UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

Software Engineering

Taivo Teder

Extracting Role-Based Access Control Models from Business Process Event Logs

Master's thesis (30 ECTS)

Supervisor(s): Raimundas Matulevičius

Fabrizio M. Maggi

TARTU 2014

Extracting Role-Based Access Control Models from Business Process Event Logs

Abstract

Today, as business processes are getting more complex and the volumes of stored data about business process executions are increasing in size, collecting information for the analysis and for the improvement of the business process security[1], is becoming a complex task. Information systems that support business processes record business process executions into event logs which capture the behavior of system usage in terms of events. Business process event logs can be used for analysing and improving the business process, but also for analysing the information security. One of the main goals of security analysis is to check the compliance with existing security requirements. Also event logs can be the basis for business process mining, or shortly process mining. Utilizing bottom-up process mining on event logs, we can extract business process-related information for security analysis. Process mining is not just only for discovering business process models, but also other models, such as security models. For this purpose, we present a possible approach to extract RBAC models (semi-)automatically from event logs in XES format. The focus is also on determining the protected business assets, such as document or other artifact data that is exchanged and accessed during business process activities. In addition, we evaluate the applicability of this approach with conformance checking where we check the compliance of a real-life event log with respect to the LTL constraints translated from RBAC model. Eventually, the purpose of the extracted RBAC models is that they provide a basis for security analysis and they can be adapted by other applications in order to implement access control mechanism.

Keywords:

business process mining, Process-Aware Information System, event log, Role-Based Access Control model

1 In the context of this work, the security of the information accessed and exchanged during business process

Rollipõhise juurdepääse kontrolli mudeli tuletamine äriprotsessi sündmuste logide põhjal

Lühikokkuvõte

Keeruliste äriprotsesside ja järjest suurenevate andmemahtude juures on väljakutsuvaks ülesandeks analüüsida ja parandada ettevõtte äriprotsessi andmeturvalisust. Infosüsteemid, mis toetavad äriprotsessi mudeli (abstraktne esitus äriprotsessist) rakendamist, registreerivad äriprotsessi tegevusi sündmustena eraldi logisse. Salvestatud sündmuste logid on aluseks äriprotsessiga seotud andmete kaevamiseks. Need andmed on vajalikud äriprotsessi analüüsimiseks ja parendamiseks, kuid neid andmeid võib kasutada ka turvaanalüüsiks. Turvaanalüüsi üheks eesmärgiks on ka kontrollida, kas nende andmete hulgas turvalisusega seotud informatsioon on kooskõlas praeguste turvanõuetega. Lisaks, äriprotsessi logide peal saab rakendada äriprotsessikaeve (uurimisvaldkond, mis ühendab andmekaeve ja äriprotsesside modelleerimise) tehnikaid, et luua äriprotsessi mudeleid. Lisaks äriprotsessi mudelitele on võimalik tuletada ka teisi mudeleid, näiteks turvamudeleid, mida saab hiljem kasutada turvameetmete tagamiseks infosüsteemis. Käesoleva töö eesmärgiks on esitada üks võimalik meetod, kuidas luua rollipõhist ligipääsukontrolli esitatavaid turvamudeleid (*Role-Based Access Control models*) XES-formaadis sündmuste logidest, mis on salvestatud äriprotsessi toetava infosüsteemi poolt. Lisatähelepanu on suunatud kaitstavate infovarade väljaselgitamiseks sündmuste logide põhjal. Need infovarad on näiteks dokumendid, dokumendiväljad, või muud andmed, mida töödeldakse äriprotsessi tegevuste jooksul. Lisaks, me hindame antud meetodi rakendatavust reaalse äriprotsessi sündmuste logi peal. Ühe võimaliku meetodina me kontrollime sündmuste logi andmete ja seoste vastavust juurdepääsu õigustega olemasoleva rollipõhise juurdepääsu kontrolli turvamudelis. Lõppkokkuvõttes võib sündmuste logidest tuletatud rollipõhist ligipääsu kontrolli mudelit võtta aluseks turvaanalüüsiks või rakendada mõnes süsteemis juurdepääsumehhanismina.

Võtmesõnad:

äriprotsessikaeve, äriprotsessilogid, infosüsteemid, rollipõhise ligipääsu kontrolli mudel

Table of Contents

1 Introduction

Today, more and more organizations are becoming dependent on information technology to facilitate their business operations, thereby meeting their business objectives. Keeping track of the performance of business activities in an organization has become important for process analysis, which is essential for the organization to stay competitive and progress in time. Most of the business process improvement decisions are driven by the knowledge stored by information system. Therefore, organizations are concerned with the security (in terms of confidentiality, integrity, and availability) of the business assets, which include information, business process itself, and other confidential data.

A possible solution to protect business assets is to establish access control mechanism, such as role-based access control (RBAC)[1]. RBAC introduces the concept of role, which is the classification of users based on their qualifications, responsibilities, or authorities within an organization. Users acquire the access (the "need") to system resources (the "know") through being a member of a role. Resent years show that RBAC has been adopted for different systems as the primary access control mechanism [2], including business process management systems. This is mainly due to the fact that RBAC reduces the complexity and simplifies the administration of access control.

Business process management systems, or in general process-aware systems (PAIS)[3], are driven by explicit business process models. PAIS-s have built-in capabilities to record business process executions as a stream of events (hence the name event log), where a single event corresponds to a specified step in business process. Usually, business processes are complex and event logs contain large volumes of data. To overcome these obstacles, a bottom-up process mining is used for getting insights about business processes from event logs. Process mining can be interpreted as an intersection between data mining and business process modeling. Event logs are not just the source for discovering business process models, but also the source for security analysis and checking the compliance with existing security requirements [4]. Additionally, utilizing the potential of process mining, we can derive other models, such as organizational models [5] or RBAC models [6]. In this work, we are particularly interested in standard RBAC models[1]. These RBAC models serve as a basis for security analysis and can be adapted by different applications in order to implement access control mechanism.

The aim of this work is to present an approach how to extract RBAC models from event logs. The main focus is on extracting information about business assets (such as different documents and document fields) that needs to be protected. This work can be considered as a complementary solution to existing approaches, which are discussed in section 3.2.

This thesis work tries to address the following research questions:

1. What data could be extracted from a business process event log and how this data could be used for creating RBAC models?

Answering to the first question we specify what kind of information is even possible to extract from event logs and present an approach to get RBAC models from extracted business process data. In this thesis, the approach is divided into three steps. In the first step, we make use of process mining technique to discover business process data from an event log. In the second step, we create in-memory object of RBAC model via applying transofrmation rules on extracted business process data. In the last step, we export RBAC model in XML-based format which structure is presented using XML Schema Definition specifically composed for

this work.

2. How applicable is this approach on real-life business process event logs?

We evaluate the approach through conformance checking. First, we compare two RBAC models created from different event logs produced by the same source information system. Secondly, we test the compliance between RBAC models and an event log. For the second conformance checking, we utilize the approach presented in [7], which makes use of Linear Temporal Logic constraints as RBAC constraints.

This work is divided into eight chapters. In chapter 2, we give theoretical overview of technologies and terminology used in this work. In chapter 3, we give the review of the state of the art and contribution of this work given the state of the art. In chapter 4 and chapter 5 we describe the approach in step-wise manner and illustrate this approach with running example. In chapter 6, we compare two RBAC models created from different event created from the same information system. We check the RBAC model, specifically role-permission relationships, compliance with an event log. In chapter 7, we summarize research and discuss some future work.

2 Background and Technology

The purpose of this chapter is to give relevant background for this thesis topic. First, we present the main concepts of process mining. Then we discuss business process event logs. In particular, we introduce the basic elements and structure of typical event log, the primary format for event log (eXtensible Event Stream (XES)) and the requirements that event log needs to satisfy in order to be used in this thesis work. We introduce Role-Based Access Control (RBAC) and the core RBAC model, which is used as the reference RBAC model in the remainder of this thesis. We determine the terminology for discussing RBAC concepts in this work. Then, we discuss how RBAC concepts are expressed through elements of event log in XES format. Lastly, we present a short overview of eXchanged Markup Language (XML) technology.

2.1 Concepts of Process Mining

In this section, we discuss the concepts of process mining [8]. In most cases business processes are modelled in top-down manner to describe the desirable version of business process. Usually, these models are presented in some modeling language, such as BPMN[9], YAWL[10], EPCs[11], Petri Nets[12]. These explicit graphical models are used to give the better understanding of the business process. Furthermore, these models can be configured into information systems, such as Process-Aware Information System (PAIS), which is defined as "a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models"[3]. PAIS records process execution data in process execution logs (denoted as event logs). PAIS instantiates model multiple times, each instantiation stores new process instance (or case) into the log. The motivation of process mining is that business processes are getting more complex, increasing in size of activities, work-flow decisions, and participants. Therefore, these models cannot be created by hand and there has to be a mature way how to get different aspects of business process from business process execution logs.

Business process mining, or simply process mining, is a research discipline that develops and provides means, such as techniques, tools (e.g., ProM[13]), algorithms (e.g., process control-flow discovery α-algorithm developed in [14]), to discover, monitor, and improve real-life business processes. There exists three types of process mining: discovery, conformance, and enhancement of business processes.

The discovery technique takes on event log and extracts information about business process from event log without having any prior information. This discovery technique is not used only for creating process models, but to also for deriving other models, such as organizational models [5] or, as in the context of this work, RBAC models [6][15].

The second type of process mining is used to measure the alignment between idealized process model and the actual real-life process model as captured in an event log. In this work, we are not checking conformance between business process model and event log, but between RBAC model (specifically role-permission assignments) and event log. We utilize the approach presented in [7] to use Linear Temporal Logic constraints generated from authorization constrains to conform event log. In [16] it is shown how to conform business process model using LTL-based constraints. Linear Temporal Logic, as the name implies, uses temporal operators in addition to classical logical operators, such as always (□), eventually (◊), until (*U*), weak next (*W*), and next (○). A subset of these operations are used in this work when creating LTL formulas from RBAC constraints in section 6.2.

The third type of process mining is the enhancement of existing business process using the data about real business process as recorded in an event log. Oftentimes, the actual process deviates from the desired process. For this reason, the enhancement technique is to modify the existing process model in order to better reflect the reality.

Process mining assumes that PAIS sequentially records business process activities as a stream of events where each event is a part of particular process instance. PAIS can include additional information with event, such as the performer or originator of the event (i.e., a person or a system executing or initiating activity), the activity (i.e., a predefined step in the process), the timestamp of the event, or the data elements of the event (e.g., cost, quantity etc). Additionally, event can contain additional information about the lifecycle of an activity, namely *event type* (the standard event life-cycle model in [8] Fig. 4.3). Event types can indicate the start or the end of an activity, examples are *start*, *schedule*, *complete*, *suspend*. The typical structure of event log is illustrated in Figure 16 (an example of such event log is given in Table 11 as tabular data). Most of the graphical business process models depict the control-flow (the ordering of activities) of business process, expressed in Petri Nets [12] or other notation, however process mining can be applied to mine different perspectives of business process other than control-flow [17]. In [8], there are discussed four main perspectives that process mining is used for analysis of the business process:

- the control-flow perspective: the ordering of activities and corresponding paths;
- the organizational perspective: people and systems who are participating in the process and how they are related; one goal is to determine the structure of an organization by classifying people in terms of roles;
- the case perspective: describes the properties of a case (process instance), for example tasks and originators working on them, or values of corresponding data elements during one process instance;
- the time perspective: the timing and frequency of events.

In this work, we use process discovery technique to get information about activities, performers, and data elements. Specifically, we are interested in the organizational perspective, i.e., how people or systems are classified into roles, and the case perspective, i.e., who are the people or systems involved with different activities and which data elements they manipulate. We pay less attention to time perspective (when) and work-flow perspective (how).

2.2 Event Logs

In this chapter, we present the general structure of business process execution logs (denoted as event logs) and describe one specific format of event log that is being used in this work. Most of the concepts about event logs are adapted from [8].

2.2.1 Basic Structure and Elements

Business process executions are recorded in PAIS as a stream of events where each event is a part of one process instance (denoted as case). For process mining, a list of information pieces needs to be present in event log in order to mine business process (as listed also in [8]):

- business process cases;
- events where each event corresponds to exactly one case;

- events in case are ordered;
- events can have attributes, such as the activity name (i.e., a well-defined step in the process), the performer, the timestamp, and additional data elements.

The structure of event log (how these pieces of information are interrelated), is illustrated in Figure 16. Oftentimes, business process activity which represents a single unit of work is called *task*, or in other words atomic activity. In this thesis work, regardless of this distinction, we use the term activity. Additionally, event can contain additional information about the event type (i.e., lifecycle transition).

2.2.2 eXtensible Event Stream (XES)

The problem with event logs is that they are vendor-dependent, i.e., different vendors and information systems define their own format for event logs. For this reason, IEEE Task Force on Process Mining[18] suggests to use a standardized generally-acknowledged logging format, called XES (eXtensible Event Stream). XES is supported by different process mining tools, such as ProM[13], OpenXES[19], Nitro[20]. In here, we elaborate on some of the syntax presented by XES standard, the full description is given in [21].

An XML-based XES document (see Figure 18) contains *log* element consisting of any number of *trace* elements (i.e., cases). Each trace consists of *event* elements corresponding to exactly one trace. Each of the log, trace, and event element can contain any number of attributes with specific type. XES standard defines attribute types, such as *String*, *Date*, *Int*, *Float*, and *Boolean*. In order to provide semantics to commonly used attributes (e.g. concept, org, lifecycle, time etc.), XES standard introduces extensions. In XES format, some of these extensions are: *concept, organizational*, *time,* and *life-cycle*, with prefixes, such as "concept", "org", "time", and "lifecycle", respectively. The keys of all attributes defined by the extension will be prepended by the extension prefix and separated by the colon [21], e.g., the name of an activity is presented using "concept:name" attribute key and this attribute is defined by *concept* extension. XES distinguishes two lists of global attributes: one for the traces and for one events. In this work, we are particularly interested in attributes that are used within the scope of events.

A mandatory part of XES is also to provide event classifiers, which are defined through a set of attributes, i.e., attributes that give an identity to each event. For example, event with *Activity* is based on "concept:name" attribute, event with *Resource* classifier is based on "org:resource" attribute.

In this work, we use XES as the primary format of event logs. An example event log in XES format is presented in Table 12. Mappings from business process data to XES elements are presented in Table 1, where l*og* element represents the entire business process. A *trace* element represents a particular process case, where element *log* contains a number of traces. Business process activity is defined through attribute with key "concept:name", the performer of an activity is defined through attribute with key "org:resource", the execution time of an activity is defined through attribute with key "time:timestamp", event type is defined through attribute with key "lifecycle:transition", and additional information is defined through attributes of types *String*, *Date*, *Int*, *Float*, or *Boolean*. All of these elements with attributes are nested in a single element *event*. A set of events is nested in one specific *trace* element.

Table 1: Representation of event log elements in XES (adapted from [21])

Process	`log` (as the top level element)
Case	`trace`
Event	`event`
Activity	`string` element attribute with key `concept:name` on the `event` level
Resource (i.e., person or system)	`string` element attribute with key `org:resource` on the `event` level
Time	`date` element attribute with key `time:timestamp` on the `event` level
Event type	`string` element attribute with key `lifecycle:transition` on the `event` level
Additional attributes	XES format allows to present attributes in five different types: `String`, `Date`, `Int`, `Float`, `Boolean`, depending on the value data type it represents.

2.2.3 Requirements for Event Log

The applicability of the approach presented in this work depends on the completeness of an event log in terms of business process elements. For this reason, events in event log in format of XES needs to meet at least minimum set of requirements:

- *Activity*: event refers to activity using attribute with key “concept:name”. Additionally, an entry in event log may include activity life-cycle information (also referred to as event type), such as a start or a completion of an activity via attribute with key “lifecycle:transition”.
- *User*: event refers to user using attribute with key “org:resource”.
- *Role*: event refers to role using attribute with key “org:role” alongside with user information. If this requirement is not met and only user informaton is provided, then additional effort is spent on classifying users into roles. Most likely, due to the large amount of users, classification is done automatically using role mining techniques (a selection of them is discussed under chapter 3). Later on, domain knowledge is required to interpret these classifications and to make necessary corrections, because these classifications may not accurately reflect the roles in the real business process. Moreover, these classifications can be different for different data sets originating from the same source information system. For convenience, we assume that the role information is existent in event log. This implies that roles are predetermined and some primitive role management is implemented into PAIS beforehand. There is also an option to define roles and assign them for each and every user manually.
- *Data elements*: event includes domain-specific data attributes, which are represented in the log using XES data element type definitions (described in section 2.2.2). Data attributes and their values can be considered as valuable business assets, such as

resource identifier, invoice number and so on. Thus, data attributes needs to be protected from unauthorized access within and also from outside of an organization.

If this information is available, we can discover business process element and relationships from an event log as basis to get RBAC model.

2.3 Role-Based Access Control

In information systems security, Role-Based Access Control (RBAC)[1] is a security model where the access to system resources is regulated through permissions assigned to roles. A motivation to use RBAC model is that system can have a lot of users. For this reason, instead of making access control decisions on the level of individual users, access control decisions are determined on the level of roles that individual users have as part of an organization. The concept of role generally implies the qualifications, responsibilities, or authorities within an organization. Roles tend to be more stable and not change as frequently, which simplifies the management of permissions.

For describing the basic concepts of RBAC, we refer to the reference RBAC model[1] proposed as NIST[2] standard. In core RBAC, a user can have one to many roles, permissions are given to the appropriate roles, and a user requests a permission by being member of a role. Users and roles have many-to-many relationship, i.e., a user can have many roles and a role can have many users. Similarly, there is many-to-many relationship between roles and permissions, i.e., a role can have many permissions and a permission can be associated with many roles. In current work, the core RBAC model (or in other words, a flat RBAC model) is used, therefore other extensions, such as hierarchical RBAC, constrained RBAC, are discarded.

