TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik

Lehrstuhl für Sicherheit in der Informationstechnik

Model-based Security Engineering of Electronic Business Processes

A framework for security engineering in the domain of business process management

Jörn Gunnar Eichler

Vollständiger Abdruck der bei der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Florian Matthes

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Claudia Eckert
2. Univ.-Prof. Dr. Helmut Krcmar

Die Dissertation wurde am 27.01.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 11.05.2015 angenommen.

Contents

Kurzfassung

Entwurf, Verwaltung, Ausführung und Analyse von Geschäftsprozessen sind vitale Herausforderungen an Organisationen, die unter dem Titel Geschäftsprozessmanagement (Business Process Management, BPM) zusammengefasst werden. Systeme für das Geschäftsprozessmanagement (Business Process Management Systems, BPMS) erlauben es Managern und Mitarbeitern, die Geschäftsprozesse der Organisation besser zu verstehen, zu entwerfen, zu simulieren und auszuführen. BPMS haben eine wichtige Funktion bei der Automatisierung von Geschäftsprozessen und somit zur Umsetzung elektronischer Geschäftsprozesse.

Geschäftsprozesse sind eng mit den Werten einer Organisation verbunden. Ausspähen, Manipulieren oder Stören von Geschäftsprozessen können diese Werte bedrohen. Trotz dieses Schutzbedarfs werden Sicherheitseigenschaften durch heutige Modellierungssprachen, Methoden und Techniken im Anwendungsbereich von BPM kaum berücksichtigt.

In dieser Arbeit entwickeln wir ein Rahmenwerk für das Security Engineering im Anwendungsbereich von BPM, welches eine Brücke von Geschäftsprozessmodellen zu dem Entwurf angemessener Schutzmaßnahmen und deren Konfiguration schlägt. Unser Vorschlag zielt auf die Einbeziehung von Personen mit geringen Sicherheitskenntnissen ab und stellt flexible Mittel für Anpassungen an die Umgebung sowie die Integration in existierende Entwicklungsprozesse und Werkzeugketten bereit. Unser Rahmenwerk umfasst:

- *Security Engineering Process Model* (SecEPM): SecEPM stellt eine Vorgehensweise und Anleitungen für das Security Engineering elektronischer Geschäftsprozesse von der Identifikation von Schutzzielen bis zur Auswahl und Konfiguration von Schutzmaßnahmen bereit.

- *Security Engineering Modeling Language* (SecEML): SecEML ist eine domänenspezifische Modellierungssprache für das Security Engineering von elektronischen Geschäftsprozessen. Die Sprache ermöglicht die konsistente und zugängliche Erfassung von Arbeitsergebnissen von SecEPM.

- *Workbench:* Eine prototypische Implementierung einer integrierten Arbeitsumgebung wird im Rahmen dieser Arbeit zusammengestellt. Die Arbeitsumgebung unterstützt die Anwendung von SecEPM und der Modellierungssprache SecEML, um sichere, elektronische Geschäftsprozesse zu entwickeln.

Unsere Untersuchungen zeigen den Nutzen des Rahmenwerks, um sichere, elektronische Geschäftsprozesse zu entwickeln. Anhand von systematisch entwickelten Bewertungskriterien können wir zudem die Vorteile unseres Rahmenwerkes gegenüber bestehenden Ansätzen verdeutlichen.

Abstract

Design, administration, enactment, and analysis of business processes are vital challenges to organizations subsumed under the term business process management (BPM). Dedicated business process management systems (BPMS) allow managers and staff to better understand the organization's business processes, to (re-) design, to simulate, and to enact business processes more easily. BPMS play an important role in the automation of business processes implementing electronic business processes.

Business processes are closely connected with assets of the respective organization. Observation, manipulation, and disruption of business processes might threaten these assets. In spite of this need, security is weakly supported by today's BPM languages, methods, and techniques.

In this thesis we develop a security engineering framework in the domain of BPM that bridges the gap between business process models on one side and the design of proper controls and their configuration on the other side. Our proposal aims at capacitating people with low security background, providing flexibility as well as means for customization, and offering integration possibilities into existing development processes and tool chains. Our framework comprises:

- *Security Engineering Process Model* (SecEPM): SecEPM provides guidance for security engineering of electronic business processes from the identification of security goals to the selection and configuration of controls.
- *Security Engineering Modeling Language* (SecEML): SecEML is a domain-specific modeling language for security engineering of electronic business processes. The language allows for a consistent and accessible capturing of work products from SecEPM.
- *Workbench:* A prototypical implementation of an integrated security workbench is provided. It supports the application of the security engineering process model SecEPM and the modeling language SecEML to develop secure electronic business processes.

Our study indicates the utility of our framework in order to develop secure electronic business processes. Furthermore, we lay out advantages of our framework over existing approaches applying systematically derived evaluation criteria.

1. Introduction

1.1. Motivation

Business processes are at the heart of every organization. They are the way organizations do their work: the set of activities carried out to accomplish a defined objective. Therefore, the design, administration, enactment, and analysis of business processes—subsumed under the term business process management (BPM)—are vital challenges to organizations. The increasing importance of information systems to organizations and the development and practice of BPM mutual influence each other. Information systems make the complex task of BPM possible while the application of information systems increase the demand to tackle business processes explicitly. [Dav05, Wes07]

To accomplish a given objective of a business process in an efficient and effective manner, the interplay between people and organizational resources, such as information systems, needs to be well aligned. BPM and accompanying systems are seen as an important facilitator for this alignment. Business process management systems (BPMSs) allow managers and staff to better understand the organization's business processes, to (re-) design, to simulate, and to enact business processes more easily. This enables organizations to become an adaptive enterprise: They are able to react faster to environmental and market changes and to proactively innovate their products and services. [Wes07, Tal08]

Consequently, BPM supported by information systems and technology has seen an ongoing development in the last decades. With respect to modeling languages and techniques a multitude of approaches have been introduced ranging from early Petri nets [Pet62] to the standardized XML Process Definition Language (XPDL) [Wor08], from business oriented languages like Event-driven Process Chains (EPC) [Sch02] to executable languages like the Business Process Modeling and Notation (BPMN) [Obj11a]. In parallel, systems to support BPM developed from simple information systems to capture and administrate process models to feature-rich BPM suites that also support simulation, execution, and controlling of business process instances.

Thus, the ability for organizations to manage business processes is well supported by today's software industry. Dedicated modeling languages and techniques as well as BPMS help organizations to automate, adapt and monitor business processes according to their needs with a minimum of technical and engineering skills. BPMS are largely applied in industry, the worldwide market for BPMS licenses is estimated at $ 9 billion[1] in 2010, and the importance of BPM is expected to increase in the next years even further. [Ric11, FS11, Cap12]

[1] This is almost one third of the worldwide market for operating system software which is estimated at $ 30 billion in 2010 [CD11].

Business processes are closely connected with assets of the respective organization. Observation, manipulation, and disruption of business processes might threaten these assets or even the existence of the organization itself. Thus, mitigation of these threats and security of business processes ought to be of high importance for every organization [HH06, Neu09, MB11].

In spite of this need for security, BPM languages and techniques provide only little support to express the security needs of an organization with regard to the modeled business process or the controls that have been established to allow for a secure process execution [WMM08, BH13]. With the success of BPM initiatives and the deployment of more and more BPMS within organizations a growing need has been declared to provide such means for BPM and its systems [KDK00, NKB06].

Several approaches have been developed to address this need for security in the domain of BPM. Techniques have been described to analyze security properties of given business process models (e.g., [SLS06, ACPP11]), languages and dialects have been developed to annotate existing business process models with security properties or to specify those properties separately (e.g., [NH08a, WMS+09, RFMTP11]). Only few proposals for a systematic procedure to develop secure electronic business processes have been published that failed to yield broad acceptance until today (e.g., [RK04, HH06, HB09]).

1.2. Problem Statement

Security engineering is about building systems to remain dependable in the face of malice, error, or mischance [And08]. As a discipline it is still in its infancy [Eck14]. Today, mostly general top-down process models from the software engineering domain are augmented with security-specific techniques and methods that are only loosely integrated. As a result, security engineering is focused on systems (not business processes), applied selectively with regard to security properties and activities, and designed to be applied by security professionals (cf. [And08, Gee10, HHJS11, For12]).

With regard to security engineering in the domain of BPM we focus on three major issues:

- Security nonprofessionals deciding and implementing security
- Heterogeneity of business process engines
- Business process environmental heterogeneity

The selection of these three major issues is inspired by the motivating observations described above: acknowledged need for security in the domain of BPM, few proposals for a systematic procedure to develop secure electronic business processes, and lack of acceptance of these proposed approaches today. The issues focused in this thesis are discussed separately in literature but not coherently considered by these proposals[2].

Two further issues have been disregarded that have been mentioned in this context as well: First, the lack of transparency with regard to costs and benefits of security investments

[2] References are provided in the following paragraphs to enhance clarity. Details with regard to the existing approaches are presented in chapter 2.

(e.g., [NKB06, IRRG09]). This important issue addresses the economic rationale of security decisions and not the application of a systematic procedure to develop secure electronic business processes. Second, insufficient (visual) representation of security aspects in existing BPM techniques, languages, and tooling (e.g., [MBS12]). This aspect has been addressed by several publications and respective results are complementary to this thesis (e.g., [NH08a, WMS+09, RFMTP11]).

Security nonprofessionals implementing security

A gap between organizational business aspects and information technology can be observed in many companies. BPM tries to narrow this gap between organization and technology, thus allowing an organization to become adaptive and agile. Important contributing factors provided by BPM are methods, standardizations, and tooling. Methods support the transfer of knowledge gained during the application of BPM and enables other organizations and practitioners to generate repeatable results. Standardization allows for a common language, facilitates the knowledge transfer further, and renders tool independence possible. Different vendors are able to offer specialized tooling to support the process models already existing in the organizations. Business process experts together with domain experts are getting the possibility to support more and more aspects of the business process life cycle. [SF03, Wes07, WH12]

This advantage from the point of view of the BPM protagonist becomes an issue with respect to security. Business process experts are mostly security nonprofessionals. As they analyze, design, and configure business processes they decide implicitly about the security needs of the organization and the controls to be deployed to secure the business process execution. Therefore, security nonprofessionals decide about security aspects of an organization. Security analysis and design are commonly seen as challenging task. If security nonprofessionals are deciding and implementing security aspects of a business process without proper guidance it is very likely that relevant security needs will not be identified, threats will not be adequately mitigated, controls will be improperly selected, and implementations will be wrongly configured. [NKB06, And08, Neu09, MBS12]

Heterogeneity of business process engines

Standardization of modeling languages and techniques enables vendors to offer tooling with common interfaces to support BPM in organizations. Business process engines from different vendors address different needs and environments. Naturally, the approaches to configure and enforce controls in business process executions differ between vendors. While some try to hide implementation details from users of their tools, others do provide interfaces to allow users to manage every detail on their own (cf. [VdA04, Sil06, Gö11]). Support for (implementation related) security standards like the Web Service Security standard [Org06] differ largely between the products as does the configuration of the respective artifacts (cf. [BB08, IRRG09]). Because the accepted BPM modeling languages and techniques do not provide elaborate support to express security properties explicitly, there is no common interface to specify all relevant security properties for BPM stakeholders [WMS+09].

Effects of this heterogeneity of business process engines with regard to provision and configuration of controls for secure business process execution are twofold. Users of BPM tools have

to learn and understand the controls provided by their tools and have to configure or implement them tool-dependent correctly. Knowledge transfer and validation becomes difficult in this situation as not only general security expertise is necessary but an in-depth knowledge of the respective tooling. In this circumstances, wrongly configured or implemented controls are very likely and might degrade the security of the business process in question. Further more, interoperability is at danger since common interfaces are not guaranteed and the change of the BPM tooling in use becomes difficult.

Business process environmental heterogeneity

BPM is applied in very different organizations with different approaches. Organizations from the military sector are applying BPM as well as companies offering financial or consulting services. Also, the role of the BPMS and the functionality actually used is diverse [VdAtHW03, Pal07, PCBV10]. Engineering methodologies and development processes applied depend on organizational habits and policies [WH12]. Approaches for security engineering that are restricted to a specific development process (e.g. traditional top-down approaches) do not fit to the needed flexibility in the BPM domain and are not likely to be adopted.

1.3. Objective and Approach

The objective of this thesis is to develop a security engineering framework in the domain of BPM that bridges the gap between business process models on one side and the design of proper controls and their configuration using the means provided by the respective business process engines and execution environment on the other side. It has to capacitate people with low security background to profit from the framework. Flexibility in it's application is necessary to allow for a customization to the needs of an organization. The possibility to integrate the framework into existing development processes and tool chains is necessary to protect existing investments.

In order to strive for our objective, we address the following research questions:

1. What requirements does a security engineering process model for electronic business processes place that copes with the prominence of security nonprofessionals, heterogeneous business process engines, and environmental heterogeneity in the domain of BPM and how could they be met?

One main part of the security engineering framework developed in this thesis is a security engineering process model that addresses the issues introduced in the problem statement.

Before we develop such process model, we have to investigate foundational aspects first: What are important concepts with regard to electronic business processes, their life cycle, and representation? What are the principles of model-driven engineering (MDE) and model-driven security (MDS)? Analogous, a consistent terminology covering core concepts for information technology (IT) security and security engineering is a foundational cornerstone. Another cornerstone are concepts and methods available for design, construction, and adaptation of development methods and techniques provided by the method engineering (ME) discipline. Second, we need to identify relevant approaches in order to reuse existing knowledge, analyze their shortcomings, and differentiate our approach.

To actually develop a suitable process model, we have to analyze requirements and develop design strategies that address the issues introduced in the problem statement. The description of structure and key entities of a corresponding process model addresses the main aspect of this research question. Prototypical tool support and an exemplary integration of the security engineering process model is needed to provide tangible artifacts that allow for an application of the process model and evaluate its properties.

2. What requirements does a domain-specific modeling language (DSML) place that supports the elaboration of main work products of the developed security engineering process model and how could they be met?

The security engineering framework developed in this thesis encompasses a DSML in order to support the creation, validation, and analysis of main work products of the process model. Hence, requirements and design strategies for such DSML have to be analyzed and depicted that match with the work products of the process model. The definition and elaboration of the metamodel as well as the concrete syntax for the DSML contribute similarly to the research question. In correspondence with the process model, a prototypical implementation of tooling to work with the DSML applying MDE techniques and practices substantiates the result.

3. What observations do we get applying the framework developed in this thesis?

The application of the security engineering framework developed in this thesis allows for a discussion of its properties. Especially, observations with regard to the issues stated in the problem statement are relevant for the validation of the framework and the contributions of this thesis. Therefore, we have to derive analysis criteria for an exemplary study from the issues stated in the problem statement. An exemplary study has to demonstrate the application of the framework. Applying the analysis criteria we are able to discuss our observations systematically.

Our research approach for this thesis rests on the design science paradigm, using a current proposal from Peffers et al. as pragmatic framework [PTRC07]. Design science has been introduced into research on information systems in the 1990s (e.g., [NJC90, MS95]). A widely accepted conceptional framework for design science in information systems research has been presented by Hevner et al. [HMPR04]. Peffers et al. complemented this and other prior research providing a pragmatic procedure for carrying out design science research [PTRC07].

Design science in general is contrasted with other paradigms focusing constructive and artificial aspects: Whereas natural and social sciences try to understand reality, design science attempts to create useful artifacts [Sim96]. Design science in information systems research is commonly understood as to create and evaluate IT artifacts [HMPR04] and addresses any designed object with an embedded solution to an understood research problem [PTRC07].

In order to carry out design science research, Peffers et al. propose a procedure comprising six activities (cf. [PTRC07]):

1. *Problem identification and motivation:* A specific research problem is identified and the value of its solution is justified. This activity requires knowledge of the state of the problem and the relevance of its solution.

2. *Define the objectives for a solution:* The objectives of a solutions are inferred from the problem definition respecting the boundaries of the environment. Resources required include knowledge of the state of problems and current solutions.

3. *Design and development:* The artifact in question is created. This activity includes determination of desired properties, its architecture, and creation of the actual artifact. Knowledge to bear in a solution is needed for this activity.

4. *Demonstration:* The use of the artifact to solve one or more instances of the problem is demonstrated. Resources required for the demonstration include effective knowledge of how to use the artifact.

5. *Evaluation:* How well the artifact supports a solution to the identified problem is observed and measured. This includes the comparison of the objectives with actual results and requires knowledge of relevant analysis techniques. At the end of the activity a decision on further iterations of the other activities can be made.

6. *Communication:* The results of the research are communicated to researchers and other relevant audiences including the problem, its importance, the artifact, and its utility. This activity requires knowledge of the disciplinary culture.

This process is presented in sequential order; however there is no expectation to follow the order as it is noted here. Peffers et al. acknowledge that research might start at almost any step and proceed as needed in the research process. [PTRC07]

The results of these activities are represented within the thesis as follows: The identification of the problem, the motivation of the utility of a solution, and the objectives for a solution are presented in sections 1.1 and 1.2 as well as in this section. Chapter 2 supplements these results in order to provide necessary background on the domains addressed in this thesis as well as existing solution approaches. The design and development of the artifacts is presented in chapters 4 and 5. Design and development is illustrated using an example depicted in chapter 3. Chapter 6 demonstrates the application of the artifacts and analyzes properties of the solution. Results of this research effort have been communicated presenting selected aspects on international conferences and other venues (cf. [Eic11a, Eic12a, Eic12b, EFL12]) and are presented considerably reviewed, detailed, and enhanced with this thesis. Furthermore, parts of the results will be provided to practitioners as technical artifacts.

1.4. Contributions

In the context of this thesis we envision a collaboration of security experts, business process experts, and domain experts that leverages the ideas of BPM to allow for an adaptive enterprise and integrates security as a first class citizen. Therefore, we propose a model-based security engineering framework in the domain of BPM that bridges the gap between business process models and properly selected and configured controls for business process engines and the execution environment of that business process. The framework developed in this thesis

supports the Analysis and Design as well as the Configuration phase of the business process life cycle.

Three ideas guide our contributions: The first idea is to specialize general security engineering approaches for the purpose of security engineering in the domain of BPM. With specialization we aim at detailing necessary activities in order to lower the skill set necessary to complete them. The second, complementing idea is to capture necessary security related facts and decisions taken in the course of the security engineering process using a DSML. This allows for technical support in the course of the engineering process, simplified validation of the results, and translation into process or tooling specific deployment artifacts. The third idea is to separate the concerns or the knowledge necessary to realize security-related engineering activities. For this purpose, security experts initially provide a setup for the security engineering process that is tailored to the needs of an organization. Based on this setup business process experts are able to realize most of the activities in the course of an actual project while security experts might validate results in a simplified manner. Developers or technical domain experts provide translations for the design decisions taken by business process experts and security experts in order to implement a secure configuration of an individual business process engine and its execution environment.

To realize these ideas, we contribute in this thesis a framework for security engineering in the domain of BPM comprising:

- *Security Engineering Process Model (SecEPM):* The SecEPM provides guidance for the security engineering from the identification of security goals to the selection and configuration of controls. It allows for an integration in different engineering or development process models and can be tailored to the needs of the organization.

- *Security Engineering Modeling Language (SecEML):* SecEML is a DSML for security engineering of business processes. The language allows for consistent and accessible capturing of work products of executed security engineering activities. Based on SecEML models, results of the security engineering process can be validated and necessary deployment artifacts can be generated.

- *Workbench:* A prototypical implementation of a workbench is provided. It supports the application of the engineering process model SecEPM and the modeling language SecEML to develop secure electronic business processes.

An overview of the framework is provided in figure 1.4.1. Grey squares represent parts of the framework contributed in this thesis. White squares depict external parts that are used by or integrated into contributed parts. Light grey squares show parts that contain external and contributed content.

Key entities of SecEPM are organized using three categories: roles, work products, and activities. Together they address the question: “Who (role) is doing what (activity) with which result (work product) in order to secure an electronic business process?” Additionally, SecEPM provides guidance artifacts including templates, checklists, examples, reference materials, and guidelines. Guidance artifacts explain how to fulfill one or more given task of an activity.

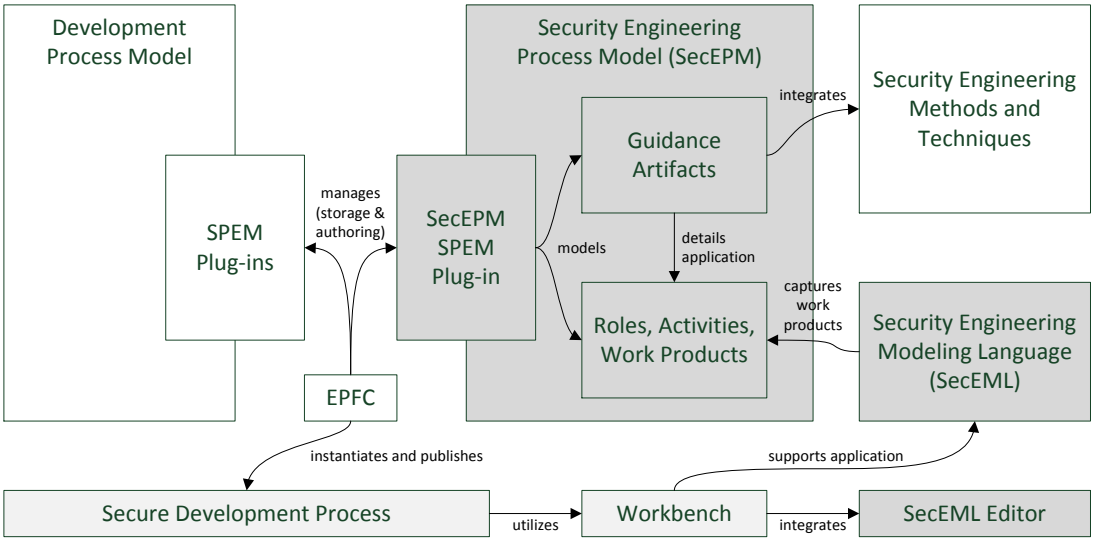

Figure 1.4.1.: Overview of the contributed framework for security engineering

Generally, they are based on existing security engineering methods and techniques. In order to integrate SecEPM into an development process model, SecEPM is provided as plug-in applying the Software & System Process Engineering Model (SPEM) [Obj08]. Utilizing the Eclipse Process Framework Composer (EPFC), method content from SecEPM as well as other development process models can be seamlessly authored and stored as well as a tailored security engineering or security-integrated development process can be instantiated and published.

SecEML is our proposal for a DSML capable of capturing work products of SecEPM and to supply a basis for the validation of work products and their transformation into deployment artifacts. An integrated workbench is provided that supports the application of SecEML. It includes an editor for work products modeled using SecEML as well as an editor for business process models and facilities to validate those models in combination.

1.5. Structure of the Thesis

This thesis is divided into seven major chapters (cf. figure 1.5.1). After this introductory chapter we provide background information in chapter 2. Since we address the security of business processes with our framework, we discuss general concepts of BPM with a focus on design, modeling, and configuration of business processes first. Furthermore, we introduce concepts of MDE and MDS which form a background to define our DSML. As a basis of the development of our process model, concepts and techniques from the ME domain are introduced and explained. Foundations of security terminology used in this thesis and its application to security engineering is provided next. As many terms and concepts in the domain of IT security are not generally agreed upon, we discuss different proposals and define a coherent terminology in alignment with our purposes. A presentation of related work presents the state

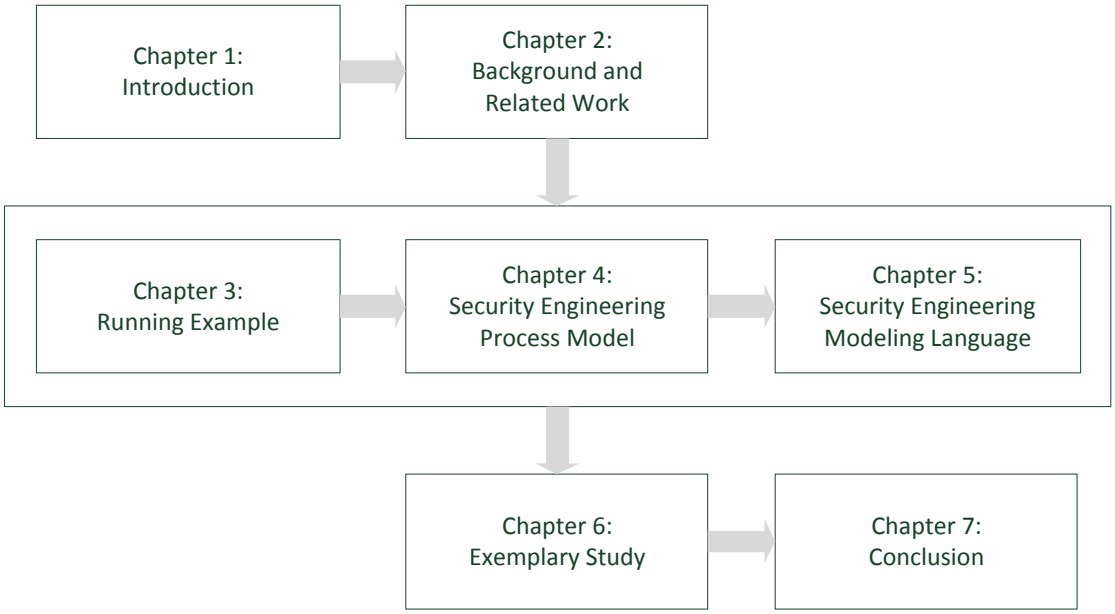

Figure 1.5.1.: Thesis structure

of research with regard to the topics addressed in this thesis and positions our contributions with regard to existing approaches.

The chapters 3, 4, and 5 detail the framework presented in this thesis. Chapter 3 introduces a running example to allow for a better understanding of the following chapters. It presents a small excerpt of a real-world business process including its background in the logistics domain. The business process model is presented and commented for later reference.

We develop SecEPM—our proposal for a security engineering process model in the domain of BPM—in chapter 4. Based on the terminology given in the previous chapters, we define the scope for SecEPM and its requirements to align our process model with the objective of this thesis. In the following, we introduce constituent entities and explain their significance for the process model. Presentation of the activities of SecEPM are at the center of the chapter. We detail constraints, responsibilities, as well as input and output relations for the activities and give exemplary guidance to apply the activities using the business process model introduced in chapter 3. The provisioning of tool support and the integration of our process model into existing development process models finalizes the chapter.

Chapter 5 is dedicated to the development of SecEML, our proposal for a modeling language supporting the application of SecEPM. Based on our process model presented in chapter 4, we elicit requirements for the modeling language and depict our design approach. A description of the modeling language follows presenting abstract and concrete syntax of the language. We discuss capturing of engineering artifacts from SecEPM first to analyze the application context of our modeling language. Second, necessary interactions with existing business process engines are examined to apply SecEML in tool chains utilizing those engines. We close the chapter describing design and implementation aspects of the workbench.

An exemplary study demonstrates the application of our framework in chapter 6. Analysis criteria are defined with regard to the research objective and the problem statement stated earlier in this introduction. SecEPM and SecEML are applied, results are analyzed as well as compared with existing approaches, and benefits, potential issues and shortcomings are discussed. The final chapter 7 summarizes the contributions of this thesis and outlines directions and research topics of future work to further improve our framework.

2. Background and Related Work

2.1. Introduction

This chapter introduces background and related work in order to tackle the first two research questions in the following chapters well founded:

1. What requirements does a security engineering process model for electronic business processes place that copes with the prominence of security nonprofessionals, heterogeneous business process engines, and environmental heterogeneity in the domain of BPM and how could they be met?

2. What requirements does a DSML place that supports the elaboration of main work products of the developed security engineering process model and how could they be met?

The chapter provides necessary knowledge for the state of problems and current solutions as needed for the definition of the objective following the research approach of our thesis (cf. section 1.3). Therefore, it introduces concepts and terms of this thesis, in particular considering BPM, ME, MDE, and security engineering. Additionally, we present and discuss related work with regard to the research objective and topics addressed in this thesis.

Section 2.2 introduces BPM concepts and terminology. Since this thesis contributes a framework for security engineering of business processes, we discuss BPM with a focus on design, modeling, and configuration of business processes. We depict primary concepts and methods for the design, construction, and adaption of development methods stemming from the ME discipline in section 2.3.2. As foundation of our modeling language SecEML we present in section 2.3 principles of MDE. The subsequent section 2.4 provides security-related terminology that will be used in this thesis. Since many terms and concepts in the IT security domain are not generally agreed upon and often somehow blurred we discuss different proposals and define a coherent terminology in alignment with our research objective. A presentation of related work in section 2.5 allows for a positioning of the contributions of this thesis with regard to current research.

2.2. Business Process Management

2.2.1. General Terminology

As the prominent researcher Thomas D. Davenport puts it, business processes are (simply) the way *"an organization does its work—the set of activities it pursues to accomplish a particular objective for a particular customer"* [Dav05, p. 2]. More precisely, a definition from Weske

includes also environment, logical relations, and constraints between the activities to be executed:

Definition 1. "A *business process* consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal." [Wes07, p. 5]

Management of business processes includes not only their representation but also all activities to support the whole life cycle as well as necessary terminology and methodology to cope with business processes.

Definition 2. "*Business process management* includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes." [Wes07, p. 5]

Automation of business processes is one of the cornerstones of BPM. As BPM inherited this aim from its predecessor workflow management (WFM), an automation of a business process is traditionally called a workflow [VdAtHW03] or an electronic business process [Rit99]. As business processes increasingly are supported by IT systems for automation, this distinction is not longer used by many authors (e.g., [KLL09]). Prevalent is a differentiation with regard to the level of a business process. Common levels are the organizational, the operational, and the implementation level [Wes07]: Organizational business processes encompass high-level processes, typically specified in textual form by their inputs, outputs, expected results, and dependencies on other organizational business processes. For operational business processes activities and their relationships are specified, but implementation aspects are (partly) disregarded. With respect to implemented business processes information on execution of activities and the technical and organizational environment is defined. We address business processes at the implementation level in this thesis and therefore omit the level. Only in case of possible ambiguities the level of a business process will be stated explicitly. To emphasize the automation of a business process we will call it an electronic business process.

BPM relies on business process models to manage business processes. Each business process model acts as a blueprint for a business process instance which forms a one-to-many relationship between a business process model and its instances.

Definition 3. A *business process model* consists of a set of activity models and execution constraints between them. A *business process instance* represents a concrete case in the operational business of an organization, consisting of activity instances. [Wes07]

If no ambiguity is possible we will use the term business process to refer to either a business process model or an instance. The enactment or execution of a business process is to be read as the creation of a business process instance from a business process model and the enactment of its activity instances following the execution constraints defined between them.

2.2.2. Business Process Life Cycle and Supporting Systems

The business process life cycle describes the various phases in support of business processes. Several generic life cycle models have been proposed [KLL09]. The overlapping between the

different models is large. Hence we stick to Weskes terminology for the description of the phases (figure 2.2.1, cf. [Wes07]). Generally, the life cycle is iterated multiple times in order to optimize the support of the business process in question [Krc10].

