES), IEEE Computer Security; 2012.
Baumgrass A, Strembeck M, Rinderle-Ma S. Deriving role engineering artifacts from business processes and scenario models. In: Proc. of the 16th ACM Symposium on access control models and technologies (SACMAT); 2011.
Benbasat I, Goldstein D, Mead M. The case research strategy in studies of information systems. MIS Quarterly 1987;11(3).
Bertino E, Ferrari E, Atluri V. The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security (TISSEC) 1999;2(1).
Bowen PL, Farrell RAO, Rohde FH. An empirical investigation of end-user query development: the effects of improved model expressiveness vs. complexity. Information Systems Research 2009;20(4).
Brun C, Pierantonio A. Model differences in the eclipse modelling framework, UPGRADE. The European Journal for the Informatics Professional 2008;IX(2).
Cavaye A. Case study research: a multi-faceted research approach for IS. Information Systems Journal 2008;6(3).
Chen Y, Douglis F, Huang H, Vo K. TopBlend: an efficient implementation of HtmlDiff in Java. In: Proc. of the World conference on the WWW and Internet (Web-Net); 2000.
Cicchetti A, Di Ruscio D, Pierantonio A. A metamodel independent approach to difference representation. Journal of Object Technology 2007;6(9).
Cobéna G, Abiteboul S, Marian A. Detecting changes in XML documents. In: Proc. of the 18th International conference on data engineering (ICDE), IEEE Computer Society; 2002.
Coyne E, Davis J. Role engineering for enterprise security management. Artech House; 2008.
de Medeiros A, Günther CW. Process mining: using CPN tools to create test logs for mining algorithms. In: Proc. of the 6th Workshop and tutorial on practical use of coloured petri nets and the CPN tools; 2005.
des Riviéres J, Wiegand J. Eclipse: a platform for integrating development tools. IBM Systems Journal 2004;43(2).
Dumas M, van der Aalst W, ter Hofstede A. Process-aware information systems. John Wiley & Sons, Inc.; 2005.
Eisenhardt K. Building theories from case study research. The Academy of Management Review 1989;14(4).
Ferraiolo D, Kuhn D, Chandramouli R. Role-based access control. 2nd ed. Artech House; 2007.
Fisler K, Krishnamurthi S, Meyerovich LA, Tschantz MC. Verification and change-impact analysis of access-control policies. In: Proc. of the 27th International conference on software engineering (ICSE), ACM; 2005.
Frank M, Buhmann JM, Basin D. On the definition of role mining. In: Proc. of the 15th ACM Symposium on access control models and technologies (SACMAT); 2010.
Fuchs L, Meier S. The role mining process model. In: Proc. of the 6th International conference on availability, reliability and security (ARES), IEEE Computer Security; 2011.
Fuchs L, Müller. Automating periodic role-checks: a tool-based approach. In: Proc. Business Services: Konzepte, Technologien, Anwendungen. 9. Internationale Tagung Wirtschaftsinformatik; 2009.
Fuchs L, Preis A. BusiROLE: a model for integrating business roles into identity management. In: Proc. of the 5th International conference on trust, privacy, and security in digital business (TrustBus); 2008.
Gallaher M, O'Connor A, Kropp B. The economic impact of role-based access control. National Institute of Standards & Technology (NIST); March 2002. Planning Report 02-1.
Giblin C, Graf M, Karjoth G, Wespi A, Molloy I, Lobo J, et al. Towards an integrated approach to role engineering. In: Proc. of the 3rd ACM Workshop on assurable and usable security configuration (SafeConfig); 2010.
Griffin C. Using EMF. available at, http://www.eclipse.org/articles/Article-Using%20EMF/using-emf.html; December 2002. retrieved on 2012-10-05.
Hevner AR, March ST, Park J, Ram S. Design science in information systems research. MIS Quarterly 2004;28(1).
Hu J, Zhang Y, Li R. Towards automatic update of access control policy. In: Proc. of the 24th International conference on large installation system administration (LISA), USENIX Association; 2010.
Hu J, Zhang Y, Li R, Lu Z. Role updating for assignments. In: Proc. of the 15th ACM Symposium on access control models and technologies (SACMAT); 2010.
Irwin K, Yu T, Winsborough W. Enforcing security properties in task-based systems. In: Proc. of the 13th ACM Symposium on access control models and technologies (SACMAT); 2008.
Jensen K, Kristensen L, Wells L. Coloured petri nets and CPN tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer (STTT) 2007;9. (Special Section CPN 04/05).
Kim M, Notkin D. Discovering and representing systematic code changes. In: Proc. of the 31st ACM/IEEE International conference on software engineering, vol. 2 (ICSE); 2009.
Kitchenham B, Pickard L, Pfleeger S. Case studies for method and tool evaluation. IEEE Software 1995;12(4).
Koch M, Mancini LV, Parisi-Presicce F. On the specification and evolution of access control policies. In: Proc. of the 6th ACM Symposium on access control models and technologies (SACMAT); 2001.
Kolovos D. 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.
Kolovos DS, Ruscio DD, Pierantonio A, Paige RF. 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.
Kolovski V, Hendler J, Parsia B. Analyzing web access control policies. In: Proc. of the 16th International conference on world wide web (WWW), ACM; 2007.
Kuhlmann M, Shohat D, Schimpf G. 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.
Kunz S, Evdokimov S, Fabian B, Stieger B, Strembeck M. Role-based access control for information federations in the industrial service sector. In: Proc. of the 18th European conference on information systems (ECIS); 2010.