The core RBAC model defines a minimum set of RBAC elements and relationships to define a role-based access control system (as depicted on Figure 1). There are five main RBAC elements: *users*, *roles*, *objects*, *operations*, and *permissions*. A *user* (also denoted as a subject) is defined as a person or a system. A *role* is defined as a job or a function within an organization. Role can refer to the authority and responsibilities that are delegated to the user assigned to the role. A *permission* is an approval to perform an operation on one or more protected objects. In most cases, "permissions are always positive and confer the ability to the holder of the permission to perform some action(s) in the system"[22]. An *operation* is an executable sequence of instructions which can be invoked to perform some task or job function for the user. The nature of the permission operations depend on the implementation details and system type, but in general, operations include *create*, *read*, *update*, and *delete* (in terms of CRUD implementation). An *object* (also denoted as a *resource*) is a document or a piece of information on which operations are performed by users, therefore it needs to be protected to prevent unauthorized access. The RBAC model embodies three relationships (as can be seen on Figure 1): *user-role* relationship, *role-permission* relationship, and *operation-object* (as permission) relationship.

2 National Institute of Standards and Technology (NIST): http://www.nist.gov/

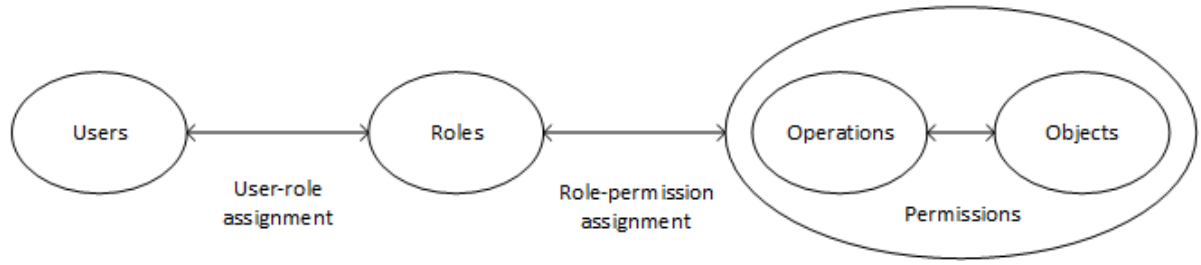

Figure 1: Role-Based Access Control model (adapted from [1])

2.4 Terms used for describing RBAC concepts

In this section, we present the terminology used for discussing RBAC concepts during this work. The RBAC model is semantically the same as presented in section 2.3, however we have introduced some small changes to it, as illustrated in RBAC model as UML[23] diagram (see Figure 2).

In business process context, users (denoted also as *subjects* in the following parts of the work) are referred to as (human) resources [8] or originators, i.e., a person or a system performing some activity. Although, in the context of this work, *resource* as material or a piece of information is used for denoting documents or other artifacts accessed during business process execution (in core RBAC definition referred to as *objects*). These documents may contain fields or data attributes (in our case *resource attributes*) which are manipulated during some business activity. In RBAC context, business activity is an *operation* on *resource*.

In most cases, the event log does not include the information to identify resources in the form of documents or artifacts, only the different data attributes as key-value pairs as a part of some document or artifact. Therefore, we assume that semantically the composition of resource attributes is a resource (as also depicted in Figure 2). Hence, protecting each resource attribute is protecting implicitly the whole resource.

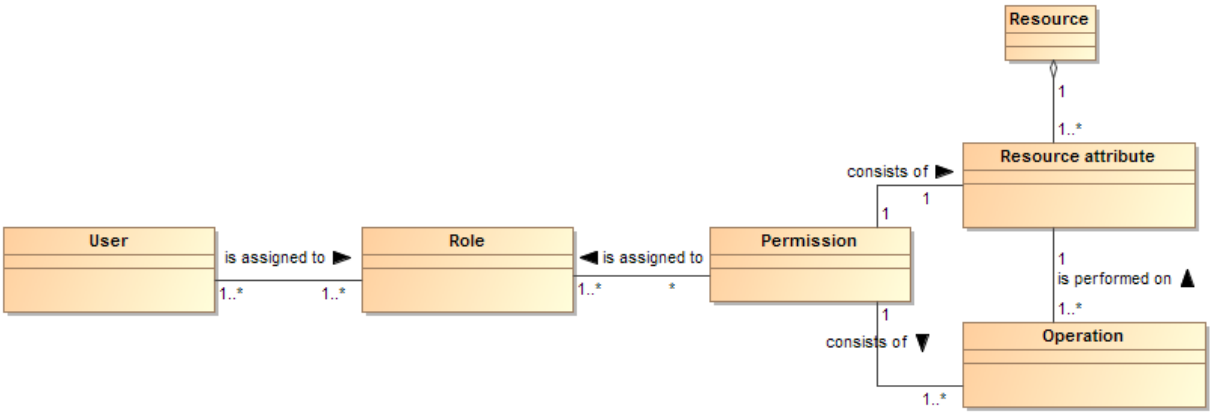

Figure 2: RBAC model as UML class diagram

2.5 RBAC Concepts Expressed through XES Elements

2.5.1 RBAC Operation

Every process instance contains a sequence of events. In the scope of a single event, business

process activity is recorded using *string* element with attribute value “concept:name”. This activity is considered as an RBAC operation on protected resource.

Event entry can contain also event type to specify business process activity, such as start or completion of an activity, which is expressed using *string* element with attribute value “lifecycle:transition”. Thus, an activity in conjuction with a event type is an RBAC operation.

RBAC operations can be also of different types. For example, when defining RBAC model in model-driven approach when using SecureUML[22], these types are referred to as action types and each type “represents a class of security relevant operations on a particular type of protected resource”. In RBAC definition, as discussed in section 2.3, we interpreted them as CRUD operations, such as create, read, update, or delete operation. However, this kind of information is not captured within event logs and it is difficult or even impossible to derive this information from event logs. Therefore, operation types are not handled explicitly in current work.

2.5.2 RBAC Subject and Role

User and role information are expressed through “Organizational” extension of XES using the extension prefix “org”[21], Hence, string elements are used which have key attributes with values “org:resource” and “org:role” for denoting user and role, respectively.

2.5.3 RBAC Resource

In current approach, data attribute keys and data attribute values which are extracted from an event log are considered as RBAC resources. Event log can also include data attributes that may not be relevant when creating RBAC model. It requires manual intervention for sorting them out by domain expert. Additionally, some of the global attributes are used not only on the level of event, but also on the level of entire process instance. For the simplicity, only data attributes are considered that are associated with particular event (specifically, with particular task). In terms of RBAC, data attributes are protected resources.

XES element (in the scope of event)	RBAC element
`string` element with attribute value „concept:name“	Operation
`string` element with attribute value „lifecycle:transition“	Specification of an operation
`string` element with attribute value „org:resource“	User (or subject)
`string` element with attribute value „org:role“	Role
All the data attributes in the scope of event	Resource attribute

2.6 Extensible Markup Language

Extensible Markup Language (XML)[24] is a markup language for documents to present its contents in a structured way. XML allows to express semantics of the elements, which makes the XML document both human-readable and machine-readable. XML is considered as one of the main data storage and exchange formats, because it is widely used as a base for

integration and communication between different applications.

The validation of XML is done using schemas, such as XML Schema Definition (XSD)[25]. XSD formally describes the purpose of the XML document, specifically the structure and constraints, which the XML document needs to conform in order to be valid.

In order to use the contents of XML document in an application, XML processors, or simply XML parsers, take the XML document and specification (for example XSD) as input and read in the information stored in XML document. If XSD or DTD (Document Type Definition) is specified, XML processor will also give the validation.

In this work, XML document is used for presenting the RBAC model, which can be adapted by other applications.

2.7 Summary

In this chapter, we presented the concepts and technologies used in this thesis. In particular, we introduced the main concepts of process mining. We discussed the business process event logs, especially the primary format for event log and the minimum requirements for event log used in this work. We introduced the RBAC model definition which is the fundamental to the remainder of this work. We discussed how the XES event log elements are interpreted as RBAC model concepts. Lastly, we gave a short overview of XML technology.

3 Related Work

In this chapter, we give an overview of the state of the art. First, we discuss the approaches regarding process mining, specifically role mining or organizational mining, which can be used for extracting data for RBAC models. Then, we give an overview of existing approaches related with extracting RBAC models from business process data stored in specification files or business process event logs. Finally, we compare the related works against the current solution and highlight the contributions of this work.

3.1 Organizational Mining

Our main interest in process mining is to extract business process related data from an event log, specifically different business process elements and their relationships which can be mapped to RBAC concepts. In process mining research field, a lot of effort has been devoted to control-flow discovery. However, there exists many process mining algorithms that mine different perspectives of business process [17] that could be potentially contribute to creating RBAC models, other than just business process control-flow. In [17], a role assignment perspective is presented, which captures the relationships between roles and activities. These relationships are discovered by algorithm that clusters subjects using similarity metric. Some of the organizational mining techniques are presented in [5] which can be used for discovering organizational perspective of business process, such as organizational structure and interactions between different organizational entities. In [5], organizational entity is defined as a set of originators (persons or machines executing business process activities) who represent some organizational unit, role, etc. The methods that are included in the mentioned approach [5] are also supported by different plug-ins of open-source process mining workbench ProM[13], such as Organizational Miner, Social network miner. However, these process mining techniques do not focus on deriving RBAC model elements, therefore they are not designed to extract RBAC data and create RBAC models from event logs.

In the field of role mining, there have been different methods proposed how to derive role information from different system configuration sources. Kuhlmann et al. uses data mining techniques (e.g., association rule algorithm, hierarchical clustering) to find roles from an existing database of cross-platform access rights [26]. Subsequently, RBAC model can be created based on these detected patterns. Molloy et al. propose an approach for finding RBAC model based on the observed usage of the permissions by system users [27]. The approach uses generative machine learning algorithms, such as Latent Dirichlet Allocation (LDA) and Author-Topic Model (ATM), which are enhanced with a discretization procedure to convert the probabilistic assignments into actual binary permission to role and role to user assignments. Another approach proposed by Molloy et al. is to mine roles with semantic meanings [28]. The authors make use of available attributes attached with user data (for example, job title, department name, location) and possible permission information (permission parameters, permission updates, permission usage) which help to create roles with semantic meanings. The idea behind this approach is that “a semantically meaningful role should correspond to a real-world concept, and a real-world concept can be described by an expression of user-attributes”[28]. In this work, we apply the same principle on the protected resources and their attributes within event log, i.e., a group of data attributes semantically form a resource.

These previous role mining techniques are not fully incorporated within this work. The main reason is that the event log in format of XES can include role information, which is annotated with “org:role” attribute key. Thus, the “org:role” attribute can be used to mine roles for

RBAC model.

3.2 Extracting RBAC Models from Business Process Data

Mengling et al. have proposed an approach [29] how to extract RBAC model from business process specification presented in Business Process Execution Language (BPEL)[30]. In particular, they present mappings from BPEL to RBAC elements. For example, BPEL *partner* and *partnerRole* represent roles. BPEL activities for incoming messages, such as *pick*, *receive*, and synchronous *invoke*, are considered as RBAC operations. BPEL *port types* (as the interfaces to actual implementation of the system) are regarded as RBAC objects. Therefore, permissions are operation-object pairs. The authors also provide extraction mechanism, using XSLT transformation script, how to transform XML-encoded BPEL process to RBAC model in XML format supported by access control policy management tool called xoRBAC[31] (a component of role engineering tool called xoRET[32]). The presented work is similar to this thesis contribution in a sense that these approaches are concerned with access control for business processes, although sources, where RBAC data is extracted from, are different.

Baumgrass presents an approach [6] of deriving current state RBAC models from event logs which is the most similar work to this thesis. In several aspects, the approach is the same, specifically when mapping event log elements in format of XES to RBAC model elements. However, there are still some conceptual differences, which needs to be highlighted. Baumgrass focuses on more role engineering and organizational mining techniques to discover roles if no role information is present in an event log. The role mining method used in that work is adapted from [5], which is assigning subjects into roles based on the similarity of performed tasks. At first, each subject has exactly one role. In the next step, subjects with similar permissions (performing similar tasks) are grouped together into single role. Eventually, these relations between different roles will result in a role hierarchy. In this work, an assumption has been made that role information is present in an event log (using XES organizational extension). The main reason is that both role and subject information are required in order to validate this approach (see chapter 6). Using this classification algorithm may give different results for different event logs. However, in conformance checking the classifications of users as roles needs to be the same for both event logs. In reality, if there is no role information, then these user classifications needs to be examined and conformed by domain worker before creating the RBAC model. This indicates that the process of creating RBAC cannot be fully automated when there is no roles specified beforehand. Therefore, in this work for simplicity, roles are identified using the “org:role” attribute in event logs. In addition, our approach is only concerned with flat RBAC models without role hierarchy, constraints, and separation of duty properties.

Another major difference is that, the only specified protected RBAC resource is information system (referred to as *source* from which the event log is extracted) in [6], whereas in this thesis, protected resources are resource attributes represented as data attributes in event logs. These data attributes can be interpreted as different data fields of documents or artifacts used during business process and they have a value to the organization. In [6], business process activities are identified as permissions, because they are performed only on one object (information system). Whereas in current work, there can be a number of resources and permissions are resource-operation pairs. In terms of Information System Security Risk Management (ISSRM)[33], the protected resource in [6] is considered as information system asset (material asset[3]) which supports business assets (immaterial assets, such as information,

3 Expect software

processes). Therefore, the approach presented [6] tries to establish access control on the level of information system, whereas current work on the level of information system resources.

The last notable difference is in the use of tool for access control policy management system, called xoRBAC[31], in the approach [1]. This tool can be integrated with software, which requires integration via C or tcl linkage. In order to fill this gap in current work, a prototype (see section 5.4) is developed, which demonstrates the applicability of current approach and allows to make preliminary adjustments to the RBAC data. xoRBAC provides also features, which support constrained RBAC. Constrained RBAC (section 3.3 in [1]) defines notions called static and dynamic separation of duties. An extension to [6], Baumgrass et al. present an approach [15] for deriving static and dynamic separation of duties as well as subject and role binding constraints from event logs. Constrained RBAC model and binding constraints are out of the scope of this work.

3.3 Comparsion of Related Works

In this section, we present a comparison between some of the related works and current work (see Table 2). We have taken into account the following characteristics: *Business process event logs* – is the approach related with business process event logs; *RBAC* – does this work serves as basis for creating RBAC models; *RBAC presented standard format* – is the RBAC model presented explicitly in some standard format; *Protected business assets* – is this work concerned with protecting business assets, such as document data. All of these selected approaches support extracting role information from system configuration files or logs, which we have left out from the characteristics.

Three approaches are not using event log as the source to extract information. For example, in [26] the roles are found from large database of cross-platform access rights, in [27] the roles are mined from access log records (as permissions usages), in [28] roles with semantical meanings are discovered from a synthesized data set. The solutions which are not focused on creating RBAC models, do not give an explicit representations of RBAC either nor do they are concerned with protecting business assets. The solutions presented in [6] and [29] include creating RBAC models from the extracted data, which are exported in XML-based format that is supported by xoRBAC[31]. One can argue, if the solution [29] is protecting business assets or not. In [29], BPEL *portTypes* are interfaces to objects, therefore it can be said that business assets are protected indirectly. As we discussed in previous section, then in [6] the only protected resource is information system, which makes unclear what are the data attributes and what operations are permitted on data resources.

The key focus of this approach is to protect business assets through access control, therefore this approach complements the work [6] discussed in section 3.2. The main novelty of this approach lies in creating RBAC model automatically based on the business process information extracted from an event log. Another contribution is storing the RBAC model in XML-based format that can be adapted easily by different applications. For example, information systems that implement RBAC or access control policy management systems.

Table 2: Comparison of related works

	Business process event logs	**RBAC**	**RBAC presented standard format**	**Protected business assets**
Organization model mining [5]	Yes	No	No	No
Role mining using data mining techniques [26]	No	No	No	No
Machine learning to detect access control policies [27]	No	No	No	No
Role mining based on the semantic meanings based on user attributes [28]	No	No	No	No
Deriving current state RBAC models from event logs [6]	Yes	Yes	Yes (XML-based format supported by xoRBAC)	No (indirectly through information system).
Creating RBAC model from BPEL based business process [29]	No	Yes	Yes (XML-based format supported by xoRBAC)	Yes
Current approach	Yes	Yes	Yes	Yes

4 An Approach to Extract RBAC Models from Business Process Event Logs

In this chapter, we present our contribution which includes the method to derive an RBAC model from an event log which consists of three main steps, as illustrated in Figure 3. In the first step, called analysis, a process mining technique is used to extract process-related data from an event log in XES format. In the next step, extracted data can be transformed into an in-memory RBAC model. Before that, minor adjustments could be made to the extracted data, so that the data and relationships would reflect actual settings of the business process. In the final step, an RBAC model is exported to the XML-based format in order to support the data exchange between different applications, e.g., information systems could implement the RBAC model or access policy management systems could be used to enhance the RBAC model.

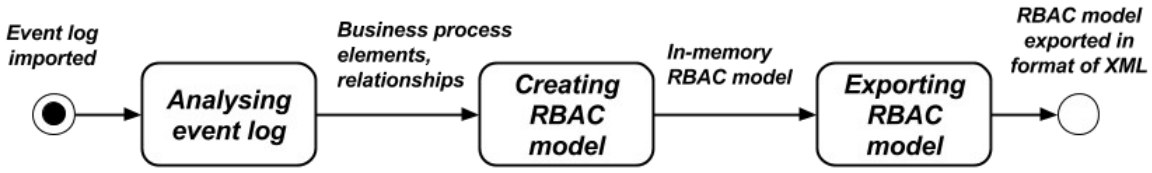

Figure 3: Step-wise approach of deriving RBAC model from an event log

4.1 Analysing Event Log

Before an event log can be imported and analyzed, it needs to meet predefined requirements (see section 2.2.3). In this work, we consider event logs represented in XES format (described in ch. 2.2.2). The XES format supports the basic business process elements and has been presented as a standard logging format by IEEE Task Force on Process Mining[18]. The major benefit of using XES is that we can avoid tackling problems that may arise when using different vendor-dependent business process logs. Analysis is important for getting insights about the business process by extracting data and data relationships from the event log before proceeding with creating an RBAC model.

The first step involves importing and analysing the event log. The analysis consists of parsing the event log and extracting information about roles, originators, activities, data attributes of the business process, and creating relationships between these elements (see Figure 4). For this purpose, an algorithm is used, described in section 4.2.