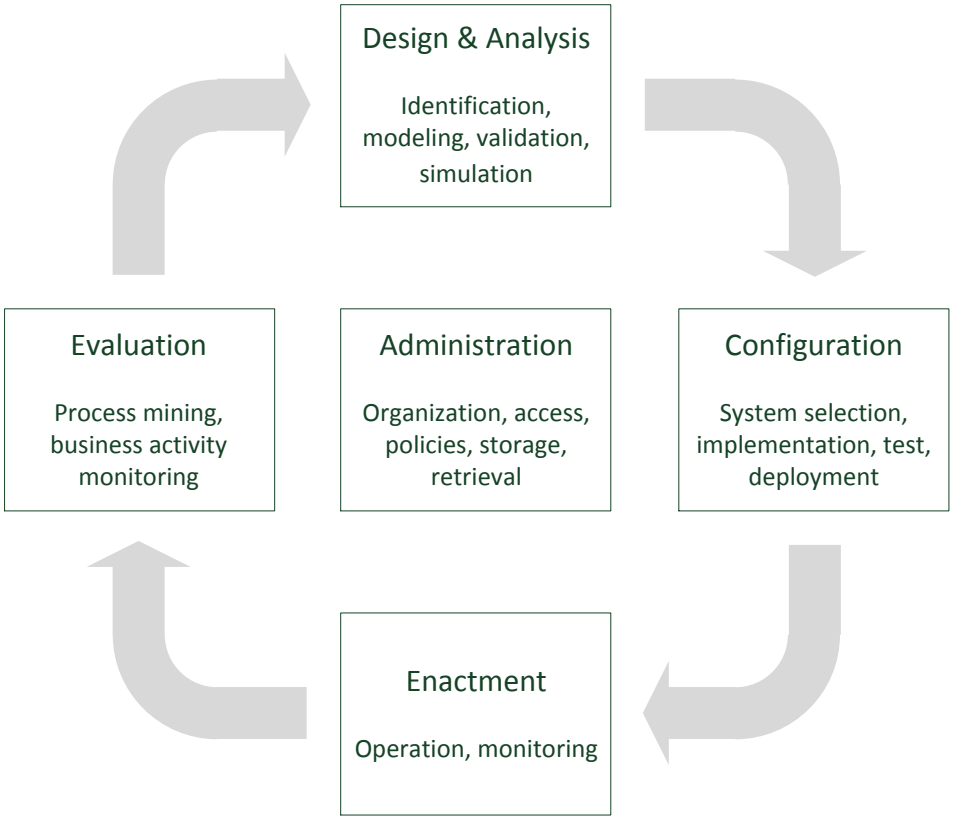

Figure 2.2.1.: Business process life cycle [Wes07, p. 12]

The *Design and Analysis* phase starts with surveys on business processes and their organizational and technical environment. Business processes are identified, reviewed, and modeled. Validation, simulation, and verification techniques are used to check the correctness of the business process models and to identify undesired runtime behavior.

The implementation of a business process is done in the *Configuration* phase. There are multiple ways to implement a business process starting with manual procedures up to the utilization of a dedicated business process management system. In case of electronic business processes, the business process model is enhanced with technical information that allow for the execution of the process with the means provided by the organizational IT infrastructure. Additional implementation work is executed as necessary, e.g. integration of existing software systems.

The *Enactment* phase encompasses the actual execution of business process instances. Execution data is gathered, typically using log files or databases. Administrative monitoring components analyzing the execution data allow for the supervision of business process instances and their accordance to the respective business process model.

Evaluation of executed business process instances and improvement of business process models are at the center of the *Evaluation* phase. Business activity monitoring and process mining techniques aim at identifying the quality of the business process models and the adequacy of the execution environment.

The *Administration* is not a phase but encompass the necessary organizational and managerial duties for all phases to allow for an efficient BPM. Structured storage and capable retrieval of artifacts regarding business process models and information on instances as well as the organizational and technical execution environment need to be taken into account.

The development of electronic business processes is subsumed as a part of general business process management or engineering methodologies and its descriptions remain mostly—from an engineering point of view—rather general (e.g. [Cha95, Wes07]). The development is structured in a sequence of phases: In a *Survey* phase the project goals are defined, the team is established, and necessary information is gathered. The business processes are modeled, validated, and the organizational and technical environment is analyzed in the *Design* phase. The technological platform on which the business process will be enacted is chosen in the *Platform Selection* phase. A following *Implementation and Testing* phase configures the IT systems necessary for the enactment, implements necessary components, and tests the electronic business process. In the *Deployment* phase, the electronic business process is transferred to target environment. The development phases can be mapped onto the business process life cycle phases: Survey and Design development phases onto the Design and Analysis life cycle phase, the Platform Selection, Implementation and Testing, and Deployment development phases onto the Configuration life cycle phase.

More detailed development processes and methodologies for electronic business process reflect the technical infrastructure and/or the architectural paradigms that are envisioned by the authors. Beside specific technical or architectural constraints those processes and methodologies resemble general software development processes and methodologies (cf. [WGHS99] as example for a development process targeting traditional workflow systems and [PvdH07, DRGRdGP09] targeting service-oriented architectures). Furthermore, BPM augmentations available for existing software development process models can be seen as supplemental indication for this observation (e.g. [JB06] and the IBM RUP for SOMA plug-in[1]). Therefore, we adopt the perspective that adequate development processes and methodologies for electronic business processes are grounded on general software development processes and methodologies that are enriched with special practices or considerations with regard to the automation of business processes.

Software systems for the coordination of activities involved in business processes and their management are called business process management systems (BPMS). Figure 2.2.2 shows a general BPMS architecture using the Fundamental Modeling Concepts (FMC) [KGT05] notation. The *Business Process Modeler* component is used for creating business process models and administrating them. Business process models are stored in the *Business Process Repository*. The *Business Process Engine* (also known as business process execution engine) is responsible for creation and controlling of the actual execution of business process instances. It uses the business process models from the repository for this purposes and logs its activities

[1] `http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/`

in the *Process Log*. The *Business Activity Monitoring* component is used for evaluation and improvement of the business process models. The *Business Process Environment* represents the organizational and technical infrastructure and environment the business processes are executed in.

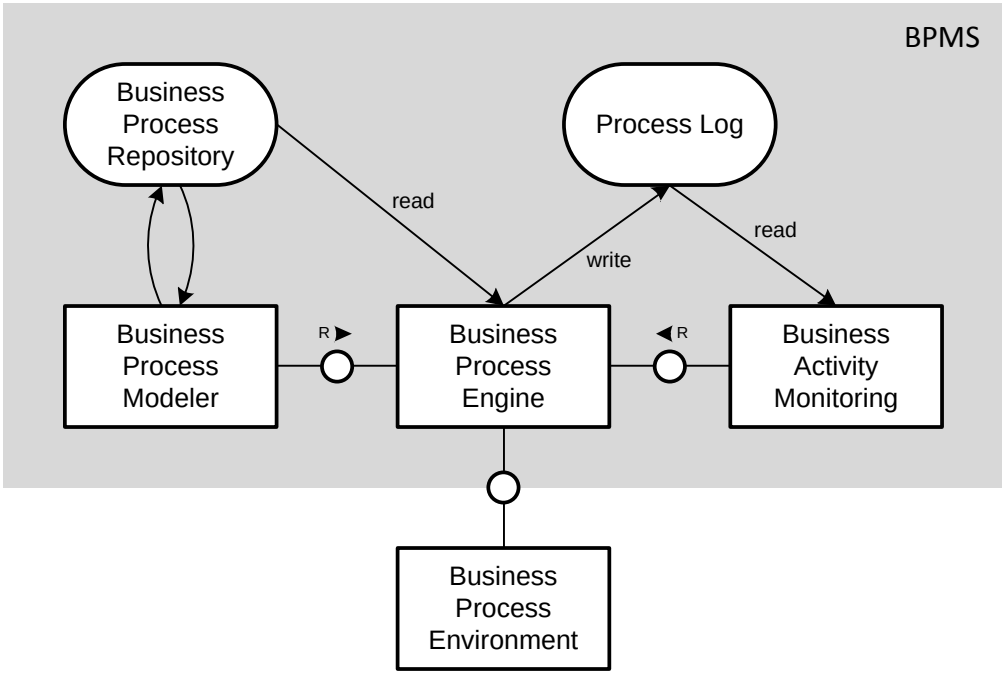

Figure 2.2.2.: Business process management systems architecture (FMC block diagram, adapted from [Wes07, Wol10])

2.2.3. Business Process Modeling

Business process modeling is a key activity in the Design and Analysis as well as the Configuration phase of the business process life cycle. It aims for the explicit representation of a business process in terms of a business process model using a process or modeling language [VdAtHW03]. Business process modeling has a long tradition. Therefore, it is supported by a variety of modeling approaches including different languages, techniques, and tools. The different approaches show different strengths and weaknesses, such as a trade-off between expressibility, ease of use, and analysis complexity. Some approaches offer rich syntax to express different business activities and their relationships while others provide few modeling entities with efficient model analysis capabilities.

Several classifications for existing business process modeling approaches have been proposed, highlighting formal foundation, syntactical richness, industrial or academic background, standardization and other criteria [VdAtHW03, AS04, LS07, KLL09, RRIG09]. A pragmatic criterion is the support of different phases of the business process life cycle as proposed by Lu et al. [KLL09]: Graphical approaches enable users to express business processes in

a diagrammatic way and generally allow for a good comprehensibility in the Design and Analysis phase. Interchange standards facilitate the portability of business process models across different BPMS especially in the Configuration phase. Execution-focused approaches address the automation of the business process enactment and the support of the Enactment phase. Approaches that are directed towards diagnosis provide administrative and monitoring capabilities for the Evaluation phase.

The BPMN standard published by the Object Management Group (OMG) in its version 2.0 provides the broadest support with regard to the criterion of supporting different phases of the business process life cycle [Obj11a]: BPMN centers around a rich graphical syntax that is specifically designed to support good comprehensibility. It provides an interchange format that can be used to exchange business process models (both domain model and diagram layout) between different tools. It specifies (informally) execution semantics and a mapping to the Web Services Business Process Execution Language (WS-BPEL) [Org07], a common language for business process execution. Furthermore, recent publications demonstrate the feasibility of the formalization of the execution semantics using graph rewrite rules to allow for a formally founded reference implementation [DVG10, VGD11]. Diagnosis is facilitated by auditing and monitoring capabilities that leverage the extensibility feature of BPMN. The industrial background, its origin in the BPM domain, and the widespread tool support and application [WH12] support the choice of BPMN as a modeling language for business processes in this thesis.

A business process can be modeled as private (executable or non-executable) business process, public process, and choreography using BPMN. A private business process defines the flow of activities within a specific organization, a public process represents the interactions between a private business process and other processes or participants, including only those activities that are used to communicate to the participants. In contrast to those business processes, a choreography formalizes the way participants coordinate their interactions and focuses on the exchange of information.

Although BPMN is a complex language providing more than 60 distinct graphical elements, BPMN offers a basic set of 15 elements clustered in five categories that provides all necessary means to model basic business processes (cf. figure 2.2.3). The five categories and their most frequently used elements are:

- *Flow Objects* are used to construct a business process, including Events, Activities, and Gateways. *Activities* represent work an organization performs. An Activity can be atomic or non-atomic (compound). *Gateways* are used to control divergence and convergence of Sequence Flows in a business process. Thus, it determines branching, forking, merging, and joining of paths. *Events* represent something that "happens" during the course of a business process execution. The three types of Events are Start, Intermediate, and End.

- *Data* is represented with four elements: Data Objects, Data Inputs, Data Outputs, and Data Stores. *Data Object*s provide information about what Activities require to be performed and/or what they produce. *Data Input* and *Data Output* provide this information for whole business processes.

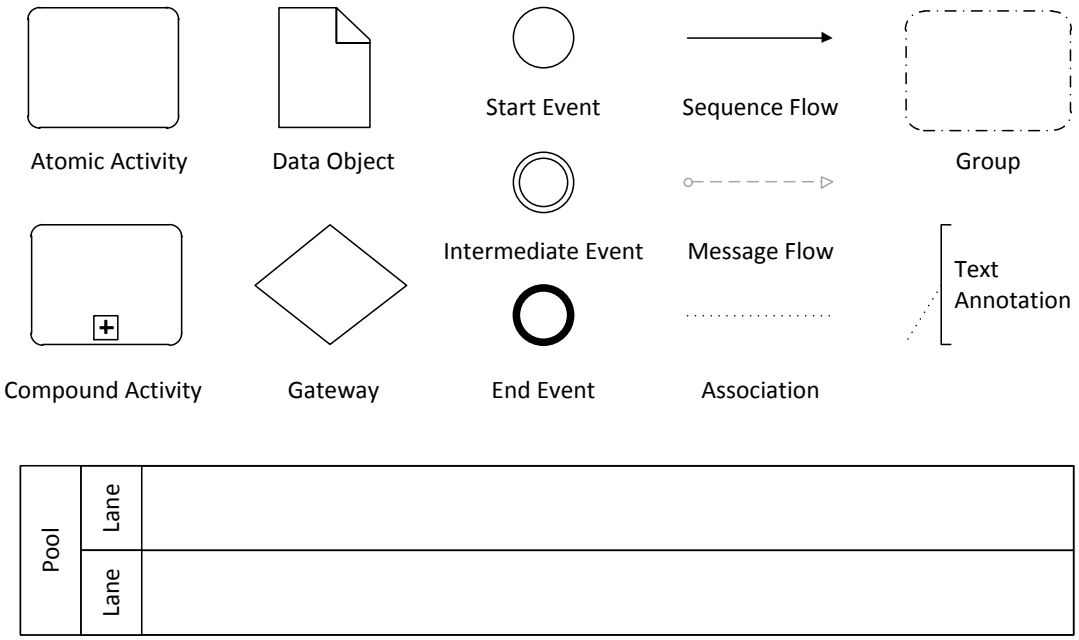

Figure 2.2.3.: BPMN core elements (cf. [Obj11a])

- *Connecting Objects* connect Flow Objects to each other or other elements. *Sequence Flows* are used to define the execution order of Activities in a business process. *Message Flows* specify the flow of messages between two participants. *Associations* are used to link elements and Artifacts like Text Annotations.
- *Swimlanes* group modeling elements. *Pools* represent a participant in a business process, e.g. a specific organization. *Lanes* sub-partition Pools and represent for example organizational entities such as roles.
- *Artifacts* provide additional information about a business process. The two standardized Artifacts are Groups and Text Annotations. *Groups* are grouping graphical elements that are within some common category without affecting the Sequence Flows. *Text Annotations* simply provide additional textual information for the reader of a BPMN diagram.

A business process diagram using BPMN is shown in figure 3.2.1 on page 49.

2.3. Software, Method, and Model-driven Engineering

2.3.1. Software Engineering

The application of models in software engineering has a long tradition. Before we detail ideas and specifics of method engineering and model-driven approaches, we introduce general terms and concepts with respect to software engineering first. The terminology is taken from international standards augmented with explanations and hints from current textbooks on this topic (cf. [Ins90, Ins97, BD09]).

Definition 4. *Software engineering* is the "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software". [Ins90, p. 67]

Software comprises computer programs, procedures, and data pertaining to the operation of a computer system. A *system* is a collection of interconnected parts to accomplish a set of functions. Modeling is a way to deal with complexity by ignoring details irrelevant to the question addressed. In the domain of software engineering the term *model* refers to any abstraction of a system. A *notation* or a *modeling language* is a graphical or textual set of rules for representing a model. Exemplary, the Unified Modeling Language (UML) [Obj11b] is a notation for representing object-oriented models.

Software engineering requires the collaboration of several people with different backgrounds and skills. The client orders and pays for the system, the project manager plans the project and controls the progress. Persons involved in the project will be referred as *participants*. A set of responsibilities in the project is called *role*. A role is associated with a set of tasks and assigned to a participant. One participant can fill several roles.

Artifacts that are produced during the development are called *work products*, e.g., a piece of software or a document. A work product for consumption within a project is an *internal work product*. A work product that must be delivered to the client is called *deliverable*.

An *activity* is a set of tasks that is performed toward a specific purpose. For example, security requirements elicitation is an activity whose purpose is to identify security requirements based on the threats and security goals analyzed before. Activities can be composed of other activities, e.g. the requirements elicitation activity includes the security requirements elicitation activity. High-level activities can also be called *phases*. A *task* represents an atomic unit of work that can be managed. Tasks consume resources, result in work products, and depend on work products produced by other tasks. *Resources* are assets that are used to accomplish work. Resources include time, equipment, material, and labor.

The *software development process* is the actual process by which user needs are translated into a software system. Models of the software development process are called *software development process models* or *software life cycle models*. A software development process model represents all activities, roles, and work products necessary to develop a software system as well as their relations.

2.3.2. Method Engineering

The development of software development process models is addressed by the method engineering discipline. Method engineering has been introduced in the 1980s as a proposal to address diverse requirements with regard to methods needed for the development of information systems. Since the one-size-fits-all approach had been regarded as unattainable the selection and assembly of method fragments from a repository in order to provide adequate methods offered a solution. [HS06, HSR10]

Definition 5. *Method engineering* is "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems." [Bri96, p. 276].

This definition is intentionally aligned with the internationally accepted definition of software engineering (cf. section 2.3.1). A major body of work within the method engineering discipline is dedicated to situational method engineering addressing adequate methods in actual organizations and projects. Situational method engineering *"encompasses all aspects of creating a development method for a specific situation"* [HSR10, p. 424].

Generally speaking, a method is a repeatable procedure that specifies the steps involved in solving a specific problem. Adapted to the method engineering discipline, a method is an *"approach to perform a software/systems development project [...] structured systematically in terms of [...] activities with corresponding [...] work products and [...] roles"* [HSR10, p. 426]. Recent publications in the field use the term method synonymously with the term process model [HSR10]. A situational method—also known as tailored process model—is a method tuned to the situation of the project at hand. The term methodology is used differently in the field. Applying etymology, some authors restrict the usage of the term methodology to the study of methods and scientific theory building (e.g., [Bri96]). Most authors in the field, though, apply the term methodology more broadly to a collection of methods for solving a class of problems and specifications how and when each method should be used or even as synonym for method [HSR10]. We will apply the latter approach within this thesis in order to align our terminology with common usage minimizing the use of the term methodology altogether.

Standardized building blocks for methods are called *method fragments*. Within the method engineering discipline, method fragments are commonly classified distinguishing the following dimensions: perspective, abstraction level, and layer of granularity [BSH98].

- The dimension perspective differentiates product and process fragments. *Product fragments* model the structures of the products of a method, e.g., diagrams and tables. *Process fragments* are models of the development process, e.g., tasks and activities.

- The abstraction level commonly distinguishes the conceptual from the technical level. The *conceptual level* addresses descriptions of development methods whereas the *technical level* addresses implementable specifications, i.e., tools.

- The compositional aspect of method engineering is explicated with the layer of granularity. Brinkkemper et al. propose to discriminate the granularity of method fragments with regard to five layers [BSH98]: method, stage, model, diagram, and concept. Method fragments on the *method layer* address a complete method. The *stage layer* entails method fragments on a segment of the life cycle. The perspective or abstraction of the system to be developed is addressed on the *model layer*. The *diagram layer* introduces representations of a view of a model layer method fragment. Concepts and associations of the method fragments on the diagram layer are covered on the *concept layer*.

While method fragments can be regarded as atomic elements of a method, different terms for relevant combinations of method fragments have been proposed by different authors. Commonly, the term *method chunk* is used for a relevant combination of process and product fragments [HSR10]. Addressing also the granularity layer of the method fragments, Sunyaev discriminates further method chains and alliances: *method chains* are combinations of method

fragments on different granularity levels, *method alliances* combine those of the same level [Sun11]. Standardized method chunks are stored in and retrieved from a so-called *method base* [Bri96].

Within the method engineering discipline, multiple approaches to support different phases within the life cycle of methods have been proposed. With regard to this thesis, especially approaches for the construction of individual method chunks and fragments from existing methods [Ral04], for the construction of methods from method chunks [RDR03], for tailoring or method configuration [ÅWK+03, KÅ04, KÅ09], and for the integration of method chunks from different method bases [RR01] are of interest.

Maturing over time and disseminating into industry, generalized frameworks and international standards addressing central aspects of method engineering have been agreed upon and respective tooling became available [FHS02, Int07, Obj08]. SPEM standardized by the OMG provides rich modeling capabilities and is supported by mature tooling from well known vendors as well as open source solutions (e.g., SPARX Enterprise Architect[2], No Magic MagicDraw[3], IBM Rational Method Composer[4], Eclipse Process Framework Composer[5]).

SPEM provides a metamodel and a conceptual framework in order to support modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes (cf. also for the following [Obj08]). The metamodel is based on the Meta Object Facility (MOF) [Obj14a] standard by the OMG and reuses the UML 2 infrastructure. Main strategies for the design of SPEM are the separation of so called method content definitions from the application of method content in a development process, independence from different life cycle models, and process variability and extensibility using a plug-in mechanism. This allows for the definition of highly sophisticated and scalable development process models and software development processes at the cost of a fairly complex specification entailing more than 60 metamodel elements.

The SPEM metamodel is organized using seven packages. We will focus on three packages: Method Content, Managed Content, and Process Structure (cf. section 4.7.2). The *Method Content* packages specifies elements for the definition of roles, tasks etc. Guidance artifacts are represented with elements from the *Managed Content* package. Elements to describe a development process build from aforementioned elements as a breakdown structure are provided in the *Process Structure* package.

- Core elements from the *Method Content* package are Work Product Definition, Task Definition, and Role Definition including their most important relations Performer, Responsibility Assignment, and Parameter In/Out/InOut. A *Task Definition* describes an assignable unit of work. A tangible work product consumed, produced or modified by tasks is specified with the help of a *Work Product Definition* and assigned to respective Task Definitions using matching *Parameter* associations (In/Out/InOut). A *Role Definition* captures a set of related skills, competencies, and responsibilities and is assigned to Task Definitions using

[2] http://www.sparxsystems.com/products/ea/
[3] http://www.nomagic.com/products/magicdraw.html
[4] http://www.ibm.com/software/awdtools/rmc/
[5] http://www.eclipse.org/epf/

the *Performer* association and to Work Product Definitions applying the *Responsibility Assignment* association.

- Main elements from the *Managed Content* package are Category and Guidance. *Categories* are used to categorize elements based on arbitrary criteria relevant for grouping, navigation, and browsing. *Guidance* elements are used to provide additional information related to other elements including guidelines, templates, checklists etc.

- The Activity element is at the center of the *Process Structure* package accompanied by the Role Use, and Work Product Use elements. An *Activity* represents a general unit of work assignable to specific performers represented by *Role Use*. An Activity can rely on inputs and produce outputs represented by *Work Product Uses*. It also represents a grouping element for all elements of a breakdown structure.

The implementation of the strategies for separation of method content definitions from their application, independence from different life cycle models, and process variability and extensibility in SPEM utilizes three central elements: Variability Element, Method Content Use, and Activity Use Kind.

- The *Variability Element* allows to describe differences (additions, changes, omissions) relative to an original of the same type using separate objects. It supports four Variability Types: contributes, extends, replaces, and extends-replaces. A Variability Element that *contributes* to another provides its properties into its base Variability Element without directly altering any of its existing properties, i.e., in an additive fashion. A special Variability Element that *extends* a base Variability Element inherits its properties but might override those with its own values. *Replaces* defines a replacement of a base Variability Element without directly changing any of its existing properties. *Extends-replaces* combines the effects of extends and replace variability; extends-replaces variability only replaces the values that have been redefined and leaves all other values of the base element as is.

- *Method Content Use* is an abstract generalization for breakdown elements that references one element from the Method Content package like Task Definition, Role Definition etc. It provides a proxy-like representation in breakdown structures of respective reference objects. It shall be provided with congruent copies of the relationships defined for the content element. However, those relationships can be modified for the particular process situation for which the Method Content Use has been created.

- The *Activity Use Kind* enumerates the nature of reuse for an Activity element. It is applied to reuse structures defined for one Activity in a second Activity without the need to physically copy its nested breakdown structure. The Activity Use Kind element allows for an extension, local contribution, or local replacement. *Extension* provides a mechanism for dynamically reusing Activity substructures in other Activities. *Local contribution* defines specific local additions to breakdown elements inherited via the Activity Use Kind “extension” within the context of the reusing Activity. *Local replacement* defines local replacements to breakdown elements inherited via the Activity Use Kind “extension”.

The elements specified by the SPEM metamodel might be subsumed applying common method engineering terminology as follows (cf. also [RMMG09]): Basic method fragments are specified using elements from the Method Content package. Especially, Work Product Definitions provide means to specify product fragments, Task Definitions for process fragments. Applying the associations Parameter, Performer, and Responsibility Assignment, basic method chunks can be defined. Method fragments on the technical level are mainly addressed by Guidance elements from the Managed Content package. Composition of method chunks are represented applying elements from the Process Structure package. In particular, the Activity element allows for the flexible composition of method chunks on different layers of granularity including method alliances and chains. The method base is organized applying elements of the Method Plugin package, that has not been covered in this section.

2.3.3. Model-driven Engineering

More than ten years ago, conceptual differences between different domain-specific concepts needed for the development of (complex) software systems came into focus. For example, to develop a system for telephone call processing, concepts from the telecommunication domain are needed to describe central parts of the problem to be solved and concepts from the software engineering domain are needed to describe central parts of the solution. Especially differences between concepts from the application domain and those from the software engineering domain draw a lot of attention and became known as the semantic gap [BBI+04]. The hypothesis has been suggested, that some of the challenges recognized for the development of contemporary applications is due to the semantic gap, i.e., the distance between the concepts of the different domains involved in the development of software systems.

2.3.3.1. Principles

Central ideas to tackle the semantic gap are direct representation, automation, and open standards [BBI+04, Bé06]. As for direct representation, the observation has been made that the more directly concepts in the application domain can be represented, the easier it becomes to specify a system. Conversely, the greater the distance between the application domain and the modeling concepts, the less value is yielded by modeling. Because of the many different application domains and their internal complexity and sophistication the systematic application of DSMLs[6] have been proposed to allow for a direct representation of the domain-specific concepts.

While direct representation using DSMLs is helpful for the use of humans, the translation of those models into models utilizing other (implementation) languages is time consuming and error-prone. Therefore, transformations between models utilizing different languages, different levels of abstraction, or different viewpoints should be automated. Automation might also introduce validation and analysis with regard to various flaws or the generation of test artifacts. Corresponding frameworks are important to support automation since transformations have to make assumptions about the application and implementation environment. Furthermore,

[6] The term domain-specific language (DSL) is also commonly used. As we focus on model-based security engineering we will stick to the term DSML.

frameworks allow for reuse, since developers do not have to implement such environmental aspects repeatedly.

Domain-specific (modeling) languages and frameworks to support their application are only likely to become common if accompanying standards support them. Open, industry-wide standards allow for the emergence of an ecosystem of tool vendors and experienced users that leverage the utility of direct representation and automation.

Approaches that address these ideas and try to provide explicit support to tackle the semantic gap are called MDE approaches. Although no common definition of MDE is apparent, the following principles are repeatedly named as essential for MDE: models as first class entities, the use of DSMLs, application of metamodels to express DSMLs, and powerful tooling to support DSML creation, application, and model transformation [Ken02, BBI+04, Bé06, Sch06, FR07, SVEH07].

Considering models as first class entities implies the systematic usage of models as central artifact in software engineering (in opposite to using models arbitrarily) as well as the treatment of any software artifact as a model (including documentation and code). While the use of DSMLs directly springs from the idea of direct representation, it is seen as a central contribution of MDE to express DSMLs by metamodels [Bé06]. A metamodel describes the various kinds of elements that a given model might contain, the way they are arranged, related, and constrained. A model is said to conform to its metamodel. A common approach to formalize models and metamodels in MDE is based on directed multigraphs (cf. [BBDF+06]). Powerful tooling again stems directly from the idea of automation. Integrating these aspects we adapt the definition of model-driven engineering from [Sch06]:

Definition 6. *Model-driven engineering* combines the application of multiple domain-specific modeling languages described using metamodels with transformation engines, generators, and respective tooling to allow for a systematic development and application of models in the domain of software engineering.

The systematic application of models with the help of DSMLs and tooling does not only comprise languages and technology but includes also the human and organizational aspects such as process descriptions for coordinated construction and evolution of models in the course of software engineering. Perhaps it is even more important to mention that MDE does not imply an all-or-nothing approach: It is valid within the boundaries of our definition to support only parts of the engineering activities with MDE techniques and methods. Introducing MDE must not outweigh the benefits of models with the burden of maintaining them. [Ken02]

The term *model-driven engineering* is often used synonymously with the term *model-based engineering*. Similar concepts and techniques are described for model-based engineering as we have discussed for model-driven engineering (e.g., [BFH+10]). We will use both terms applying the definition provided in this section and assign an individual focus to each term: The application of the term *model-based* indicates a focus on DSMLs described using metamodels and respective tooling to allow for systematic development and application of models. The term *model-driven* indicates a focus on transformation engines and generators in the same context. An example is the application of the term model-based to attribute our framework for security engineering (focusing our DSML SecEML, cf. chapter 5). An alternative example

is the application of the term model-driven to attribute one of the frameworks for the implementation of plug-ins for our workbench focusing the generation of source code: the Eclipse Modeling Framework (EMF).

2.3.3.2. Model-driven Architecture

Model-driven Architecture (MDA) [Obj03] is one of the best known MDE approaches for software and system development. It is an initiative started and standardized by the OMG. Foundation of MDA is the application of the divide and conquer strategy (or separation of concerns) with regard to architectural aspects: The specification of a system should be separated from the details how that system uses the capabilities of its platform. Therefore, MDA provides means to specify a system independently of the platform that supports it, specify platforms, choosing a particular platform for a system, and transforming the system specification into one for a particular platform. The primary goals of MDA are portability, interoperability, and reusability.

MDA defines different viewpoints and corresponding models to specify a system: computational independent model, platform independent model, platform model, and platform specific model. The computational independent model (CIM) captures the environment and the requirements of the system; details of the structure and processing of the system are hidden or undetermined. A CIM is also called the domain model and describes the system in the problem domain. The platform independent model (PIM) specifies the system omitting aspects that are specific with regard to a given platform. The PIM is suitable for use with a number of different platforms of similar type. A common technique for achieving platform independence is to target a system model for a technology-independent virtual machine. A platform model provides a set of technical concepts representing the parts that make up a platform and the services provided by the platform. A platform specific model (PSM) is the combination of the specifications in the PIM with the details that specify the usage of a particular platform.

Models are converted from one model to another model of the same system with the help of transformations. A MDA mapping provides all necessary specifications for transformations of a PIM into a PSM for a particular platform. The platform model will determine the nature of the mapping. Ideally, the MDA mapping is given in a way that allows for a transformation from the PIM into an implementation without manual intervention. Nevertheless, MDA defines different mappings to provide means for different application scenarios; not necessarily all of the transformations specified by the mappings are executable. Figure 2.3.1 shows a MDA metamodel transformation that converts a PIM to a PSM utilizing a mapping specified on the basis of the metamodels of the PIM and the PSM.