Lin D, Rao P, Bertino E, Li N, Lobo J. EXAM: a comprehensive environment for the analysis of access control policies. International Journal of Information Security 2010;9.
Lin D, Rao P, Bertino E, Lobo J. An approach to evaluate policy similarity. In: Proc. of the 12th ACM Symposium on access control models and technologies (SACMAT); 2007.
Mazzoleni P, Crispo B, Sivasubramanian S, Bertino E. XACML policy integration algorithms. ACM Transactions on Information and System Security (TISSEC) 2008;11(1).
Mendling J, Strembeck M, Stermsek G, Neumann G. 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); 2004.
Mens T. A state-of-the-art survey on software merging. IEEE Transactions on Software Engineering 2002;28(5).
Miller G. WordNet: a lexical database for English. Communications of the ACM 1995;38(11).
MOF 2.0/XMI Mapping specification, version 2.1.1. The Object Management Group (OMG). available at, http://www.omg.org/technology/documents/formal/xmi.htm; December 2007, formal/2007-12-01.
Moody DL. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering 2009;35(6).
Myers MD, Newman M. The qualitative interview in IS research: examining the craft. Information and Organization 2007;17(1).
O'Connor A, Loomis RJ. Economic analysis of role-based access control. National Institute of Standards and Technology (NIST); December 2010. Final Report.
Oh S, Park S. Task-role-based access control model. Information Systems 2003;28(6).
Ohst D, Welle M, Kelter U. 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); 2003.
OMG. Business process modeling notation (BPMN). The Object Management Group. available at, http://www.omg.org/spec/BPMN/2.0/Beta2/, version 2.0-Beta 2, dtc/2010-06-04; May 2010.
Rahm E, Bernstein PA. A survey of approaches to automatic schema matching. The VLDB Journal 2001;10.
Reijers HA, Mendling J. A study into the factors that influence the understandability of business process models. IEEE Transactions on Systems, Man, and Cybernetics – Part A 2011;41(3).
Rembert AJ, Ellis CS. An initial approach to mining multiple perspectives of a business process. In: Proc. of the 5th Richard Tapia celebration of diversity in computing conference (TAPIA); 2009.
Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 2009;14(2).
Sandhu R, Coyne E, Feinstein H, Youman C. Role-based access control models. IEEE Computer 1996;29(2).
Schefer S, Strembeck M, Mendling J, Baumgrass A. Detecting and resolving conflicts of mutual-exclusion and binding constraints in a business process context. In: Proc. of the 19th International conference on cooperative information systems (CoopIS), Lecture notes in computer science (LNCS), vol. 7044, Springer-Verlag; 2011.
Schipper A, Fuhrmann H, Hanxleden R.V. Visual comparison of graphical models. In: Proc. of the 14th IEEE International conference on engineering of complex computer systems (ICECCS), IEEE Computer Society; 2009.
Schlegelmilch J, Steffens U. Role mining with ORCA. In: Proc. of the 10th ACM Symposium on access control models and technologies (SACMAT); 2005.
Senat der Wirtschaftsuniversität Wien. Anhang 6 Habilitationsrichtlinien des Senats. Satzung der Wirtschaftsuniversität Wien (WU). available at, http://www.wu.ac.at/structure/lobby/professorsassociation/habilrichtlinien_9.5.2012.pdf; May 2012.
Siegel S. Nonparametric statistics. The American Statistician 1957;11(3).
Song M, van der Aalst W. Towards comprehensive support for organizational mining. Decision Support Systems 2008;46(1).
Steinberg D, Budinsky F, Paternostro M, Merks E. EMF: eclipse modeling framework. 2nd ed. Addison-Wesley Professional; 2008.
Stevens SS. On the theory of scales of measurement. Science 1946;103(2684).