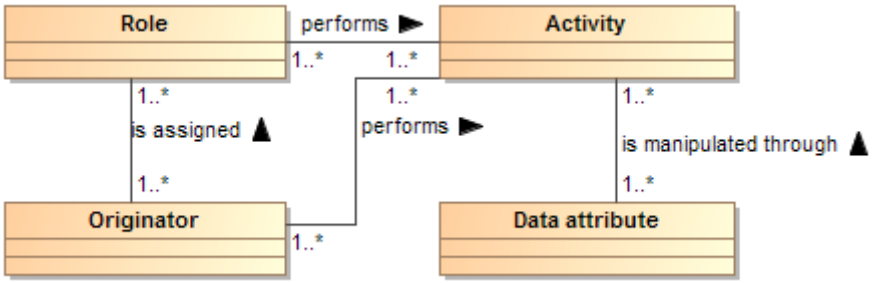

Figure 4: Business process elements and relationships elicited form an event log

When analysing the event log, no information about the secured resource is present in it, but only a set of data attributes. Semantically, a composition of these data attributes describes the

resource. The resource attributes are interpreted as protected resources. We assume that they have potentially a business-critical importance to the organization. Additionally, in this work, activities in conjunction with event type (such as start or completion of an activity) are considered activities.

4.2 Algorithm to Extract Business Process Elements and Relationships

In this section, we describe the high-level algorithm (see Algorithm 1) for discovering business process elements and their relationships. First, we need to have an event log as input. Then we can begin with the procedure by instantiating variables, where:

1. R is a finite set of roles present in the event log,
2. O is a finite set of originators present in the event log,
3. A is a finite set of activities present in the event log,
4. and D is a finite set of data attributes (in here, we consider data attribute keys) present in the event log.

We define projections, such as $\pi_A(e)=a$, $\pi_O(e)=o$, $\pi_R(e)=r$, $\pi_D(e)=d$ on every event. An event is defined as a 4-tuple $e=(a,o,r,d)\in E$, where a is an activity, o is an originator, r is a role, d is a data attribute recorded with the event, and E is the universe of the events contained in the event log. For simplicity, we define only one projection for data attribute (i.e., data attribute key) d , however there can be more different data attributes captured with the event.

We also define different assignments as follows:

a) *Role-to-originator assignments*: $R_O=\{(r,orig)\in R\times O|\exists e\in E,\pi_R(e)=r\wedge\pi_O(e)=orig\}$. R_O relation means that if at least one event $e\in E$ with originator $orig\in O$ and role $r\in R$ is recorded in the event log, then we assign role r to originator $orig$.

b) *Data-to-activity assignments*: $D_A=\{(d,a)\in D\times A|\exists e\in E,\pi_D(e)=d\wedge\pi_A=a\}$. D_A relation means that if at least one event $e\in E$ with data attribute $d\in D$ and activity $a\in A$ is recorded in the event log, then we assign data attribute d to activity a , i.e., this data attribute d can be accessed during activity a .

c) *Activity-to-role assignments*: $A_R=\{(a,r)\in A\,x\,R|\exists e\in E,\pi_A(e)=a\wedge\pi_R(e)=r\}$. A_R relation means that if at least one event $e\in E$, where role $r\in R$ executes activity $a\in A$, is recorded in the event log, then we assign activity a to role r , i.e., activity a can be executed by an originator with role r .

During the procedure we iterate over every event $e\in E$ recorded in event log and check if the event has any of the required elements attached. At every step we update the appropriate value sets and relations. Finally, after completing the procedure, we have an in-memory business process model, i.e., business process elements with data and relationships between these elements.

Algorithm 1: High-level algorithm to elicit business process elements and their relationships

```
Input: event log in XES format
Name: discovering business process elements and relationships
Output: in-memory business process model

set   R ← Ø , O ← Ø , A ← Ø , D ← Ø
for each event   e ∈ E   do
     // get activity, originator, role, and data attribute from   e
      a ← π_A(e)   ,   orig ← π_O(e)   ,   r ← π_R(e)   ,   d ← π_D(e)
     // add elements to appropriate sets
       A ← A ∪ {a}   ,   R ← R ∪ {r}   ,   O ← O ∪ {orig}   ,   D ← D ∪ {d}
     // add relations as assignments
       R_O ← R_O ∪ {(r , orig)}   ,   D_A ← D_A ∪ {(d , a)}   ,   A_R ← A_R ∪ {(a , r)}
endfor
```

4.3 Creating RBAC Model

Before proceeding with the second step, minor refinements can be made to the extracted information. The adjustments might include changing *role-subject*, *role-activity* relationships, and if necessary changing *role* names and *data attributes* (excluding the ones that are not necessary). The rationale behind this refinement is to confirm assignments of users to roles, actual activities performed by role, and select data attributes which are considered valuable business assets to the organization. This is a manual activity that needs to be carried out by a domain expert to assure that the information accurately reflects the actual real life process.

In order to create an RBAC model from an in-memory business process as a set of business process elements with values and relationships, we introduce transformation rules to translate those business process elements and relationships (as in Figure 4) into RBAC model elements and relationships (as in Figure 2). During this step, we take all the knowledge collected about the business process and place it in the context of security constraints in the form of an RBAC model. In general, most of the concepts are the same for business process models and RBAC models. Thus, transformation rules take care of renaming business process elements to RBAC model elements. The main value of transformation rules is to create permissions for different roles based on the extracted business process information. As illustrated in Table 3, we take business process elements (including element values) *role*, *originator*, *activity*, and *data attribute*, and create respective RBAC model elements (with values): *role*, *subject*, *operation*, and *resource attribute*. In the RBAC model, we do not create relationships between role and activity or originator and activity. Instead, we have a role-permission assignment, where permission is an operation-resource attribute mapping (or activity-data attribute, in the of business process terminology).

Table 3: Tranformation rules for transforming business process elements and relations to RBAC model elements and relationships

Business process	RBAC model	Comment
Role	*Role*	*Business process roles are translated to RBAC roles.*
Originator	*Subject*	*Originators (as users) are translated to RBAC subjects.*
Activity	*Operation*	*Business process activities are translated to RBAC operations.*
Data (or *data attribute*)	*Resource* (or *resource attribute*)	*Data (or data attributes) are translated to RBAC resources (or resource attributes).*
Role-Activity	-	*No direct role-operation in the RBAC model.*
Originator-Activity	-	*No direct subject-operation in the RBAC model.*
Activity-Data attribute	*Operation-Resource attribute*	*Activities that access or manipulate data attributes are translated to operations on resource attributes. In the of RBAC terminology, this relationship is called as* ***permission****.*
-	*Role-Permission*	*This relationship is created based on the Role-Activity and Activity-Data attribute relationships from the business process.*

4.4 Algorithm to Transform Business Process Relationships to RBAC Model Relationships Using Transformation Rules

In this section, we describe the high-level algorithm (see Algorithm 2) to transform business process elements and relationships to RBAC model elements and relationships using the transformation rules presented in Table 3.

First, we need to have as input an in-memory business process. In-memory business process consists of a set of roles R, a set of originators O, a set of activities A, a set of data attributes D, and assignments R_O, D_A, and A_R (descriptions and informal definitions are given in section 4.2).

We begin the procedure by instantiating the variables for the RBAC model:

1. R_{RBAC} is a finite set of RBAC roles;
2. S_{RBAC} is a finite set of RBAC subjects;
3. O_{RBAC} is a finite set of RBAC operations;
4. RA_{RBAC} is a finite set of RBAC resource attributes;
5. and PER is a finite set of RBAC permissions.

We also define different assignments:

1. *Role-to-subject assignments*: $R_{S_{RBAC}}=\{(r,s)\in R_{RBAC}\times S_{RBAC},(r,s)\in R_O\}$. $R_{S_{RBAC}}$ relation means that for every role-originator assignment R_O in business process model, there is role-to-subject assignment $R_{S_{RBAC}}$ in the RBAC model.
2. *Resource-to-operation assignments*: $RA_{O_{RBAC}}=\{(res,o)\in RA_{RBAC}\times O_{RBAC}|\exists(res,o)\in D_A\}$. $RA_{O_{RBAC}}$ relation means that for every data-to-activity assignment D_A in the business process model, there is a resource-to-operation assignment $RA_{O_{RBAC}}$ in the RBAC model.
3. *Permission assignments*: $PER=\{(res,o,r)\in RES_{RBAC}\times O_{RBAC}\times R_{RBAC},(o,r)\in A_R\wedge(res,o)\in D_A\}$. PER relation means that for every such resource-activity assignment $(d,a_1)\in D_A$ and activity-to-role assignment $(a_2,r)\in A_R$ where $d\in D$, $a_1,a_2\in A$ and $a_1=a_2$, $r\in R$ in business process model, there exists permission $(res,op,r)\in PER$ in RBAC model, where $res\in RA_{RBAC}$ is protected resource attribute, $op\in O_{RBAC}$ is RBAC operation, $r\in R_{RBAC}$ is role, and $res\in RA_{RBAC}$ corresponds to $d\in D$, $op\in O_{RBAC}$ corresponds to a_1 and a_2 , and $r\in R$ corresponds to $r\in R_{RBAC}$.

Finally, after applying the transformation rules on the in-memory business process, we get the in-memory RBAC model, whose components and relationships correspond to the RBAC model definition presented in section 2.3.

Algorithm 2: High-level algorithm to transform a business process model into a RBAC model

```
Input: in-memory business process model
Name: algorithm to transform business process model into RBAC model
Output: in-memory RBAC model

set   R_RBAC ← ∅ , S_RBAC ← ∅ , O_RBAC ← ∅ , RA_RBAC ← ∅ , P_RBAC ← ∅

for each activity   a∈A
      op_RBAC ← a
      O_RBAC ← O_RBAC ∪ {op_RBAC}
endfor

for each originator   o∈O
      subject_RBAC ← o
      S_RBAC ← S_RBAC ∪ {subject_RBAC}
endfor

for each role   r∈R
```

$r_{RBAC} \leftarrow r$
$R_{RBAC} \leftarrow R_{RBAC} \cup \{r_{RBAC}\}$
endfor

// Role-subjects assignments
for each role-to-originator $(r, o) \in R_O$
$ro_{RBAC} \leftarrow (r, o)$
$R_{S_{RBAC}} \leftarrow R_{S_{RBAC}} \cup \{ro_{RBAC}\}$
endfor

for each data attribute key $d \in D$
$d_{RBAC} \leftarrow d$
$RES_{RBAC} \leftarrow RES_{RBAC} \cup \{d_{RBAC}\}$
endfor

// Resource-operation assignments
for each data-to-activity $(d, a) \in D_A$
$da_{RBAC} \leftarrow (d, a)$
$RES_{O_{RBAC}} \leftarrow RES_{O_{RBAC}} \cup \{da_{RBAC}\}$
endfor

// Permission assignments
for each resource $res_{RBAC} \in RA$
for each operation $o_{RBAC} \in O_{RBAC}$
for each role $r_{RBAC} \in R_{RBAC}$
if $(res_{RBAC}, o_{RBAC}) \in D_A \wedge (o_{RBAC}, r_{RBAC}) \in A_R$
then
$per \leftarrow (res_{RBAC}, o_{RBAC}, r_{RBAC})$
$RA \leftarrow RA \cup \{per\}$
endif
endfor
endfor
endfor

4.5 Exporting RBAC Model

After the RBAC data is extracted from an event log, we can create an XML document which presents the RBAC model. We decided to use the XML format, because it is supported in many environments, and is a base for integration and communication between different applications. XML is platform-independent and is not only used for showing data values but also constraints and relationships between data. Therefore, it is suitable for capturing RBAC model elements and relationships. In this work, an XML document is used for making RBAC models available to other applications, for example when implementing an RBAC mechanism into the information system that supports business processes or when importing it into an access control policy management system.

The structure of the XML document is provided in XML Schema Definition (XSD), which is graphically represented in Figure 5 (the content of XSD file is given in Table 13). The XSD for RBAC models defines the structure and necessary RBAC components (such as roles, subjects, operations, resources or data resources), and the corresponding relationships (for example role to subject assignments, operations on resources). There is a separate element in

the XML document for permissions. XML document does not only contain permissions, but also declares RBAC component instances in a separate list with unique identifier. These instances are referred in other parts of the document with this unique identifier. The XSD also defines sub-elements *operations* and *values* for *resources*, i.e., permissible operations on resource and possible values of resource. *Permissions* element is divided into sub-elements by *resource*, which in turn consists of sub-elements *permission*. Permission has attributes like *role* and *operation* (representing the role-to-operation assignment). Permissions could have also been presented as a list of flat elements, all having attributes *resource*, *role*, and *operation*, but this is just a matter of representation.

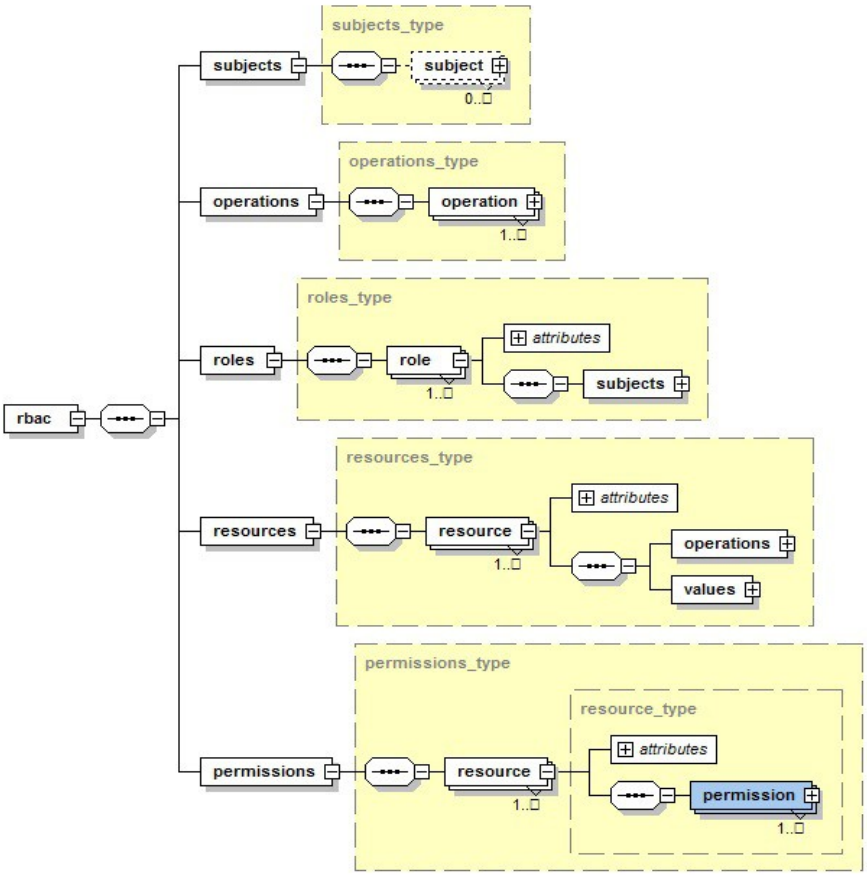

Figure 5: XML Schema Definition for RBAC model

4.6 Discussion

An event log can include information about user groups or departments (using "org:group" key in XES), which should not be interpreted as roles. This kind of categorization is used for dividing users into departments or larger user groups based on the organizational goals. Therefore, this information can be used to develop an enterprise-wide RBAC model as an extension of the standard RBAC model. In this work, we consider only simple RBAC models without the mentioned extension.

In the core RBAC definition (section 2.3), the user has a permission for an operation that changes a single resource. However, when we are dealing with business process event logs, then during a business process activity, multiple data attributes as data resources are manipulated. We have agreed that a composition of data attributes within an event is a

resource and modifying one or many data attributes implies that whole resource is also modified, although no information about whole resource is explicitly captured in event log. Another difference with the standard RBAC definition is missing information about operations with action types, such as create, read, update, delete. Action type information is impossible to be automatically extracted from a log. Determining action types of different operations would require insights about the business process. For the sake of simplicity, we have not considered action types in our RBAC model, but they can be seen as an improvement of the current approach.

5 Proof of Concept

In this chapter, we illustrate the step-wise approach on a running example, which is an adjusted scenario from [8]. The scenario describes a process for handling a request for ticket compensation within an airline (the process model is depicted in Figure 6). In section 5.1, we present the results of analysing the example event log. In section 5.2, we present the results of creating RBAC model from the business process information extracted in the analysis step. In section 5.3, we present the output of exporting the created RBAC model in XML format. In section 5.4, we explain the usage of the developed prototype for this thesis. The purpose of this prototype is to demonstrate the applicability of the approach to create RBAC models from event logs (semi-)automatically.

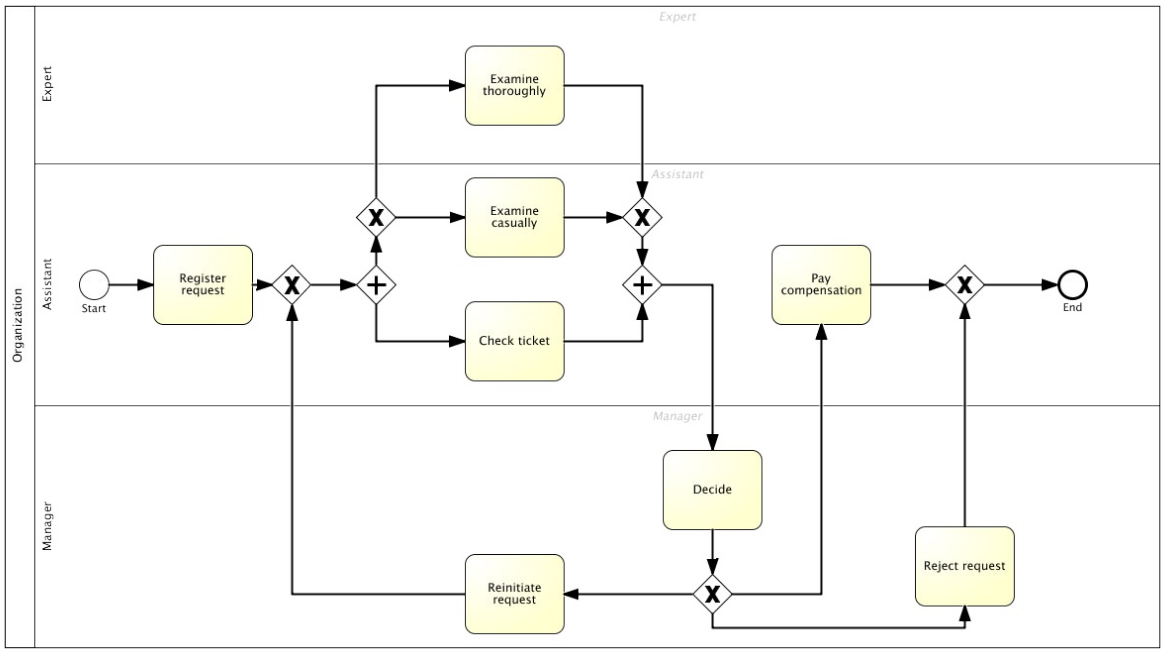

Figure 6: Business process that corresponds to the example scenario

5.1 Analysing Event Log

Prerequisite for the first step is that we have an event log of the business process, which is possibly generated by a non-RBAC information system. The event log needs to be in XES format and meets the predefined requirements introduced in section 2.2.3. Once we have the event log, we import the event log into the prototype application for analysis. The analysis includes extracting business process related data and relationships from an event log using a process mining technique (the algorithm is given in Algorithm 1). As a result, we get different elements and relationships of the business process (in [17], they are referred to as business process entities and relationships) needed for constructing RBAC model.