2.3.3.3. Model-driven Security

The problem of the semantic gap addressed by MDE approaches is prevalent in the security domain as well: First, security models and other software engineering models are typically disjoint and expressed using different notations. Second, security requirements and imple-

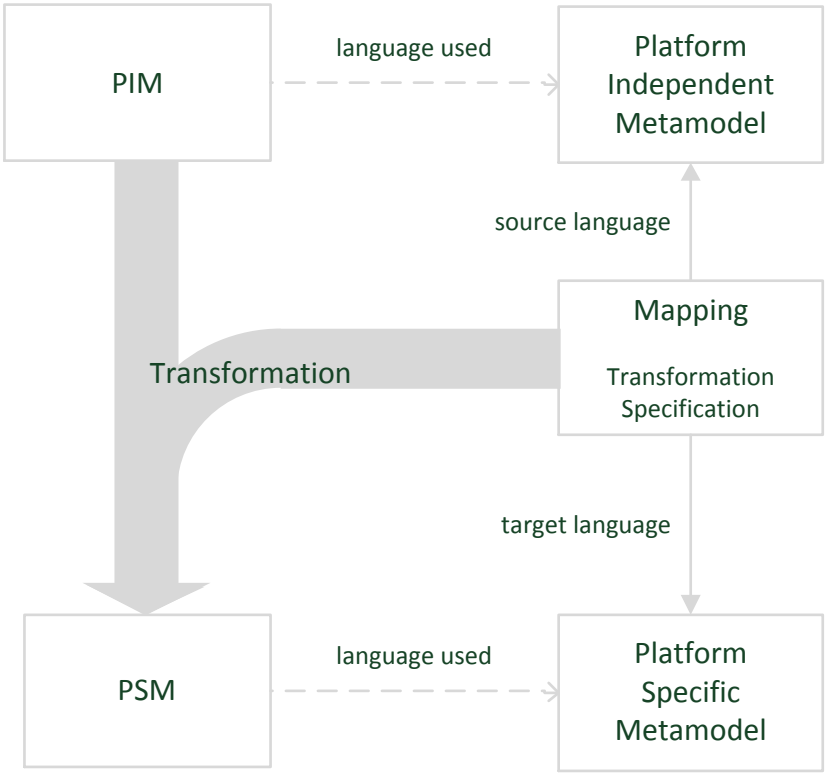

Figure 2.3.1.: MDA metamodel transformation [Obj03, p. 3-9]

mentation of corresponding controls commonly are not connected and are expressed using (very) different means. As a result, security is often integrated into systems in a post hoc or discretionary manner, which degrades the security and maintainability of the resulting systems. [BDL06]

Consequently, the adoption of MDE techniques and ideas in the security domain is an active research topic, often addressed as model-driven security (MDS, e.g., [BDL06, HB09, WMS+09, MRS09, RdGFMP10, MM10, BCE11]) and well recognized in industry [McD07]. The general idea of many approaches for MDS is to specify security aspects at an adequate level of abstraction using a DSML, to transform those models to enriched design models on the same or lower levels of abstraction, and to generate security-related artifacts necessary for implementation using a proper platform model. An exemplary approach from Basin et al. proposes to define security requirements considering access control constraints using a DSML for that purpose, link them with the corresponding system design model, and transform the resulting model into a security configuration model specifying low-level access control policies [BDL06].

An important challenge for the integration of security aspects in MDE approaches is the relation of the (to be defined) security model and the (existing) requirements, design, and implementation models. We distinguish four integration strategies: language extension, dialect definition, entity reference, and model weaving (cf. also [MM10]).

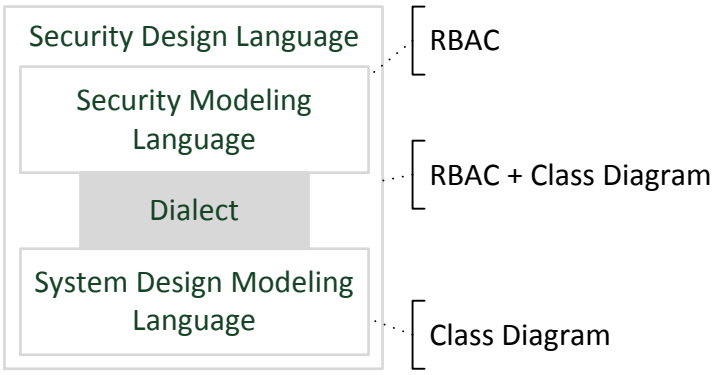

Figure 2.3.2.: Dialect definition strategy (cf. [BDL06])

The first strategy, language extension, enhances a given modeling language in order to express also relevant security aspects. Several languages define extension points that can be used for this purpose. For instance, UML provides stereotypes and tags to extend UML modeling elements. This variant of the strategy is known as lightweight language extension: The effort for the extension is generally rather small and existing tooling supporting the language extension points will not be broken. Another variant of this strategy is the extension of the metamodel of a modeling language. The syntax and semantics of the language is enriched introducing new elements and additional attributes for existing elements. This variant is also known as heavyweight language extension since existing tools are mostly not capable to handle the changed language. Consequently, resulting efforts are higher. An example for lightweight language extension is the extension of BPMN for the inclusion of security goals in business process models [WMS+09] or the definition of an UML Profile for the specification of security requirements in activity diagrams [RFMTP11]. Heavyweight language extension in the context of BPMN has been used in [RFMP07a].

The dialect definition strategy has been proposed by [BDL06] for the specification of access control policies and has been used for other purposes as well [MM10]. The dialect definition strategy proposes a modular schema for building DSMLs in the context of MDS. The schema depicted in figure 2.3.2 is parametrized using three languages: a security modeling language, e.g., for expressing security policies (here: role based access control (RBAC) [Ame04]), a system design modeling language, e.g., for constructing component models (here: UML class diagram), and a dialect, which provides a bridge between those two languages by defining connection points. Using different instantiations of the three parameters, different languages tailored for the specific needs can be defined based on a common set of concepts. Thus, this schema defines families of modeling languages (called security design languages in [BDL06]).

We will call the specification of references between models using different languages entity reference strategy. Precondition is the possibility to reference entities in a language-independent manner. If the dependency of two given models is unidirectional, only one of the languages used has to provide those referencing possibilities. Therefore, the independent language, models specified using this language, and tooling to work with those models are

untouched. An example for the use of the entity reference strategy to model security policies in the context of UML models is given in [AHB07].

If none of the languages to be integrated provide language-independent entity references and the modeling languages should not be extended or changed, the model weaving strategy provides a solution [BBDF+06]. Model weaving defines a modeling language for model composition that provides language-independent entity references. Entities from arbitrary models can be related using weaving models specified in a model composition language. The integration of security concepts and business process models using model weaving has been demonstrated in [Eic10].

2.4. Security

In the domain of IT the term security and accompanying concepts have different definitions and are not used consistently in literature (cf. [AW04, McG06, FGH+10]). To provide a consistent terminology for the thesis, section 2.4.1 introduces several proposals, discusses similarities and differences, and provides definitions and explanations of relevant concepts. Section 2.4.2 introduces current interpretations of security engineering, respective concepts, and relates them with the approach taken in this thesis.

2.4.1. General Terminology

In everyday language, the term *security* has the meaning of a "*state of being protected or safe from harm*" and *"things done to make people or places safe"* [MW14]. Therefore, the term denotes a specific state as well as means to achieve it.

The application of the concept of security in the IT domain is called IT security. The term IT security shares the ambiguity of the general concept, but further meanings have been assigned to it. An early definition focuses distinctive means to achieve security with respect to information systems: *"The term 'security' describes techniques that control who may use or modify the computer or the information contained in it."* [SS75, p. 1280] A little shift can be observed in the following definition: *"Security [is a] system condition that results from the establishment and maintenance of measures to protect the system."* [Shi07] Still means ("measures") are in the focus of the definition, but IT security becomes a system condition and protection is more general than usage and modification of the computer or information contained on it. Commonly, IT security is defined enumerating specific objectives, e.g., *"security [is] the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources"* [Nat96, p. 5]. Practitioners often focus adversaries: *"Security [is the idea of] engineering software so that it continues to function correctly under malicious attack [...]."* [McG06, p. 3]

To allow for a systematic approach to define our terminology with regard to IT security, we will focus on proposals from three main sources: the commonly referred standard 13335-1 from the International Organization for Standardization (ISO) [Int04], the influential terminology from the Common Criteria (CC) standardized as ISO 15408-1 [Int14], and a proposal from

academia by Fabian et al. [FGH+10]. The first two sources represent international standardized as well as commonly referenced and applied terminology that stem from two different viewpoints. ISO 13335-1 provides its terminology in the context of IT security management, the definitions in the CC address IT security evaluation. The proposal from Fabian et al. addresses the development of secure systems and introduces a conceptual framework for security requirements engineering. Furthermore, it provides a mapping from the conceptual framework to other approaches that are frequently referenced in academia. Therefore, the terminology from Fabian et al. synthesizes many proposals from academia and allows for a systematic positioning of our terminology with regard to related literature. Additional sources will be used to incorporate further relevant aspects not focused in the three main sources. In the following, we will omit the attribution "IT" of "IT security" if no misunderstandings are possible.

A very broad definition of security is given in ISO 13335-1: *"Information and communications technology (ICT) security [is defined as] all aspects related to defining, achieving, and maintaining confidentiality, integrity, availability, non-repudiation, accountability, authenticity, and reliability, of ICT."* [Int04, p. 3] Therefore, the definition highlights two aspects: the life cycle of security (from definition to maintenance) and seven enumerated properties that security has to tackle. The definiendum of security itself is sketched very broadly with the term "all aspects" that are related to the life cycle of the enumerated properties. Compared with ISO 13335-1, the definition of the CC seems to be more clear-cut: IT security is concerned with the *"[...] protection of assets from unauthorised disclosure, modification, or loss of use. The categories of protection relating to these three types of failure of security are commonly called confidentiality, integrity, and availability, respectively"* [Int14, p. vi]. Three of the seven properties that have been named in the definition of the ISO 13335-1 are picked up as well and assets shall be protected with regard to these properties to achieve security. The idea of protecting assets is further abstracted by Fabian et al. but not explicated very well: *"We consider security to be a system property."* [FGH+10, p. 9] Following Fabian et al., security is achieved if all specified security properties for given resources of a system hold, i.e., there is no violated security property. To substantiate the aspect of security being a system property, the following definition is helpful that has been recognized by Fabian et al. as well: *"Security is the property of a save system to take only those states that do not induce unauthorized information modification or disclosure."* [Eck14, p. 6, translation by the author] Therefore, security interpreted as a state of a system implies that states with specified properties must not be taken by the system.

With our definition of security we will basically follow the proposal of Fabian et al. (augmented with the precision from [Eck14]) but highlight the relation of the protection of assets with regard to the security intentions of their stakeholders:

Definition 7. *Security* is the property of a system to take only those states that do not violate the security goals of the assets affected by the system.

To make our definition of security fully transparent we need to further explain the terms assets and security goals. Often, assets are defined as *"anything that has value to the organization"* [Int04, p. 1]. As an organization generally will not consider or manage anything that it does place any value upon, this definition does not allow for a clear-cut separation of things

that are assets from other things. The CC is somewhat more precise in its definition: Assets are *"entities that the owner of the [target of evaluation] presumably places value upon"* [Int14, p. 2]. Therefore, the CC introduces a stakeholder (the owner of the target of evaluation) in opposite to a more general organization. Fabian et al. generalize and narrow this idea in their definition that we will use as well:

Definition 8. *"*An *asset* is any entity that a stakeholder puts a value upon with respect to security." [FGH+10, p. 11, markup by the author]

This definition introduces the security perspective taken in the valuation of entities. Furthermore, it allows for a multilateral perspective on security that acknowledges conflicting interests of different stakeholders with regard to security [RPM99, Ran00]. A stakeholder in this definition is an individual, a group, or an organization that has an interest in the system in question.

The term security goal or security objective is used mainly with two different meanings. Prominently, security goals are defined as an enumerated list of properties: *"The five security goals are confidentiality, availability, integrity, accountability, and assurance."* [SHF04, p. A-3]. This definition introduces the issue of addressing general properties as goals. Therefore, security goals following this definition lack some context like subject and object that must be introduced later (e.g., as security requirement). In contrast, other authors define security goals as expression of (high-level) security needs with regard to an asset: *"A stakeholder's security goal expresses his or her security concerns towards an asset."* [FGH+10, p. 11] We will build our definition on the latter proposal:

Definition 9. A *security goal* expresses the stakeholder's concerns towards an asset with regard to a security goal class.

Our definition narrows *"security concerns"* from the definition of Fabian et al. to *"concerns [...] with regard to a security goal class"*. Therefore, we introduce those properties that are commonly understood to be of interest in the domain of IT security as security goal classes:

Definition 10. A *security goal class* is a classification of security goals. Within the scope of this thesis we will address the following classes of security goals: confidentiality, integrity, availability, and non-repudiation.

Our distinction between security goals and security goal classes also allows for a flexibility in the definition of the scope of security considerations without loosing precision. The CC is applying a similar approach subsuming confidentiality, integrity, and availability as *"categories of protection"* [Int14, p. vi]. Fabian et al. address this distinction in their conceptual framework as well: *"Security goals are traditionally classified into* integrity, confidentiality, *and* availability *goals."* [FGH+10, p. 12].

Definitions for security goal classes are often taken from ISO 13335-1, also in academia. Therefore, we will use those definitions as well:

- Confidentiality: *"The property that information is not made available or disclosed to unauthorized individuals, entities, or processes"* [Int04, p. 8]

- Integrity: *"The property of safeguarding the accuracy and completeness of assets"* [Int04, p. 9]
- Availability: *"The property of being accessible and usable upon demand by an authorized entity"* [Int04, p. 8]
- Non-repudiation: *"The ability to prove an action or event has taken place, so that this event or action cannot be repudiated later"* [Int04, p. 3]

Violations of security goals are commonly associated with threats. ISO 13335-1 introduces threats as *"a potential cause of an incident that may result in harm to a system or organization"* [Int04, p. 4]. Therefore, threats are not manifest as they are "potential" but might harm a system or an organization in causing unwanted incidents. The relation of the harm potentially caused by a threat to assets is focused in the definition of the CC: *"A threat consists of an adverse action performed by a threat agent on an asset. [...] [Adverse] actions influence one or more properties of an asset from which that asset derives its value."* [Int14, p. 43] Following CC, a threat agent is simply an *"entity that can adversely act on assets"* [Int14, p. 8]. The CC focuses actions that might be performed by an entity in opposite to a causation of possible harm. Thus, the CC limits threats to willingly executed acts that might devalue an asset and ignores omissions or other inactive causation for asset devaluations. Another proposal diminishes the relevance of threats and puts the definitional weight on vulnerabilities: *"[A ...] violation [of a security property] can be caused by a vulnerability, which could be potentially be exploited by a threat [...]."* [FGH+10, p. 13]. Fabian et al. assign causation of violated security properties to vulnerabilities. Vulnerabilities—following Fabian et al.—are everything that causes violations of security properties. More commonly, vulnerabilities are defined as a deficiency of assets or systems. ISO 13335-1 defines a vulnerability as *"a weakness of an asset or group of assets that can be exploited by one or more threats"* [Int04, p. 4], the CC as a *"weakness in the [target of evaluation] that can be used to violate the [security functional requirements] in the operational environment for the [target of evaluation]"* [Int14, p. 17].

We will assign causation of violated security properties with threats as causation is generally assigned with driving forces instead of deficiencies (the apple falls from the tree because of gravity, not because the tree has failed to keep it) [Joh10]. Furthermore, we will allow not only actions to violate security properties but also omissions:

Definition 11. A *threat* is a potential cause of the violation of a security goal.

As threats are potential causes, actual causes of deliberate violations of security goals are called attacks: An attack is an exploitation of a vulnerability, realizing a threat (cf. [FGH+10]). Vulnerabilities are defined as deficiencies of a system in question:

Definition 12. A *vulnerability* is a weakness of a system that can be exploited by a threat.

A security goal expresses the concerns of a stakeholder towards an asset with regard to the security goal classes. A threat potentially violates a security goal. Refinements of security goals are commonly called security requirements: *"A security requirement refines one or more security goals. It refers to a particular piece of information or service that explicates the meaning*

of the asset it concretizes in the context of the system under construction." [FGH+10, p. 12] In CC terminology, security needs of a stakeholder are refined in security objectives, necessarily addressing threats. Security objectives are then further refined in security requirements using a standardized language: A security objective is a *"statement of an intent to counter identified threats"* [Int14, p. 7]. A security requirement is a *"requirement, stated in a standardised language, which is meant to contribute to achieving the security objectives for a [target of evaluation]"* [Int14, p. 7]. A different perspective is introduced by Moffet et al., defining security requirements as *"constraints on [...] functional requirements"* [MHN04, p. 18]. As we do not qualify necessarily all security requirements as constraints, we will stick to the proposal from Fabian et al., integrating the relation of security requirements and threats proposed by the CC:

Definition 13. A *security requirement* is the refinement of security goals and states the intent to counter threats.

Until now we did not touch the second part of the meaning of security in everyday language: the guarding measures. In IT security different terms are used to address such guarding measures: safeguard, countermeasure, and control are the most frequently used terms. ISO 13335-1 defines safeguards as *"practices, procedures, or mechanisms that may protect against a threat, reduce a vulnerability, limit the impact of an information security incident, detects incidents, and facilitate recovery"* [Int04, p. 15]. Therefore, safeguards are not only technical mechanisms but also human practices and organizational procedures. Furthermore, their purpose covers not only prevention, but also *"deterrence, limitation, detection, correction, recovery, monitoring, and awareness"* [Int04, p. 15]. The CC uses the term countermeasure and control with a comparable meaning, referencing the enumeration of controls provided by the ISO 27001 standard (cf. [Int13a, Int14]). Another approach links controls with security goals and their classes: *"[Security controls are] management, operational, and technical controls (i.e., safeguards or countermeasures) prescribed for an information system to protect the confidentiality, integrity, and availability of the system and its information."* [Nat11, p. 171] Fabian et al. simply define a countermeasure as mitigation of vulnerabilities [FGH+10, p. 13]. We synthesize from those approaches our definition as:

Definition 14. (Security) *controls* are practices, procedures, or mechanisms that mitigate threats with regard to security requirements.

Hence, we will use the term control or—in case of any ambiguity possible—security control and subsume human practices, organizational procedures, and technical mechanisms under it. The purpose of a control is the mitigation of one or more threats referenced by a security requirement. The purposes can be further detailed in prevention, deterrence, limitation, detection, correction, recovery, monitoring, and awareness. Controls are assigned to security requirements to relate every control with one or more security goals that are pursued by the control.

Not every security requirement can be satisfied introducing controls to a system. Furthermore, most (technical) controls rely on environmental circumstances to be effective. Therefore, the term assumption is important. In software (requirements) engineering, assumptions *"constrain the environment of the machine"* [FGH+10, p. 13]. In the domain of

IT security, assumptions are used similar: *"Assumptions [...] are made on the operational environment in order to be able to provide security functionality."* [Int14, p. 43] Fabian et al. detail the processing of assumptions: *"At the implementation level, assumptions are refined to organizational procedures [...] that prescribe how the implemented machine must be used in order to achieve security."* [FGH+10, p. 13] Therefore, we define assumptions as follows:

Definition 15. (Security) *assumptions* constrain the environment of a system in order to achieve security.

Assumptions (or security assumptions in case of any ambiguity possible) are related to security requirements and threats, as they constrain the environment of a system in order to satisfy security requirements and mitigate threats not countered by the system itself.

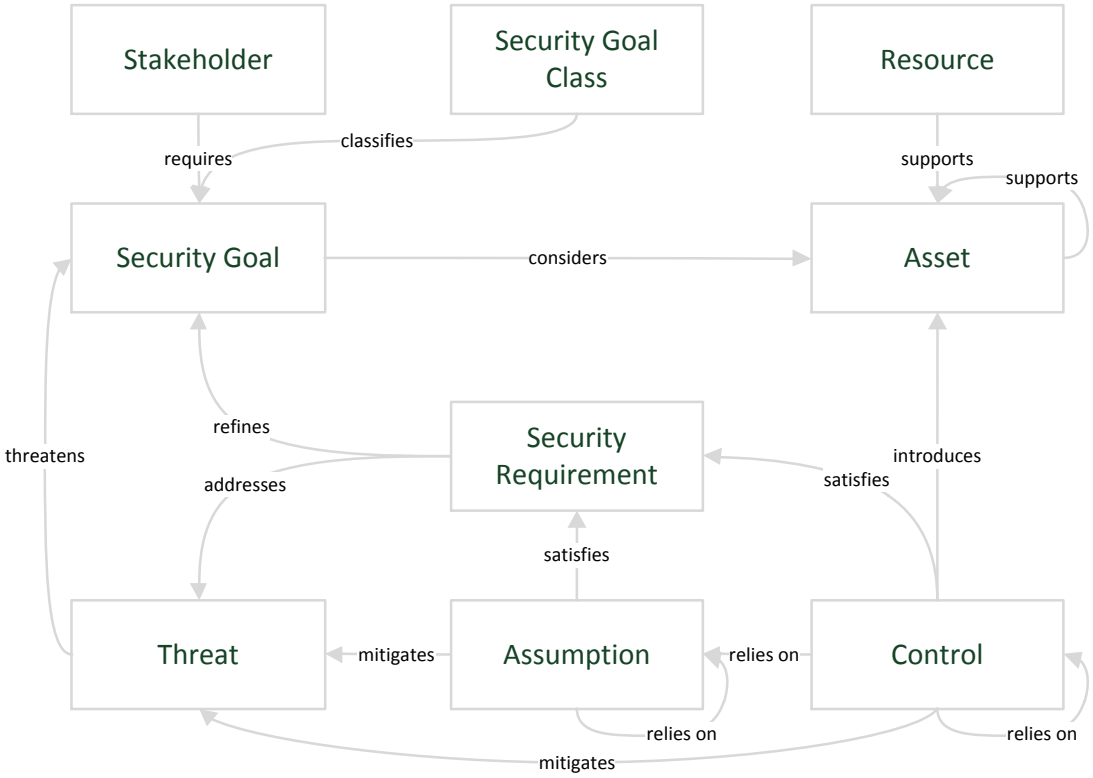

Figure 2.4.1.: Core security concepts and their relations

Dependencies between the core concepts of IT security as we have introduced them can be summarized as follows (cf. figure 2.4.1): A *security goal* expresses the *stakeholder's* concern towards an *asset*. *Security goal classes* that will be considered in this thesis are confidentiality, integrity, availability, and non-repudiation. *Assets* might be related to other assets such as one *asset* is supported by another *asset*, i.e., if the supportive *asset's security goals* are violated, those of the supported *asset* are very likely to be violated as well. *Threats* are potential causes of the violation of *security goals*. *Security requirements* refine *security goals* and address *threats* that threaten its *security goals*. *Controls* mitigate *threats* in order to satisfy *security requirements*

(possibly in conjunction with other *controls*). *Controls* might introduce new *assets* into the system that must be considered with regard to their *security goals* and *threats* as well. In order to work properly, *controls* might rely on other *controls* or *assumptions*. *Assumptions* constrain the environment of the system in order to mitigate *threats* and satisfy *security requirements*. Most likely, *assumptions* will be refined in organizational procedures etc. at the implementation level.

2.4.2. Security Engineering

A common definition of security engineering is not available. This might be due to the fact that security engineering is considered to be still in its infancy [MJF06, Eck14]. One often cited definition is provided by Anderson: *"Security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves."* [And08, p. 3] This definition focuses four aspects: security engineering is constructive, security engineering focuses those aspects that have to do with activities of adverse stakeholders, security engineering has to tackle (most phases of) the system development life cycle, and security engineering is about technology as well as human activities. Similar aspects are considered in a current definition from the National Institute of Standards and Technology (NIST): Following NIST, security engineering is an *"interdisciplinary approach and means to enable the realization of secure systems. It focuses on defining customer needs, security protection requirements, and required functionality early in the systems development life cycle, documenting requirements, and then proceeding with design, synthesis, and system validation while considering the complete problem"* [Nat11, p. 171]. With regard to this definition, security engineering is also constructive, addresses the security of systems, integrates different domains, and has to address the system development life cycle.

The difference between security engineering and secure system (or software) development is blurred. Often those terms are used interchangeably. The same proposals are cited to be approaches and methods for security engineering, secure system development, or secure software development (e.g., [JM05, KS06, CZ08, Jü09]). Sometimes security engineering is used in a broader sense, i.e., approaches for secure system/software development are subsumed under security engineering and the development approaches align methods, tools, and techniques from security engineering in a specific configuration or arrangement (e.g., [MJF06]). Another interpretation relates approaches for secure system or software development with specific development methodologies that integrate security-related activities in opposite to an integration of all security-related activities in a consistent cross-cutting discipline called security engineering (e.g., [VFMP05]).

Reconsidering the definition of software engineering[7], we distinguish secure system or software development from security engineering taking the following perspective: Security engineering focuses security-related activities and provides systematic approaches for the

[7] Software engineering is the *"application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software"* [Ins90, p. 67], cf. section 2.3.1.

integration and application of security-related activities. Secure system or software development focuses system or software development and integrates also security-related activities (cf. [Bis02, p. 484]). We call all those (parts of) activities within a system or software life cycle security-related, that explicitly consider or apply security concepts and gain a unique viewpoint, technique, or practice in doing so. In accordance to software development process models, we call a representation of all activities, roles, and work products necessary to augment traditional software development process models in order to build secure and trusted systems a *security engineering process model*.

With regard to the topic of this thesis—model-based security engineering for electronic business processes—we contribute a framework to integrate and apply security-related activities from asset identification to the configuration of controls for electronic business processes systematically. Following our distinction between security engineering and secure system or software development, approaches for security engineering need to be integrated into approaches for system or software development to be applicable. Therefore, our approach has to be integrated in existing software development approaches as it is demonstrated in section 4.7.

Methods and techniques that are applied in security engineering can be classified as security principles, patterns, best practices, formal modeling, verification methods, and process models.

Security principles stem from experiences. It has been recognized very early, that the development of secure systems is difficult. To guide the engineering of secure systems, principles have been formulated that generalize practical experience. Most prominently, design principles for secure systems from Saltzer et al. [SS75] have been recognized in this area.

Most principles from Saltzer et al. address recommended application of controls: *fail-safe defaults* to restrict on explicitly defined access permissions, *complete mediation* of every access attempt, *separation of privilege* to avoid single points of failure, *least privilege* to minimize the impact of vulnerabilities, and *least common mechanism* to minimize information paths. Some principles consider human factors: *economy of mechanism* to allow for an easy understanding of the design, *open design* to allow for public criticism of design decisions, and *psychological acceptability* to avoid circumvention of controls in everyday usage.

The formulation of security principles to guide secure system development has been repeatedly adopted by many authors and detailed with regard to (sub-) topics like software assurance (e. g., [Lan01, Red07]). The NIST proposed additional engineering principles for IT security integrating risk-based considerations (risk reduction, trade-off identification between risk, cost, and operational effectiveness) and organizational issues (developer training) [SHF04].

Security patterns have been introduced to security engineering focusing solutions [YB97, Sch03, HHS07]. Security patterns inherit their principle from the work provided by Gamma et al. [GHJV95]: Their software engineering patterns capture solution blueprints, their relationship to the problem, necessary context, and constraints they impose. A frequently cited publication is the collection of security patterns by Schumacher et al. [SFBH+05]. Following Schumacher et al., a security pattern describes a recurring security problem in a given context and presents a generic solution for it. The generic solution can be adapted according to the needs of the context. The pattern catalog provides more that 50 patterns that cover topics like enterprise security, access control, accounting, and secure internet applications. Many

of the patterns address security problems analyzed and solved before, demonstrate their interaction, and unify the description. Therefore, the pattern catalog can be seen as a best practice collection comparable to [And08], integrated and homogenized using the pattern approach.

Formalization of security problems is a well known practice in security engineering [vO06]. Following Bau et al. [BM11], formal approaches to security generally build upon three components: a system model, a threat model, and a formal definition of the security properties in question. The security models provide a basis for security analysis, i.e., the process of evaluating whether the desired security properties hold with regard to the system and the threat model. Formal approaches leverage the usage of tools to automate steps in the analysis. Frequently applied tooling includes model checkers and theorem provers. Common problem areas for formal methods are access control (e.g., [BL73, SCFY96]), information flow (e.g., [Den76, MSK07]), and cryptographic protocols (e.g., [MMS97, BCH10]). Providing important contributions to the field of security and security engineering, formal methods are nonetheless considered to be limited to address restricted problems. In general, larger systems cannot be tackled completely with formal methods due to under-specification and increased complexity.

The term security best practices denotes methods and techniques that are deemed useful in the domain of IT security. Generally, they spring from experience, are driven by practical considerations, and cover a large set of topics and means including stepwise guidance to general rules of thumb, checklists to sophisticated tooling. A comprehensive presentation of security engineering best practices is provided by Anderson [And08]. Anderson presents a set of security building blocks like protocol analysis, cryptography, and access control first. After that, application areas and scenarios are analyzed to demonstrate the application of security best practices to understand security problems and solution approaches based on building blocks. Application areas encompass banking and bookkeeping, nuclear command and control, telecommunication system security, and others.

Security principles, patterns, formal methods, and best practices provide methods, techniques, and tooling to augment standard software development processes with regard to security. Process models for security engineering integrate security-related activities and provide guidance on how to apply those methods and practices systematically. Process models for security engineering are a vivid field of research. General requirements for process models are analyzed in [BBH+03], process models based on specific standards are provided in [Whi01, VWW02, MFMP07], and individual phases in the development life cycle are highlighted in [MHS05, HHS07].

2.5. Related Work

This section provides an overview of existing approaches considering security engineering of electronic business processes. First, approaches that focus (electronic) business processes are presented emphasizing those approaches that propose a model-based, methodological procedure for the development of secure electronic business processes (section 2.5.1). Second, general model-based security engineering approaches are depicted that might be applied to electronic business processes as well (section 2.5.2). A closing discussion positions the

contributions of this thesis with respect to the approaches presented (section 2.5.3). All approaches will be presented using the terminology developed in section 2.4 to allow for better comprehensibility and comparison.

2.5.1. Approaches for Security Engineering of Electronic Business Processes

2.5.1.1. Surveys

Only very few surveys have been conducted that collocate approaches for security engineering of (electronic) business processes. An analysis of 17 development approaches for information system security by Siponen highlights two approaches that follow a business process paradigm [Sip05]. Both approaches address conceptual analysis and are considered as non-supportive for actual information system development by Siponen. A subsequent proposal by one of the authors of those approaches is discussed in some detail later in this section [HH06]. Outlining main contributions in the area of security for workflow systems, Atluri et al. touch the topic of security engineering [AW08]. Focusing access control, several approaches for access control specification, analysis, and validation are presented and compared. None of them provide a methodical procedure for the development of secure electronic business processes. Jakoubi et al. provide an overview of scientific research efforts to integrate security and risk considerations into BPM [JTGQ09]. Comparing nine approaches, they highlight risk valuation of business processes and identify several aspects missing in the approaches presented such as adequate support for probabilities and some security properties, analysis of efficient resource allocation, and metrics for security. Two of the proposals mentioned by Jakoubi et al. will be covered in some detail in the following as they more or less directly address security engineering of business processes [Rö03, NKB05], another will be touched later as it covers only selected phases of the development [RFMP06].

2.5.1.2. Process Oriented Security Model (POSeM)

One of the early proposals for security engineering of business processes is the Process Oriented Security Model (POSeM) provided by Röhrig et al. [Rö03, RK04]. The objective of the POSeM approach is to support the selection of appropriate security controls for a pre-defined business process. To meet this objective, POSeM provides three building blocks: first, the Security Enhanced Process Language (SEPL) to capture security requirements of business processes, second, two rule bases to check the consistency of the security requirements and to support the identification of appropriate controls, and third, a description of the systematic procedure to apply the approach.