Strembeck M. A role engineering tool for role-based access control. In: Proc. of the 3rd Symposium on requirements engineering for information security (SREIS); 2005.
Strembeck M. Scenario-driven role engineering. IEEE Security & Privacy January/February 2010;8(1).
Strembeck M, Mendling J. Generic algorithms for consistency checking of mutual-exclusion and binding constraints in a business process context. In: Proc. of the 18th International conference on cooperative information systems (CoopIS), Lecture notes in computer science (LNCS), vol. 6426, Springer-Verlag; 2010.
Strembeck M, Mendling J. Modeling process-related RBAC models with extended UML activity models. Information and Software Technology 2011;53(5).
Sweller J. Cognitive load during problem solving: effects on learning. Cognitive Science: A Multidisciplinary Journal 1988;12(2).
Tan K, Crampton J, Gunter CA. The consistency of task-based authorization constraints in workflow systems. In: Proc. of the 17th IEEE Workshop on computer security foundations (CSFW); 2004.
Vaidya J, Atluri V, Guo Q, Adam N. Migrating to optimal RBAC with minimal perturbation. In: Proc. of the 13th ACM Symposium on access control models and technologies (SACMAT); 2008.
Vaidya J, Atluri V, Warner J, Guo Q. Role engineering via prioritized subset enumeration. IEEE Transactions on Dependable and Secure Computing 2010;7(3).
van den Brand M, Hofkamp A, Verhoeff T, Protić Z. Assessing the quality of model-comparison tools: a method and a benchmark data set. In: Proc. of the 2nd International workshop on model comparison in practice (IWMCP); 2011.
van den Brand M, Protić Z, Verhoeff T. Fine-grained metamodel-assisted model comparison. In: Proc. of the 1st International workshop on model comparison in practice (IWMCP), ACM; 2010.
van der Aalst W, Reijers HA, Song M. Discovering social networks from event logs. Computer Supported Cooperative Work (CSCW) 2005;14(6). Springer-Verlag.
Wainer J, Barthelmess P, Kumar A. W-RBAC – a workflow security model incorporating controlled overriding of constraints. International Journal of Cooperative Information Systems (IJCIS) 2003;12(4).
Wang Y, DeWitt DJ, Cai J-Y. X-diff: an effective change detection algorithm for XML documents. In: Proc. of the 19th International conference on data engineering (ICDE), IEEE Computer Society; 2003.
Warner J, Atluri V. Inter-instance authorization constraints for secure workflow management. In: Proc. of the 11th ACM symposium on access control models and technologies (SACMAT); 2006.
Wenzel S. Scalable visualization of model differences. In: Proc. of the 2008 International workshop on comparison and versioning of software models (CVSM), ACM; 2008.

Williams JR, Kolovos DS, Polack FAC, Paige RF. Requirements for a model comparison language. In: Proc. of the 2nd International workshop on model comparison in practice (IWMCP), ACM; 2011.

Wolter C, Schaad A. Modeling of task-based authorization constraints in BPMN. In: Proc. of the 5th International conference on business process management (BPM), Lecture notes in computer science (LNCS), vol. 4714, Springer-Verlag; 2007.

Xing Z, Stroulia E. UMLDiff: an algorithm for object-oriented design differencing. In: Proc. of the 20th IEEE/ACM International conference on automated software engineering (ASE); 2005.

Zhang D, Ramamohanarao K, Ebringer T. Role engineering using graph optimisation. In: Proc. of the 12th ACM symposium on access control models and technologies (SACMAT); 2007.

Zhang D, Ramamohanarao K, Ebringer T, Yann T. Permission set mining: discovering practical and useful roles. In: Proc. of the 2008 annual computer security applications conference (ACSAC), IEEE Computer Society; 2008.