First, we find role-to-originator assignments, i.e., roles, originators, and relationships between them (as shown in Table 4). In our example, every user has exactly one role, however a user can have more than one role (based on the RBAC definition given in section 2.3). We can see from the table there are 3 different roles: *Assistant*, *Manager*, and *Expert*; and there are in total 5 different users: *Pete*, *Ellen*, *Sara*, *Mike*, *Sean*, where each subject is assigned to a role.

Table 4: Role-to-originator assignments discovered from the event log represented in Table 12

Originator	Role
Pete	Assistant
Ellen	Assistant
Sara	Manager
Mike	Assistant
Sean	Expert

Next we find data and data attributes being processed during the business activities. A list of data attributes and activities extracted from the example event log is given in Table 5. There are 3 data attributes (*cost, cid* (client id), and *status)* and there are in total 8 activities (i.e., *Register request*, *Check ticket*, *Examine thoroughly*, *Examine casually*, *Decide*, *Reinitiate request*, *Reject request*, and *Pay compensation)*.

In business process, one activity can manipulate multiple data resource attributes at once (as we can see also from the table), which is different from the standard RBAC, where one operation (an activity) can change only one data resource. However, as discussed in section 4.6, we consider compositions of data attributes resulting in a single resource. Therefore, when we speak in terms of RBAC, then it is acceptable to change multiple data resource attributes of a single data resource (a document or an artifact which is comprised of different data fields) during one single operation.

Table 5: Activities on different data attributes discovered from the example event log

Data attribute(s)	Activity
cost, cid, status	Register request
cost, cid, status	Check ticket
cost, cid, status	Examine thoroughly
cost, cid, status	Examine casually
cost, cid, status	Decide
cost, cid, status	Reinitiate request
cost, cid, status	Reject request
cost, cid, status	Pay compensation

Lastly, we find roles executing particular activities (see Table 6). Each activity is performed by only one role, although a role can be assigned to many users and an activity can be performed by many users. From the business point of view, one can also make conclusions about what are the business tasks that different roles are responsible for.

Table 6: Role and activity assignments discovered from the example event log

Activity	Role
Register request	Assistant
Check ticket	Assistant
Examine thoroughly	Expert
Examine casually	Assistant
Decide	Manager
Reinitiate request	Manager
Reject request	Manager
Pay compensation	Assistant

5.2 Creating RBAC Model

Prerequisite for creating an RBAC model is to have different perspectives of business process as an in-memory business process model (discovered in the first step). Once we have an in-memory business process, we can transform it into an in-memory RBAC model using the transformation rules implemented in the prototype. The transformation rules (also presented in Table 3) and the outcomes are presented in Table 7.

Table 7: Results of transforming an in-memory business process model into an in-memory RBAC model using transformation rules

Transformation rule	Business process model	RBAC model
Role -> Role	Assistant Manager Expert	Assistant Manager Expert
Originator -> Subject	Pete Ellen Sara Mike Sean	Pete Ellen Sara Mike Sean
Role-Originator -> Role-Subject	Presented in Table 4, only in the business process context.	Presented in Table 4, only in the RBAC model context.
Data (or *data attribute*) -> *Resource* (or *resource attribute*)	cost cid status	cost cid status
Role-Activity -> Role-Operation	The same element contents and relationships as presented in Table 6, only in the business process context.	-
Activity-Data attribute -> Operation-Resource attribute (*permission*)	The same element contents and relationships as presented in Table 5, only in the	The same element contents and relationships as presented in Table 5, only in the RBAC

Transformation rule	Business process model	RBAC model
Role -> Role	Assistant Manager Expert	Assistant Manager Expert
	business process context.	model context.
Role-Permission	-	Role-Permission assignments are presented in Table 14.

After applying the transformation, we get the following RBAC components and contents:

- subjects: *Pete*, *Ellen*, *Sara*, *Mike*, *Sean* (the same individuals already identified within the business process);
- roles: *Assistant*, *Manager*, and *Expert* (the same roles already identified within business the process);
- role-to-subject relationships: *Assistant* - *Pete*, *Ellen*, *Mike*; *Manager* - *Sara*; *Expert* - *Sean* (the same relationships already identified in Table 4);
- protected resource attributes: *cost*, *data, status*;
- permissions (a flat representation is given in Table 14).

We visualize the in-memory RBAC model using SecureUML[22] (see Figure 7). The diagram shows that *TicketCompensationRequest* is a protected resource as a composition of the discovered resource attributes *cost, cid,* and *status*. In reality, information about resource (specifically the name of the resource) may not be present in event log. Therefore, in order to get protected resource and resource attributes as valuable assets for the business, they need to be recorded in the event log, so that they can be used as a basis when creating the RBAC model.

In the SecureUML diagram (see Figure 7), role-to-user assignments are presented as associations with stereotype <<*secuml.roleAssignment*>>. A permission, such as *ManagerTicktetCompensationRequestPermission*, is presented as an association class with stereotype <<*secuml.permission*>> between role *Manager* (a role class annotated with stereotype <<*secuml.role*>>) and resource *TicketCompensationRequest* (a resource class annotated with stereotype <<*secuml.resource*>>). Based on the running example, *Manager* has permissions to perform operations *decide*, *reinitiateRequest*, and *rejectRequest* on resource *TicketCompensationRequest*.

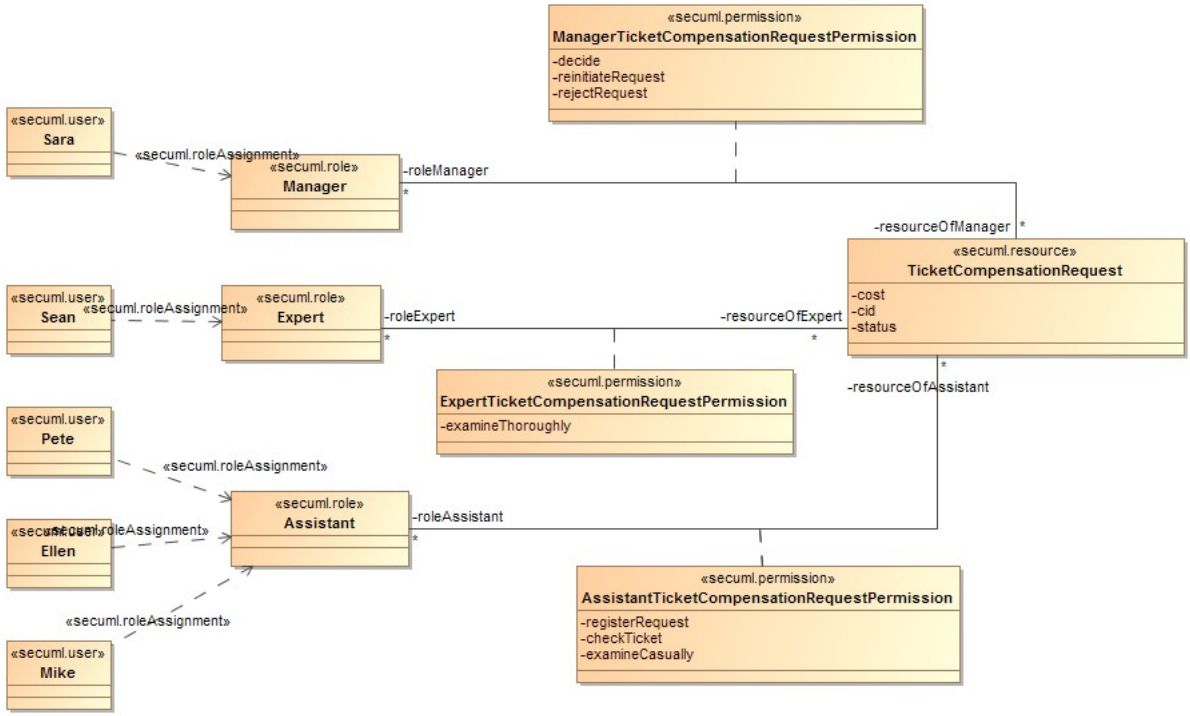

Figure 7: TicketCompensationRequest resource permissions presented using SecureUML

5.3 Exporting RBAC Model

Prerequisite for exporting RBAC model in XML format is to have created an in-memory object of RBAC model using our prototype application. Once we have an in-memory RBAC model, we can use the XML writer implemented in our prototype to generate our RBAC model (in the form of declarative access control policies) as an XML document, whose structure and data types are defined using XML Schema (represented in Table 13).

A fragment of the XML document generated for our running example is given in Figure 8. As we can see, the XML document contains a complete list of RBAC data, such as subjects, roles, resources, and permissions, and including all of the data relationships, such as the role-to-subject, resource-operation assignments, and permissions. Most importantly, we can capture the RBAC model components and relationships in the XML document (the full contents of the XML document for our running example is given in Table 15) as they are represented in our in-memory RBAC model (as illustrated in Figure 7 using SecureUML). The RBAC model in XML format can be used by different applications for implementing access control mechanisms for information systems that support the underlying business process.

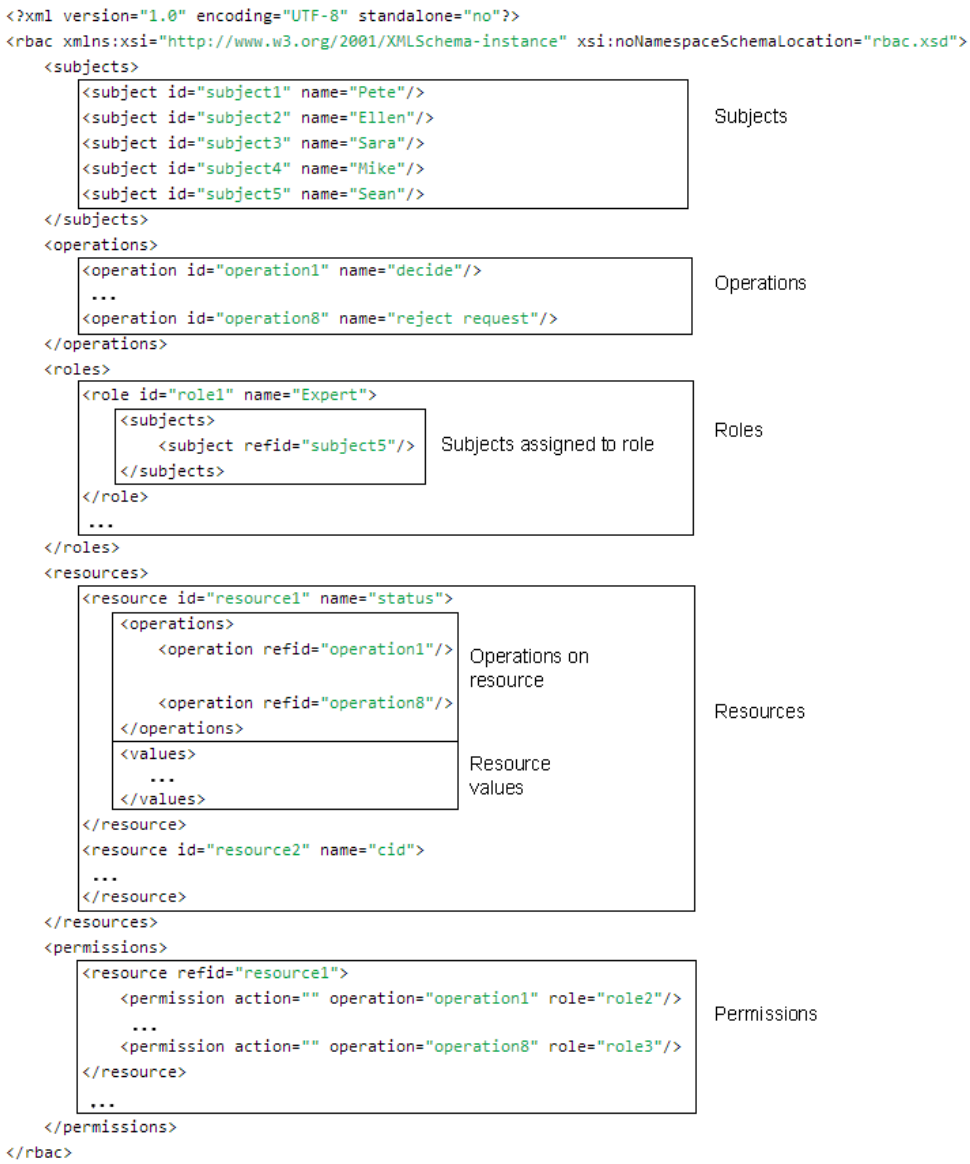

Figure 8: A fragment of the XML document representing the RBAC model

5.4 Using The Prototype

In this section, we describe the overall process of using the developed prototype for creating RBAC models from event logs. Event log can be opened using *Open event log* in the *File* menu. The only supported event log file type is XES. Once we have the event log imported, we can run the analysis by clicking on the *Analyze* button. Event log file is internally processed and an algorithm is applied to discover business process elements and relationships.

After analysis, different views of discovered business process information are displayed in three tabs: *Roles*, *Attributes*, and *Operations* (see Figure 9). These three different views show role-to-originator (or role-to-subject) relationships, attribute-activity (or resource-operation) relationships, and role-activity (or role-operation) relationships derived from the extracted business process information.

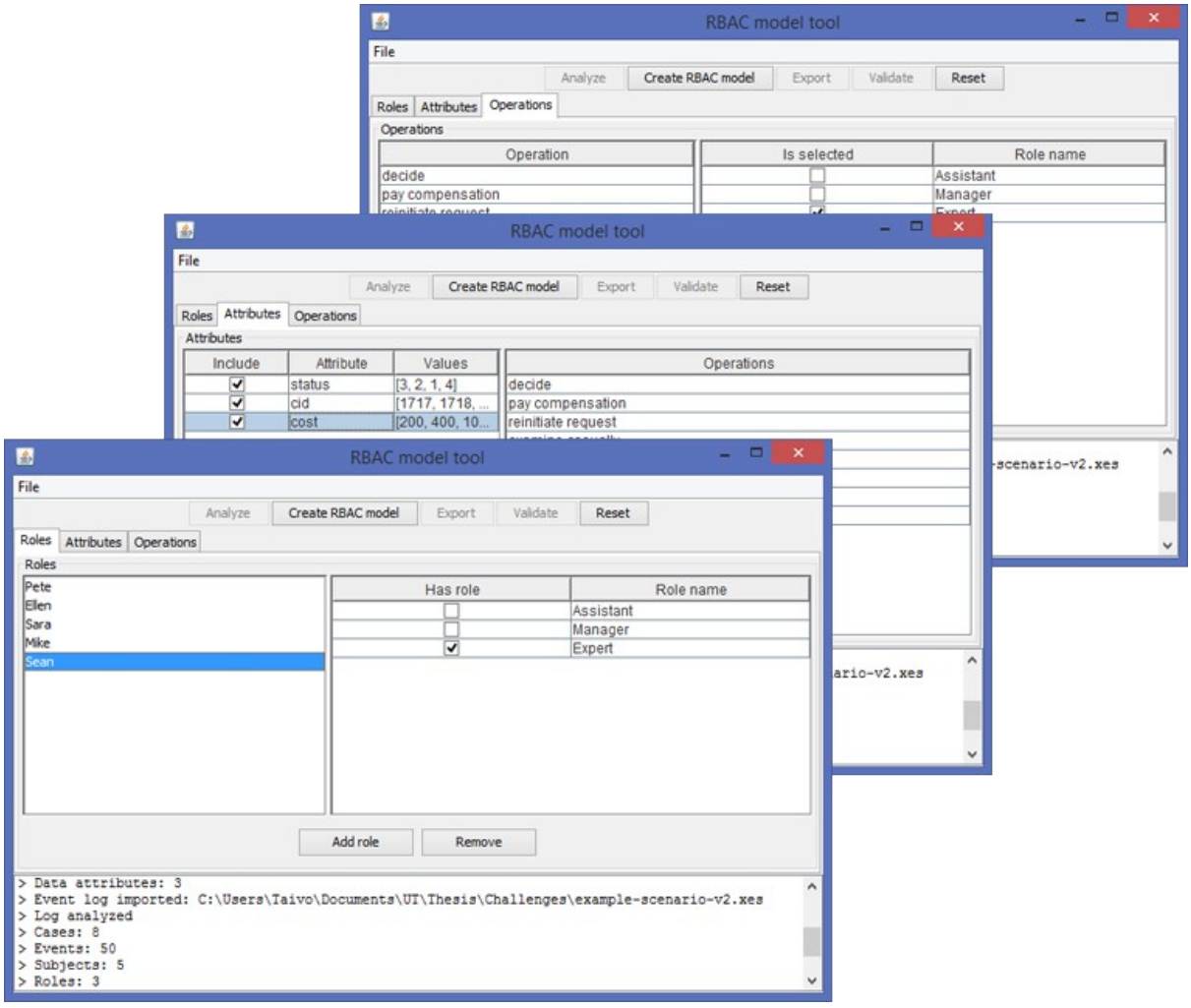

Figure 9: An analyzed event log which different views are presented in three tabs: Roles, Attributes, and Operations

Minor adjustments can be introduced to discovered data. For instance, we can rename discovered roles in *Roles* tab, change assignments of role-to-subject in *Roles* tab and role-operation in *Operations* tab. We can choose data attributes in *Attributes* tab which we think are considered as protected resources. New roles can be added to RBAC model using *Add role* button in *Roles* tab. New role can be assigned to admissible operations in the *Operations* tab. Unfortunately, this prototype does not allow to add any other new relationships between discovered business process elements.