SEPL is a simplified version of the Workflow Process Definition Language (WPDL) standardized by the Workflow Management Coalition (WfMC) that introduces additional security markups. It enables assignment of a security tag to each element of the process definition specifying clearance or security levels with regard to security goal classifications. The first rule base specifies checks to ensure the consistency of the security tags in the SEPL process definition by comparing clearance and security levels of participants, tasks and necessary resources. The second rule base provides rules to derive generic controls from SEPL models

based on two IT security management standards (BS 7799 and ISO 13335-4, now succeeded by ISO 27002 and 27005 [Int13b, Int11b]). The procedure consists of four activities:

1. Definition of security goals: Relevant assets are grouped and assigned with security goal classes and ratings (e.g., the asset group "company-internal data" is assigned with the security goal class "confidentiality" and a corresponding level of necessary protection with the rating "high").

2. Refinement of security goals: The process is modeled in SEPL and each element of the process model is assigned with security tags based on the security goals defined in the preceding activity (e.g., the data element "order" is regarded as "company-internal data" and therefore assigned with the security tag "high confidentiality").

3. Consistency analysis: The SEPL model is checked for consistency utilizing the first rule base; inconsistencies are solved by redefining ratings of the security goals or changing the process definition (e.g., splitting one task in two separate tasks assigned to different participants).

4. Derivation of generic controls: For each element of the process model appropriate controls are derived utilizing the second rule base (e.g., applying symmetric encryption to the "order" entity in transit); doublets and obsolete controls are removed afterwards.

A fifth activity—the mapping of generic controls onto implementations—is envisioned in the POSeM approach but not detailed.

2.5.1.3. Modeling Security Semantics of Business Processes (MoSSBP)

An approach from Herrmann et al. named Modeling Security Semantics of Business Processes (MoSSBP) [HH06] provides a framework to handle business process security requirements from their specification to their realization. To meet this objective, two approaches are combined: Modeling Security Semantics (MoSS) [HP99] addressing the business process expert and the more general Object-oriented Security Analysis approach [HK01] utilizing graph rewriting to enable automated model refinement. The MoSSBP approach rests on three main pillars: a notation to specify security goals within business processes, repositories, and a systematic procedure to apply the approach. The repositories provide (1) business process modifications according to security goals, (2) security controls, and (3) security control implementations.

The MoSS notation allows to assign security goal classes and specific security related terms like "copyright" to certain elements of the business process that are considered as assets. The repository for business process modifications entails graph rewrite rules to modify business process models according to security goals assigned to business process elements. The repository for security controls is used to assign controls according to the modified process models. Elements of the repository are specified using the language ALMO$T (A Language for Modeling Secure Business Transactions, [RHP99]). The repository for security control implementations maps security controls to their respective implementations. The procedure comprises four activities:

1. Identify security goals: Security goal classes are assigned to elements of the business process modeled as UML Activity Diagrams; checks ensure the validity of the assignments.

2. Check for process modifications: Checks are executed to identify security goals that might not be addressed by controls in the repository; for those security goals graph rewrite rules are executed to modify the process model to allow for an application of existing controls.

3. Assign security controls: Security controls from the repository are assigned to the security goals; missing controls are prompted to the security expert.

4. Assign control implementations: Control implementations from the repository are assigned to the selected security controls.

2.5.1.4. ProSecO

A security engineering approach for service-oriented architectures has been developed by Hafner et al. [HB09]. Its application to electronic business processes integrates a previously developed framework for high-level development and management of workflows based on web services called SECTET [HBAN06, AHB07] and a security analysis method called ProSecO [BHIOW08].

The SECTET framework follows the MDA approach sketched in section 2.3.3.2. An UML Activity Diagram is used to specify a global workflow model as PIM. Security goals are specified for elements of the global workflow model utilizing a DSML called SECTET Policy Language (SECTET-PL). Security goals are provided as UML Notes. SECTET transforms the global workflow model to local workflow models (PSMs) applying an UML Profile. A further—partly manual—transformation generates executable workflow models as implementation specific model (ISM) specified using the WS-BPEL and security configuration artifacts specified using the Extensible Access Control Markup Language (XACML) [Org13]. Transformations of the security goals are labeled as patterns in SECTET and provided for message integrity, confidentiality, and non-repudiation generating XACML policies specifying symmetric encryption and message signing.

The security analysis method ProSecO introduces a language to model functional and security views. Those views describe model elements that are interrelated with regard to security. The functional view captures dependencies of assets and stakeholders on different levels of abstraction (e.g., business process, information, service, local component, node). The security view relates security concepts (security goals, threats, controls) with elements of the functional view. The elements of the security view themselves are specified using natural language. The ProSecO analysis process specifies six activities that should be executed iteratively:

1. Create or adapt the functional view: Assets and stakeholders are identified based on the business process specification (e.g., a stakeholder “Research Group” and a process “Retrieve Patient Data”).

2. Define security goals: Security goals for assets are specified (e.g., the possibility to retrieve patient data anonymously).

3. Identify dependencies: Based on the security goals, dependencies of elements in the functional view are analyzed and modeled (e.g., the stakeholder “Research Group” uses the process “Retrieve Patient Data”).

4. Refine security goals: Security goals are refined with regard to dependent and depending assets (e.g., the process “Retrieve Patient Data” must not access personal data).

5. Threat and risk analysis: Threats to assets of the security goals are identified and evaluated with regard to probability and possible impact (e.g., the possibility to personalize access to patient data); security requirements are elaborated relating threats to security goals.

6. Design of controls: Controls are chosen that satisfy the security requirements (e.g., the application of strong pseudonyms for authentication).

2.5.1.5. Automated Risk and Utility Management (AURUM)

An approach that combines risk assessment with security requirement elicitation and interactive selection of controls is presented by Neubauer et al. [NKB05, NH08a, NH08b]. It has later been labeled as Automated Risk and Utility Management (AURUM) [EFN09, NP10]. AURUM is intended as method for security experts for a structured risk assessment process. It does not provide any technical solution but supports stakeholders in finding an appropriate control portfolio for a given business process. AURUM provides three activities each comprising several tasks:

1. Modeling and identification: The first activity identifies and models necessary entities for the risk assessment. First, a business process model is created specifying the process in question. Using the business process model, assets represented in the model are identified and classified as IT system, machinery, communication device, data, or other assets (e.g., intangible assets). Vulnerabilities as well as threats for that assets are identified and documented. To support vulnerability and threat analysis, the authors propose consultation of security standards such as ISO 17799 (since 2007 succeeded by ISO 27002 [Int13b]). The last task of the first activity is the identification of possible and actual security controls.

2. Risk Assessment: A workshop follows after the initial activity to relate modeled entities with each other and to allow for a quantitative trade-off analysis. Risks are identified relating assets, vulnerabilities, and threats. Controls for the identified risks are analyzed and assigned. Cost/benefit categories are defined for rating controls. Risks and controls are quantified depending on the chosen categories (e.g., acquisition cost, maintenance cost, probabilities). After that, additional constraints and dependencies are specified with regard to the available controls, e.g., maximum number of controls per risk.

3. Interactive selection of controls: The third activity executes an interactive decision making process using the modeled entities, their dependencies and the quantified cost/benefit categories. Initially, Pareto-efficient portfolios of proposed controls are generated. After that, interactive modifications of constraints or individual selections are executed to determine an optimal control portfolio for the business process.

2.5.1.6. Other Approaches

Computer-aided security engineering focusing business processes is envisioned by Mana et al. [MMRV03]. The approach called Formal Methods and Modeling Language Framework (FML) proposes the extension of UML models to capture formally specified security requirements and the application of security patterns to assist developers in the design of secure business processes. Implementation details of the approach are not provided.

Ciuciu et al. address a comparable objective with their approach to introduce semantic support for security-annotated BPMN business process models [CZM+11]. Security constraints for business process are specified with the help of a simple language that utilizes BPMN Text Annotations [MvSB11b]. A tool supports the definition of security constraints employing a predefined security ontology to match security intents of a business process expert with valid security annotations. The security constrains are transformed to implementation artifacts that enable an adapted business process engine to enforce the security constraints [MvSB11a, MB11].

A model-driven approach from Wolter et al. centers on access control for electronic business processes [WSM07, WMM08, WMS+09]. Business process experts annotate security goals to business process models specified using BPMN. To validate an annotated business process model, it is translated into the intermediate Protocol Meta Language (ProMeLa) and processed using the model checker SPIN [Hol97]. Model markings for different enforcement components allow for a transformation of the annotated business process model to configuration artifacts for that components. The transformation into XACML authorization policies and Apache Rampart[8] configuration files has been demonstrated so far.

Security requirements engineering for electronic business processes is addressed by Rodríguez et al. [RFMP06, RFMP07b, RdGFMP10, RFMTP11]. Their approach defines an UML Profile named Business Process Security (BPSec) to annotate business processes specified using UML Activity Diagrams. The profile has also been translated for the use with BPMN [RFMP07a]. A corresponding Method for Business Process Security (M-BPSec) provides guidance for the elicitation of security requirements of business processes and the application of BPSec. Tools support the transformation of annotated process models to analysis models. BPSec allows for tagging of specific elements of UML Activity Diagrams in order to annotate stakeholder needs with regard to access control, attack detection, auditing, integrity, non-repudiation and privacy. M-BPSec considers four stages: a construction activity to model the business process, an activity to annotate the business process using BPSec, a refinement activity to further detail the security annotations, and a transformation activity to generate

[8] Apache Rampart is the security module of the Apache Axis 2 Web Service Stack (`http://axis.apache.org/axis2/java/rampart/`).

UML Use Cases and Class Diagrams from the annotated UML Activity Diagram. A description of necessary roles, work products, and their dependencies completes M-BPSec.

Not equivalently detailed is the approach from Lopez et al. for security requirement engineering [VML03, LMV+05]. They propose the application of a general "security" tag as UML Stereotype for classes that should be attached to business process specifications detailing the specific security aspect as class name (e.g., security goals like confidentiality of the respective element), attributes (e.g., specifying data that should be protected) and methods (e.g., specifying activities that should be audited).

The Business Process with Service Level Agreements (BP&SLA) approach presented by Frankova et al. [FSG+11] reformulates security engineering as engineering of service level agreements (SLAs). BP&SLA provides a method to define SLAs to be signed in order to guarantee a secure operation of a web service-based business starting with high-level requirements analysis. At first, informal security requirements provided by domain experts are formalized using Secure Tropos [MG07] and an indicator-based trustworthiness model is introduced to reflect the compliance of each participant with the requirements. Intermediate models are derived from the requirements model as well as trust indicators. Together, they enable a comparison of different service providers with regard to compliance and to specify a hierarchy of local process models utilizing the Secure Business Process Execution Language (Secure BPEL, [FMS07]). Finally, a constraint system is build navigating recursively the intermediate models and a set of solutions is identified using constraint propagation for an acceptable security level.

Badr et al. present a comparable approach annotating service descriptions with required controls to generate security enhanced SLAs called protection level agreements [BBT11]. To identify security goals, assets are classified as public, restricted, private. A given set of security goals is assigned to every asset depending on the respective class. Analogous, security controls are assigned to each service description depending on the assets it is processing or utilizing and their security goals. An adapted implementation of the open source enterprise service bus (ESB) Petals[9] analyzes and enforces the protection level agreements specified for each service.

The integration of risk assessment and BPM is addressed by the Risk-Oriented Process Evaluation (ROPE) methodology presented by Jakoubi et al. [JTQ07, JT09, JTGK10, TJG+11]. To allow for an analysis, ROPE refines business process models analyzing dependencies between process activities and their resources and environment first. Secondly, threats, their impact on resources and the environment of an activity, and mitigating controls are investigated and documented. A formalization of the relations between threats, controls, and business process activities allows for a simulation of different control set selections as well as impact analysis of threats that can be utilized in the security engineering process.

Finally, several approaches for the integration of (security) risk assessment and business process management have been proposed, addressing information security management instead of security engineering and therefore will not be covered here (e.g., [KH10, CKE+10]).

A large body of work covers the analysis of business processes focusing on their security properties. Integration of security requirements and their verification with regard to business process models is addressed in the approach from Backes et al. [BPW03]. It is based on a

[9] http://petals.ow2.org/

probabilistic model of reactive networks expressing and analyzing cryptographic protocols. Business processes, security requirements, and control specifications are translated into such probabilistic models and formal verification of the models is applied. One of the main observations of the authors is the complexity even of small process models that renders formal verification of larger process models impossible. An authorization model with a set of invariants for the analysis of electronic business processes is proposed by Hung et al. [HK03]. Based on a multilayered state machine the invariants of the authorization model are analyzed and the process specification is deemed secure if they hold. Similarly, Armando et al. analyze business process specifications under authorization constraints [AGMP12]. They apply an action-based language to specify authorization constraints to allow for a number of reasoning tasks including checks whether the execution might violate the authorization constraints. Addressing more general security goals and procedural controls for business processes, Arsac et al. present a validation approach that employs model checking techniques that interact directly with the business process expert [ACPP11]. The business process expert specifies so called security desiderata such as dual control graphically within a BPMN business process model and a transformation of the annotated model is checked with regard to the security desiderata. The approach of Weldemariam et al. introduces procedural security analysis to assess the security of business processes [WV11]. Based on the formalization of asset flows specified by the business process and possible threats, the NuSMV symbolic model checker [CCG+02] is applied to identify security-related deficiencies of the current process specification.

2.5.2. Approaches for Model-based Security Engineering

Although security engineering is considered an immature discipline, a large body of work has been published in this field. This section reviews shortly related surveys and sketches general model-based approaches for security engineering that might be applied to electronic business processes as well.

An early survey on security engineering analyzing the application of models is provided by Baskerville [Bas93]. Comparing a number of academic and industrial approaches, Baskerville identifies three generations of methods and characterizes them as checklist methods, mechanistic engineering methods, and logical-transformational methods. The latter methods employ analysis and design models abstracting the security problem and possible solutions. Therefore, third generation approaches overcome oversimplifications and limitations in comparison to first and second generation approaches. Unfortunately, third generation methods at that time exhibited lack of practical applicability and (industrial) experience. Addressing industrial security engineering practices, Vaughn et al. conclude that only very few model-based approaches are applied in industry [VHF02]. Following their analysis, reasons for this lack of application include immaturity of the (scientific sound) approaches, (inappropriate) need of sophisticated experts and considerable investments for the specific approaches, and absence of integrative approaches and integrated results. A recent survey by Jensen et al. focuses model-driven approaches in security engineering [JJ11]. From the five approaches considered to be relevant in this field (filtered out of 2844 candidate publications), three of the approaches

are covered in previous sections [BDL06, RFMP06, HB09], another is briefly presented in the following paragraphs [MRS09], and one approach does only apply to data warehouses.

A prominent example of model-based security engineering is an approach based on the UML extension UMLsec presented by Jürjens [Jü05]. UMLsec provides an UML Profile to specify security goals and assumptions: UML Stereotypes are used together with UML Tags to annotate model elements with security goals. Constraints provide criteria that determine whether security goals are met by the annotated UML models. The semantics of UMLsec is defined on the basis of so-called UML Machines utilizing a notation similar to Abstract State Machines (ASMs) [Gur95]. UMLsec has been applied to electronic business processes (manually) annotating UML models on different levels of abstraction in order to validate security goals for annotated models [Jü01] or to support security risk analysis [TJ08].

The development of (security critical) smart card applications is focused with the SecureMDD approach presented by Moebius et al. [MRS09, MSR10]. Applying the SecureMDD approach, the developer designs the system under development with the help of UML models. A security expert annotates those design models using the SecureMDD UML Profile. The UML Profile provides means to express some general controls such as encryption and hashing. In a next step, cryptographic protocols applied to the system under development are modeled with the help of UML Activity Diagrams and the Model Extension Language (MEL) . The extended UML models (PIMs) are then transformed applying either a Java Card or a formal ASM platform model. The resulting PSMs are then further transformed to Java sources or ASM specifications (implementation model). Using the ASM specifications, the coherent application of the controls can be validated.

An approach for model-driven development of access control infrastructure artifacts as well as analysis of the specification models is presented by Basin et al. [BDL06, BCDE09]. They propose a DSML to specify access constraints for UML models called SecureUML and apply SecureUML utilizing the dialect definition strategy described in section 2.3.3.3. Transformations are provided to generate access control infrastructure artifacts for Enterprise Java Beans (EJBs), Java Servlets, and Microsoft Enterprise Services for .NET. Properties of the specification models are further analyzed using Object Constraint Language (OCL) [Obj14b] expressions.

Model-based security analysis is focused by the CORAS approach from Lund et al. [DRR+02, dBDG+03, dBHL+07, LSS11]. CORAS provides a compact process model comprising seven activities to identify assets, to describe the system and its environment, to identify, refine, and evaluate threats and vulnerabilities, and to depict appropriate countermeasures. Results are modeled using a graphical DSML. Secure design is not at the heart of CORAS. More formally, Wimmel provides a model-based method for the specification of security requirements, validation of security architectures, and the generation of security test cases [Wim05].

Several approaches have been presented that aim at integrating security assessment and security engineering. Especially the representation of security analysis and security design models based on concepts provided by the CC and accompanying instructions for the application of such models have gained some interest. Vetterling et al. present a secure system development process integrating activities and documents from the CC [VWW02]. A systematic approach to develop a security architecture based on CC concepts is presented by Whitmore [Whi01]. Keblawi et al. report their experiences in specifying and analyzing security requirements in large systems using CC system-level protection profiles [KS06]. Require-

ments engineering based on CC is also covered by the approach of Mellado et al. [MFMP07]. Experiences applying such CC-based approaches in industry are reported by Sharp [Sha09].

Resting on proprietary specification and execution requirements, Eckert et al. describe an early approach for the development of secure applications [EM97]. Security properties in terms of information flows and access restrictions are formally specified and transformed into statements using the programming language INSEL+. The resulting application is executed in a dedicated distributed environment. Other early approaches encompass a method by Salter et al. based on attack trees [SSSW98]. Similarly, more recent approaches center on threat models and resulting risk considerations [EW05, EYZ10]. Several approaches address only selected security engineering activities or properties [MBSFM10], e.g., the SQUARE methodology provides an approach for security requirements engineering [MHS05], privacy engineering is discussed by Spiekermann et al. [SC09], a method to engineer access control mechanisms is provided by [Pop05].

2.5.3. Discussion

The large body of work reviewed in the preceding sections demonstrates impressively the active research that is related to the topic of this thesis. For a discussion of the approaches and to elaborate the distinction with regard to our contributions we will refer to the problem statement and general solution ideas (cf. sections 1.2 and 1.4).

In our problem statement we focused on three major issues with regard to security engineering in the domain of BPM: security nonprofessionals deciding and implementing security, the heterogeneity of (the security configuration of) business process engines, and the environmental heterogeneity including the application of different engineering and development methods. Our general solution ideas included the specialization of general security engineering methodologies for their application in the BPM domain, the provision of a DSML to capture resulting artifacts, and the separation of security-knowledge intense (preparatory) activities and other activities detailed with detailed guidance in a way that domain or business process experts are able to execute them (cf. section 1.4).

Most of the approaches for security engineering in the domain of BPM address security professionals. AURUM, ProSecO, and FML directly assign security experts to most activities. POSeM envisions close cooperation between security professionals and nonprofessionals. MoSSBP reflects the participation of nonprofessionals and provides means tool-based support them. Nevertheless, the latter approaches assign recurring activities to security professionals as well. Focusing the collaboration of security professionals and nonprofessionals in order to lighten the skill set needed for security engineering activities, Wolter et al. come closest to our idea [WMS+09]. Nevertheless, their approach covers only access control. General model-based security engineering approaches like UMLsec, SecureMDD, SecureUML, CORAS and CC-based all address security experts, mostly in collaboration with domain experts. Additionally, most of the security engineering approaches do not provide sufficient guidance to reproduce examples or apply the approaches unescorted.

Approaches that bridge between (executable) process definitions, proper design of controls, and their configuration supporting different business process engines are not available at the moment. AURUM does not address technical details at all, FML provides only a conceptual

description but lacks actual implementations. POSeM only sketches mappings from controls onto control implementations but does not elaborate them. MoSSBP does not reveal design and implementation of transformations from control design into implementation artifacts described in the approach. SECTET focuses Web Service environments and does not support any business process engine directly. Other approaches support only specific aspects such as access control or do not necessarily utilize business process models, e.g., the approach from Wolter et. al. and SecureUML.

Likewise, only very few approaches address the environmental heterogeneity and the integration of security engineering in existing development process models and tooling. One exception is den Braber et al. discussing the integration of CORAS into the Open Unified Process (OpenUP) [dBDG+03]. Approaches to augment development process models with general security engineering activities that do not rest on elaborated approaches for security engineering are provided by [PH07, AS08] for the Rational Unified Process (RUP) and by [Bez03, CPG05] for agile approaches.

In summary, a large body of work discusses approaches and techniques for security engineering, some of them addressing the application of security engineering in the domain of BPM. None of the existing approaches jointly integrates means to lower the skill set necessary to complete the recurring engineering activities enough, bridges between (executable) process models, proper design of controls, and their configuration, and demonstrates their applicability in different organizational and technical environments. Additionally, a study from Siponen and Heikka suggests that existing methods for security engineering do not provide adequate modeling support which is seen as crucial requirement for such methods by the authors [SH08].

Nevertheless, several approaches discussed in this chapter provide valuable insights that guide this thesis. To name just some of them, approaches for MDS from Basin et al. and Möbius et al. paved the way for high-level security goal or requirement specification and derivation of implementation artifacts utilizing DSMLs [BDL06, MRS09]. In the area of secure electronic business processes Hafner et al., Röhrig et al., and Wolter et al. addressed the needs in the domain of BPM [RK04, HB09, WMS+09]. Providing detailed guidance for their approaches and discussing integration in existing development process models and approaches den Braber et al. and Popp have been useful [dBDG+03, Pop05]. Last but not least, application and industry reports from Clavel et al., Dhillon et al., and Geer proved helpful to consider the needs in the field [CdSBE08, Gee10, Dhi11].

2.6. Summary

This chapter introduced necessary knowledge for the state of problems and current solutions as well as central concepts and terms. Main areas of interest have been BPM, ME, MDE, and security engineering.

Section 2.2 introduced BPM concepts and terminology. Our thesis addresses security engineering of implemented and automated business processes that we call electronic business processes. Our security engineering process model presented in chapter 4 rests on general process models and allows for integration into (business process aware) software development

process models. Within the life cycle of business processes we focus the Design & Analysis as well as the Configuration phase. Explicit representation of business processes necessary for their automation is addressed by business process modeling. BPMN in its version 2.0 provides good support for all phases of the business process life cycle, is well recognized in academia and industry, and offers good comprehensibility despite of its rich semantics. Thus, we will use BPMN in the following chapters as notation for business process models.

ME as the discipline for the design and adaptation of methods for the development of information systems has been sketched in section 2.3. Core idea of ME is the systematic provision, selection, and assembly of method fragments from a repository in order to provide adequate methods. SPEM provides a metamodel and a conceptual framework in order to support ME. It is standardized by the OMG, provides rich modeling capabilities and is supported by mature tooling. We apply SPEM in chapter 4 to model our process model SecEPM and integrate SecEPM in development process models utilizing SPEM.

MDE is an approach to tackle the semantic gap between different domain-specific concepts necessary for the development of (complex) software systems. Central ideas of MDE are direct representation of problems and solutions using DSMLs, automation of the translation of the resulting models, and interoperability by the application of open standards. Therefore, MDE combines the application of multiple DSMLs based on metamodels with tooling necessary for specification, management, and transformation to allow for a systematic development and application of models in the domain of software engineering.

The ideas of MDE have been picked up with regard to security coining the term MDS. MDS aims at modeling security aspects on an adequate level of abstraction using one or more DSMLs and transforming those models into security-related artifacts, e.g., access control policies. Our process model SecEPM presented in chapter 4 rests on business process models, chapter 5 introduces with SecEML a DSML utilizing the entity reference strategy to model work products of SecEPM and to allow for transformations into other necessary work products.

To provide a solid base for our security engineering framework, section 2.4 discussed common definitions in the domain of IT security and provided definitions of important concepts and terms. All of these concepts and terms will be reflected in SecEML to capture work products of SecEPM (cf. chapter 5). Security engineering focuses activities to establish security for systems and provides systematic approaches to apply those activities. We call a representation of all activities and work products necessary to augment traditional software development process models in order to build secure systems a security engineering process model.

Section 2.5 presented existing approaches for security engineering of electronic business processes and model-based security engineering respectively. A discussion of the existing approaches revealed that none of the existing approaches addresses all challenges focused on in our problem statement in section 1.2. Most of the approaches address security professionals, do not bridge process models, proper design of controls, and their configuration, and do not provide enough flexibility to cope with existing environmental heterogeneity.

3. Running Example: The Replan Process

3.1. Introduction

In order to develop a framework for security engineering of electronic business processes, the previous chapter introduced concepts and terminology with regard to BPM, ME, MDE, and IT security. This chapter presents a real world business process from the logistics domain—the Replan Process. The Replan Process serves as running example for the development of the security engineering process model SecEPM in chapter 4 and the DSML SecEML in chapter 5. SecEPM activities will be motivated and explained using the Replan Process, work products modeled using SecEML will reference entities from the Replan Process. Furthermore, the Replan Process will be used in the exemplary study in chapter 6 to demonstrate the feasibility of the framework developed in this thesis. The continuity in using the Replan Process enables us to concentrate on security engineering aspects as well as explanation and examination of the framework.

The following section 3.2 introduces the Replan Process, provides background information on its purpose and application, and comments on the business process model of the Replan Process. Section 3.3 summarizes briefly the results of this chapter.

3.2. Background, Application, and Business Process Model

The Replan Process is part of a business process chain to implement door-to-door delivery of shipments around the world operated by a large provider of integrated logistics services. Such world-wide delivery of shipments are subject of an initiative by the International Air Transport Association (IATA) called Cargo 2000[1]. The objective of the Cargo 2000 initiative is to provide a transportation scheme backed by quality standards in order to measure and improve the efficiency of air cargo. With the help of the quality standards provided by Cargo 2000 the performance of logistics providers can be compared. Therefore, the Cargo 2000 initiative increases the pressure on individual logistics providers to adapt their business processes in order to improve their service quality.

One important challenge with regard to existing business processes implementing world-wide door-to-door shipments is the separation of business processes and activities of individual participants. Therefore, logistics providers often acquire status information about their shipments only at transfer points between participants. For example, delays by a freight forwarder are often noticed after they reach the next stop in their route map. This individual delays might lead to global delays of the shipment and therefore decrease the quality of the

[1] `http://www.iata.org/whatwedo/cargo/cargo2000/`

service offered by the logistics provider. To allow for a proactive intervention by the logistics provider in case of delays (or other deviations from the current planning), continuous monitoring of the shipment is desirable.

The Replan Process focuses on such proactive intervention. Status data from the freight forwarder is continuously monitored by the logistics provider. In case of any deviations from the current planning by given thresholds alternatives are calculated, approved by the dispatcher, and transmitted to the freight forwarder. In the course of the research project Alliance Digital Flow of Goods (Allianz Digitaler Warenfluss, ADiWa[2]) sponsored by the Federal Ministry of Education and Research in Germany the Replan Process of a logistics provider has been re-engineered to allow for better adaptability and service quality applying wireless sensors, communication and BPM.

The business process model of the Replan Process is depicted in figure 3.2.1 on the next page. It entails the interactions between the freight forwarder (Pool "P2") and the logistics provider (Pool "P1") after pickup of shipments and before arrival at the shipments destination. Each Pool is divided in two Lanes: Lane "L11" represents the dispatcher responsible for the routing of the shipments, Lane "L12" the IT system from the logistics provider, Lane "L21" the on-board unit (OBU) of the fright forwarder, and Lane "L22" the driver responsible to transport the shipments to the next destination.

The logistics provider receives status messages (Event "E11", Message "M1") at regular intervals from the OBU of the freight forwarder with regard to the actual position of the truck and possibly further sensor data like temperature or vibrations. The logistics system checks whether the status of the shipments conforms to the planned values (Task "T11"). If the status is OK (Gateway "G12"), the electronic process terminates (Event "E12"). Otherwise, the logistics system identifies shipments that are at risk with regard to their service level agreements (Task "T12"). Alternative route proposals for the shipments at risk are calculated to reduce the risk (Task "T13"). The dispatcher reviews the proposals manually and selects the best alternative (Task "T14"). The logistics system transfers the new routing information to the driver (Message "M2", Task "T15") and the electronic process terminates (Event "E13"). The driver manually accepts the new route plan (Task "T22").

Market pressure on the logistics providers and consecutive re-engineering of the Replan Process to achieve better adaptability and service quality are typical phenomena that BPM addresses. From a security point of view, the re-engineering introduces new risks to the logistics provider. After re-engineering, the Replan Process relies on external sensor data that is transmitted over the air, involves different administrative districts coupled by IT technology, communicates potential sensitive routing information, and includes human interaction. Manipulation of the enactment of the electronic business process like message blocking, tapping, or tampering as well as unauthorized access might lead do service level degradation, contractual fines, loss of goods, information disclosure, and other unwanted results. Therefore, security aspects of the electronic business process must be carefully analyzed and necessary mitigation of new risks must be considered and enforced. Furthermore, re-engineering of business processes utilizing BPM does apply frequently to multiple business processes and

[2] `http://adiwa.net/`

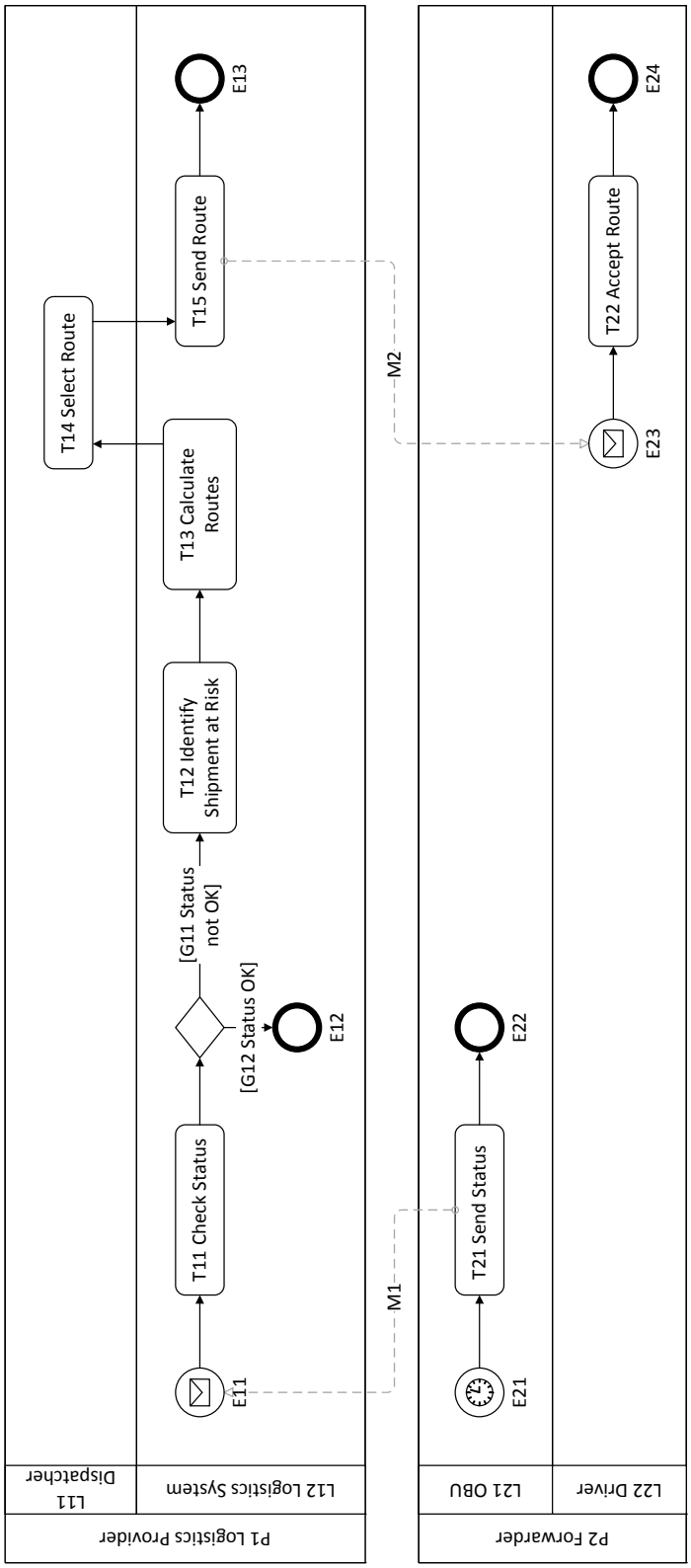

Figure 3.2.1.: The Replan Process (BPMN process diagram)

therefore have to utilize resources efficiently. The security engineering framework proposed in this thesis addresses these issues. Hence, the Replan Process is a good example to illustrate and explain the framework as well as to demonstrate its feasibility.