Once the analysis is completed, RBAC model can be created by clicking on the *Create RBAC model* button. Internally, in-memory object of RBAC model is created after the transformation rules are applied on the business process data and relationships. Next we can export the RBAC model in XML format via *Export* (also in the *File* menu there is an *Export* button) and save it on the local hard drive. The XML file includes necessary RBAC model elements, such roles, subjects, data, resources, and permissions, and relationships between these elements.

Additionally, already existing RBAC model in XML format can be imported by clicking on the *Import* in the *File* menu. Optionally, this imported data can be modified, as discussed before, and new RBAC model can be created with the new adjustments. Otherwise, by

clicking on the *Create RBAC model*, new in-memory object is created based on the imported data. The idea is to allow to modify and use already existing RBAC model instead of creating RBAC from an event log again.

5.4.1 Comparison of RBAC Models With The Prototype

We can validate the created in-memory RBAC model against different RBAC model For the validation, the other RBAC modal needs to be in XML format (follows the XML Schema definition for our RBAC model as presented in Table 13) and can be imported by clicking on the *Validate* button. If there are no differences, the application will notify the user. Otherwise, a separate window opens where user can handle the discrepancies between different RBAC models (see Figure 10). The user can resolve the differences (potential violations), which is accepting or rejecting the differences, by clicking on the *Resolve and export* button. In the background, these two RBAC models are merged together into new resolved RBAC model with changes accepted by the user.

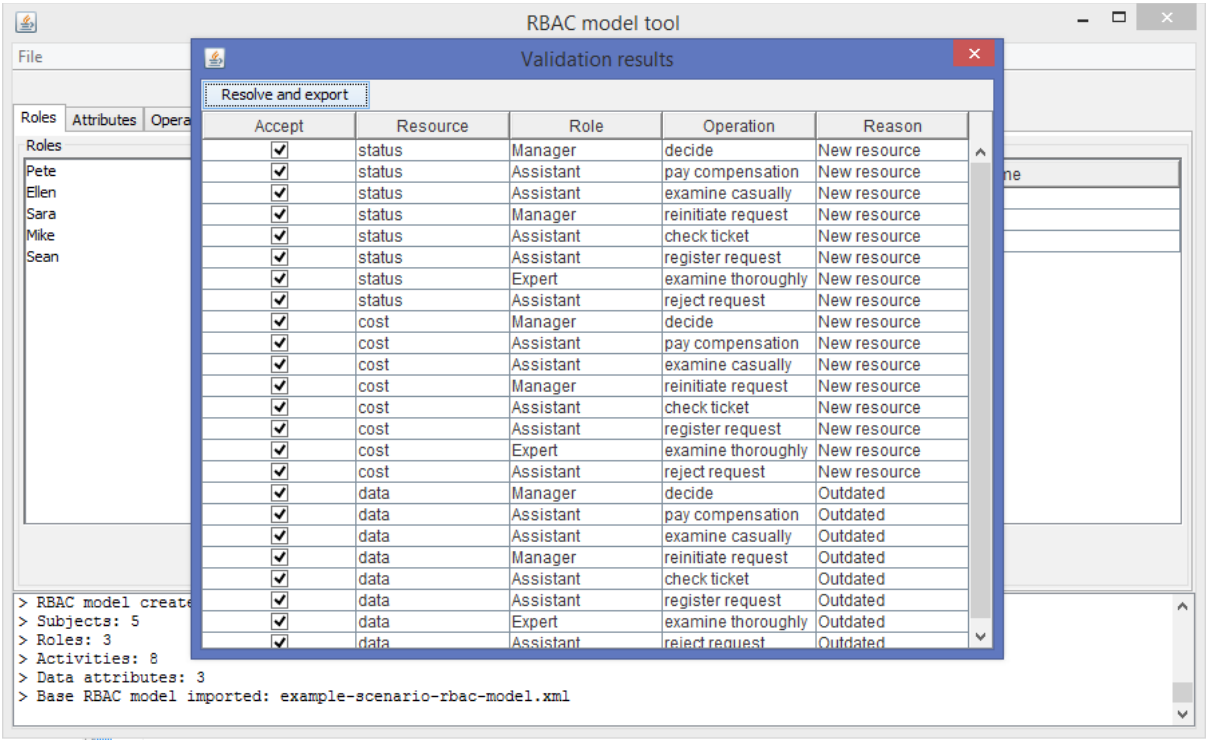

Figure 10: Violation handling between different RBAC models

5.4.2 Exporting LTL Constraints

For the validation of current approach, additional functionality was implemented into the prototype to create Linear Temporal Logic formulas from RBAC model constraints (specifically role-permission assignments). When in-memory RBAC model is created, we can export LTL formulas based on the RBAC constraints by clicking on the *Export LTL formulas* in the *File* menu (see Figure 11). In the background, an algorithm is applied on the RBAC model constraints to get LTL formulas, which then are exported as LTL file. The generated LTL can be used for checking conformance between RBAC model constraints and an event log, as will be discussed in section 6.2.

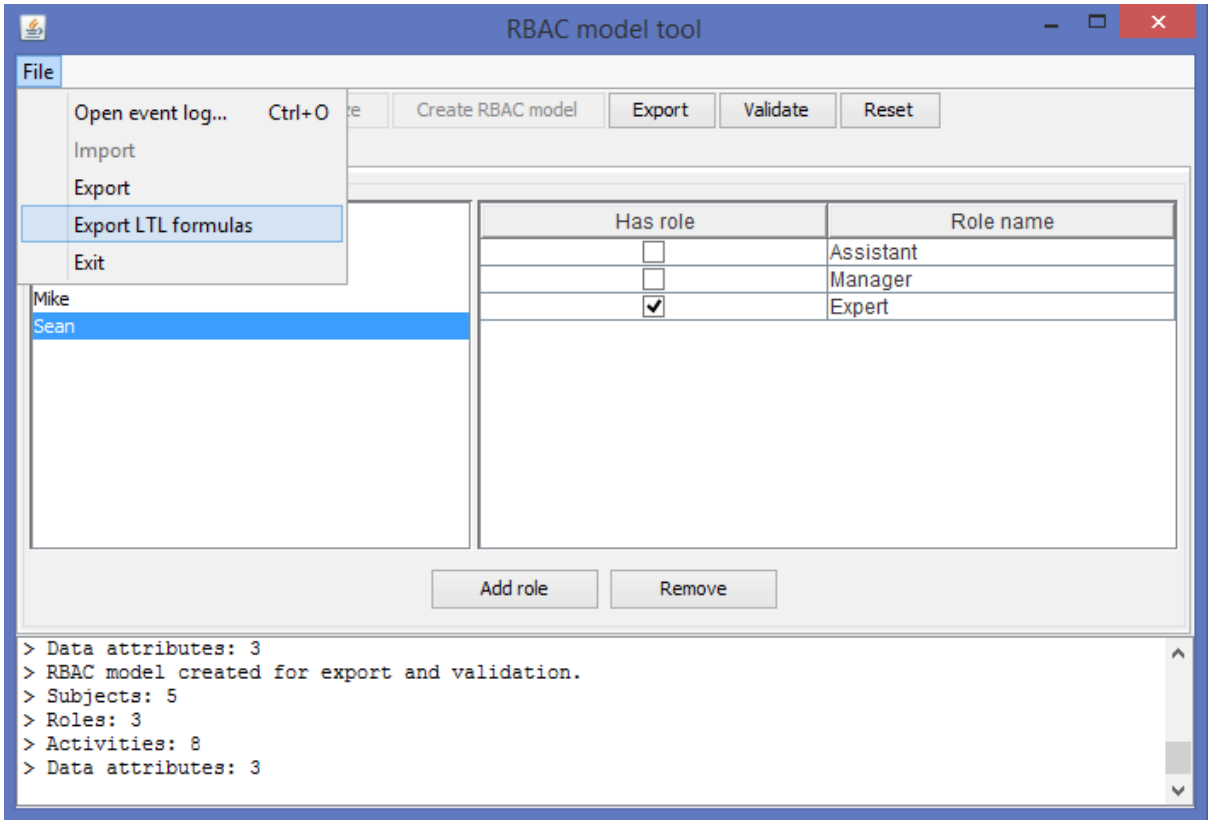

Figure 11: Exporting RBAC model contraints as LTL contraints

5.5 Discussion

In here we give some reasoning regarding running example and RBAC model concepts. Business process activities can compose of more complex tasks than just manipulating data attributes. In addition, the activity names can be also too ambiguous and do not indicate what data attributes are actually accessed or changed, for example „decide“, „examine thoroughly“, „examine casually“. For this reason, we only concentrate on the data attributes being captured in an event log and the activities, despite their names, associated with these attributes.

In most cases, event logs do not include information about action types (such as create, read, update, delete) information, therefore it is impossible to automatically detect action types for RBAC operations without such information, For this reason, we have not added any action type information to our RBAC models and have not presented this information in SecureUML diagram (see Figure 7). This is different to the standard SecureUML metamodel [22] where every permission includes possible types of actions on every resource attribute. In addition, for the sake of simplicity, we have not explicitly presenting any resource attribute specific methods in the SecureUML diagram or in our RBAC models, because they are mostly getting and setting data attribute values.

6 Evaluation of The Approach Through Conformance Checking

In this chapter, we evaluate the approach (presented in chapter 4) through conformance checking. The conformance checking is divided into two parts. In the first part (see 6.1), we show how we compare two RBAC models extracted from different event logs produced by the same source information system and how we resolve the differences between these RBAC models. In the second part (see 6.2), we show how we check the conformance of the event log and the LTL formulas, which are translated from RBAC models. Lastly, we provide a running example how the conformance checking is done on two real-life event logs.

6.1 Comparison of RBAC Models from Different Event Logs

A real-life scenario would be that the RBAC model is implemented into the information system. However, we might still want to check if RBAC policies are enforced correctly and if there are any deviations as possible vulnerabilities that might pose a threat to the security of the information system and its data. If the system is process-aware and stores business process executions as a stream of events in format of XES, then we can use this event log to extract RBAC data, as described in section 4.1. After that, we can confirm if there are any deviations with the previous RBAC model and newly extracted RBAC model.

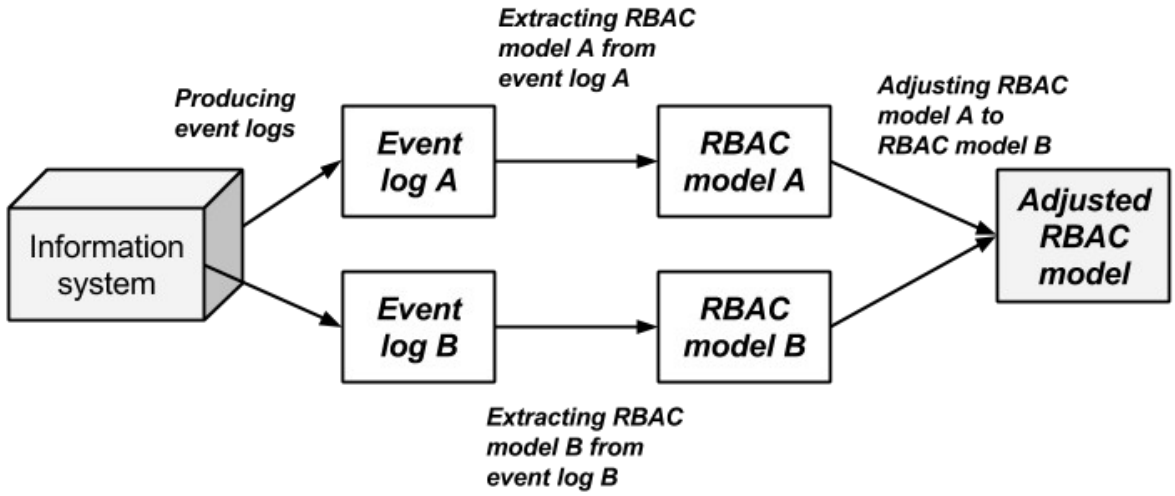

Figure 12: Extracting RBAC model A and RBAC model B from different logs and adjusting RBAC model A to RBAC model B

The first part of the conformance checking is illustrated in Figure 12. Prerequisite for this comparison is that we have two different event logs which are produced by the same information system. These event logs are the input for creating RBAC models using the approach presented in chapter 4. The first RBAC model extracted from the first event log is considered as the *base RBAC model*, denoted as RBAC model *A*. The second RBAC model extracted from the second event log (assumingly the newest event log) is considered as the recent RBAC model, denoted as RBAC model *B*, which is validated against RBAC model *A*. For the validation of these two RBAC models, we compare RBAC model constraints, i.e., permissions assigned to some particular role. If there are discrepancies in these constraints then we interpret them as violations. In order to determine these violations, a two-way check is performed: the RBAC model *B* is tested against the RBAC model *A* and vice versa. The

check needs to be done in both directions in order to get permissions introduced by the second RBAC model *B* and permissions that are in the first RBAC model *A* but not in the second RBAC model *B*. For this reason, there can be different causes for these violations:

- the base RBAC model *A* can be outdated containing permissions that are not in effect anymore;
- business process and responsibilities within an organization has changed, therefore new permissions are introduced into the RBAC model *B*;
- the extracted RBAC model *B* contains permissions that are not defined in the base RBAC model *A* nor these permissions are introduced by organization (possible violations).

As a conclusion, not every case of violation is intentional. Violation handling needs to be performed by knowledge worker who then distinguishes whether a violation is intentional or not. After the knowledge worker has resolved all of the discrepancies, the adjusted RBAC model can be generated which takes into account new changes.

6.2 Conformance Checking with LTL Formulas

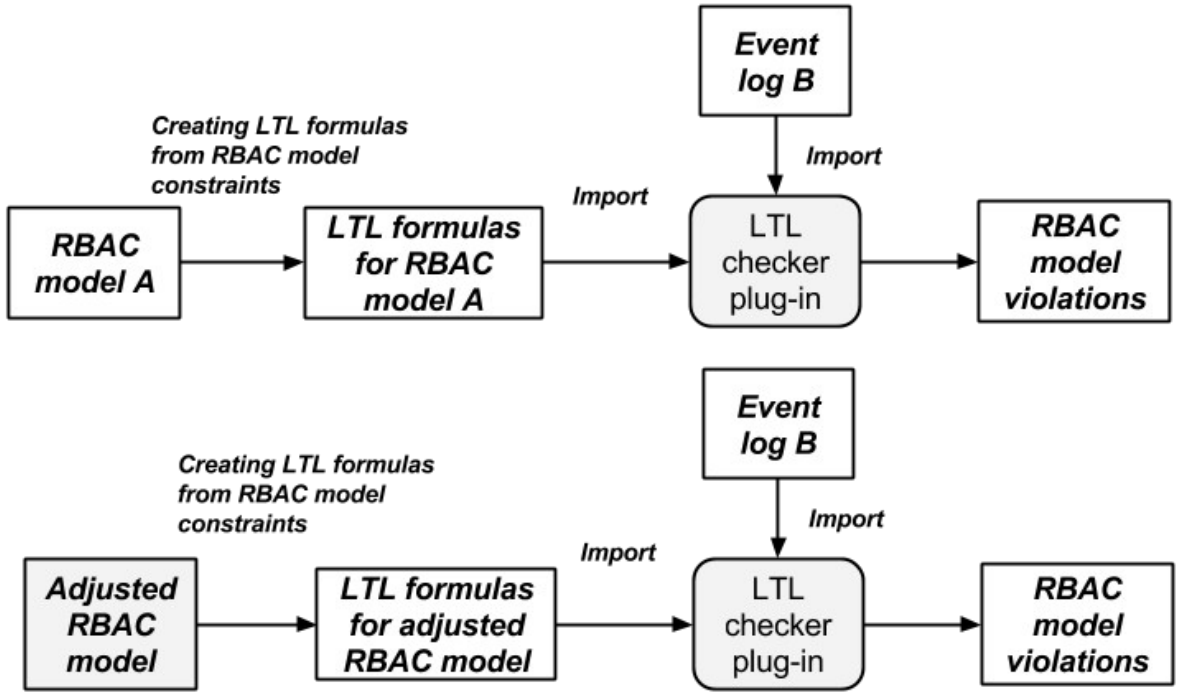

Figure 13: Creating LTL formulas based on the RBAC model A and the adjusted RBAC model. These LTL formulas are used to check the event log conformance against the RBAC models.

The second part of the conformance checking is illustrated in Figure 13. The rationale behind this conformance checking is that the RBAC model constraints might not always be fully compliant with the business process executions as they are captured in an event log. Therefore, we want to show that if we improve the RBAC model by introducing new RBAC permissions, which is extracted from the event log, then the results could be better, too.

For this purpose, we use RBAC model *A* as the RBAC model, which is not fully compliant

with the event log *B*. In order to improve the RBAC model *A*, we use the method described in the previous section 6.1. We extract the RBAC model *B* from the event log *B*. After the comparison of the RBAC model *A* and RBAC model *B*, we adjust the RBAC model *A* to RBAC model *B*, which is essentially creating new RBAC model with the accepted differences. As a result, we get the adjusted RBAC model, which can be interpreted as the improved RBAC model.

For checking of the RBAC model *A* and the adjusted RBAC model constraints (role-permission assignments) and the business process executions in the event log *B*, we utilize the approach presented in [7] where LTL formulas are generated from business process-related RBAC models.

We create an LTL formula for every role-permission assignment, the formula template is presented in Table 8. The formula specifies if always some event with data attribute named *ATTR* and operation *ACTIVITY* (optionally with *EVENT_TYPE*) occurs, then it is executed by the authorized subject *P* out of *n* authorized subjects. These set of subjects belong to respectful roles which have the permission. An excerpt of LTL file with example formula is given in Table 9. We use subjects instead of roles in LTL formulas, because for LTL checker plug-in there is no standard definition for role and only subject declarations (i.e., "ate.originator") are supported. Another reason could be that in most event logs there are no role information included.

Table 8: LTL formula to check role-permission assignment

```
[](((ATTR != "" /\ activity == ACTIVITY /\ eventType == EVENT_TYPE) ->
(subject == P1 \/ (subject == P_N-1 \/ subject == P_N ...))));
```

Table 9: LTL formula to check role-permission assignment for activity „Examine casually"

```
formula RO_Assistant_attr1_examine_casually () := {}
[](((attr1 != "" /\ activity == "examine casually") -> (subject ==
"Mike" \/ (subject == "Ellen" \/ subject == "Pete"))));
```

If this statement does not hold for some event in the sequence of events, then we can say that the event in the sequence of the events is not consistent with this formula and could be a possible violation with respect to RBAC model.

As illustrated in Figure 13, we use LTL checker plug-in to perform the conformance checking. LTL checker plug-in expects an event log (possibly in format of XES) and LTL file with the LTL formulas as the input in order to start analysing event log and checking the log against LTL formulas. First, we check the conformance of the RBAC model A and the business process executions in the event log *B*. Next, we check the conformance of the adjusted RBAC model and the business process executions in the event log *B*. Ideally, the results for the adjusted RBAC model should be improved, i.e., the adjusted RBAC model is more compliant with the event log *B* with respect to the RBAC model *A*.

6.3 A Running Example of Conformance Checking with LTL Rules

In this section, we illustrate the conformance checking with LTL formulas on a running example. For this purpose, we use two different real-life event logs which originate from the incident and problem management system of Volvo IT Belgium, called VINST[34], and were prepared for the BPI Challenge 2013 [35]. The first event log is about open problems, and the second event log is about closed problems. These event logs meet the requirements presented

in section 2.2.3.

We create the base RBAC model *A* from the event log *A,* closed cases, and the RBAC model *B* from the event log *B,* open cases. We want to know how much the RBAC model *A* conforms to the business process executions in the event log *B* and check if how much the conformance improves when adjusting the RBAC model *A* to RBAC model *B*.

We have implemented RBAC model validation into the developed prototype. We use the prototype to validate RBAC model *B* against the RBAC model *A*. An extract of the validation results are given in Table 10 (for full list see Table 16), which consists of different examples of violations. Violations are categorized as follows:

- *Not allowed* – no such authorization constraint is defined in the base RBAC model *A,* although role, operation, and resource definitions are the same. For example, an individual with role *C_5* is not allowed to perform operation *Completed\nClosed* on *product*, because this is not permitted according to the RBAC model *A*, although such combination is present in the RBAC model *B*. Remark: Activities and event types of activities are separated with "\n".
- *New role*, *new resource*, *new operation* – the RBAC model *B* introduces new role, resource, or operation definitions. For example, resource (or resource attribute) *organization involved* is not present in base RBAC model descriptions.
- *Outdated* – authorization constraint defined in base RBAC model is not used anymore in other BRAC model.

Once we have validated the RBAC model *B* against the RBAC model *A*, we can create the adjusted RBAC model.

Table 10: An extract of validation results of two RBAC models for open and closed cases

Role	**Permission**		**Reason**
	Operation	**Resource**	
C_5	Completed\nClosed	product	Not allowed
C_7	Completed\nClosed	product	New role
E_9	Accepted\nIn Progress	product	New role
C_5	Completed\nClosed	resource country	Not allowed
C_5	Completed\nClosed	organization country	Not allowed
C_7	Accepted\nAssigned	organization country	New role
E_9	Accepted\nWait	organization country	New role
E_1	Completed\nClosed	organization involved	New resource
C_7	Completed\nClosed	organization involved	New resource, new role
C_5	Accepted\nWait	impact	Outdated

Next, we create LTL formulas based on the RBAC model *A* and the adjusted RBAC model constraints, specifically role-permissions assignments. For this purpose, a high-level algorithm (see Algorithm 3) is implemented into the developed prototype to generate LTL

formulas. The algorithm takes input: RBAC data attribute as protected resource attribute, RBAC operation (or activity in business process), and a list of RBAC subjects. These three attributes represent the role-permission relationship, where the role is a set of users and the permission is the pair of attribute and operation. The output of the algorithm is the assembled formula. These formulas are gathered into LTL file (with *ltl* file extension), which can be used by the LTL checker plug-in[36] of the process mining workbench ProM[13]. The contents of an example LTL file is given in Table 17).

Once we have created the LTL formulas for the RBAC model *A* and for the adjusted RBAC model, we make the use of the LTL checker plug-in which expects two inputs: an LTL file which constraints LTL formulas and an event log which is checked against these LTL formulas. We perform two conformance checks with LTL plug-in, first with the RBAC model *A* and the event log *B*, and then with the adjusted RBAC model and the event log *B*.

Algorithm 3: High-level algorithm to compose LTL formula for role-permission assignment

```
function createRolePermissionFormula(attribute, operation, subjects)
      formula = ''
      formulaA = ''
      eventType = extractEventTypeFromOperation(operation)
      if eventType != '' then
            formulaA = '(attribute != "" /\ activity == "operation"  /\
eventType == "eventType")'
      else
          formulaA = '(attribute != "" /\ activity == "operation")'
      endif
      prevFormulaS = ''
      formulaS = ''
      for every subject in subjects
            formulaS = 'subject == "subject"';
            if prevFormulaS != '' then
                  formulaS = '(formulaS \/ prevFormulaS)'
            endif
            prevFormulaS = formulaS
      endfor
      formula = '[]((formulaA -> formulaS))';
      return formula
endfunction
```

The results of the first check are presented in Figure 14. The metrics used here to measure the conformance is satisfied and unsatisfied LTL formulas. LTL checker gives us that there are 16 formulas out of 35 unsatisfied. For example, a subject named *Habib* is executing operation *Accepted* (with event type *Assigned*) on attribute *product* (alias *attr1*), which has no permission definition in the RBAC model *A*, therefore it is considered as a violation. This can be also unintentional violation due to various reasons as explained in section 6.1.

The results of the second check are presented in Figure 15. LTL checker shows that the second event log is fully compliant with the adjusted RBAC model, i.e., there are no unsatisfied LTL formulas. Also fom the results, we can see that the subject named *Habib* has now the permission to execute operation *Accepted* (with event type *Assigned*) on attribute *product*.

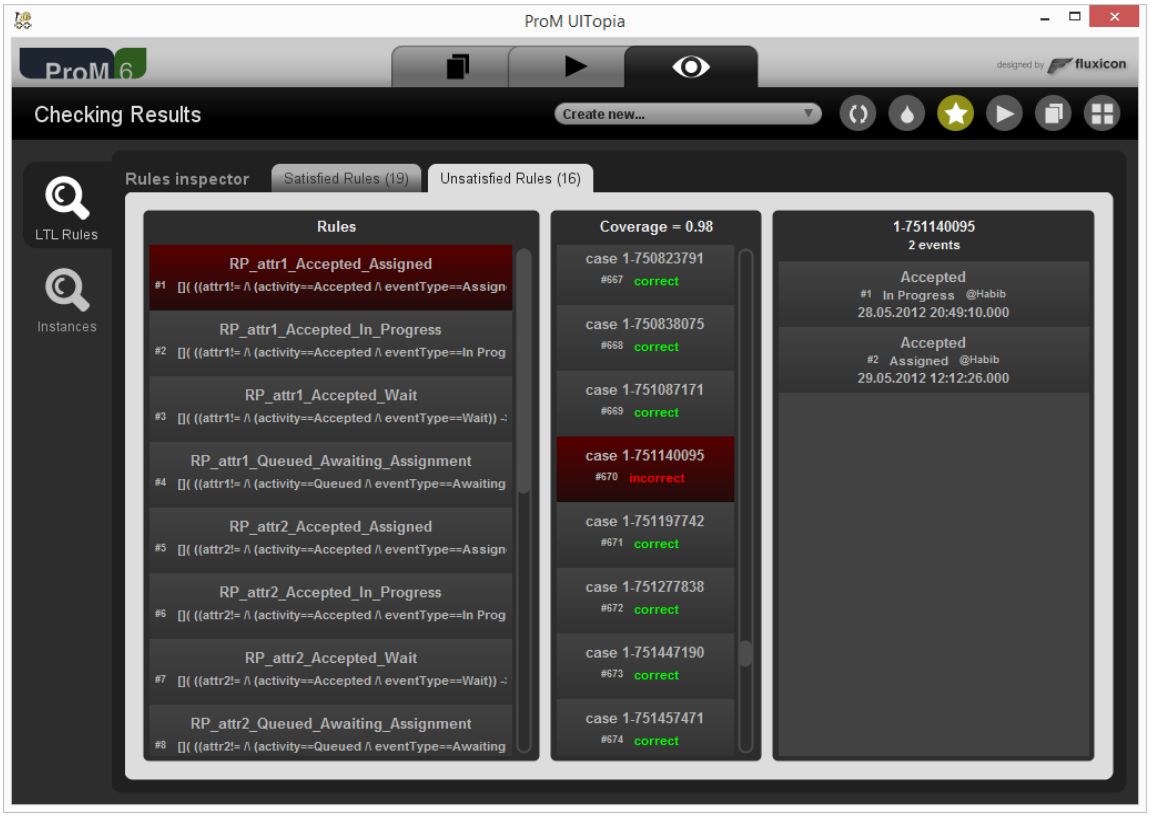

Figure 14: LTL checker results of conformance checking

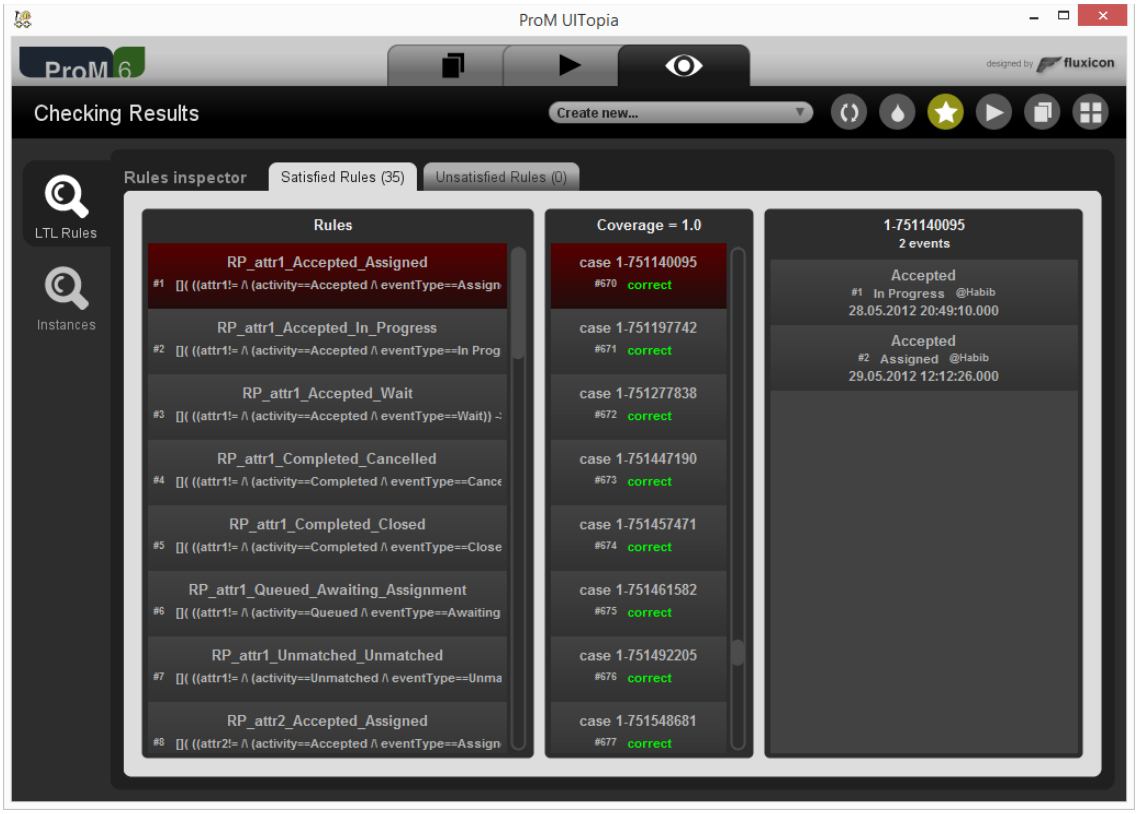

Figure 15: Full conformance of RBAC model and an event log

6.4 Discussion

In this chapter, we have shown that extracting RBAC models from two real-life event logs is applicable. In particular, we evaluated the approach with conformance checking, i.e., how much the role-permission assignments defined in RBAC model conform to the business process executions as captured in an event log. Additionally, we used the LTL checker to measure the conformance of RBAC model and an event log in terms of satisfied and unsatisfied LTL formulas.

In section 6.1, the different event logs may not capture the entire business process, which inevitably will lead to deviations between RBAC models. This means that these differences needs to be resolved manually by domain worker. Additionally, we used different logs instead of doing the conformance checks with one event log. For obvious reasons, the RBAC model will be fully compliant with the event log from which the RBAC model was extracted. Therefore, the goal was also to show that we can improve the extracted RBAC models. The improvement was measured in terms of satisfied and unsatisfied LTL constraints.

In section 6.2, we define formula (see Table 8) which checks only for role-permission assignments, i.e., the LTL formula is satisfied if the permission exists in RBAC model and it is performed by authorized subject defined in RBAC model or the permission does not exist in RBAC model, then the formula does not check any further and the formula is satisfied automatically. For the latter case, a (sub)formula is needed to check if operation and resource attribute relationship is also in accordance with the RBAC model.

7 Conclusions and Future Work

In this thesis, we have proposed an approach how to extract RBAC models (semi-)automatically from event logs in XES format, where the focus is also on determining the protected business assets, such as document or other artifact data that is exchanged and accessed during business process activities. We divided the approach into three consecutive steps. The first step involves in analysing event log, which is utilizing business process mining technique to extract business process elements and relationships from an event log. The results of the first step are the input for creating in-memory RBAC model in the second step. We create RBAC model by applying transformation rules on the extracted business process elements and relationships. In the last step, we export in-memory RBAC model to XML document, whose structure is specifically defined by XML Schema Definition for this work. In addition, we evaluated the approach through conformance checking.

7.1 Limitations

In this work, we have required that both user and role information is present in event log, although user information is not entirely needed when creating RBAC model. This requirement is only set for conformance checking using Linear Temporal Logic constraints.

The developed prototype allows to make primitive changes to the extracted business process data and relationships, such as changing role-user relationships, role-activity relationships, and including or excluding data attributes as protected resources. It does not allow to introduce new users, new activities and thereby new RBAC permissions.

We are not including RBAC operation types in our RBAC model, such as create, read, update, delete. Mainly because they are difficult or even impossible to extract from event logs automatically without knowing the underlying business process or PAIS implementation.

7.2 Conclusions

We have built our thesis around two research questions (formulated in chapter 1):

1. What data could be extracted from a business process event log and how this data could be used for creating RBAC models?

When answering to the first question we addressed what is the data that could be extracted from an event log and how this data could be used for creating standard RBAC models. We selected the XES as the primary format of event logs in this thesis work, which can include the minimum set of business process elements, such as users, roles, activities, and data attributes, to derive RBAC model components. In order to extract the data from potentially large event logs, we used the process mining technique (an algorithm presented in Algorithm 1). We divided the approach of creating RBAC model from an event log into 3 steps. The first step is analyzing event log (as discussed in section 4.1), which is extracting business process data from an input event log. The outcome of the first step is the business process elements and relationships. The second step involves in creating RBAC model (as discussed in section 4.3) based on the extracted business process elements and relationships using transformation rules (presented in Table 3). We get in-memory object of RBAC model as the output. Lastly, we can present this RBAC model explicitly by exporting the in-memory object of RBAC model as XML document (as discussed in section 4.5). The XML document follows the structure and data type constraints of XML Schema Definition proposed for our RBAC models.

2. How applicable is this approach on real-life business process event logs?

When answering to the second question we addressed the possible methods how to validate the approach. In this work, we evaluated the approach through conformance checking. For that reason, we used a real-life event log from BPI Challenge 2013, which was split into two parts, open and closed cases. The first conformance checking included the comparison of RBAC models created from different log samples of an event log. The second conformance checking was testing the compliance of RBAC model constraints in the form of the Linear Temporal Logic formula against an event log.

As a result, we have shown a possible approach how to create RBAC models from event logs generated by information systems that support business processes. The aim was not to only get RBAC roles, but to also focus on extracting information about business assets that needs to be protected. The contribution of this work also includes developed prototype application (as discussed in chapter 5) to demonstrate the applicability of our approach and to also provide means of the approach evaluation, such as validating and merging two different RBAC models and creating LTL constraints based on RBAC model constraints.

7.3 Future Work

Although, we have required that role information needs to be included in the event logs for this approach, but typically there are no role information in event logs. The next opportunity would be to extend this work with discovering roles based on the characteristics of event log in XES format (or in MXML[37], an alternative but not so preferred logging format for business processes). A possible solution would be to apply classification algorithms. Some existing ideas were presented in the section of related work.

In this work, we use our XML Schema Definition for presenting RBAC models. Apparently, there are more sophisticated technologies how to specify RBAC model in XML-based format. We see an opportunity to develop an extension to export RBAC models created from event logs also in XACML[38] or in X-RBAC[39].

Additionally, due to the limited capabilities of managing extracted RBAC data with the developed prototype, improvements can be made to the current solution. Alternatively, this tool can be integrated with existing access control policy management systems, for example xoRBAC[31].

References

[1] D. F. Ferraiolo, R. Sandhu, D. R. Kuhn, R. Chandramouli. *Proposed NIST Standard for Role-Based Access Control*, *ACM Transactions on Information and System Security (TISSEC)*, Volume 4 Issue 3, pp. 224-274. August, 2001.

[2] A. C. O'Connor, R. J. Loomis. *Economic Analysis of Role-Based Access Control*, 2010.