3.3. Summary

This chapter introduced the Replan Process as running example for the subsequent chapters. The Replan Process is a real world electronic business process supporting a logistics provider to continuously monitor its shipments and to proactively intervene in case of delays or disturbances.

The business process model of the Replan Process entails a collaboration between two participants (the freight forwarder and the logistics provider) exchanging status messages considering the current position and status of the freight forwarder and route messages providing new route plans in case of critical delays and disturbances. As the Replan Process relies on external sensor data that is transmitted over the air, involves different administrative districts, and includes human interaction it exhibits interesting properties with regard to security.

The next chapters will use this example to illustrate the security engineering process model (chapter 4) and the corresponding modeling language (chapter 5). it also provides the background for the exemplary study (chapter 6).

4. Security Engineering Process Model

4.1. Introduction

The previous chapter detailed a real-world business process from the logistics domain, the Replan Process. Our proposal to support security engineering for electronic business processes such as the Replan Process is presented in this chapter. It addresses the development aspects of the first research question and documents parts of the results of the design and development activity of our research approach (cf. section 1.3):

1. What requirements does a security engineering process model for electronic business processes place that copes with the prominence of security nonprofessionals, heterogeneous business process engines, and environmental heterogeneity in the domain of BPM and how could they be met?

Our framework addresses three major issues as it is sketched in the research question: a restricted skill set and resources available to secure electronic business processes, heterogeneous capabilities and configuration options with regard to security controls of existing business process engines, and a heterogeneity of the environment with regard to engineering methodologies and development process models.

This chapter introduces our Security Engineering Process Model (SecEPM) that consistently integrates security-related activities in the course of the development of secure electronic business processes and provides guidance on how to apply relevant methods and practices systematically. SecEPM focuses the Analysis and Design as well as the Configuration phase of the business process life cycle and details necessary activities, roles, and work products to support security engineering from the identification of security goals to the selection and configuration of controls. Early versions of the requirement analysis for our process model as well as an appropriate design approach and key entities have been published before [Eic12a, Eic12b]. With respect to these publications, the results presented in this chapter have been considerably reviewed, detailed, and enhanced.

The presentation of SecEPM is structured as follows: The next section 4.2 confines the scope of our process model and analyzes key requirements derived from existing process models and the issues detailed in the problem statement (cf. section 1.2). Section 4.3 details guiding principles for the design of SecEPM. An overview of the key entities of our process model as well as their grouping and relations is given in section 4.4. Section 4.5 describes the activities covered by SecEPM providing information on purpose, necessary tasks, preconditions, results, related roles, and work products. Examples for (optional) guidance artifacts and methods to support certain tasks in the course of the security engineering activities are provided in section 4.6. Authoring, storage, retrieval, instantiation, and integration of SecEPM in different

development process models is discussed in section 4.7. A summary of the main results concludes this chapter.

4.2. Requirements

In order to analyze key requirements for SecEPM, the scope of the process model needs to be confined first. As it is detailed in the research objective (cf. section 1.3), SecEPM is part of a security engineering framework that bridges the gap between business process models and the design of proper controls and their configuration using means provided by the respective business process engine and its execution environment.

Therefore, SecEPM includes only security-related activities during development of electronic business processes. With regard to the business process life cycle, the phases Analysis and Design as well as Configuration are addressed by SecEPM. The process model presented in this thesis focuses on actual electronic business processes represented by (executable) business process models. Security goals, requirements, and controls are analyzed and designed in the course of the application of SecEPM utilizing a business process model that might get refined and detailed in the course of the supported life cycle phases. In other words, SecEPM supports the development of a security analysis model (as representation in the problem domain) for a given business process model, the creation of a matching security design model (as representation in the solution domain) and a mapping of the security design model onto the implementation.

SecEPM excludes several aspects of the BPM and security engineering discipline. First of all, as only the gap between business process models and the design and configuration of proper controls is addressed by SecEPM, the phases Enactment as well as Evaluation of the business process life cycle covering operation and optimization are not considered here. Furthermore, complementing security considerations and measurements addressing aspects like service security, operating system security, network security, and physical security are not part of SecEPM. Analysis and definition of further organizational measurements and plannings like emergency response planning are not included. The business process models provided as starting point and central artifact for SecEPM are not subjected to major changes, a business process re-engineering perspective is not taken by SecEPM. Nevertheless, different design alternatives for the same business process might be compared with regard to necessary security controls and assumptions utilizing SecEPM.

Key requirements for SecEPM originate from two sources: requirements that have been explicated for matching security engineering process models as well as requirements addressing the major issues detailed in the problem statement stemming from the application of security engineering in the domain of BPM. References are provided to facilitate tracing of the requirements. Unfortunately, requirements for security engineering process models have been explicated rarely. Therefore, some of the requirements are taken indirectly from existing process models.

The structure of our process model is given by definition of a security engineering process model (cf. section 2.4.2):

1. Structure: The process model must describe activities, roles, work products, and their relations that are necessary for augmenting traditional development process models in order to build secure electronic business processes.

Activities covered by different security engineering approaches for electronic business processes vary slightly[1]. POSeM elaborates four activities: definition of security goals, refinement of security goals, consistency analysis, and deviation of generic controls [Rö03]. ProSecO details six activities: create or adapt the functional view, define security goals, identify dependencies, refine security goals, threat and risk analysis, and design of controls [HB09]. AURUM collapses the prescribed tasks into three activities: modeling and identification, risk assessment, interactive selection of controls [NP10].

In the case of SecEPM, a functional view is provided by the business process model. Therefore, a creation of this view is not necessary. In order to cover at least the activities provided by the named security engineering approaches and to fulfill the aim of bridging the gap between business process models and the design and configuration of proper controls, we identify the following requirement with regard to coverage[2]:

2. Coverage: The process model must cover at least activities for the assessment of security goals, the elicitation of security requirements, threat modeling and prioritization, as well as control design and configuration.

General requirements that are stated commonly for security engineering process models entail the separation of requirements and controls, traceability, and the integration with different development approaches (cf. [BBH+03, Sip05, HB09, Jü09]).

In software engineering, it is a well known strategy to separate the problem from the solution domain [BD09]. This is equally important for an effective security engineering process. This separation is a precondition to consider different solution approaches and to compare them with regard to a common set of requirements. Furthermore, security controls possibly introduce accompanying requirements themselves and cover more than one requirement. Also, they may become effective only in combination with other controls.

3. Separation of Problem and Solution Domain: The process model must allow to separate activities and work products related to the problem domain from those related to the solution domain.

Traceability is the ability to describe and follow the life of (software) artifacts [WvP10]. Although traceability is considered an emerging discipline, it is crucial for successful security engineering: To validate security requirements as well as security control design and implementation, the dependencies between those entities established in the course of the security engineering activities need to be traceable. Also activities not included in SecEPM like incident and impact analysis rely on traceability.

[1] We continue to use the terminology given in section 2.4.1 instead of the diverse original terminology to allow for a better comprehensibility and comparability.

[2] The term coverage within this thesis has a different meaning than the term code coverage: Requirement 2 specifies the scope for the process model in terms of necessary activities that need to be covered by the process model. In contrast, code coverage is a measure to describe the degree to which the source code of a program is tested by a particular test suite.

4. Traceability: The process model shall foster traceability of work products in order to trace security aspects from high-level security goals to control configurations and vice versa.

Security engineering process models cover only security related activities in the course of a development project. For an effective application, the integration in at least one development process model or approach is necessary. Furthermore, only if the security engineering process model might be integrated into different development approaches (e.g., waterfall or agile approaches) it is flexible enough to cover heterogeneous environments and project types.

Therefore, this requirement also addresses the issue of business process environmental heterogeneity stated in the problem statement. It reflects the needs stemming from the application of security engineering practices in the domain of BPM (cf. section 1.2). The integrability requirement includes the ability to tailor the security engineering process model for different development approaches, environments, and project types. It also addresses the need for flexibility in the application of the process model, e.g., with regard to the sequence of activities. The environmental heterogeneity also requires an independence from a specific technology or tooling—excluding general BPM components—necessary to apply the security engineering process model.

5. Integrability: The process model must integrate into different development process models or approaches.

6. Independence from Development-time Technology: Beside general BPM components, the process model should not need specific tooling in order to be applied.

A related requirement stems from the heterogeneity of business process engines. A security engineering process model must take those differences into account and avoid to overspecialize with regard to a specific engine or security configuration but also support stakeholders to bridge the gap from a general security control design to specific security configuration artifacts.

7. Independence from Runtime Technology: The process model must allow to use different business process engines in different runtime environments.

The final requirement addresses the current situation focusing on business process experts together with domain experts in order to execute most of the activities in the development of electronic business processes. The resulting impairment of skill sets available and necessary for existing security engineering approaches in the domain of BPM needs specific consideration.[3]

8. Restricted Skill Sets: The process model must enable security nonprofessionals to execute as much of its activities as possible without the presence of security experts.

[3] Also in the general security engineering community a need for the consideration of restricted security skills in the course of security engineering activities is getting some voice [Gee10].

4.3. Design Approach

To meet the requirements detailed in the preceding section, several strategies and principles are applied for the design of SecEPM. These strategies take proposals from the method engineering community into account that address the construction of method chunks and fragments from existing methods [Ral04], the construction of methods from those chunks [RDR03], and the tailoring or method configuration [ÅWK+03, KÅ04, KÅ09]. The following paragraphs describe three key strategies, depict decisions taken with regard to the structuring of the activities, and highlight some principles that guided the development of our process model.

The design of SecEPM rests on three main strategies:

1. Specialization (of existing process models and practices)
2. Separation of Concerns
3. Decoupling (of activities)

The Specialization strategy aims at using knowledge and experience condensed in existing approaches but restricting the skill set necessary to complete activities as well as focusing on imminent necessary activities. Most prominently, the proposal from Breu et al. for a formally based security engineering model and its refinement provided with ProSecO prepares a foundation of SecEPM [BBH+03, HB09]. Furthermore, the largely applied IT Baseline Protection Methodology (IT-BPM) [Bun08] provides supplemental activities and guidance to augment SecEPM.

Some examples for the application of the Specialization strategy are sketched in the following. Details are provided in the next sections, especially sections 4.5 and 4.6. Similar to the given approaches, a representation of the functional aspects of the system is the starting point for SecEPM. More specific, not a generic enterprise model is taken as starting point but a business process model. Analogously, SecEPM takes an asset-oriented perspective first and identifies and interrelates assets represented in the business process model. To allow for a comparably restricted skill set, detailed steps are provided on how to analyze the business process model in order to identify and relate assets represented in the business process model. Additionally, detailed guidance is provided that allows to tailor the security engineering process to meet the available resources and to provide the necessary depth.

The Separation of Concerns strategy aims at the identification and separation of different areas of interest and is applied with regard to several aspects of our process model. First, security-related tasks that require security experts have been separated from tasks that might be executed by other participants with sufficient guidance provided. Mainly, those tasks have been concentrated in a preparatory activity in SecEPM. For example, threats and their impact on electronic business processes are collected, analyzed, and classified by an security expert in a preparatory task in order to provide a threat catalog for the business process expert including application constraints for the instantiation of threat classes.

Second, treatment of conceptual and implementation aspects have been separated as well. For instance, elicitation of the security requirements and conceptual design of proper controls

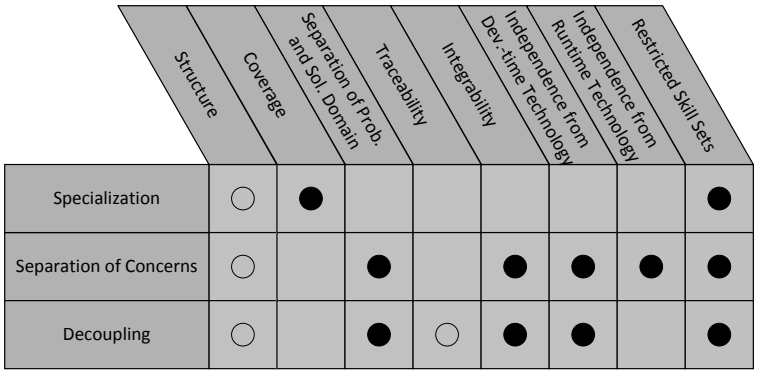

Figure 4.3.1.: Design strategy vs. requirement matrix for SecEPM

is separated from individual capabilities of a given business process engine. Also, application of security controls for a business process engine is separated from its configuration details.

Third, activities and work products addressing the problem domain are separated from activities in the solution domain. Exemplary, identification, detailing, and modeling of requirements are comprised in an activity for the elicitation of requirements opposed to the design of proper controls covered by another activity.

Fourth, activities are separated from guidance artifacts. Activities are restricted to provide the backbone for the security engineering process. They describe aim, intention, participating roles, work products, and tasks necessary to achieve the aim. Guiding artifacts explain how to fulfill one or more given task, provide supportive templates and examples, and point out alternatives in the course of the security engineering process. Exemplary, the activity for the identification of assets details tasks in order to provide a set of interrelated business assets and their resources represented in the business process model. In contrast, a respective guidance artifact describes detailed step-by-step instructions that might be followed in order to separate assets and resources and proposes criteria to interrelate individual assets. Applying this separation, SecEPM becomes more flexible as different methods can be applied within SecEPM providing different guidance artifacts.

While the Separation of Concerns strategy is applied mainly in order to guide the structuring of elements of the process model, the Decoupling strategy aims at relaxing constraints of the process model with regard to the execution sequence of activities. Hence, the Decoupling strategy focuses pre- and postconditions of activities. As development approaches apply development activities differently (e.g., serial vs. iterative application), decoupling of activities allows for a flexible configuration of a SecEPM-based development process. This is especially important for agile approaches, that generally do not provide a given sequence of activities but push the decision of an optimal execution time to the responsible stakeholder. Examples for the application of the Decoupling strategy include the design of activities for threat modeling and security goal assessment that might be executed in an order decided by a process model expert or iteratively in order to focus on high value goals.

Figure 4.3.1 depicts contributions of design strategies in order to meet the requirements detailed in section 4.2. Filled circles denote direct contributions, empty circles indirect contri-

butions. The Specialization strategy addresses the Coverage requirement 2 as it specializes process models that cover the named activities and mainly the Restricted Skill Sets requirement 8 as it narrows the skill set necessary for the individual activities. The Separation of Concerns strategy addresses the Separation of Problem and Solution Domain requirement 3, the Integrability and Independence from Development-time requirements 5 and 6, the Independence from Runtime Technology requirement 7, and the Restricted Skill Sets requirement 8. The Decoupling strategy contributes to the Separation of Problem and Solution Domain requirement 3 as decoupled actvities allow for an easier separation, the Integrability and Independence from Development-time requirements (5, 6) as decoupled activities are easier to integrate and allow for the application of different tooling, and the Restricted Skill Sets requirement 8. The Decoupling strategy indirectly addresses the Traceability requirement 4 as it allows for an independent treatment of the respective work products of the decoupled activities. All strategies contribute indirectly to the Structure requirement 1.

Additional guidelines for the design of SecEPM include a focus on explicit modeling and early steps in the development life cycle. The focus on explicit modeling is related to the goals of MDE and their major means: direct representation, automation, and open standards (cf. section 2.3). The design of SecEPM is aligned with the application of one or more DSMLs in the course of the security engineering process. A proposal for an applicable DSML named SecEML is provided in chapter 5.

Our focus on early steps in the development life cycle addresses a need for applicable methods that has been stated by Anderson and others: *"Security requirements engineering is often the most critical task of managing secure system development, and can also be the hardest. [...] The available methodologies have consistently lagged behind those available to the rest of the system engineering world."* [And08, p. 834]

This section presented our major design considerations: the application of the Specialization, the Separation of Concerns, and the Decoupling strategies as well as our focus on explicit modeling and early steps in the development life cycle. The following section introduces SecEPM and discusses the application of these considerations.

4.4. Structure

SecEPM is our proposal for a security engineering process model that integrates security-related activities in the course of the development of secure electronic business processes. This section provides an overview of the key entities of SecEPM as well as their grouping and relations.

Key entities of SecEPM are organized using three categories: roles, work products, and activities (cf. Structure requirement 1 and definitions provided in section 2.3.1). Together they address the question: "Who (role) is doing what (activity) with which result (work product) in order to secure an electronic business process?" Additionally, SecEPM provides guidance artifacts including templates, checklists, examples, reference materials, and guidelines. Guidance artifacts explain how to fulfill one or more given task of an activity in the course of the security engineering process.

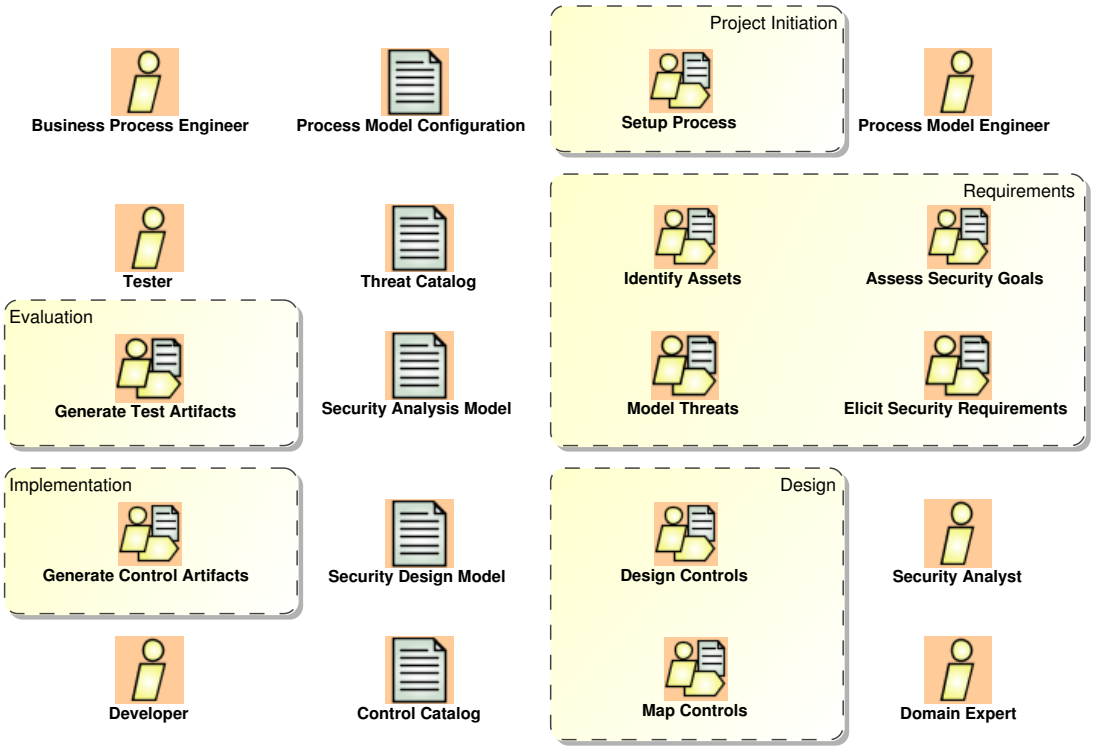

Figure 4.4.1.: SecEPM overview (SPEM notation)

Figure 4.4.1 on the facing page depicts an overview of SecEPM. Activities of SecEPM are grouped following recommendations of the Institute of Electrical and Electronics Engineers (IEEE) using the activity groups Requirements, Design, Implementation, and Evaluation [Ins97]. The *Requirements* activity group comprises the activities *Identify Assets*, *Assess Security Goals*, *Model Threats*, and *Elicit Security Requirements*. These activities structure the security engineering process from the identification of business assets and related security goals, the rating and prioritization of security goals with regard to potential damage, the assessment of threats to those security goals, and the refinement of security requirements from security goals. The *Design* activity group includes the activities *Design Controls* and *Map Controls*. They detail the security engineering process from the identification and selection of appropriate controls and their mapping onto an actual runtime environment. The activity groups *Implementation* and *Evaluation* each cover only one activity: the *Generate Control Artifacts* and *Generate Test Artifacts* activity respectively. They address the generation of implementation and test artifacts. A preparatory activity *Setup Process* is assigned to the activity group Project Initiation. All SecEPM activities comprise several method chunks relating process and product fragments. Details are presented in section 4.5.

SecEPM distinguishes six main work products: Security Analysis Model, Security Design Model, Process Model Configuration, Threat Catalog, Control Catalog, and Runtime Capability Model. Table 4.1 on the next page provides an overview of the work products. Core work products of SecEPM are the Security Analysis Model and the Security Design Model. The *Security Analysis Model* defines the security problem to be solved and is created and updated by the activities in the Requirements activity group. Activities in the Evaluation activity group use the Security Analysis Model in order to validate design and implementation artifacts. The *Security Design Model* depicts a solution of the security problem and describes the controls to be applied as well as their mapping onto the runtime environment. Activities in the Design activity group create and update the Security Design Model that is consumed by activities in the Implementation activity group.

The work products *Process Model Configuration*, *Threat Catalog*, *Control Catalog*, and *Runtime Capability Model* are mainly created upfront a specific security engineering process and reused within several comparable projects. They cover the individual configuration of the security engineering process for a project type or an individual project, catalogs to support security engineering activities, and a description of the capabilities of a BPMS or its runtime environment. Several additional work products are necessary as input for SecEPM activities or result as output that is not considered further by any SecEPM activity. These include the *Information Security Policy*, the (executable) *Business Process Model*, *Test Artifacts*, and *Implementation Artifacts*. Those work products are not represented in the aforementioned figure 4.4.1 on the facing page.

Figure 4.4.2 on page 61 displays the relations between artifacts and work products: filled circles note a produce relation (i.e., an activity creates or updates a work product), empty circles note a consume relation (i.e., an activity uses a work product).

We keep the number of roles in SecEPM pragmatically small. SecEPM distinguishes Business Process Engineers, Developers, Domain Experts, Process Model Engineers, Security

Name	Description	Assignment in SecEPM
Business Process Model	Definition of the electronic business process	Starting point and input for most activities
Control Catalog	Catalog of control classes detailing mitigation possibilities for one or more threat class, application consequences, and relation to other control classes	Supports the design of controls and repeatability of results
Implementation Artifacts	Necessary artifacts for the implementation of the electronic business process excluding the business process model	Configuration of the runtime environment and implementation of controls
Information Security Policy	Definition of high-level security policies, asset classification schemata, and rating criteria	Basis for the Threat and Control Catalog, supports asset identification and security goal assessment
Process Model Configuration	Individual configuration of the security engineering process	Tailoring of SecEPM
Runtime Capability Model	Configuration and implementation alternatives for controls supported by a specific runtime environment	Basis for the mapping of controls onto a selected runtime environment
Security Analysis Model	Description of the security problem	Documentation of assets, security goals, threats, security requirements, and their relations
Security Design Model	Description of the solution to the security problem documented in the Security Analysis Model	Documentation of selected controls and their mapping onto the capabilities of the runtime environment
Test Artifacts	Description of tests and corresponding artifacts to validate the quality of the implementation	Tests to assess the functioning of the controls
Threat Catalog	Catalog of threat classes that endanger security goals including entry or applicability conditions, relation to other threat classes and consequences of successful manifestations	Supports threat analysis for given business process models and repeatability of results

Table 4.1.: SecEPM work products overview

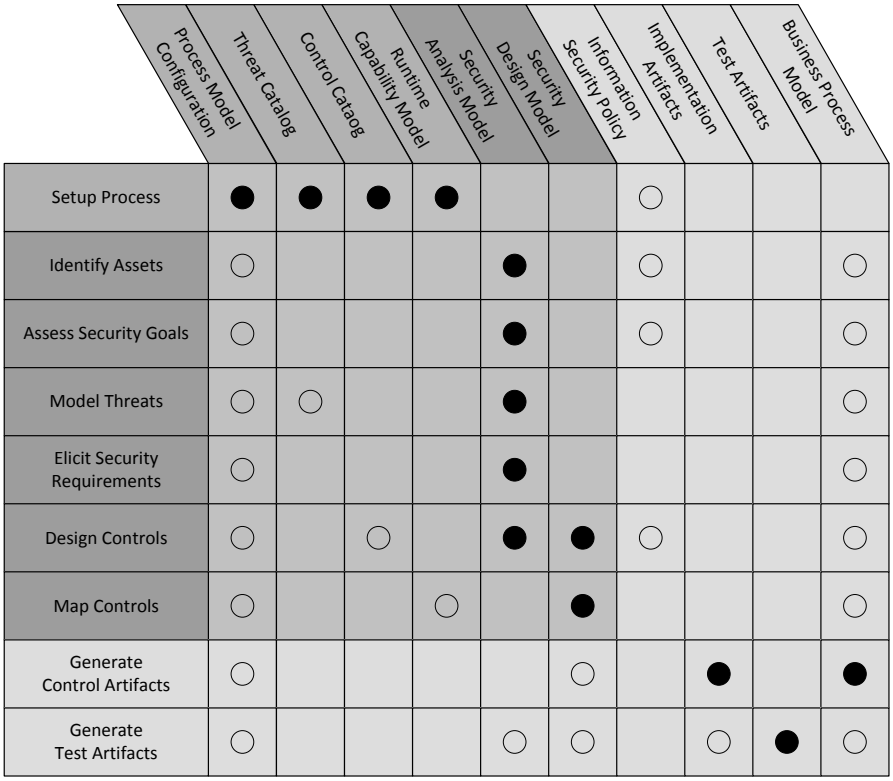

Figure 4.4.2.: Activity vs. work product matrix

Analysts, and Testers. As SecEPM augments development process models, roles from SecEPM complements those from the development process model (cf. section 4.7).

Process Model Engineers are experts for development processes and their instantiation. They tailor SecEPM to the needs of an organization and an individual project. In SecEPM, Process Model Engineers account for the work product Process Model Configuration. *Business Process Engineers* translate business requirements into business process models and support their configuration in order to enact them. In accordance to common practices in the BPM domain, Business Process Engineers carry out most of the activities in SecEPM supported by Domain Experts, Developers, Testers, and Security Analysts [SF03, Wes07]. *Domain Experts* provide knowledge from the application domain of a business process. In SecEPM they contribute to the work product Threat Catalog, support the identification of assets and the assessment of security goals. *Security Analysts* are familiar with the analysis of security problems and their solution. In SecEPM they provide the Threat Catalog (supported by the Domain Expert), the Control Catalog, and detailed guidance for all activities. Furthermore, Security Analysts validate the work products Security Analysis Model and Security Design Model. *Developers* contribute their technological know-how. In SecEPM they provide the work product Runtime Capability Model (supported by the Security Analyst) and support the Business Process Engineer in mapping controls to runtime capabilities and configuring the executable business process model. *Testers* ensure the quality of the electronic business process. In SecEPM they

Name	Description	Focus in SecEPM
Business Process Engineer	Elicit business requirements, translate requirements into business process models, and support business process model configuration	Execute most activities
Developer	Contribute technological know-how with regard to business process model automation and specific runtime environments	Provide Runtime Capability Model, support control mapping and business process model configuration
Domain Expert	Provide knowledge from the application domain of a business process	Contribute to the Threat Catalog, support asset identification and assessment of security goals
Process Model Engineer	Customize individual security engineering processes, enhance process models based on experiences	Provide Process Model Configuration, adopt SecEPM
Security Analyst	Analyze security problems and develop solutions in the domain of electronic business processes	Provide Threat and Control Catalog as well as detailed guidance, validate Security Analysis and Design Model
Tester	Ensure the quality of electronic business process implementations	Generate Test Cases, execute them, and interpret the results

Table 4.2.: SecEPM roles overview

generate the work product Test Artifacts and execute the test cases contained against the developed solution. Table 4.2 summarizes the roles in SecEPM.

Compared to activities, guidance artifacts might vary largely between different instantiations of SecEPM. As they contribute detailed support on how to complete an activity or a certain task, they depend on the organizational environment, the skills available in a given project, and the runtime environment in question. Guidance artifacts include checklists, examples, guidelines, references, and templates. In section 4.6 several sample guidance artifacts are presented and explained.

SecEPM does not prescribe the sequence of activities and does not define completeness criteria for necessary input artifacts (work products) in order to restrict the execution of an activity. Nevertheless, SecEPM defines logical interdependence between entities within work products applying the terminology presented in section 2.4.1. Therefore, regardless of the individual creation process of a work product, full validation of work products becomes possible only if related tasks have been accomplished and the dependencies between the work products are fulfilled. A brief comparison of (idealized) security engineering or evaluation processes following the ProSecO, the CC, and the SecEPM approach might highlight these dependencies.

- ProSecO specifies six activities that are executed sequentially and iteratively (cf. section 2.5.1): 1) a functional view is created, 2) security goals for assets are specified, 3) dependencies between security goals and elements of the functional view are identified, 4) security

goals are refined with regard to dependent and depending assets, 5) threats to assets are identified and evaluated, and 6) controls are chosen that satisfy the security requirements.

- The CC evaluation process specifies the following sequence of activities (as a part of the whole evaluation process, cf. [Int14]): 1) analysis of threat classes, 2) specification of assets and security goals, 3) derivation of security requirements (including a traceability matrix between threats and respective security goals).

- SecEPM starts with the 1) identification of assets, 2) the assessment of security goals, 3) the analysis of threats, 4) the derivation of security requirements, and 5) the design as well as 6) mapping of appropriate controls.

All given approaches take an asset-oriented perspective in the beginning and try to identify security goals for those assets. Differences exist with regard to the role of threats. ProSecO does not explicitly differentiate between security goals and requirements (according to our terminology). Threat analysis is applied to evaluate risks and prioritize security goals. CC focuses on threats as security requirements have to address an analyzed threat (no requirement exists if no threat has been identified threatening a given security goal). With regard to this aspect, SecEPM takes the perspective of the CC: Only if a threat has been identified, a security requirement has been validly specified. Therefore, a logical dependency exists between assets, security goals, threats, security requirements, and controls that must be fulfilled in order to validate the Security Analysis Model and the Security Design Model that capture those entities.