[3] M. Dumas, W.M.P. van der Aalst, and Hofstede. *Process-Aware Information Systems*. Wiley. 2005.

[4] R. Accorsi, T. Stocker, On the Exploitation of Process Mining for Security Audits: The Process Discovery Case, 2012

[5] M. Song, W. M. P. van der Aalst. *Towards Comprehensive Support for Organizational Mining*, *Decision Support Systems*, Volume 46, Issue 1, pp. 300-317. , 2006.

[6] A. Baumgrass. *Deriving current state RBAC models from event logs*. In: *International Workshop on Security Aspects of Process-aware Information Systems (SAPAIS), Proc. of the 6th International Conference on Availability, Reliability and Security (ARES)*. Vienna, 2011.

[7] A. Baumgrass, T. Baier, J. Mendling, M. Strembeck, *Conformance Checking of RBAC Policies in Process-Aware Information Systems*. In: F. Daniel, K. Barkaoui, S. Dustdar, editors. *Business Process Management Workshops: BPM 2011 International Workshops, Revised Selected Papers, Part II*, pp. 435-446. Springer. 2011.

[8] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance, and Enchancement of Business Processes*. Springer. 2011.

[9] Object Management Group, Business Process Model and Notation (BPMN), version 2.0: http://www.omg.org/spec/BPMN/2.0/ (last visited: 01.05.2014)

[10] Yet Another Workflow Language (YAWL): http://www.yawlfoundation.org/ (last visited: 01.05.2014)

[11] A. Scheer, O. Thomas, O. Adam, *Process Modeling Using Event-Driven Process Chains*. In: M. Dumas, W. M. P. van der Aalst, and A. H. M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*, pp. 119-145. Wiley. 2005.

[12] W. M. P. van der Aalst, C. Stahl. *Modeling Business Processes: A Petri Net-Oriented Approach*. MIT Press. 2011.

[13] Process Mining workbench ProM, v6.3: http://www.promtools.org/prom6/ (last visited: 11.05.2014)

[14] W. M. P. van der Aalst, A. J. M. M. Weijters, L. Maruster.*Workow Mining: Which processes can be rediscovered?*. In: *Knowledge and Data Engineering, IEEE Transactions on*, Volume 16, Issue 9, pp. 1128-1142. 2004.

[15] A. Baumgrass, S. Schefer-Wenzl, M. Strenbeck. *Deriving Process-Related RBAC Models from Process Execution Histories*. In: *4th IEEE International Workshop on Security Aspects in Processes and Services Engineering (SAPSE)*. Izmir, Turkey, 2012.

[16] W. M. P. van der Aalst, H. de Beer, B. van Dongen. *Process Mining and Verification of Properties: An Approach based on Temporal Logic*. In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science (LNCS)*. Springer-Verlag Berlin, Heidelberg, 2005.

[17] A. J. Rembert. *An Initial Approach to Mining Multiple Perspectives of a Business Process*. In: *he Fifth Richard Tapia Celebration of Diversity in Computing Conference:*

Intellect, Initiatives, Insight, and Innovations. New York, NY, USA, 2009.

[18] IEEE Task Force on Process Mining: http://www.win.tue.nl/ieeetfpm (last visited: 01.05.2014)

[19] OpenXES Developer Guide version 1.9: http://www.xes-standard.org/_media/openxes/openxesdeveloperguide-1.9.pdf (last visited: 01.05.2014)

[20] NITRO, process mining tool to convert tabular data to business process logs: https://fluxicon.com/nitro/ (last visited: 25.05.2014)

[21] C. W. Günther, E. Verbeek. *XES Standard Definition (version 1.4)*. October, 2014.

[22] T. Lodderstedt, D. Basin, J. Doser. *SecureUML: A UML-Based Modeling Language for Model-Driven Security*, 2002.

[23] Unified Modeling Language (UML): http://www.uml.org/ (last visited: 01.05.2014)

[24] The Extensible Markup Language, version 1.1: http://www.w3.org/TR/xml11/ (last visited: 25.05.2014)

[25] W3C XML Schema: http://www.w3.org/XML/Schema (last visited: 22.05.2014)

[26] M. Kuhlmann, D. Shohat, G. Schimpf. *Role mining - revealing business roles for security administration using data mining technology*. In: *Proc. of the 7th ACM Symposium on Access Control Modelsand Technologies (SACMAT)*. NY, USA, 2003.

[27] I. Molloy, Y. Park, S. Chari. *Generative Models for Access Control Policies: Applications to Role Mining Over Logs with Attribution*. In: *SACMAT '12 Proceedings of the 17th ACM symposium on Access Control Models and Technologies*. Newark, NJ, USA, 2012.

[28] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, J. Lobo. *Mining Roles with Semantic Meanings*. In: *SACMAT '08 Proceedings of the 13th ACM symposium on Access control models and technologies*. New York, NY, USA, 2008.

[29] J. Mendling, M. Strembeck, G. Stermsek, G. Neumann. *An Approach to Extract RBAC Models from BPEL4WS Processes*. In: *Proceedings. WETICE '04 Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Washington, DC, USA, 2004.

[30] Business Process Execution Language (BPEL), short for Web Services Business Process Execution Language (WS-BPEL): http://bpel.xml.org/ (last visited: 15.05.2014)

[31] Access control policy management tool xoRBAC: http://wi.wu-wien.ac.at/home/mark/xoRBAC/index.html (last visited: 01.05.2014)

[32] Role engineering tool xoRET: http://wi.wu-wien.ac.at/home/mark/xoRET/index.html (last visited: 01.05.2014)

[33] É. Dubois. P. Heimans, N. Mayer, R. Matulevičius, *A Systematic Approach to Define the Domain of Information System Security Risk Management*. In: S. Nurcan, C. Salinesi, C. Souveyet, J. Ralyté. *Intentional Perspectives on Information Systems Engineering*, pp. 289-306. Springer. 2010.

[34] Incident and problem management system VINST: http://www.win.tue.nl/bpi/_media/2013/vinst_manual.pdf (last visitied: 25.05.2014)

[35] 9th International Workshop on Business Process Intelligence 2013: http://www.win.tue.nl/bpi/2013/challenge (last visited: 25.05.2014)

[36] H. de Beer. *The LTL Checker Plugins: A Reference Manual*. September 20, 2007.

[37] B. F. Van Dongen, W. M. P. van der Aalst, A Meta Model for Process Mining Data, 2005

[38] eXtensible Access Control Markup Language (XACML) v3.0, OASIS Standard:

http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.pdf (last visited: 22.05.2014)

[39] J. B. Doshi, R. Bhatti, E. Bertino, A. Ghafoor.*X-RBAC: An Access Control Language for Multi-Domain Environments*. In: *Internet Computing, IEEE*, Volume 8, Issue 6, pp. 40-50. 2004.

Appendix A – Example event log as tabular data

Table 11: Example event log as tabular data

#PID	#EID	#DATETIME	#Activity	#Resource	#Role	#Cost	#Customer ID	#Status
1	25654521	30-12-2013 15:10	register request	Pete	Assistant	50	1123	1
1	25654523	30-12-2013 15:10	examine casually	Mike	Assistant	400	1123	1
1	25654524	30-12-2013 15:10	check ticket	Ellen	Assistant	100	1123	1
1	25654525	30-12-2013 15:10	decide	Sara	Manager	200	1123	2
1	25654528	30-12-2013 15:10	reinitiate request	Sara	Manager	200	1123	2
1	25654530	30-12-2013 15:10	examine thoroughly	Sean	Expert	400	1123	1
1	25654531	30-12-2013 15:10	check ticket	Pete	Assistant	100	1123	1
1	25654532	30-12-2013 15:10	decide	Sara	Manager	200	1123	2
1	25654533	30-12-2013 15:10	pay compensation	Ellen	Assistant	200	1123	3
2	25654534	30-12-2013 15:10	register request	Pete	Assistant	50	1717	1
2	25654535	30-12-2013 15:10	check ticket	Mike	Assistant	100	1717	1
2	25654536	30-12-2013 15:10	examine thoroughly	Sean	Expert	400	1717	1
2	25654537	30-12-2013 15:10	decide	Sara	Manager	200	1717	2
2	25654538	30-12-2013 15:10	reject request	Ellen	Assistant	200	1717	4
3	25654539	30-12-2013 15:10	register request	Ellen	Assistant	50	1718	1
3	25654540	30-12-2013 15:10	examine casually	Mike	Assistant	400	1718	1
3	25654541	30-12-2013 15:10	check ticket	Pete	Assistant	100	1718	1
3	25654541	30-12-2013 15:10	decide	Sara	Manager	200	1718	2
3	25654543	30-12-2013 15:10	reinitiate request	Sara	Manager	200	1718	2
3	25654544	30-12-2013 15:10	check ticket	Ellen	Assistant	100	1718	1
3	25654545	30-12-2013 15:10	examine casually	Mike	Assistant	400	1718	1
3	25654546	30-12-2013 15:10	decide	Sara	Manager	200	1718	2
3	25654547	30-12-2013 15:10	pay compensation	Ellen	Assistant	200	1718	3
4	25654550	30-12-2013 15:10	register request	Mike	Assistant	50	1900	1
4	25654555	30-12-2013 15:10	check ticket	Ellen	Assistant	100	1900	1
4	25654556	30-12-2013 15:10	examine casually	Mike	Assistant	400	1900	1
4	25654559	30-12-2013 15:10	decide	Sara	Manager	200	1900	2
4	25654560	30-12-2013 15:10	pay compensation	Ellen	Assistant	200	1900	3

Appendix B – Example event log in format of XES

Table 12: Example event log in format of XES

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7"
xmlns="http://www.xes-standard.org/">
    <trace>
        <string key="concept:name" value="1"/>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Pete"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="decide"/>
        </event>
        <event>
            <string key="org:resource" value="Mike"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="400"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="examine casually"/>
        </event>
        <event>
            <string key="org:resource" value="Sean"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Expert"/>
            <int key="cost" value="400"/>
```

```
            <int key="cid" value="1123"/>
            <string key="concept:name" value="examine thoroughly"/>
        </event>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="3"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="pay compensation"/>
        </event>
        <event>
            <string key="org:resource" value="Pete"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="50"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="register request"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1123"/>
            <string key="concept:name" value="reinitiate request"/>
        </event>
    </trace>
    <trace>
        <string key="concept:name" value="2"/>
        <event>
            <string key="org:resource" value="Mike"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1717"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1717"/>
            <string key="concept:name" value="decide"/>
        </event>
        <event>
            <string key="org:resource" value="Sean"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Expert"/>
```

```
            <int key="cost" value="400"/>
            <int key="cid" value="1717"/>
            <string key="concept:name" value="examine thoroughly"/>
        </event>
        <event>
            <string key="org:resource" value="Pete"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="50"/>
            <int key="cid" value="1717"/>
            <string key="concept:name" value="register request"/>
        </event>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="4"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1717"/>
            <string key="concept:name" value="reject request"/>
        </event>
    </trace>
    <trace>
        <string key="concept:name" value="3"/>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Pete"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="decide"/>
        </event>
        <event>
            <string key="org:resource" value="Mike"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
```

```
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="400"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="examine casually"/>
        </event>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="3"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="pay compensation"/>
        </event>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="50"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="register request"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1718"/>
            <string key="concept:name" value="reinitiate request"/>
        </event>
    </trace>
    <trace>
        <string key="concept:name" value="4"/>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="100"/>
            <int key="cid" value="1900"/>
            <string key="concept:name" value="check ticket"/>
        </event>
        <event>
            <string key="org:resource" value="Sara"/>
            <int key="status" value="2"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Manager"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1900"/>
            <string key="concept:name" value="decide"/>
        </event>
        <event>
            <string key="org:resource" value="Mike"/>
            <int key="status" value="1"/>
```

```
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="400"/>
            <int key="cid" value="1900"/>
            <string key="concept:name" value="examine casually"/>
        </event>
        <event>
            <string key="org:resource" value="Ellen"/>
            <int key="status" value="3"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="200"/>
            <int key="cid" value="1900"/>
            <string key="concept:name" value="pay compensation"/>
        </event>
        <event>
            <string key="org:resource" value="Mike"/>
            <int key="status" value="1"/>
            <date key="time:timestamp" value="2013-12-30T15:10:00+02:00"/>
            <string key="org:role" value="Assistant"/>
            <int key="cost" value="50"/>
            <int key="cid" value="1900"/>
            <string key="concept:name" value="register request"/>
        </event>
    </trace>
</log>
```

Appendix C – Event log structure

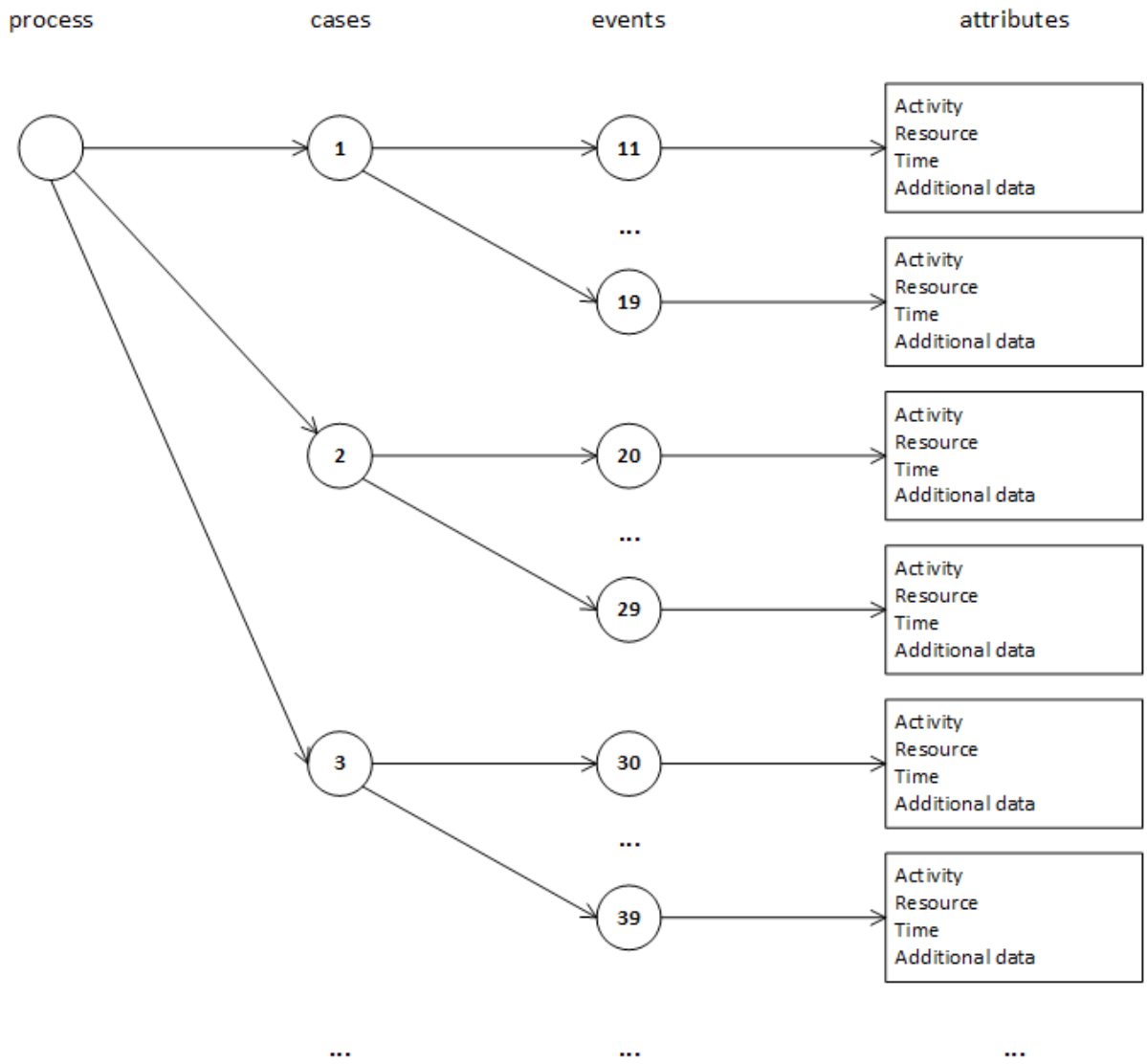

Figure 16: Event logs structure (adapted from [8] p. 100)

Appendix D – XML Schema Definition for RBAC model

Table 13: XML Schema Definition for RBAC model

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- edited with XMLSpy v2014 rel. 2 (x64) (http://www.altova.com) by Taivo Teder
(University of Tartu) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <!-- RBAC element -->
    <xs:element name="rbac">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="subjects" type="subjects_type"/>
                <xs:element name="operations" type="operations_type"/>
                <xs:element name="roles" type="roles_type"/>
                <xs:element name="resources" type="resources_type"/>
                <xs:element name="permissions" type="permissions_type"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!-- Subjects complex type -->
    <xs:complexType name="subjects_type">
        <xs:sequence>
            <xs:element name="subject" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="id" type="xs:string"/>
                    <xs:attribute name="name" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <!-- Operations complex type -->
    <xs:complexType name="operations_type">
        <xs:sequence>
            <xs:element name="operation" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="id" type="xs:string"/>
                    <xs:attribute name="name" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <!-- Roles complex type -->
    <xs:complexType name="roles_type">
        <xs:sequence>
            <xs:element name="role" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="subjects">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="subject" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:complexType>
```

```
                                                <xs:attribute name="refid"
type="xs:string"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:string"/>
                        <xs:attribute name="name" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <!-- Resources complex type -->
        <xs:complexType name="resources_type">
            <xs:sequence>
                <xs:element name="resource" minOccurs="1" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="operations">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="operation"
minOccurs="0" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:attribute name="refid"
type="xs:string"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="values" type="values_type" />
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:string"/>
                        <xs:attribute name="name" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
        <!-- Values complex type -->
        <xs:complexType name="values_type">
            <xs:sequence>
                <xs:element name="value" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <!-- Permissions complex type -->
        <xs:complexType name="permissions_type">
            <xs:sequence>
                <xs:element name="resource" type="resource_type" minOccurs="1"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <!-- Resource complex type -->
        <xs:complexType name="resource_type">
```

```
        <xs:sequence>
            <xs:element name="permission" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="operation" type="xs:string"/>
                    <xs:attribute name="action" type="xs:string"/>
                    <xs:attribute name="role" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="refid" type="xs:string"/>
    </xs:complexType>
</xs:schema>
```