A brief alignment of SecEPM with the concepts CIM, PIM, platform model, and PSM from MDA might further detail the dependencies between some of its work products (cf. section 2.3.3.2). SecEPM introduces the Security Analysis Model that might be associated with the role of a CIM. Similarly, the Security Analysis Model does only capture requirements for the security design. The Security Design Model might be associated with the role of a PIM—appropriate controls are selectable independent of a specific runtime environment. The Platform Capability Model might be associated with the platform model as it provides technical concepts and services provided by the runtime elements. The Implementation Artifacts might be associated with the PSM as they include not only design decisions but also details that specify the usage of a particular platform.

This alignment must not be confused with an identification: As SecEPM only addresses security aspects of an electronic business process, it does not take the same perspective as MDA that addresses software and system development in general. Furthermore, SecEPM starts with a definition of an electronic business process that potentially includes platform specific aspects in the very beginning of the security engineering process and therefore might be associated with a PSM. Nevertheless, the similar procedure of successive inclusion of platform specific details and design decisions supports the alignment.

4.5. Activities

This section details the activities of SecEPM. An activity in SecEPM is a set of tasks that is performed towards a specific purpose in order to develop secure electronic business processes

(cf. section 2.3.1). Every activity is introduced explaining its objective and its relations to other entities of the process model. The execution of the tasks is exemplified using the Replan Process (cf. chapter 3). A tabular overview summarizes information on aim and use as well as necessary work products consumed by the activity and created and updated work products. Furthermore, involved roles are provided and the tasks of the activity are listed. Details on how to work on specific tasks are described in section 4.6 introducing guidance artifacts.

4.5.1. Setup Process

4.5.1.1. Overview

Before a security engineering process is started, the process needs to be set up. The preparatory Setup Process activity subsumes all tasks that are necessary to tailor the process model to its application environment. Depending on the size of the organization, the number of application domains, and the heterogeneity of the development projects, this activity has to be executed before every project, upfront a set of similar projects, or while it is introduced to the organization. Although it is necessary to setup the security engineering process in advance, every task might be rerun in the course of a project. This enables the provision of additional information, detailing of overgeneralized aspects, or removal of unnecessary items from the Process Model Configuration.

A successful security engineering process depends on a comprehensive preparation. Therefore, a close cooperation of Business Process Engineers, Developers, Domain Experts, Security Analysts, and the responsible Process Model Engineer is necessary for providing all required information. The individual configuration of the process model is compiled in the Process Model Configuration. This work product does not only encompass basic definitions but includes also selected guidance artifacts, descriptions of preferred practices and tooling, and hints on the integration of the security engineering process into the development process. Additionally, the Threat and Control Catalog are generated that enable security nonprofessionals to instantiate relevant threats and to select adequate mitigation. A description of the capabilities of runtime components is collected in the Runtime Capability Model that provides means to map the selected controls onto actual infrastructure. Table 4.3 provides an overview of the activity Setup Process.

4.5.1.2. Provide Basic Definitions

An early task in the Setup Activity is the provision of a basic framework for the definition and assessment of security goals, security requirements, and controls. It aims at aligning the security engineering process with the overall information security management and making a coherent set of classifications available. This task is generally assigned to the Security Analyst, supported by the other roles assigned to the superordinate activity.

The minimal set of definitions to be provided encompasses security goal classes, security requirement purposes, and rating scales for security goals, threats, controls, and control implementations. The definition of security goal classes provides the scope of the security analysis: Inclusion or exclusion of specific goal classes allow for a focused analysis or a broader

Name	Setup Process
Aim	Preparation of the security engineering process
Use	Tailoring of the security engineering process for an organization, the application domain, or a development project
Input	Information Security Policy
Output	Process Model Configuration, Threat Catalog, Control Catalog, Runtime Capability Model
Roles	Process Model Engineer (responsible), Business Process Engineer, Developer, Domain Expert, Security Analyst
Tasks	• Provide Basic Definitions • Define Threat Classes • Define Control Classes • Provide Capabilities of Runtime Environment • Provide Guiding Artifacts

Table 4.3.: Activity: Setup Process

perspective. The definition of control purposes provides a classification of intentions for the application of controls. The definition of rating scales allows to prioritize goals, threats, and requirements as well as to assess controls and their implementations. Depending on the risk assessment methodology applied, rating scales might be simple, ordinal differentiations or elaborated assessment formula. Generally, existing methods or best practices will be applied in the course of the security engineering process covered with SecEPM. Therefore, many of these definitions might originate in those methods and practices. Appropriate guidance artifacts should be included to the Process Model Configuration to elaborate the application of the definitions provided with this task (cf. task Provide Guiding Artifacts on page 67).

Exemplary, the security goal classifications provided in the preceding chapters are chosen here: “Confidentiality”, “Integrity”, “Availability”, and “Non-repudiation” (cf. section 2.4.1). Control purposes are specified applying the differentiations provided by an ISO standard defining “Prevention”, “Detection”, “Limitation”, and “Monitoring” [Int04]. If we apply the risk assessment approach from the IT-BPM, rating scales for security goals are defined as simple, ordinal scales ranging from “Normal” as limited and calculable impact to “Very high” as impact threatening the survival of the organization [Bun08]. For threats, controls, and control implementations no rating scales are provided by the IT-BPM. For the application of the selected risk assessment approach damage scenarios and corresponding rating criteria have to be defined as well (cf. section 4.6). Exemplary, the damage scenario “Financial consequences” is defined including the rating criterion “Limited financial losses” assigning the rating “Normal” to the condition “Financial losses are below 50,000 EUR”.

4.5.1.3. Define Threat Classes

The identification and assessment of relevant threats is regarded as challenging task in the course of a security engineering process (e.g., [Sho14]). SecEPM utilizes a Threat Catalog providing a pre-compiled set of threat classes relevant in the application domain and aligned with the BPM environment to facilitate threat modeling and to allow for comparable, repeatable

results. The preparation of the Threat Catalog is assigned to the Security Analyst supported by the other roles assigned to the superordinate activity.

A threat class defines a set of possible threats for an electronic business process. It specifies security goal classes that are violated if an instance of the threat class becomes effective, applicability conditions of the threat class, affected elements, and possible impact. Furthermore, threat classes might interrelate either in detailing (e.g., one threat is decomposed in several sub-threats) or influencing each other (e.g., one threat leverages the impact of another threat).

Exemplary, the Security Expert starts with rather general threat classes adopting the Spoofing, Tempering, Repudiation, Information disclosure, Denial of service, Elevation of privileges (STRIDE) [SS04, Sho14] method for the threat analysis of electronic business processes. Every adverse action proposed by the method is applied either to processes or data elements of a business process model. One threat class is defined as "Tampering Execution Sequence" endangering security goals of the class "Integrity" and applies to elements of the type Pool from a BPMN business process model. Affected elements are those of the type Sequence Flow of the Pool in question manipulating the incoming or outgoing connection. In the course of the security engineering process, this or other threat classes might be further refined separating different attack vectors, e.g., using attack trees [Sch99] or refined versions of the STRIDE method [SB12].

4.5.1.4. Define Control Classes

Complementary to the Threat Catalog, a set of pre-compiled control classes is utilized in SecEPM to facilitate the design of appropriate controls and—similarly—to allow for comparable and repeatable results. The definition of control classes is also a duty of the Security Analyst supported by the other roles assigned to the superordinate activity.

A control class specifies a set of generic controls that might be applied in order to satisfy security requirements for an electronic business process. The control class specification includes the enumeration of addressed security goals and threat classes, control purposes it might serve, assets that are introduced if a control of that class is applied, and protected entities. Analogous to threat classes, control classes might be interrelated either hierarchically or with regard to other dependencies.

Most likely, an existing control catalog will be used as a starting point and aligned by the Security Analyst, e.g., a control catalog provided by the NIST [RSP+10]. A control class that is derived from the control "AU-12 Audit Generation" is "Log Process Execution" that addresses the security goal class "Integrity" and mitigates the threat class "Tampering Execution Sequence" providing means to detect attacks. The control class introduces log files as new assets and protects Control Flow entities.

4.5.1.5. Provide Capabilities of Runtime Environment

In order to implement controls that have been selected in the course of the security engineering process, capabilities of the runtime components of the target environment need to be formulated. The Developer in collaboration with the Business Process Engineer and the

Process Model Engineer collect and document the capabilities creating the Runtime Capability Model.

Capabilities are specified documenting the runtime component providing this capability, the control classes that are implementable using a capability, and preconditions for the implementation like dependencies on other capabilities. Additionally, proprietary information on the configuration alternatives for capabilities are noted. Differences with regard to affected and protected elements between referenced control classes and a capability are documented.

An exemplary BPMS to execute the Replan Process is Activiti[4]. It is an open source business process engine and management system. A starting point to identify capabilities are security manuals provided by the software vendor. One feature of Activiti is its logging functionality. Therefore, a capability "Activiti logging" can be specified that is assigned to the runtime component "Activiti" and implements the control class "Log Process Execution". The property "Log Level" and its domain is annotated as proprietary information with regard to the configuration of the capability.

4.5.1.6. Provide Guiding Artifacts

An important design decision for SecEPM is the application of the Separation of Concerns strategy (cf. section 4.3). In order to separate decisions on what to do in the course of the security engineering process from the decisions on how to execute a given task, activities are separated from guiding artifacts. This differentiation allows for integration of different methods and best practices in SecEPM instead of providing different process models.

Guiding artifacts encompass checklists, examples, guidelines, references, and templates. Every guiding artifact is classified with regard to these classes and assigned to at least one or more tasks. Guiding artifacts might also reference each other and other elements of the SecEPM such as responsible roles or work products. The application of guiding artifacts in the context of SecEPM needs to be described and necessary adoptions have to be documented. The selection of guiding artifacts should be carefully considered, as they need to be precise enough to allow security nonprofessionals to execute the related tasks but shall not overwhelm the practitioner with detail or alternatives.

In the course of the preceding tasks of the Setup Process activity, several guiding artifacts have been mentioned already. Exemplary, the risk assessment approach applied for the rating of security goals needs to be integrated as reference as well as a guideline on how to apply it in the context of the activities Setup Process, Identify Assets, and Assess Security Goals. Also, the STRIDE method might be provided as reference and as guideline for the activity Model Threats [SS04, Sho14]. Some of these artifacts are described in greater detail in the following section 4.6.

[4] `http://www.activiti.org/`

Name	Identify Assets
Aim	Identification of assets that are impacted by the electronic business process
Use	Assignment of assets and entities from the electronic business process, preparation of security goal assessment, and integration in overall risk management
Input	Process Model Configuration, Information Security Policy, Business Process Model
Output	Security Analysis Model
Roles	Business Process Engineer (responsible), Domain Expert
Tasks	• Identify Business Assets • Identify Modeled Assets • Identify Supporting Resources • Analyze Dependencies Between Assets and Resources

Table 4.4.: Activity: Identify Assets

4.5.2. Identify Assets

4.5.2.1. Overview

As mentioned in the preceding section 4.3, SecEPM takes an asset-oriented perspective. Assets are entities that a stakeholder puts value upon with respect to security (cf. definition 8). The aim of this activity is to identify, decompose, and relate all elements of value that might be impacted by an electronic business process. Therefore, assets provide a basic link between the functional representation of the electronic business process (documented in the Business Process Model) and the security analysis (documented in the Security Analysis Model).

The general approach for this activity is to identify general business assets, decompose them to enable assignment with modeled elements of the Business Process Model, and to document necessary resources and dependencies between assets and resources. The Identify Assets activity is a good starting point for a security engineering process as it allows to focus on relevant aspects of the electronic business process from the business point of view. Nevertheless, several tasks of this activity are very likely to be executed multiple times in the course of a security engineering process, e.g., the analysis of assets introduced by controls selected in other activities.

All tasks of this activity are assigned to the Business Process Engineer supported by a Domain Expert. Definitions and guidelines provided in the Process Model Configuration are applied within this activity, general business assets are aligned with the Information Security Policy. Table 4.4 provides an overview of the activity Identify Assets.

4.5.2.2. Identify Business Assets

First, general business assets related to the electronic business process are identified to assess security needs. Mostly, these business assets reflect the main purpose of the business process in question and depict the value at risk if the business process becomes compromised. A business value represented by a business asset has to be documented in order to justify any kind of control that is applied to the electronic business process,.

Business assets are identified using the Business Process Model and the Information Security Policy. Descriptions of the added value provided by a business process, objectives, and results of a business process are good candidates for business assets. Business assets might be abstract, i.e., they are not always tangible goods but a benefit or other immaterial goods. Business assets are documented providing at least a label, a description, and a stakeholder for that asset.

Exemplary, the Replan Process has the purpose to monitor shipments with the purpose of proactive intervention if thresholds are not met in the current situation. The overarching value at risk is the successful proof of delivery (POD), i.e., the confirmation that a shipment has been transported in time to the designated destination without damage. Therefore, "Successful POD" with the logistics provider as stakeholder is documented as business asset for the Replan Process.

4.5.2.3. Identify Modeled Assets

Business assets are not necessarily directly represented in the Business Process Model. In order to enable the analysis of security goals and threats for a given electronic business process, the business assets have to be related to the Business Process Model. Therefore, the business assets are decomposed into modeled assets and related to elements of the Business Process Model.

Different approaches have been proposed for the identification of assets for electronic business processes (e.g., [WMM08, HB09, JTGK10]). Every elements of a given type might be regarded as modeled asset, e.g., elements of the type Message, or elements that fulfill certain criteria, e.g., the inclusion of an element in an administrative district. The Information Security Policy as well as guidance artifacts from the Process Model Configuration provide details for this task. In addition to documentation requirements for general business assets, modeled assets provide a link to related elements of the Business Process Model.

For the Replan Process, "Message M1" (status message), "Message M2" (updated route plan), and "Pool P1" (process in the administrative district of the logistics provider) are considered as modeled assets. This selection follows from the purpose of the Replan Process: a proactive intervention in order to reach a successful POD. The elements Message "M1", Message "M2", and Pool "P1" provide value with regard to security for the logistics provider and directly influence the business assets "Successful POD".

4.5.2.4. Identify Supporting Resources

Security goals for modeled assets have to be assessed and related to respective threats to elicit requirements systematically (cf. the following sections). Some threats might not directly effect an asset but effect an asset indirectly via other elements that have been modeled in the Business Process Model. These elements are regarded as resources in SecEPM. A resource supports one or more assets.

The identification of supporting resources depends on the threat identification approach chosen in the Setup Process activity. Similar to the preceding task, for every resource label, description, and related entity from the Business Process Model are documented.

Exemplary, “Pool P2” (the process outside the administrative district of the logistics provider), or Message Flows between the freight forwarder and the logistics provider representing communication channels might be regarded as resources.

4.5.2.5. Analyze Dependencies Between Assets and Resources

In security analysis, the protection of the weakest link is an often cited principle [VM02]. Analysis of dependencies as well between different assets as between assets and resources is crucial to understand the propagation of security goals and to allow for their assessment, . Integrity of a return address of a shipment might not be considered as a security goal at all or only as minor concern. But if a manipulation of this return address endangers the successful POD because of customs formalities, integrity of the return address becomes a major concern.

Dependencies with regard to security goals and their assessment between assets as well as those between assets and resources are captured as a directed relation “supports” starting with the supporting element and ending with the supported element. Primarily, dependencies might be recorded in the course of the identification of modeled assets and resources. Within this task, the dependencies should be reconsidered, consolidated, and documented in the Security Analysis Model.

Exemplary, a “supports” relation between “Pool P1” and “Successful POD” is documented as security goals for the successful POD are endangered by violations of the security goals of the process modeled with Pool “P1”. Similarly, relations between “Message M1” and “Pool P1”, “Message M2” and “Pool P1”, “Pool P2” and “Pool P1” are documented.

4.5.3. Assess Security Goals

4.5.3.1. Overview

Security goals express stakeholders concerns towards an asset with regard to a security goal class (cf. definition 9). The aim of this activity is the identification and evaluation of security goals for assets that might be impacted by the electronic business process. The result of this task are security goals and their ratings that support prioritization of further analysis effort and bridge assets and security requirements.

In order to assess security goals, security needs for every asset are identified and rated using the rating scales provided in the Process Model Configuration and general security policies provided in the Information Security Policy. The Business Process Model provides context information for the specification of security goals. Analyzing dependencies between assets, ratings for respective security goals are adjusted to reflect those dependencies. The assessment of security goals should be executed, whenever new assets are included in the Security Analysis Model or dependencies between assets change. All tasks of this activity are assigned to the Business Process Engineer supported by the Domain Expert. Table 4.5 provides an overview of the activity Assess Security Goals.

Name	Assess Security Goals
Aim	Identification and evaluation of security goals for assets
Use	Preparation of threat modeling and prioritization of security goals
Input	Process Model Configuration, Information Security Policy, Business Process Model
Output	Security Analysis Model
Roles	Business Process Engineer (responsible), Domain Expert
Tasks	• Specify Security Goals • Rate Individual Security Goals • Analyze Dependencies Between Security Goals • Adjust Security Goal Ratings

Table 4.5.: Activity: Assess Security Goals

4.5.3.2. Specify Security Goals

Security goals capture security needs of stakeholders with respect to assets. With the specification of security goals the Business Process Engineer (supported by the Domain Expert) distinguishes between relevant and insignificant security aspects: Only those aspects that are documented by a security goal will be considered in the security analysis.

For every asset security needs are considered applying the security goal classes defined in the Process Model Configuration. An individual security goal is documented detailing the asset considered by the security goal, the security goal class of the security goal, and a (short) description of the security need.

With respect to the Replan Process, the process operated for the monitoring of shipments has been identified as an asset ("Pool P1") because a disturbance of the process execution endangers a successful POD. Analyzing this asset, the security need can be detailed: The sequence of activities must not be disturbed by an unauthorized party. Therefore, the security goal "P1 Process Integrity" is specified for the asset "Pool P1" and classified as "Integrity" security goal.

4.5.3.3. Rate Individual Security Goals

In order to prioritize security goals and to allow for decisions considering focus areas of the security analysis and design, security goals are rated. The rating procedure depends largely on the respective methodology selected and documented in the Process Model Configuration. After completion of this task, for every security goal an individual rating has to be documented in the Security Analysis Model.

Using the IT-BPM, the Process Model Configuration defines a number of rating criteria grouped by different damage scenarios. Following the IT-BPM, for every security goal and damage scenario at least one criterion is assigned to the security goal. The individual rating of the security goal is derived applying the maximum rating to the security goal. Exemplary, the criterion "Limited financial losses" from the damage scenario "Financial Consequences" is assigned to the security goal "P1 Process Integrity" and the derived rating is "Normal".

4.5.3.4. Analyze Dependencies Between Security Goals

The activity Identify Assets includes a task to analyze dependencies between assets that are documented in the Security Analysis Model (cf. section 4.5.2). These dependencies are now further refined. The objective is to document, whether security goals of supported assets influence security goals of supporting assets or vice versa. Similarly to the preceding task, the procedure depends largely on the respective assessment method. After completion of this task, for every security goal the influence of security goal ratings of supporting or supported assets have to be documented in the Security Analysis Model.

The IT-BPM method for the rating of security goals proposes the application of three aggregation strategies as it is described in the corresponding guideline (cf. section 4.6.2.5). Exemplary, the aggregation strategy for the security goal "P1 Process Integrity" is set to the maximum strategy as security goal ratings from supported assets have to be propagated to this security goal.

4.5.3.5. Adjust Security Goal Ratings

Ratings for individual security goals have to be adjusted to reflect dependencies between assets and their security goals. Therefore, the influence of security goal ratings of supporting or supported assets is evaluated and ratings for individual security goals are adjusted accordingly. The process for the adjustment of the ratings depends largely on the respective assessment method documented in the Process Model Configuration.

Exemplary, if a security goal of the class "Integrity" for the asset "Successful POD" has been rated as "High", the initial rating of the security goal "P1 Process Integrity" will be adjusted from its individual rating "Normal" to "High" as the maximum strategy has been chosen for this security goal and the asset "Pool P1" supports the asset "Successful POD".

4.5.4. Model Threats

4.5.4.1. Overview

The Model Threats activity explicates security challenges for the electronic business process in terms of threats. Threats are potential causes for the violation of security goals (cf. definition 11). The analysis of threats provides means to refine security requirements from security goals: Only if a security goal is endangered by one or more threats, a security requirement might be refined for that security goal. Therefore, threats enable participants to focus further on relevant aspects of the electronic business process in the security engineering process.

The analysis and modeling of relevant threats starts with the identification of candidate threats from the Threat Catalog, proceeds with the selection of relevant threats from the candidate list, and ends with an analysis of the impact of the selected threats using the Business Process Model. The Process Model Configuration provides guidance for selected threat modeling and assessment techniques including rating scales and definitions. Threat modeling is needed whenever changes to the Business Process Model apply that might introduce new threats or further security goals are created or updated. All tasks of this activity are assigned

Name	Model Threats
Aim	Analysis of threats that endanger security goals
Use	Preparation of requirement elicitation
Input	Process Model Configuration, Threat Catalog, Business Process Model
Output	Security Analysis Model
Roles	Business Process Engineer (responsible), Domain Expert
Tasks	• Identify Candidate Threats • Select Relevant Threats • Analyze Threat Impact

Table 4.6.: Activity: Model Threats

to the Business Process Engineer supported by the Domain Expert. Table 4.6 provides an overview of the activity Model Threats.

4.5.4.2. Identify Candidate Threats

SecEPM applies a collection of threat classes provided with the Threat Catalog to support the Business Process Engineer identifying threats. Therefore, threat classes need to be identified from the Threat Catalog that apply to the actual Business Process Model. Applicable threat classes are instantiated and provided as candidate threats.

For every threat class the respective applicability conditions are checked to identify candidate threats. If the applicability conditions are fulfilled with regard to specific parts or elements of the Business Process Model, a new candidate threat is instantiated in the Security Analysis Model, providing a label, description, and a reference on the threat class as well as a reference to the respective part or element of the Business Process Model.

Exemplary, the threat class “Tampering Execution Sequence” is checked for the Replan Process. The threat class applies for the Pool “P1” and the Pool “P2” as it might be applied to elements from the type Pool. Therefore, two candidate threats “Tamper P1 Execution Sequence” and “Tamper P2 Execution Sequence” are instantiated and documented in the Security Analysis Model.

4.5.4.3. Select Relevant Threats

Relevant threats that have to be considered for the refinement of security requirements provide two properties: First, they apply to the Business Process Model, i.e., they endanger certain aspects of the functionality or quality of the electronic business process. Second, they endanger a security goal of a stakeholder of the process. The first property is evaluated during the preceding task providing candidate threats. Now, security goals need to be checked with regard to the candidate threats to select only relevant threats. An example for this distinction might be the threat of tapping a communication channel: This threat applies to every communication channel. It is only relevant, if confidential information can be gained by tapping a communication channel. If this is the case, a corresponding security goal of the class “Confidentiality” for the data transmitted has to be documented.

Relevant threats are selected from the list of candidate threats by checking for every candidate threat, whether the parts or elements referenced by the candidate threat touch one or more security goals. For every match, the security goal classes of the threat and the respective security goal are compared. If both match, the threat is relevant for the security analysis and the matching security goals are assigned to the threat. Candidate threats that do not touch a security goal are marked as irrelevant or removed from the Security Analysis Model.

From the list of candidate threats, only "Tamper P1 Execution Sequence" touches a security goal: "P1 Process Integrity". The threat addresses the "Integrity" security goal class as well as the security goal. Therefore, the security goal is assigned to the threat and it is documented in the Security Analysis Model.

4.5.4.4. Analyze Threat Impact

The aim of this task is to analyze the impact of threats. This analysis includes the questions, which elements of the Business Process Model would be affected by a successful realization of threats as well as whether threats can be rated in order to prioritize further analysis and resulting security requirements.

For each threat the affected elements are derived using the information provided by the respective threat class. Similarly, the impact on the affected elements is evaluated and documented for each threat. The rating procedure for the threats depend largely on the selected assessment method. Ratings for the threats are documented in the Security Analysis Model.

Exemplary, the threat "Tamper P1 Execution Sequence" affects all Sequence Flow elements contained in the Pool "P1", manipulating the incoming or outgoing connection of those elements. As no rating procedure for threats has been included in the Process Model Configuration, threats remain unprioritized. Alternatively, the tasks Provide Basic Definitions and Provide Guiding Artifacts of the Setup Process activity might be re-executed in order to integrate a respective method, e.g. from the Microsoft Secure Development Lifecycle (MS SDL) [Mic12].

4.5.5. Elicit Security Requirements

4.5.5.1. Overview

Security requirements refine security goals and state the intent to counter threats (cf. definition 13). The activity Elicit Security Requirements synthesizes security goals and threats to specify concise security requirements: Security goals alone do not require necessarily corresponding controls. Similarly, threats state only negative instead of positive propositions. As result of this activity, security needs for the electronic business process are specified in a way that supports the design of appropriate controls and the mitigation of respective threats.

Security requirement elicitation covers identification of candidate requirements, selection of security requirements, consolidation of selected security requirements, and validation of security requirements with regard to the modeled security goals and threats. Requirement elicitation uses security goals and threats documented in the Security Analysis Model in combination with the Business Process Model as major input and applies definitions and practices provided in the Process Model Configuration. Requirements are specified in the

Name	Elicit Security Requirements
Aim	Refinement of security requirements based on threats and security goals identified
Use	Specification of concise security requirements as basis for the design of controls
Input	Process Model Configuration, Business Process Model
Output	Security Analysis Model
Roles	Business Process Engineer (responsible), Domain Expert, Security Analyst
Tasks	• Identify Candidate Security Requirements • Select and Detail Security Requirements • Consolidate Security Requirements • Validate Security Requirements

Table 4.7.: Activity: Elicit Security Requirements

Security Analysis Model. Requirement elicitation is executed by the Business Process Engineer, supported by the Domain Expert to align decisions with the application domain. The Security Analyst validates the resulting requirement specification. Table 4.7 provides an overview of the activity Elicit Security Requirements.

4.5.5.2. Identify Candidate Security Requirements

The refinement of security goals adds information to the specification of security needs for the electronic business process in the Security Analysis Model. Elicitation of requirements includes not only explication of inherently contained propositions but also decisions on alternative formulations of the security needs (and the resulting security problem). Nevertheless, requirement elicitation must not be confounded with problem solving, i.e., the design of adequate controls. Therefore, SecEPM distinguishes the identification of candidate security requirements and the selection of (actual) security requirements. This task identifies candidate security requirements that can be elicited from security goals and threats.

Every threat and security goal are analyzed in combination to identify candidate security requirements. Results are documented as candidate security requirements in the Security Analysis Model providing label, description, references on respective threats, and security goals.

Exemplary, in order to address the threat “Tamper P1 Execution Sequence” for the security goal “P1 Process Integrity” the candidate security requirement “Check P1 Execution Sequence in Retrospect” with the respective relations is specified. An alternative candidate security requirement analyzing the same security goal and threat might be “Check P1 Process Model Conformance Before Activity Execution”.

4.5.5.3. Select and Detail Security Requirements

This task selects (actual) security requirements from candidate security requirements and provides further details for these requirements. It aims at documenting decisions with regard to the refinement of security goals and their interpretation. The meaning of the documented

security goals is narrowed executing this task distinguishing actual from discarded security requirements.

To select security requirements, candidate requirements are grouped by security goal. Supported by the domain expert, security requirements that conform to the intended interpretation of the security goal are selected from the candidate security requirements. Candidate security requirements that are not selected are marked as discarded or removed from the Security Analysis Model. The selection process should be supported by guidance artifacts providing a systematic approach including selection criteria. For selected security requirements, the Business Process Model is analyzed and elements or parts of the Business Process Model that are addressed by a requirement are documented.

For the Replan Process, the candidate security requirements “Check P1 Execution Sequence in Retrospect” and “Check P1 Process Model Conformance Before Activity Execution” apply to the security goal “P1 Process Integrity”. The requirement “Check P1 Execution Sequence in Retrospect” is selected and all Control Flow elements in the Pool “P1” are assigned to the requirement.

4.5.5.4. Consolidate Security Requirements

The tasks for security requirement identification and selection do not consider their commonalities and conflicts. As both tasks start with individual security goal and threats, it is possible that resulting security requirements addressing different security goals specify similar needs for the same parts or elements of the Business Process Model. Those overlapping requirements should be merged maintaining all relations to security goals, threats, and elements of the Business Process Model to provide a better overview and comprehensibility. Similarly, it is possible that resulting security requirements specify conflicting needs. Either a compromise for conflicting security requirements must be identified or predominant security requirements must be selected.

For the merging, security requirements are analyzed with regard to the specified functionality or quality. If the required functionality or quality is identical the requirements are merged. Merging requirements are documented in the Security Analysis Model, referencing the superset of security goals, threats, security requirements purposes, and elements of the Business Process Model as well as the merged security requirements. Security requirements are analyzed with regard to the assigned elements of the Business Process Model to identify conflicts. The required functionality or quality of security requirements assigned to the same elements is analyzed. In case of conflicts, a compromise is formulated or predominant security requirements are selected and documented in the Security Analysis Model.

No security requirements are specified for the Replan Process that require the same functionality. Nevertheless, it is possible to think of a security requirement stemming from a non-repudiation security goal for the Pool “P1” that requires the possibility to check the execution sequence of the Pool “P1” after the Activity “T14” in retrospect. This security requirement would then be merged with the security requirement “Check P1 Execution Sequence in Retrospect”, referencing the superset of security goals, threats, elements of the Business Process Model, as well as the merged security requirements.

4.5.5.5. Validate Security Requirements

The elicitation of security requirements is an important activity to design secure, electronic business processes. In order to provide a sound foundation for the security design, the security requirements are validated by the Security Analyst. Primarily, validation includes two aspects: All security goals have to be covered by security requirements and all threats have to be covered by security requirements. Additionally, the Security Analyst checks the security requirements with regard to best practices for their specification.

To validate coverage of security goals and threats by the security requirements, (at least two) traceability matrices are derived enumerating all security goals and referenced elements of the Business Process Model as well as threats. Security requirements and the assigned elements of the Business Process Model are enlisted and coverage is marked in the matrices. The compliance check depends on the best practices selected in the Process Model Configuration.

Exemplary, for the Replan Process the security goal “P1 Process Integrity” and the threat “Tamper P1 Execution Sequence” (impacting all Sequence Flow elements contained in Pool “P1”) have been identified. The security requirement “Check P1 Execution Sequence in Retrospect” covers the security goal “P1 Process Integrity” and the threat “Tamper P1 Execution Sequence”. The elements referenced by the security requirement and the threat of the Business Process Model are identical. Therefore, the validation succeeds.

4.5.6. Design Controls

4.5.6.1. Overview

During this activity, controls are introduced to mitigate threats identified for assets of the electronic business process and to meet the specified security requirements. As SecEPM focuses on the gap between business process models and the design of proper controls, complementing security considerations and measurements addressing aspects like service, operating system, network, and physical security are not considered by SecEPM (cf. section 4.2). Therefore, not every security requirement might be met with controls provided by the BPMS. For these requirements, assumptions are formulated that have to be fulfilled by the environment.