Appendix E – Business process elements in event log in format of XES

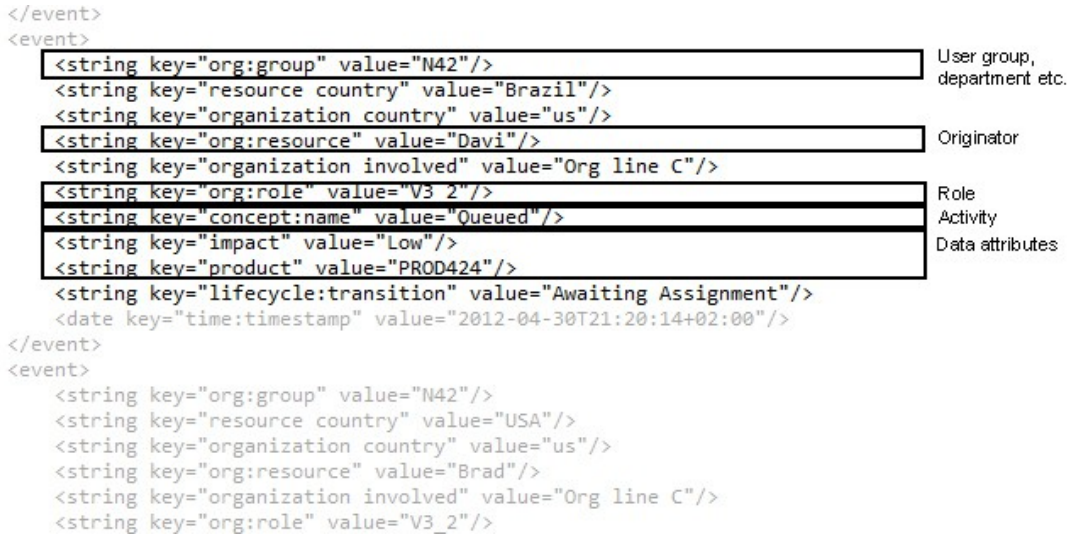

Figure 17: Business process elements presented in the event log in format of XES

Appendix F – RBAC permissions derived from example event log

Table 14: RBAC permissions derived from example event log

Resource (or resource attribute)	Operation	Role
cost	Register request	Assistant
	Check ticket	Assistant
	Examine thoroughly	Expert
	Examine casually	Assistant
	Decide	Manager
	Reinitiate request	Manager
	Reject request	Manager
	Pay compensation	Assistant
cid	Register request	Assistant
	Check ticket	Assistant
	Examine thoroughly	Expert
	Examine casually	Assistant
	Decide	Manager
	Reinitiate request	Manager
	Reject request	Manager
	Pay compensation	Assistant
status	Register request	Assistant
	Check ticket	Assistant
	Examine thoroughly	Expert
	Examine casually	Assistant
	Decide	Manager
	Reinitiate request	Manager
	Reject request	Manager
	Pay compensation	Assistant

Appendix G – An example RBAC model in format of XML

An example RBAC model in format of XML that follows the XSD presented in Table 13.

Table 15: An example RBAC model in format of XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rbac xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="rbac.xsd">
    <subjects>
        <subject id="subject1" name="Pete"/>
        <subject id="subject2" name="Ellen"/>
        <subject id="subject3" name="Sara"/>
        <subject id="subject4" name="Mike"/>
        <subject id="subject5" name="Sean"/>
    </subjects>
    <operations>
        <operation id="operation1" name="decide"/>
        <operation id="operation2" name="pay compensation"/>
        <operation id="operation3" name="examine casually"/>
        <operation id="operation4" name="reinitiate request"/>
        <operation id="operation5" name="check ticket"/>
        <operation id="operation6" name="examine thoroughly"/>
        <operation id="operation7" name="register request"/>
        <operation id="operation8" name="reject request"/>
    </operations>
    <roles>
        <role id="role1" name="Expert">
            <subjects>
                <subject refid="subject5"/>
            </subjects>
        </role>
        <role id="role2" name="Manager">
            <subjects>
                <subject refid="subject3"/>
            </subjects>
        </role>
        <role id="role3" name="Assistant">
            <subjects>
                <subject refid="subject1"/>
                <subject refid="subject2"/>
                <subject refid="subject4"/>
            </subjects>
        </role>
    </roles>
    <resources>
        <resource id="resource1" name="status">
            <operations>
                <operation refid="operation1"/>
                <operation refid="operation2"/>
                <operation refid="operation3"/>
                <operation refid="operation4"/>
                <operation refid="operation5"/>
                <operation refid="operation6"/>
                <operation refid="operation7"/>
```

```
                <operation refid="operation8"/>
            </operations>
            <values>
                <value>3</value>
                <value>2</value>
                <value>1</value>
                <value>4</value>
            </values>
        </resource>
        <resource id="resource2" name="cost">
            <operations>
                <operation refid="operation1"/>
                <operation refid="operation2"/>
                <operation refid="operation3"/>
                <operation refid="operation4"/>
                <operation refid="operation5"/>
                <operation refid="operation6"/>
                <operation refid="operation7"/>
                <operation refid="operation8"/>
            </operations>
            <values>
                <value>200</value>
                <value>400</value>
                <value>100</value>
                <value>50</value>
            </values>
        </resource>
        <resource id="resource3" name="cid">
            <operations>
                <operation refid="operation1"/>
                <operation refid="operation2"/>
                <operation refid="operation3"/>
                <operation refid="operation4"/>
                <operation refid="operation5"/>
                <operation refid="operation6"/>
                <operation refid="operation7"/>
                <operation refid="operation8"/>
            </operations>
            <values>
                <value>1717</value>
                <value>1718</value>
                <value>1900</value>
                <value>1123</value>
            </values>
        </resource>
    </resources>
    <permissions>
        <resource refid="resource1">
            <permission action="" operation="operation1" role="role2"/>
            <permission action="" operation="operation2" role="role3"/>
            <permission action="" operation="operation3" role="role3"/>
            <permission action="" operation="operation4" role="role2"/>
            <permission action="" operation="operation5" role="role3"/>
            <permission action="" operation="operation7" role="role3"/>
            <permission action="" operation="operation6" role="role1"/>
            <permission action="" operation="operation8" role="role3"/>
```

```
        </resource>
        <resource refid="resource2">
            <permission action="" operation="operation1" role="role2"/>
            <permission action="" operation="operation2" role="role3"/>
            <permission action="" operation="operation3" role="role3"/>
            <permission action="" operation="operation4" role="role2"/>
            <permission action="" operation="operation5" role="role3"/>
            <permission action="" operation="operation7" role="role3"/>
            <permission action="" operation="operation6" role="role1"/>
            <permission action="" operation="operation8" role="role3"/>
        </resource>
        <resource refid="resource3">
            <permission action="" operation="operation1" role="role2"/>
            <permission action="" operation="operation2" role="role3"/>
            <permission action="" operation="operation3" role="role3"/>
            <permission action="" operation="operation4" role="role2"/>
            <permission action="" operation="operation5" role="role3"/>
            <permission action="" operation="operation7" role="role3"/>
            <permission action="" operation="operation6" role="role1"/>
            <permission action="" operation="operation8" role="role3"/>
        </resource>
    </permissions>
</rbac>
```

Appendix H – Conformance checking results

Table 16: Validation results of two RBAC models for open and closed cases (first 50 rows)

Role	Permission		Reason
	Operation	Resource	
C_5	Completed\nClosed	product	Not allowed
C_7	Completed\nClosed	product	New role
E_9	Completed\nClosed	product	New role
C_7	Accepted\nIn Progress	product	New role
E_9	Accepted\nIn Progress	product	New role
A2_4	Completed\nCancelled	product	Not allowed
A2_3	Completed\nCancelled	product	Not allowed
E_5	Completed\nCancelled	product	Not allowed
A2_2	Unmatched\nUnmatched	product	Not allowed
A2_3	Unmatched\nUnmatched	product	Not allowed
A2_1	Unmatched\nUnmatched	product	Not allowed
C_6	Unmatched\nUnmatched	product	Not allowed
E_3	Unmatched\nUnmatched	product	Not allowed
E_9	Accepted\nWait	product	New role
C_3	Accepted\nWait	product	Not allowed
E_2	Accepted\nWait	product	Not allowed
C_5	Accepted\nAssigned	product	Not allowed
C_7	Accepted\nAssigned	product	New role
E_9	Accepted\nAssigned	product	New role
C_3	Accepted\nAssigned	product	Not allowed
E_5	Accepted\nAssigned	product	Not allowed
C_7	Queued\nAwaiting Assignment	product	New role
D_1	Queued\nAwaiting Assignment	product	Not allowed
E_3	Queued\nAwaiting Assignment	product	Not allowed
V8_1	Queued\nAwaiting Assignment	product	New role
C_5	Completed\nClosed	org:group	Not allowed
C_7	Completed\nClosed	org:group	New role
E_9	Completed\nClosed	org:group	New role
C_7	Accepted\nIn Progress	org:group	New role
E_9	Accepted\nIn Progress	org:group	New role

A2_4	Completed\nCancelled	org:group	Not allowed
A2_3	Completed\nCancelled	org:group	Not allowed
E_5	Completed\nCancelled	org:group	Not allowed
A2_2	Unmatched\nUnmatched	org:group	Not allowed
A2_3	Unmatched\nUnmatched	org:group	Not allowed
A2_1	Unmatched\nUnmatched	org:group	Not allowed
C_6	Unmatched\nUnmatched	org:group	Not allowed
E_3	Unmatched\nUnmatched	org:group	Not allowed
E_9	Accepted\nWait	org:group	New role
C_3	Accepted\nWait	org:group	Not allowed
E_2	Accepted\nWait	org:group	Not allowed
C_5	Accepted\nAssigned	org:group	Not allowed
C_7	Accepted\nAssigned	org:group	New role
E_9	Accepted\nAssigned	org:group	New role
C_3	Accepted\nAssigned	org:group	Not allowed
E_5	Accepted\nAssigned	org:group	Not allowed
C_7	Queued\nAwaiting Assignment	org:group	New role
D_1	Queued\nAwaiting Assignment	org:group	Not allowed
E_3	Queued\nAwaiting Assignment	org:group	Not allowed
V8_1	Queued\nAwaiting Assignment	org:group	New role
C_5	Completed\nClosed	resource country	Not allowed

Appendix I – Example LTL file

Table 17: Example LTL file with RBAC role-permission assignments as LTL formulas

```
string ate.status;
rename ate.status as attr1;
string ate.cid;
rename ate.cid as attr3;
string ate.cost;
rename ate.cost as attr2;
set ate.EventType;
set ate.Originator;
set ate.WorkflowModelElement;
rename ate.Originator as subject;
rename ate.WorkflowModelElement as activity;
rename ate.EventType as eventType;
formula RP_attr1_decide () := {}
[](((attr1 != "" /\ activity == "decide") -> subject == "Sara"));
formula RP_attr1_pay_compensation () := {}
[](((attr1 != "" /\ activity == "pay compensation") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr1_examine_casually () := {}
[](((attr1 != "" /\ activity == "examine casually") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr1_reinitiate_request () := {}
[](((attr1 != "" /\ activity == "reinitiate request") -> subject == "Sara"));
formula RP_attr1_check_ticket () := {}
[](((attr1 != "" /\ activity == "check ticket") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr1_register_request () := {}
[](((attr1 != "" /\ activity == "register request") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr1_examine_thoroughly () := {}
[](((attr1 != "" /\ activity == "examine thoroughly") -> subject == "Sean"));
formula RP_attr1_reject_request () := {}
[](((attr1 != "" /\ activity == "reject request") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr2_decide () := {}
[](((attr2 != "" /\ activity == "decide") -> subject == "Sara"));
formula RP_attr2_pay_compensation () := {}
[](((attr2 != "" /\ activity == "pay compensation") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr2_examine_casually () := {}
[](((attr2 != "" /\ activity == "examine casually") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr2_reinitiate_request () := {}
[](((attr2 != "" /\ activity == "reinitiate request") -> subject == "Sara"));
formula RP_attr2_check_ticket () := {}
[](((attr2 != "" /\ activity == "check ticket") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr2_register_request () := {}
[](((attr2 != "" /\ activity == "register request") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr2_examine_thoroughly () := {}
[](((attr2 != "" /\ activity == "examine thoroughly") -> subject == "Sean"));
formula RP_attr2_reject_request () := {}
[](((attr2 != "" /\ activity == "reject request") -> ( subject == "Mike" \/
```

```
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr3_decide () := {}
[](((attr3 != "" /\ activity == "decide") -> subject == "Sara"));
formula RP_attr3_pay_compensation () := {}
[](((attr3 != "" /\ activity == "pay compensation") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr3_examine_casually () := {}
[](((attr3 != "" /\ activity == "examine casually") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr3_reinitiate_request () := {}
[](((attr3 != "" /\ activity == "reinitiate request") -> subject == "Sara"));
formula RP_attr3_check_ticket () := {}
[](((attr3 != "" /\ activity == "check ticket") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr3_register_request () := {}
[](((attr3 != "" /\ activity == "register request") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
formula RP_attr3_examine_thoroughly () := {}
[](((attr3 != "" /\ activity == "examine thoroughly") -> subject == "Sean"));
formula RP_attr3_reject_request () := {}
[](((attr3 != "" /\ activity == "reject request") -> ( subject == "Mike" \/
( subject == "Violator" \/ ( subject == "Ellen" \/ subject == "Pete" )))));
```

Appendix J – XES meta model

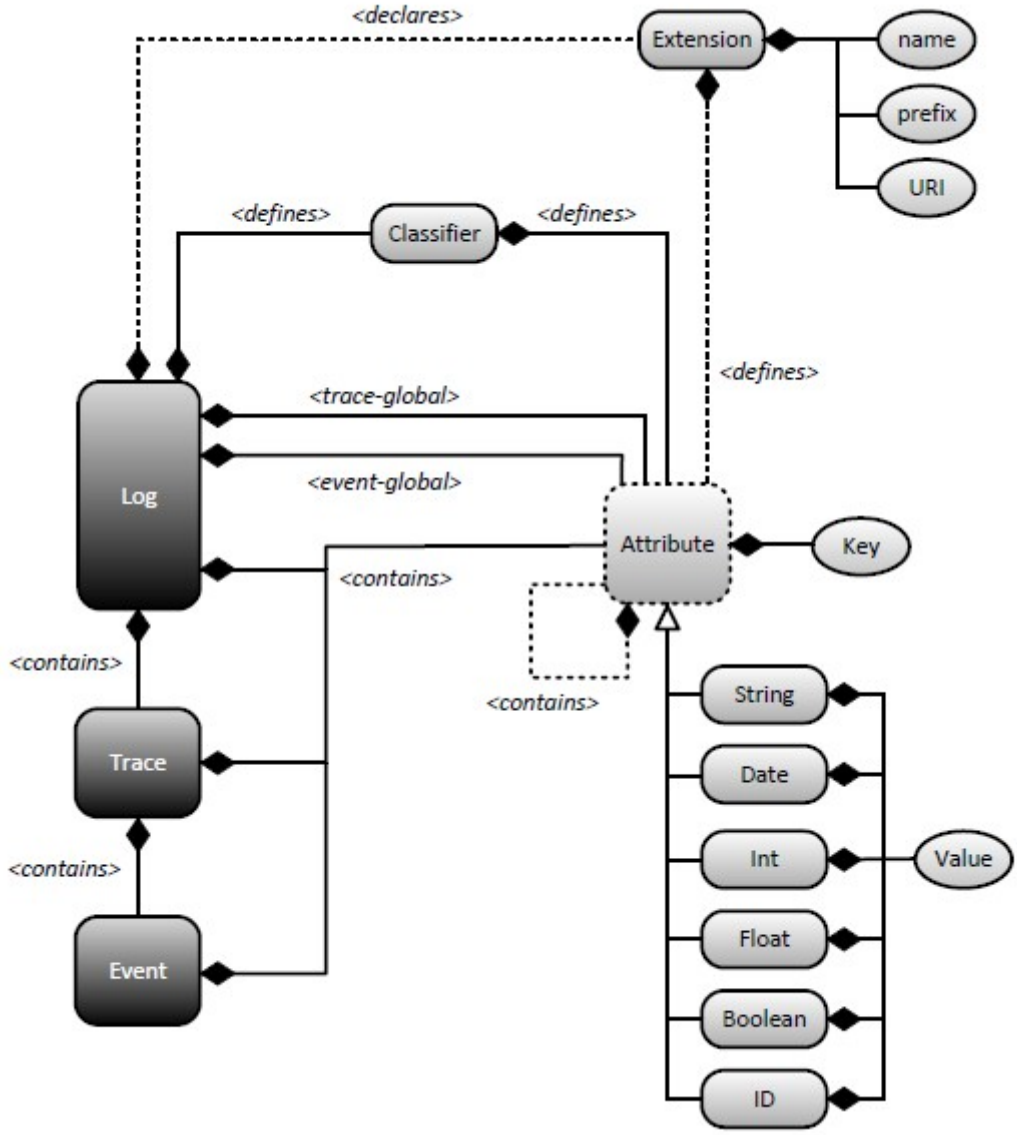

Figure 18: XES meta model (adapted from [21])

Appendix K – Prototype

The prototype source code is submitted with this thesis work in an archive file **RbacModelTool.zip**. The prototype is written in Java language, therefore Java needs to be installed in order to run the application. Also, the prototype source code is maintained in Git repository and is accessible at address: https://bitbucket.org/tefkon/rbac-model-tool. The copy of the Git repository can be cloned into the local Git project with the following command (requires that Git is installed into the computer):

```
git clone https://tefkon@bitbucket.org/tefkon/rbac-model-tool.git
```

Building the project is done through Maven with the command `mvn package` which creates the executable jar-file into the project's `target/` folder.

Non-exclusive licence to reproduce thesis and make thesis public

I, Taivo Teder, (date of birth: 14.04.1989),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Extracting Role-Based Access Control Models from Business Process Event Logs

supervised by Raimundas Matulevičius and Fabrizio M. Maggi,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 26.05.2014