Similar to requirements elicitation, the design of controls separates identification of candidate controls as well as selection and detailing of actual controls. Selected controls are consolidated to avoid redundancy. Assets that are introduced by the selected controls have to be included in the security analysis. Remaining security requirements and threats not covered by controls are specified with assumptions. The resulting set of controls and assumptions are validated to check that all security requirements are addressed. The Business Process Engineer is responsible for the execution of the Design Controls activity supported by the Security Analyst. The activity consumes the Security Analysis Model as major input and creates or updates the Security Design Model. As assets might be introduced by controls, the Security Analysis Model gets updated accordingly. The Control Catalog provides a precompiled set of applicable controls supporting the identification of candidate controls. The Process Model Configuration and the Information Security Policy provide guidelines and definitions, the Business Process

Name	Design Controls
Aim	Design of security controls to fulfill the security requirements
Use	Selection and tailoring of controls to cover the security requirements
Input	Process Model Configuration, Control Catalog, Information Security Policy, Business Process Model
Output	Security Analysis Model, Security Design Model
Roles	Business Process Engineer (responsible), Security Analyst
Tasks	• Identify Candidate Controls • Select Controls • Consolidate Controls • Integrate Assets Introduced by Controls • Provide Assumptions • Validate Controls

Table 4.8.: Activity: Design Controls

Model the context for the selection of appropriate controls. Table 4.8 provides an overview of the activity Design Controls.

4.5.6.2. Identify Candidate Controls

The Business Process Engineer identifies candidate controls selecting applicable control classes from the Control Catalog. A control class is applicable if it meets a security requirement with regard to (a subset of) security goal classes, threat classes, and protects elements referenced by the security requirement. Applicable control classes are instantiated and provided as candidate controls.

For every control class from the Control Catalog and security requirement from the Security Analysis Model security goal classes and threat classes are compared to identify candidate controls. If a control class addresses at least a subset of the enumerated elements for each aspect, the referenced elements of Business Process Model from the security requirement are compared with the elements that might be protected by the control class. If (a subset of) the referenced elements can be protected by the control class, a new candidate control is instantiated in the Security Design Model providing at least label, description and a reference on the security requirement. If an instance of the control class requires instances of other control classes, respective candidate controls are instantiated and references are documented for the dependent candidate control.

Exemplary, the control class “Log Process Execution” addresses the security goal class “Integrity” and mitigates threats of the class “Tampering Execution Sequence” providing means to detect attacks. These aspects correspond to those of the security requirement “Check P1 Execution Sequence in Retrospect”. As the control class protects elements of the type Control Flow and the security requirement references all Control Flow elements of Pool “P1”, a candidate control “Log P1” is instantiated from the control class “Log Process Execution” referencing the security requirement “Check P1 Execution Sequence in Retrospect”.

4.5.6.3. Select Controls

Generally, controls affect not only security properties but other properties of an electronic business process as well such as maintainability, performance, usability etc. Therefore, candidate controls have to be analyzed with respect to their impact on those properties to minimize unintended impact. Furthermore, different sets of controls might provide similar protection levels and meet the same security requirements. Hence, from the list of candidate controls a set of controls is selected that minimizes negative impact on the electronic business process and meets the security requirements appropriately.

The analysis of the impact of candidate controls on the electronic business process is not within the scope of SecEPM. Guidance artifacts should be provided to assess the impact and provide respective selection criteria (e.g., adapting [EY07, NEF08, KSS12]). Similarly, criteria for the selection of comparable sets of controls should be provided in the Process Model Configuration to allow for an effective selection. Candidate controls that are not selected are marked as discarded or removed from the Security Control Model. For the Replan Process, only one candidate control is instantiated that is now selected as actual control.

4.5.6.4. Consolidate Controls

Several controls instantiating the same control class might have been documented in the process of identification and selection of controls. If those controls meet security goals with similar ratings they might be merged to avoid redundancies. Merged controls maintain all relations to respective security requirements, depending controls, and protected elements in the Business Process Model.

For the merging, controls from the same control class that reference security requirements refining security goals with similar ratings are identified. A new, merging control is specified in the Security Design Model, referencing the superset of security requirements, depending controls, and protected elements as well as the merged controls. No controls instantiating the same control class are documented for the Replan Process.

4.5.6.5. Integrate Assets Introduced by Controls

Controls might introduce new assets if they are applied for an electronic business process. Those assets need to be considered in the security analysis the same way as other assets. Therefore, they are included in the Security Analysis Model, respective security goals and requirements are identified, assessed, and specified, and it is checked, whether new controls are necessary for the protection of those assets.

For every control that is documented in the Security Design Model introduced assets are identified. If already existing assets are identical with the asset to be introduced, these assets are merged. Resulting assets are documented in the Security Analysis Model referencing the respective controls.

Exemplary, the control “Log P1” introduces the new asset “P1 Log File”. This asset is documented in the Security Analysis Model referencing the control “Log P1”.

4.5.6.6. Provide Assumptions

Not every security requirement might be met with a corresponding control class from the Control Catalog. Furthermore, controls might rely on further (e.g., organizational) controls that are not covered by control classes provided in the Control Catalog. Security assumptions specify constraints on the environment in order to meet those security requirements or preconditions (cf. definition 15).

To provide necessary assumptions security requirements in the Security Analysis Model are identified that are not addressed by (at least) one control. Furthermore, controls are identified that rely on other controls that are not documented in the Security Design Model. An assumption is specified in the Security Design Model for every identified security requirement and missing control, providing label, description and references on the respective security requirement or control.

In case of the Replan Process, every security requirement is met in the Security Design Model. Nevertheless, it is (very) likely that (e.g., physical) access to the BPMS has to be restricted. This security requirement would be addressed by an assumption specified in the Security Design Model.

4.5.6.7. Validate Controls

The security of the electronic business process relies on the appropriate selection of of controls. Therefore, the controls are validated by the Security Analyst. The validation concentrates on coverage with regard to security requirements and controls as well as assumptions. Every security requirement has to be met by at least one control or assumption. Additionally, the Security Analyst checks plausibility and feasibility of selected controls and assumptions.

For every security requirement referencing controls and assumptions are identified to validate the coverage of security requirements. For every control, security goal classes and protected elements of the Business Process Model are compared with the respective elements of the security requirement. If all elements are covered, the security requirement is met. If some of the elements are covered, the security requirement is partly met. If an assumption is assigned to the security requirement, the security requirement is considered. If every security requirement is at least considered, the validation succeeds. Nevertheless, for every security requirement that is only partly met, the Security Analyst has to confirm sufficient coverage.

The Security Analysis Model specifies the security requirement “Check P1 Execution Sequence in Retrospect”. The referencing control “Log P1” and the security requirement specify the same security goal and the same elements in the Business Process Model. Therefore, the control meets the security requirement and the validation succeeds.

4.5.7. Map Controls

4.5.7.1. Overview

The activity Map Controls highlights the nexus between design and implementation. Capabilities of runtime components are analyzed to implement the controls specified in the Security

Name	Map Controls
Aim	Map controls onto runtime environment capabilities
Use	Provide platform specific details to allow for a translation of the generic security design into implementation artifacts
Input	Process Model Configuration, Runtime Capability Model, Business Process Model
Output	Security Design Model
Roles	Business Process Engineer (responsible), Developer
Tasks	• Identify Candidate Control Implementations • Select Control Implementations • Configure Control Implementations • Check Control Coverage

Table 4.9.: Activity: Map Controls

Design Model. Matching capabilities are selected and mappings between controls and their implementations utilizing those capabilities are specified.

Similar to the design of controls, the mapping of controls separates the identification of candidate control implementations and the selection of control implementations. Control implementations are detailed selecting configuration options and necessary information considering the target environment. Finally, the coverage of all controls with control implementations is checked. The Business Process Engineer is responsible for the mapping of controls supported by the developer that provides detailed technical knowledge with regard to the runtime component and the target environment. Major input work products are the Security Design Model providing the controls to be implemented and the Runtime Capability Model describing possible control implementations for the runtime component. The Process Model Configuration and the Business Process Model provide guidelines and definitions as well as context for the selection of appropriate control implementations. Table 4.9 provides an overview of the activity Map Controls.

4.5.7.2. Identify Candidate Control Implementations

The Business Process Engineer identifies candidate control implementations analyzing control implementation classes provided by the Runtime Capability Model of the runtime component that has been selected for the target environment. Matching control implementation classes are instantiated as candidate control implementation.

In order to identify candidate control implementations, for every control implementation class provided in the Runtime Capability Model those controls are selected that instantiate a control class referenced by the control implementation class. If (a subset) of threat classes and security requirement purposes referenced by the control implementation class match with those of the selected controls, a candidate control implementation is created in the Security Design Model. It is providing at least label, description, and a reference to the matching controls. If the control implementation class references other required control implementation classes, respective candidate control implementations are instantiated that are referenced by the dependent candidate control implementation.

Exemplary, the control implementation class “Execution Log” for the runtime component “Actitivi” references the control class “Log Process Execution”, the threat class “Tampering Execution Sequence”, and the control purpose “Detection”. Therefore, the control “Log P1” matches the control implementation class. A candidate control implementation “Activiti Log P1” is instantiated from the control implementation class “Execution Log” referencing the control “Log P1”.

4.5.7.3. Select Control Implementations

Comparable with the task Select Controls from the activity Design Controls, control implementations affect not only security properties of an electronic business process. Hence, candidate control implementation have to be analyzed with regard to their impact on the runtime component, other candidate control implementations, and the target environment. The Business Process Engineer supported by the Developer selects from the candidate control implementations a set of control implementations minimizing the negative impact on the electronic business process.

The analysis of the impact of candidate control implementations on the electronic business process is not within the scope of SecEPM. Guidance artifacts should be provided to assess the impact and supply criteria for the selection of appropriate control implementations. Candidate control implementations that are not selected are marked as discarded or removed from the Security Design Model. Generally, the consolidation of controls in the activity Design Controls should prevent redundancy. Nevertheless, a consolidation of the selected control implementation might be useful to improve clarity and conciseness.

For the Replan Process, the candidate control implementation “Activiti Log P1” is selected as control implementation.

4.5.7.4. Configure Control Implementations

A control implementation specifies the utilization of a capability of a particular runtime component to implement a control for an electronic business process. Configuration details and specifics of the target environment have to be provided for control implementations to become effective. The Developer provides those details for the configuration of all specified control implementations.

In the case of the control implementation “Acitiviti Log P1”, the scope for the logging is the execution of all activities by the Acitiviti business process engine. The property “Log Level” specified for the control implementation class has to be set to “Fine” to provide logging information on start and ending of an activity.

4.5.7.5. Check Control Coverage

To validate the mapping of controls onto capabilities of runtime components, the coverage of all controls with respective control implementations has to be checked. For every control specified in the Security Design Model respective control implementations are identified and analyzed. If all controls are covered, the check is successful.

Name	Generate Control Artifacts
Aim	Create artifacts to implement the controls
Use	Transform the Security Design Model into Implementation Artifacts
Input	Process Model Configuration, Security Design Model
Output	Implementation Artifacts, Business Process Model
Roles	Developer (responsible), Business Process Engineer
Tasks	• Provide Target Environment • Transform Control Artifacts • Deploy Control Artifacts

Table 4.10.: Activity: Generate Control Artifacts

For the Replan Process, the control “Log P1” is covered by the control implementation “Activiti Log P1”. Therefore, the check succeeds.

4.5.8. Generate Control Artifacts and Test Cases

The activities Generate Control Artifacts and Generate Test Cases address the transformation of specified control implementations into Implementation Artifacts and their testing. As both activities depend largely on the target environment and the development-time tooling, this section provides only a general overview. Tables 4.10 and 4.11 summarize the information on these activities.

To generate artifacts for the implementation of the controls, the target environment has to be specified or access to the respective infrastructure has to be provided (e.g., current configuration of the components of the BPMS). Then, control implementations specified in the Security Design Model are transformed into corresponding Implementation Artifacts. The resulting artifacts are then deployed to the target environment. The activity is assigned to the Developer, supported by the Business Process Engineer. Major input work products are the Security Design Model specifying the control implementations and the Process Model Configuration providing information on the target environment. Results are the Implementation Artifacts and potentially an adopted Business Process Model including information on control implementation.

The generation of test artifacts is very similar to the generation of implementation artifacts. The differences primarily address the information about the test environment instead of the target environment and the assignment of the activity to the Tester instead of the Developer.

Depending on the environment and the tooling, individual tasks of both activities are not necessarily fully automated. Also, generated artifacts may not always be ready to use binaries or configurations. Test and Implementation Artifacts may also include skeleton implementations, partial configurations, and aids to support or test the actual implementation of controls.

Name	Generate Test Artifacts
Aim	Create artifacts to test the functioning of the controls
Use	Transform the Security Design Model together with the Security Analysis Model into Test Artifacts
Input	Process Model Configuration, Security Analysis Model, Security Design Model, Implementation Artifacts, Business Process Model
Output	Test Artifacts
Roles	Tester (responsible), Business Process Engineer, Developer
Tasks	• Provide Test Environment • Transform Test Artifacts • Deploy Test Artifacts

Table 4.11.: Activity: Generate Test Artifacts

4.6. Guidance

The preceding section detailed activities of SecEPM, i.e., descriptions of the tasks that are proposed to be performed in order to develop secure electronic business processes as well as respective roles and work products. This section focuses on explanations on how to execute these tasks and presents exemplary guidance provided by SecEPM. Useful guidance artifacts vary largely between different security engineering processes based on SecEPM since they depend on environmental conditions. Therefore, the selection of exemplary guidance presented in this section demonstrates the concept and its application.

We detail two examples in this section. The first example provides guidance on how to prepare and integrate guidance artifacts, the other example explains the application of an existing method for the rating of security goals in a SecEPM-based security engineering process. Generally, guidance encompasses several artifacts such as one or more guidelines, examples, checklists, templates, and references. Each example is introduced explaining its objective and its relation to other entities of SecEPM. A summary of the respective method or technique is provided including relevant references. Guidelines are documented briefly, accompanying artifacts (e.g., simple checklists and templates) are listed but not represented here. Sample applications are sketched in the guidelines and included in the description of the respective tasks. More details on guidance artifacts are provided in the exemplary study in chapter 6 and in the SecEPM repository (cf. section 4.7).

4.6.1. Provide Guidance Artifacts for Existing Methods

4.6.1.1. Overview

Guidance artifacts complement activities defined for SecEPM: Activities describe a set of tasks that is performed to aim at a specific purpose and detail what has to be done (by whom with which result). In contrast, guidance artifacts explain and demonstrate how to fulfill one or more given task. Therefore, guidance artifacts allow for the integration of existing (security engineering) methods and techniques into SecEPM-based security engineering processes. As this variability and integrability is an important property of SecEPM, this section details

Name	Provide Guidance Artifacts for Existing Methods
Aim	Integration of established methods and techniques into security engineering processes based on SecEPM
Related entities	• Tasks: Setup Process/Provide Guiding Artifacts • Work products: Process Model Configuration • Roles: Process Model Engineer, *
Guidelines	• Provide Overview • Provide Specific Guidelines • Provide Additional Guidance Artifacts
Additional artifacts	• Template: Tabular guidance overview • Reference: EPF Method Development Practice

Table 4.12.: Guidance: Provide Guidance Artifacts for Existing Methods

some guidance for the provision of guidance artifacts with the purpose of integrating existing (security engineering) methods.

This guidance on the provision of guidance artifacts is split into four major sections: Beside this overview three guidelines are detailed that describe different aspects of the provision of guidance artifacts. All guidelines support the execution of the task Provide Guiding Artifacts of the activity Setup Process. The guidelines address primarily the Process Model Engineer responsible for simplicity, coherence, and quality of the resulting guidance artifacts. Nevertheless, the individual descriptions are most likely provided by method experts (e.g., the Security Analyst). Guidance artifacts are integrated in the SecEPM repository and alter the Process Model Configuration.

The provision of guidance artifacts with the purpose of integrating existing methods has to consider at least the following aspects: An overview has to be provided that introduces the respective method and assigns the guidance artifacts to relevant SecEPM entities. The description of the application of the method should be provided as specific guidelines. Additional guidance artifacts like the explanation and definition of concepts, templates, examples, and checklists should be provided as necessary.

4.6.1.2. Provide Overview

For every method that gets integrated into SecEPM, an overview of this method and its integration into SecEPM should be supplied. This overview is prepared by a method expert (e.g., the Security Analyst) and checked and integrated by the Process Model Engineer in the task Provide Guiding Artifacts from the activity Setup Process. At first, this overview should name the integrated method (and possibly its source) and detail a short motivation considering aim and purpose of the method and the integration into SecEPM.

Next, the overview should name the different guidance artifacts that are provided to integrate the method into SecEPM. At least, an overview and a specific guideline should be supplied that details the application of the method in the course of the respective security engineering tasks. All (major) guidance artifacts should be related to respective entities of SecEPM, e.g., the guidelines with the respective tasks, roles, and work products.

The overview should also include an introduction to the method to facilitate comprehension of the general procedure and the relevance of the other guidance artifacts. In order to summarize these information, a tabular overview might be generated. A template for this overview is entailed in the SecEPM repository, an example is provided for the overview to this guideline (cf. table 4.12 on the preceding page).

4.6.1.3. Provide Specific Guidelines

The description and explanation of the application of the method to be integrated is detailed by one or more method specific guideline. These guidelines are created by a method expert (e.g., the Security Analyst) and reviewed and integrated by the Process Model Engineer in the task Provide Guiding Artifacts from the activity Setup Process.

The decision with regard to the provision of one or more guidelines for one method should consider the cohesion of the resulting guideline (i.e., it should not lump together too many different aspects or split into too many very closely related guidelines) and the coupling with the different entities from SecEPM, especially the tasks. As a rule of thumb, a guideline should not be assigned to tasks from different activities and have more than two to three roles.

A guideline should reference the tasks it supports, the necessary roles, and the work products it alters. Further guidance artifacts supporting the application of the guideline should be referenced as well. Core of a guideline is the explanation of the steps to take to apply the method in the context of the referenced tasks. The Method Development practice from the Eclipse Process Framework (EPF) provides further practical hints on the definition of method fragments including guidance artifacts[5].

4.6.1.4. Provide Additional Guidance Artifacts

Additional guidance artifacts might ease the application of the method and the respective guidelines. These additional guidance artifacts like examples, templates, checklists, references etc. are provided by a method expert (e.g., the Security Analyst) and checked and integrated by the Process Model Engineer in the task Provide Guiding Artifacts from the activity Setup Process. Similar to guidelines, related entities from SecEPM should be referenced by every guidance artifact, although these artifacts might mainly get referenced from other entities (especially guidelines).

4.6.2. Rate Security Goals Adapting IT-BPM

4.6.2.1. Overview

In order to prioritize efforts and investments aiming at the security of an electronic business process, specified security goals have to be differentiated with regard to the impact of their violation. This rating of security goals should be supported by a systematic approach since the results should be repeatable and transparent for all stakeholders. The IT-BPM provides such a

Name	Rate Security Goals Adapting IT-BPM
Aim	Systematic rating of security goals
Related entities	• Tasks: Setup Process/Provide Basic Definitions, Identify Assets/Analyze Dependencies Between Assets and Resources, Assess Security Goals/* • Work products: Process Model Configuration, Security Analysis Model • Roles: Security Analyst, Domain Expert, Business Process Engineer
Guidelines	• Define Damage Scenarios and Rating Criteria • Analyze Dependencies • Apply Rating Criteria • Aggregate Security Goal Ratings
Additional artifacts	• Example: List of damage scenarios and rating criteria • References, concepts

Table 4.13.: Guidance: Rate Security Goals Adapting IT-BPM

method. Guidance for the adaptation and application of this method in a security engineering process based on SecEPM is detailed in this section.

Beside this overview, four guidelines are supplied describing the following tasks (cf. table 4.13): The first guideline supports the definition of damage scenarios and rating criteria in the task Provide Basic Definitions from the activity Setup Process. Another guideline details the analysis of dependencies for the task Analyze Dependencies Between Assets and Resources from the activity Identify Assets. The last two guidelines describe the application of rating criteria in order to rate security goals as well as the aggregation of security goal ratings depending on their dependencies (tasks Rate Individual Security Goals, Analyze Dependencies Between Security Goals, and Adjust Security Goal Ratings from the activity Assess Security Goals). The influenced or altered work products are the Process Model Configuration entailing basic definitions and the Security Analysis Model specifying identified dependencies and actual ratings. The first guideline addresses the Security Analyst and aims at supporting the provision of necessary definitions during the Setup Process activity. The other guidelines address mainly the Business Process Engineer and partly the Domain Expert supporting the named tasks. Additionally, a list of exemplary damage scenarios and rating criteria is delivered with the SecEPM repository as starting point for the Security Analyst. Also, references to the original method and definitions of specific concepts like damage scenario and aggregation strategy are given in the SecEPM repository.

The IT-BPM is a methodology for the management of information security. It is published as a series of standards by the Bundesamt für Sicherheit in der Informationstechnik (BSI) (German Federal Office for Information Security) as “IT-Grundschutz” [Bun08] in order to establish and maintain an appropriate level of protection for information assets of an organization. As it aims at the management of information security it is only weakly linked to security engineering in the first place. Three properties qualify the rating approach from the IT-BPM as a good candidate for application in SecEPM-based security engineering processes: First, information security management relies similar to security engineering on repeatable and transparent ratings of security goals. Second, one of the most important objectives of IT-BPM

[5] http://www.eclipse.org/epf/

is the reduction of expenses of the security process within an organization. It addresses therefore not only security professionals but also IT and project managers and provides extensive documentation. Third, IT-BPM is (at least in German-speaking countries) widely applied and results from those efforts can be partly reused applying the method in security engineering (e.g., definitions of damage scenarios and rating criteria). Consequently, resulting ratings are aligned between information security management and security engineering.

The method to rate security goals from the IT-BPM proposes the following procedure: First, damage scenarios are defined that provide the analysis background in order to assess possible negative consequences of violations of security goals. Second, rating criteria capturing individual negative consequences are defined that assign a specific rating from an ordinal scale to a security goal. Third, as ratings of security goals might be influenced by dependencies between assets, those dependencies are explicated and an aggregation strategy is defined for the rating of each (dependent) security goal encompassing maximum, cumulation, or distribution strategy. Fourth, the actual rating is derived from the criteria and the dependency structure. The following guidelines describe how to apply the rating method within the course of a SecEPM-based security engineering process.

4.6.2.2. Define Damage Scenarios and Rating Criteria

In order to rate security goals following the IT-BPM the Security Analyst supported by the Domain Expert provides necessary definitions for the Process Model Configuration within the task Provide Basic Definitions from the activity Setup Process.

At first, damage scenarios are identified. A damage scenario describes a typical scenario that might result from the violation of security goals. Damage scenarios are stakeholder-centered, i.e., they describe negative consequences from the point of view of a relevant stakeholder (which will often be the business point of view) and not from a technical perspective. Damage scenarios need not to be disjunctive, the violation of one security goal might address several damage scenarios. Nevertheless, the set of defined damage scenarios should cover all typical negative consequences. Damage scenarios are defined providing at least a unique label and a short description. Best practice from the BSI recommends to capture relevant aspects of every damage scenario by formulating one or more questions for each security goal class and damage scenario. Typical damage scenarios encompass “Violations of laws, regulations or contracts”, “Physical injury”, “Financial consequences” and others. One question to capture a relevant aspect of the damage scenario “Financial consequences” with regard to the security goal class “Confidentiality” might be: “Could the publication of confidential information lead to compensation claims?”

In the next step, for every damage scenario and rating one or more criteria are defined that explicate conditions that subsume a security goal under this damage scenario and rating. Criteria must be disjunctive with regard to one damage scenario and different ratings. They determine the limits of each damage scenario and the respective rating. They should be easy to decide from the stakeholder’s perspective and exclude personal opinions or other subjective factors as much as possible. It is often helpful to decide for one or more similar aspects of a damage scenario addressed by the corresponding questions on limits to distinguish between the different ratings and note them as criterion. For the damage scenario “Financial

consequences" the financial loss might be qualified (and also quantified) as "Acceptable loss (< 100,000 EUR)" (assigned rating: "Normal"), "Considerable loss (>= 100,000 EUR and < 10,000,000 EUR)" (assigned rating: "High"), "Critical loss (>= 10,000,000 EUR)" (assigned rating: "Very high").

An additional guidance artifact provided in the SecEPM repository provides a list of exemplary damage scenarios, rating criteria, and questions extracted from [Bun08].

4.6.2.3. Analyze Dependencies

The dependencies between the ratings of security goals are considered by the IT-BPM analyzing dependencies between assets in a top-down manner (relating assets from different levels of abstraction or perspectives like business process, application, network etc.). The result is an a-cyclic dependency graph with assets as nodes captured in the Security Analysis Model. This guideline explains the adoption of the technique for the application on electronic business processes in the task Analyze Dependencies Between Assets and Resources from the activity Identify Assets.

In order to support the generation of a-cyclic dependencies between the assets, an ordered list of asset types is defined by the Business Process Engineer supported by the Domain Expert starting from the most general or abstract one. Depending on the technique for asset identification, the following asset types might apply: business assets, process assets (matching Pool entities), service assets (matching Activity entities), and message assets (matching Message entities).

In a second step, the Business Process Engineer supported by the Domain Expert analyzes and documents dependencies between the identified assets. They start with assets from the first type in the ordered asset type list and assets from the next type in the list (the "lower" assets support matching "higher" assets). One asset might support several other assets. This creation of dependencies is repeated for every step in the list of asset types. For modeled (non-business) assets, the dependency might be generally be deduced from an existing "contains" or "connected with" relation of the respective element in the business process model. There might be multiple resulting (a-cyclic) dependency graphs, every asset not contained in an dependency graph is counted as single dependency graph.

In order to trace security goals, requirements, and controls back to business assets, every dependency graph should include at least one business asset. Therefore, dependency graphs not containing a business asset (orphan graphs) need revision and have to be related to existing (non-orphan) dependency graphs connecting assets of the "highest" type in the orphan graph with assets of an even "higher" (or preceding) type in the other graph.

If the following assets have been identified: "Successful POD", "Pool P1", "Message M1", and "Message M2", the following dependency graph is established: "Successful POD" is supported by "Pool P1" which is supported by "Message M1" and "Message M2".

4.6.2.4. Apply Rating Criteria

In order to document the basis for an actual rating of a security goal, the Business Process Engineer supported by the Domain Expert assigns to every security goal all matching criteria

from all damage scenarios defined in the Process Model Configuration in the task Rate Individual Security Goals from the activity Assess Security Goals. The assignments are stored in the Security Analysis Model.

Every security goal specified for a business asset needs at least one criterion to be assigned to it. It might not be reasonable to assign rating criteria to modeled assets. Those security goals will be determined by the ratings for the security goals for the supported business assets. Rationale or comments might be added as notes to the assignments. To derive an individual rating for a security goal, the maximum rating from all assigned criteria is evaluated as resulting rating.

If the criterion “Limited financial losses” is assigned to the security goal “Successful POD” the rating is evaluated as “Normal”.

4.6.2.5. Aggregate Security Goal Ratings

The dependencies between the security goals are considered analyzing the dependencies between the corresponding assets. In a first step, the dependencies between the assets are qualified for each security goal by the Business Process Engineer with regard to an aggregation strategy in the task Analyze Dependencies Between Security Goals. The resulting ratings for the security goals are derived in a second step applying these aggregation strategies to the individual ratings of the security goals in the task Adjust Security Goal Ratings (both tasks are from the activity Assess Security Goals, results update the Security Analysis Model).

Aggregation strategies define how ratings of supported assets influence the rating of the supporting asset. The IT-BPM proposes the application of one of three strategies to consider the influences of asset dependencies: the maximum, the cumulation, and the distribution strategy. If the maximum strategy is applied, the rating of a security goal is set to the maximum of ratings of security goals of supported assets and its individual rating. If the cumulation strategy is applied, the rating of a security goal is increased (e.g., on the subsequent level) after the application of the maximum strategy. If the distribution strategy is applied, the rating of a security goal is decreased (e.g., on the preceding level) after application of the maximum strategy.

For every security goal, the initial aggregation strategy is set to the maximum strategy. After that, the Business Process Expert might apply the cumulation strategy to individual security goals, if a cumulation of different supported (business) assets occur to a security goal, e.g., if (low-rated) security goals from multiple business assets depend on the confidentiality of one message. Inversely, the distribution strategy might be applied to a security goal, if only a joint violation of several security goals affect a security goal of a supported asset, e.g., if a (high-rated) security goal from one business asset is only endangered by the violation of the integrity of a message and the integrity of the execution sequence of a process.

4.7. Tool Support and Integration

The preceding sections introduced the structure and key entities of SecEPM, presented its security engineering activities, and demonstrated guidance artifacts that support the execution of the activities. In order to apply SecEPM, two aspects are still missing: tool support for

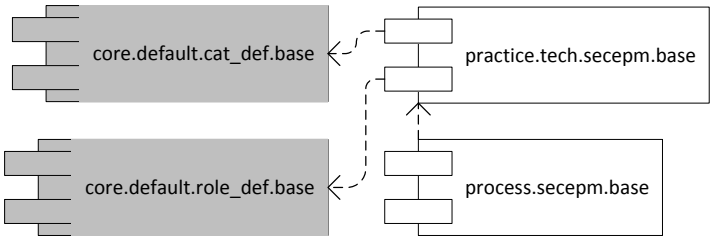

Figure 4.7.1.: Plug-in structure of SecEPM SPEM (UML component diagram)

authoring, storage, retrieval, and instantiation of SecEPM resources and integration of SecEPM into existing software development process models. The following section 4.7.1 describes tooling and its application. After that, integration of SecEPM into existing software engineering process models is discussed and an example integration is detailed in section 4.7.2. Work in this section is partly based on a master's thesis written in the context of this thesis [Mol12]. Results presented here have been largely reviewed, re-engineered, and enhanced compared to that thesis.

4.7.1. Tool Support to Tailor a Security Engineering Process

In order to apply a process model like SecEPM an actual security engineering process tailored to the needs of the development project has to be defined. To facilitate this procedure, the application of supportive tooling providing a computer aided method engineering (CAME) environment is recommended in literature (e.g., [Bri96, HSR10]). Tooling to provide a CAME environment might range from simple editors supporting given method engineering languages to integrated tool chains with sophisticated retrieval, validation, and generation facilities to automate recurring tasks in the situational method life cycle [NR08].

Following Arni-Bloch, a comprehensive CAME environment should support at least a repository as well as tooling for authoring and instantiation [AB05]. The repository is used to store and retrieve method fragments from the method base. Method fragments are defined, assembled, and tailored using the authoring tooling. The tooling for instantiation is used to generate and publish the actual security engineering process and supportive artifacts.

We selected the EPF Composer[6] in order to support storage and authoring of SecEPM as well as instantiation of SecEPM-based security engineering processes. The EPF Composer is an Eclipse-based rich client available under the Eclipse Public License. It integrates the functionality demanded by Arni-Bloch within an extensible integrated development environment (IDE) and is considered as advanced tool in the field [SHAAR12]. Several software development process models have been published using the EPF Composer based on SPEM. Hence, the entities of SecEPM have been modeled using the EPF Composer applying SPEM.

Packaging and naming of modeled elements for SecEPM is largely determined by the library management approach of SPEM and the EPF Composer. Figure 4.7.1 depicts the plug-in

[6] `http://www.eclipse.org/epf/`

structure and their most important dependencies. Plug-ins with gray background are provided by the EPF.

- core.default.cat_def.base: This plug-in defines common categories for SPEM-based process models. These include activity groups such as “requirements” that are used to categorize process fragments as well as so called domains like “architecture” that are applied to product fragments. Matching categories are reused to model SecEPM categories.

- core.default.role_def.base: Common roles are defined in this plug-in. This includes the roles project manager, developer, and tester. Matching roles are reused to model SecEPM roles.

- practice.tech.secepm.base: This plug-in entails basic method fragments and chunks of SecEPM. This includes role, work product, and task definitions as well as their relations. Similarly, guidance artifacts are defined within this plug-in.

- process.secepm.base: Compositional method chunks such as activities and process patterns are defined in this plug-in. Therefore, method fragments from the plug-in practice.tech.secepm.base might be reused without any dependencies to this plug-in.

Figure 4.7.2 displays the modeling of individual SecEPM elements and their relations using the task Identify Business Assets as example (cf. section 4.5.2.2). The task is modeled applying the Task Definition element from the metamodel. The Business Process Engineer and the Domain Expert are modeled applying the Role Definition element and assigned to the Identify Business Assets element using the Performer association. Work Product Definitions apply to the relevant work products. They are assigned to the defined task using the respective Parameter association and to the matching roles with the Responsibility Assignment association. The Identify Assets activity is modeled using the Activity element from the metamodel. The Role Use and Work Product Use relations are omitted for simplicity. SecEPM itself is modeled as Process Pattern representing one possible application of the defined activities.

The authoring of SecEPM using the EPF Composer is shown in figure 4.7.3. It displays the preview of one specific method fragment (the task Identify Business Assets, cf. section 4.5.2.2) that has been detailed as well as parts of the method base (“Library”) and the current tailoring (“Configuration”).

Figures 4.7.4 and 4.7.5 depict two exemplary artifacts that have been instantiated using the EPF Composer: Figure 4.7.4 on page 94 displays a web page that provides all information about the actual development process utilizing SecEPM. Such web pages are commonly published in the intranet of developing organizations to disseminate relevant documentation, terminology, templates etc. to all stakeholders. A project management template for Microsoft Project is displayed in figure 4.7.5 on page 95. Such templates speed up project initiation based on tailored development processes and harmonize the actual management of the development projects.

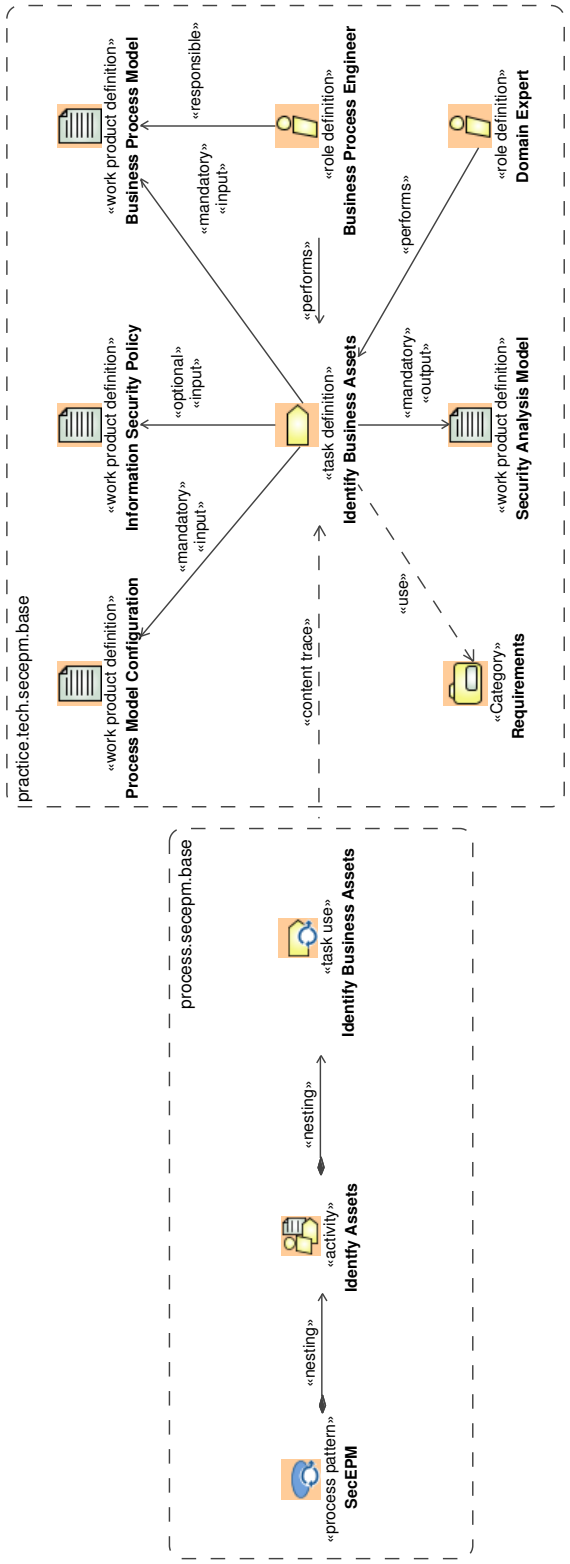

Figure 4.7.2.: Modeling of an exemplary SecEPM task (SPEM notation)

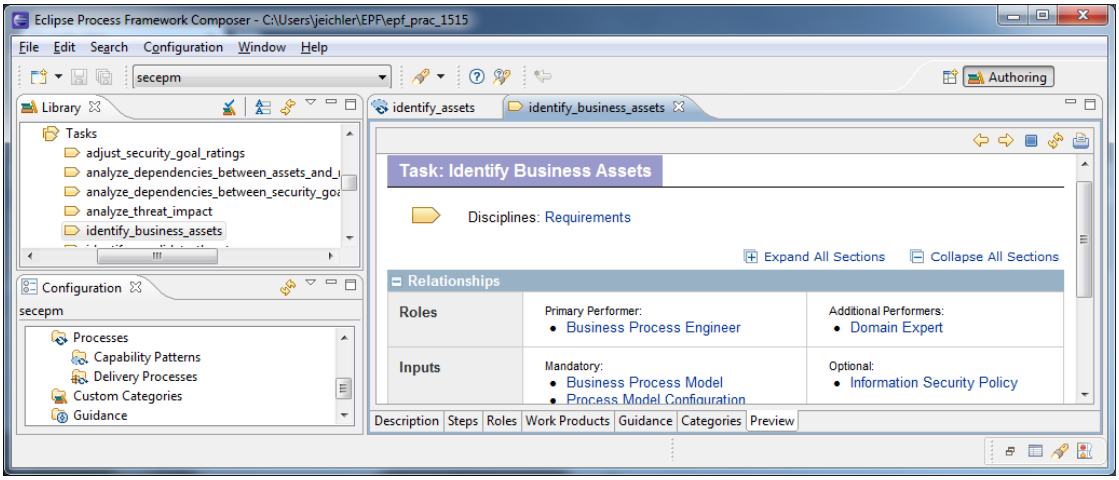

Figure 4.7.3.: Authoring the SecEPM process model using the EPF Composer

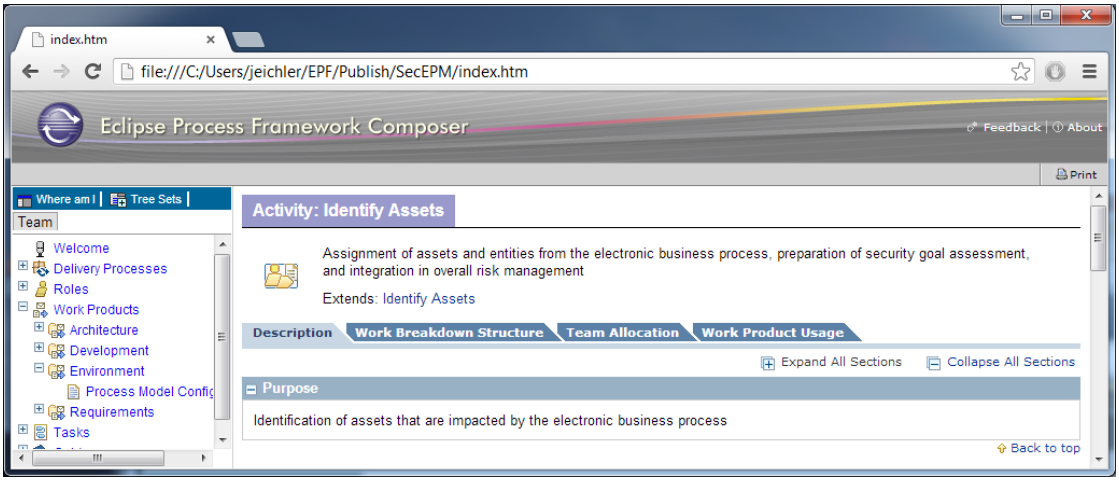

Figure 4.7.4.: Documentation of a SecEPM-based security engineering process

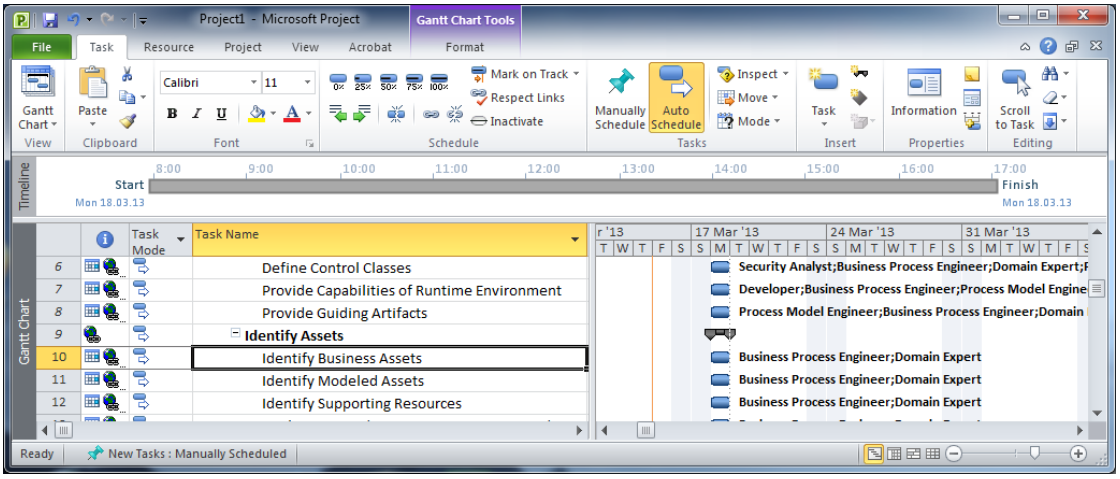

Figure 4.7.5.: A Microsoft Project template for a SecEPM-based security engineering process

4.7.2. Integration into Software Development Process Models

SecEPM is a process model for security engineering. It provides an integrated representation of security-related activities, roles, and work products in order to support the development of secure electronic business processes from asset identification to the configuration of controls. Being restricted to security-related entities, SecEPM needs to be integrated into existing software development approaches as it has been discussed before (cf. section 2.4.2).

Two important aspects have to be addressed to integrate a security engineering process based on SecEPM into an existing software development approach systematically: First, the integration of method fragments and chunks from two different method bases into one method base. Second, the integration of method chunks from the integrated method base into one development process. We will focus on the first aspect in this section as the second is largely dominated by requirements of the development project and not so much from general considerations.

The integration of method fragments and chunks from different method bases is regarded as part of the method engineering discipline. In this section, we adopt an approach presented by Ralyté et al. [RR01]. Ralyté et al. consider two main intentions with regard to the integration of product and process fragments: adaptation of method fragments and construction of integrated method fragments. Main strategies to address these intentions are name unification, merge, specialization, generalization, remove, and addition. The *name unification* strategy is applied to adjust the terminology used for method fragments. Similar terms with different semantics are disambiguated, different terms with similar semantics are unified. Method fragments with similar semantics and structure are consolidated applying the *merge* strategy. The similarity of semantics and structure does not only address the imminent meaning and the dimensions introduced before (perspective, layer of granularity, and level of abstraction, cf. section 2.3.2) but also include specialization and generality. Complementary, the *specialization* strategy is applied to associate more general fragments with more specialized ones. Two elements with different specialization might be generalized introducing a new generalized

fragment applying the *generalization* strategy. Redundant or useless method fragments are eliminated using the *remove* strategy. The *addition* strategy is used to introduce new fragments especially in order to streamline the integrated method fragments.

We demonstrate the integration of SecEPM into OpenUP[7], a software development process model that comprises an iterative and incremental approach within an structured live cycle published applying the Eclipse Public License [Gus08]. Both process models are compliant with SPEM and available for the EPF Composer. As SPEM is designed to support method adaptation, integration, and tailoring it provides several means to apply the integration strategies explicated above (cf. section 2.3.2). Therefore, we present our general approach to apply integration strategies proposed by Ralyté et al. utilizing different mechanisms provided by SPEM first. Main decisions with regard to the integration of SecEPM and OpenUP are summarized subsequently.

The integration strategies have to be applied with regard to different kinds of method fragments. Generally, we consider two distinctions to differentiate our approach: product versus process fragments and basic versus composite method chunks. The latter adopts the perspective of SPEM differentiating elements from the Method Content and other packages, especially the Process Structure package (cf. section 2.3.2).

- The application of the *name unification strategy* is independent of the different kinds of method fragments: Method fragments with ambiguous or equivocal naming are renamed using a Variability Element applying the “extends-replace” Variability Type. For example, the Category “Design” for activities from SecEPM is unified with the Category “Architecture” from OpenUP renaming the former to the latter.

- The application of the *merge strategy* might get complex. Basic method fragments or chunks are merged utilizing a Variability Element with the Variability Type “contributes”, i.e., one method fragment or chunk contributes to the other. In order to merge Activities Method Content Use and/or Activity Use Kind elements are utilized. Basic breakdown elements of Activities are merged providing alternative Method Content Use elements that adapt necessary relationships. Nesting Activities are merged extending one Activity utilizing “local contribution” and “local replacement” for appropriate nested Activities. Merging of basic and composite method chunks might need a two step approach utilizing both a Variability Element and an Activity Use Kind. Exemplary, the Task “Tailor the Process” from OpenUP is merged with the Activity “Setup Process Model” from SecEPM contributing the content from the Task “Provide Guiding Artifacts” to the Task “Tailor the Process” first (utilization of a Variability Element with Variability Type “contributes”). Furthermore, it extends the Activity “Prepare Environment” from OpenUP that includes the application of the Task “Tailor the Process” contributing the remaining tasks from “Setup Process Model” subsequently (utilization of an Activity Use Kind element with “local contribution”).

- The *specialization strategy* is applied for product fragments introducing a Work Product Definition Relationship. For other method fragments a Variability Element is utilized commonly that contributes the relationship between those fragments. If specialization

[7] `http://epf.eclipse.org/wikis/openup/`

includes containment, the more general fragment might simply nest the more specialized fragment for appropriate fragments (e.g., applying the Activity Use Kind element providing "local contributions"). Analogous to the application of the merge strategy, specialization of basic and composite method chunks might need a two step approach. As a simple example, the Activity "Elicit Security Requirements" from SecEPM is a specialization of the Activity "Identify and Outline Requirements" from OpenUP and is nested into it (utilization of an Activity Use Kind element with "local contribution").

- For the application of the *generalization strategy* a new method fragment or chunk is defined that is integrated using Method Content Use, a Variability Element with the Variability Type "replace", or the existing specialized fragments are nested into the new method chunk. The generalization strategy is not applied for the integration of SecEPM into OpenUP.

- In order to apply the *remove strategy* individual method fragments or chunks are not deleted from the method base but either the containing Method Plugin is removed from the Method Library or the method fragment is removed from the Method Configuration applied. The remove strategy is not applied for the integration of SecEPM into OpenUP.

- The *addition strategy* simply introduces a new method fragment or chunk. Exemplary, a Guidance element explaining the application of a SecEPM-based security engineering process within an incremental development process might be introduced.

For the actual integration, we take the viewpoint of an integration of SecEPM into OpenUP. Therefore, OpenUP concepts and structures dominate within the integration process. Other alternatives encompass the integration of OpenUP into SecEPM or the integration of OpenUP and SecEPM into an additional process model. Table 4.14 on the following page summarizes main decisions.

Since both process models are SPEM compliant, and SecEPM reuses common SPEM categories, not many high-level terms needed unification (exception: activity group vs. discipline). Similarly, reuse of common SPEM roles in SecEPM reduced the integration effort with regard to these as well. Beside name unification for the role Process Model Engineer only the Domain Expert from SecEPM has been integrated as specialization of the general Analyst role from OpenUP utilizing the variability element (VE) with the Variability Type "contributes". Most work products of SecEPM are either complementary to OpenUP work products (e.g., Threat Catalog) or have been integrated as specializations of existing work products utilizing Work Product Definition Relationships (WPDRs) (e.g., Security Analysis Model and System-wide Requirements). Similarly, most activities and tasks of SecEPM detail those of OpenUP and have been integrated applying the specialization strategy utilizing VE or Activity Use Kind (AUK) elements.

A mapping of one standard software development process model onto existing security engineering approaches has been proposed by Lee et al. [LLL02]. As that mapping includes all phases of the software life cycle as well as all security-related activities published by different authors it is far more coarse grained than the the mapping presented in this section. Moreover, as the proposal from Lee et al. does not address specific process models it does not consider

Type	SecEPM	OpenUP	Strategy
Term	Activity Group	Discipline	Name unification
Role	Domain Expert	Analyst	Specialization (VE, contributes)
	Process Model Engineer	Process Engineer	Name unification
Activity / Task	Setup Process Model	Prepare Environment	Merge (AUK, local contribution)
	Provide Guiding Artifacts	Tailor the Process	Merge (VE, contributes)
	Identify Assets Assess Security Goals Model Threats Elicit Security Requirements	Identify and Refine Requirements	Specialization (AUK, local contribution)
	Identify Assets Assess Security Goals	Identify and Outline Requirements	Specialization (VE, contributes)
	Model Threats Elicit Security Requirements	Detail System-Wide Requirements	Specialization (VE, contributes)
	Design Controls Map Controls Generate Control Artifacts	Develop Solution Increment	Specialization (AUK, local contribution)
	Design Controls Map Controls	Design the Solution	Specialization (VE, contributes)
	Generate Control Artifacts	Implement Solution	Specialization (VE, contributes)
	Generate Test Artifacts	Implement Tests	Specialization (VE, contributes)
Work Product	Process Model Configuration	Project Defined Process	Specialization (WPDR)
	Security Analysis Model	System-wide Requirements	Specialization (WPDR)
	Security Design Model	Design	Specialization (WPDR)
	Implementation Artifacts	Implementation	Name unification
	Test Artifacts	Test Cases Test Scripts	Specialization (WPDR)
	Business Process Model	Implementation	Specialization (WPDR)

Table 4.14.: Integration of SecEPM into OpenUP

integration strategies and their application explicitly. A security extension for the OpenUP has been published by Ardi et al. [AS08]: A set of security-related activities for the development of general purpose software is directly designed for OpenUP. No integration is necessary as concepts and definitions have been taken directly from the OpenUP. Likewise a security extension for the RUP has been described from Paes et al. [PH07]. The integration of a specific security engineering process model into a software development process model and the methodical generalization of the integration procedure is not provided by any of the authors.

4.8. Summary

This chapter introduced SecEPM, our proposal for a security engineering process model that integrates security-related activities in the course of the development of secure electronic business processes.

We started by narrowing the scope of the process model to be developed: It should cover the development of a security analysis model for a given business process model, the creation of a matching security design model and a mapping of the security design model onto an implementation in order to bridge the gap between executable business process models and properly selected and configured controls for corresponding business process engine.

Based on this scoping, we identified eight major requirements in section 4.2 covering the structure of the process model (1), the coverage with regard to security engineering activities (2), the separation of problem and solution domain (3), traceability from high-level security goals to control configurations (4), integrability into different development process models (5), independence from development-time and runtime technology (6, 7), and the ability to be applied in environments with restricted skill sets available with regard to security (8).

We applied three key strategies for the design of SecEPM to meet these requirements, (cf. section 4.3): specialization of existing process models and practices to incorporate existing knowledge and experience but restricting the skill set necessary to complete the activities (1), separation of concerns to realize a modular and flexible process model (2), and decoupling aiming at relaxing constraints of the process model with regard to execution sequence and preconditions of the activities (3).

We presented the resulting process model introducing the structure covering four activity groups of the respective IEEE standard [Ins97], nine activities focusing the early phases for the development, two core work products (Security Analysis Model and Security Design Model), four additional work products, and six roles highlighting the role of the Security Analyst and the Business Process Engineer (cf. section 4.4).

Activities describing tasks to be performed in order to develop secure electronic business processes and their relations to other elements of the process model have been detailed in section 4.5. Support for the execution of these tasks is provided by guidance artifacts. Two topics—provisioning of guidance artifacts as well as rating of security goals adapting a method from the IT-BPM—have been covered with exemplary guidance artifacts, detailing relations with other elements of the process model and presenting respective guidelines (cf. section 4.6).

Authoring, storage, retrieval, and instantiation of usable resources for project participants of SecEPM-based security engineering processes using adequate tooling has been documented in section 4.7. This section also covered the methodical integration of SecEPM into existing software development process models providing an exemplary integration of SecEPM into OpenUP.

5. Security Engineering Modeling Language

5.1. Introduction

SecEPM, our process model for security engineering, has been presented in the previous chapter. SecEPM utilizes several work products. Exemplary, the Security Analysis Model and the Security Design Model are central to many SecEPM activities. Intentionally, the process model does not prescribe how to note and manage these work products. To minimize prerequisites necessary to apply SecEPM, the choice of notations and tooling is left to the Process Model Engineer and other participating stakeholders.

Nevertheless, in order to apply a security engineering process effectively an adequate modeling language and supportive tooling is considered helpful [SH08]. This chapter introduces SecEML, our proposal for a DSML to capture work products of SecEPM and to provide a basis for the validation of work products and their transformation into deployment artifacts. It addresses the second research question and documents the remaining part of results of the design and development activity of our research approach (cf. section 1.3):

2. What requirements does a DSML place that supports the elaboration of main work products of the developed security engineering process model and how could they be met?

SecEML has a two faced relation with the ideas of MDE. First, as a DSML it provides means to capture work products of the security engineering process applying concepts of the domain (i.e., security engineering) minimizing the semantic gap. Therefore, it allows for a model-based security engineering approach applying the ideas of MDE in the domain of security engineering.[1] Second, tooling supporting the application of SecEML is provided applying model-driven techniques and frameworks. Hence, SecEML and its tooling is situated in a model-driven environment as result of the application of MDE. Together, SecEML and supportive tooling complement our approach to bridge the gap between (executable) business process models and the design of proper controls and their configuration.

An early version of the modeling language has been published before as well as experiences developing tools to support the application of the language using model-driven techniques and frameworks [Eic11a, Eic11b, EFL12]. The results presented in this chapter have been considerably complemented, detailed, and enhanced compared with those publications.

SecEML is presented as follows: The next section 5.2 defines the scope of SecEML and analyzes key requirements starting with the issues identified in the problem statement and

[1] Please consider the different focuses of the terms *model-driven* engineering and *model-based* engineering detailed in section 2.3.3.

discussing criteria from literature. Section 5.3 presents strategies applied for the design of SecEML. The language is introduced in section 5.4 providing abstract and concrete syntax. Section 5.5 discusses the application of SecEML and details guidance using SecEML to capture work products of SecEPM. Additionally, the development of tooling for the application of SecEML is covered in this section. A summary of the main results in section 5.6 concludes this chapter.

5.2. Requirements

SecEML aims at supplying an adequate DSML for the application of SecEPM. A DSML to support the application of SecEPM might address a broad range of issues. We limit the scope of SecEML to capture core work products of SecEPM focusing the Security Analysis Model and the Security Design Model. Additional work products like the Threat and Control Catalog as well as the Runtime Capability Model and the Process Configuration Model are considered as much as their interaction with these main work products is addressed and no conceptual changes to SecEML would be necessary for this interaction. The intended purpose of capturing work products using a DSML is to allow for model-based security engineering fostering a shared understanding of the results of the activities of SecEPM and an iterative and incremental execution of the respective tasks.

The formal analysis or proof of certain properties of electronic business processes is considered outside the scope of SecEML. Several languages for this purpose have been discussed in the sections on related work (cf. section 2.5) and might include proposals from Arsac et al., Weldemariam et al., and Armando et al. [ACPP11, WV11, AGMP12]. Similarly, high-level representation and validation of runtime policies or the configuration of individual security mechanisms is not at the heart of SecEML. Approaches concentrating on those topics include proposals from Basin et al., Moebius et al., and Wolter et al. [BDL06, MRS09, WMS+09]. They are briefly covered in sections 2.3.3.3 and 2.5. SecEML is not intended as replacement for those approaches. Instead, SecEML is intended as complimentary DSML that provides a security engineering process point of view and might reference artifacts and elements of those approaches.

As SecEML focuses on support for the application of SecEPM, key requirements for SecEML address the major issues detailed in the problem statement. The first key requirements address mainly the impairment of skills available in BPM development projects with regard to security and general aspects like the coverage of necessary concepts.

1. Coverage: Concepts and relations defined and required by SecEPM for the description of its core work products must be supported by the DSML.

In order to leverage also security nonprofessionals, not only all relevant concepts and relations have to be covered by SecEML but also the concrete syntax has to be easy to learn, to read, and to use. Therefore, it should provide a familiar appearance also for security nonprofessionals.

2. Accessible Concrete Syntax: The concrete syntax should be easy to learn and to apply by security nonprofessionals.

Even if a syntax is easy to use and the relevant concepts are directly represented by a DSML, the creation, validation, and analysis of models applying that DSML is laborious and error-prone if there are no tools to support the user executing these tasks. Automation as one of the core ideas of MDE addresses this topic similarly, although we do not require an automation of the execution of the activities of SecEPM.

3. Technical Assistance: Tools should be provided that support users of the DSML creating, validating, and analyzing models applying the DSML.

Also, SecEML has to tackle the heterogeneity of BPMS. Therefore, it can not be required that a given notation is supported by a BPMS in order to apply SecEML. Furthermore, security controls, their configuration, and runtime capabilities of BPMS differ largely between different vendors. Insofar a user is able to use SecEML to describe runtime capabilities and to specify their configuration, specific runtime capabilities must not be presupposed in order to apply SecEML.

4. Independence of Notation: The DSML must be applicable regardless of the notation that is used for the specification of the electronic business process.

5. Independence of Runtime Capabilities: The DSML must not require specific runtime capabilities of BPMSs.

The environmental heterogeneity that SecEML has to face is addressed by two further requirements: Integration in Tool Chains and Coexistence with Existing Models (cf. requirements 6, 7). The creation, validation, transformation, and management of models using SecEML should not require a new infrastructure and conceptually deviating tooling. As the environment differs between the different organizations applying BPM the possibility to use standard tools to handle SecEML models reduces cost, effort, and risk of using SecEML and increases the likelihood of an successful application. Similarly, if the application of SecEML does not require changes to existing models but SecEML models profit from existing models by establishing relations between elements from those models, introduction of SecEML does not break existing tool chains and processes but allows for an enhancement of tool chains and processes.

6. Integration in Tool Chains: The application of the DSML and the integration in existing tool chains does not require extensive prelimiaries but might reuse standard tooling.

7. Coexistence with Existing Models: The DSML does not require changes to existing models using different notations or their tooling but elements from integration-friendly notations can be related to elements of models using the DSML.

A consensus on general requirements for DSMLs in the domain of security engineering is not known to the author. Therefore, the following paragraphs discuss criteria applied in literature to evaluate security modeling approaches in the context of the key requirements and those criteria will be used as checks for the validity of the key requirements.

Diallo et al. compare three approaches to specify security requirements [DRMSR06]. They introduce and apply five criteria or requirements for approaches to specify security requirements: lernability, usability, solution inclusiveness, clarity of output, and analyzability. *Learnability* addresses the question, how long typical project participants take to learn and use

the approach. *Usability* asks for the complexity of the application of the approach (e.g., comparing number and difficulty of necessary steps). *Solution inclusiveness* requires that the compared approaches not only specify threats but also cover (possible) controls. *Clarity of output* addresses the ease of understanding of resulting artifacts for typical project participants (in opposite to the complexity of the application of the approach). *Analyzability* requires the possibility and ease to analyze results with regard to typical questions, e.g., whether all identified threats have been considered.

Another work on the usability of approaches for the specification of security properties focusing on UML artifacts is presented by Talhi et al. [TML+09]. To evaluate the usability of different approaches they propose four criteria: expressiveness, tool support, verifiability, and complexity. *Expressiveness* addresses the provision of means to specify all necessary (security) propositions. *Tool support* requires availability of tooling to support specification and analysis of resulting models. *Verifiability* is defined as the ability to "verify the design against the security requirements" [TML+09, p. 13], i.e., the possibility to prove (security) properties of the system (design) that has been specified. *Complexity* evaluates the amount of necessary information that has to be added by the approach in question and its impact on the readability of the original UML artifacts.

We address the criteria lernability, usability, clarity of output, complexity, and tool support with our requirements Accessible Concrete Syntax and Technical Assistance (requirements 2 and 3). An accessible concrete syntax supports learnability and clarity of output especially in comparison with approaches focusing a compressed or machine-friendly concrete syntax. Additionally, it addresses the complexity criteria: As SecEML does not annotate models using different notations, the readability of those models is not endangered by the application of SecEML. Technical assistance addresses similar issues as tool support. It addresses also usability since technical assistance can reduce the difficulty of tackling a given task. Analyzability is partly addressed by the requirements Technical Assistance and Coverage (requirements 1 and 3) as tooling should support validation and analysis of the entities represented in the DSML. Expressiveness is restricted to the purpose of SecEML: the modeling of core work products of SecEPM covering all relevant concepts (cf. Coverage requirement 1). Solution inclusiveness is also addressed by this requirement since controls are part of the Security Design Model. Verifiability is excluded from our requirements, since formal analysis of properties of electronic business processes has been excluded from the SecEML's scope. The criteria do not exclusively address requirements of a DSML. Also the design approach of a DSML addresses some of the criteria and will be covered in the following section.

5.3. Design Approach

We applied the language engineering approach of Kleppe for the development of SecEML [Kle08]. Kleppe proposes an iterative incremental engineering process focusing on the metamodel of the DSML (cf. section 2.3.3). The main steps of the engineering process are:

1. Create an abstract syntax (here: metamodel)
2. Define and map a concrete syntax

3. Test definitions with some example models
4. Define semantics
5. Create tools to support the user

The steps are repeatedly applied with increasing comprehensiveness, i.e., the first iteration might include only the first three steps and the results are revised in following iterations before semantics and tooling are tackled.

Within this engineering process, we applied the following main design strategies in order to meet the key requirements detailed in the preceding section:

1. Direct Representation
2. Modularization
3. Reuse
4. Model-driven Techniques

The Direct Representation strategy follows one of the main ideas of MDE (cf. [BBI+04, Bé06] in section 2.3.3): In order to minimize the semantic gap, concepts of the application domain are directly represented by the DSML. The terminology defined in section 2.4.1 and applied to specify elements and relations in the Security Analysis Model and the Security Design Model from SecEPM become entities in SecEML. Therefore, security goals, threats, controls etc. are directly represented and specifiable using SecEML.

The Modularization strategy aims at the provision of sparsely dependent compartments that constitute the modeling language instead of one monolithic block. It addresses several aspects of the language design, mainly at the level of the structuring of the abstract syntax. Exemplary, the application of the Modularization strategy (partly in combination with the Reuse strategy) introduces name spaces in order to specify parts of the model independently from other parts, provides means to separate generic specifications from specializations for project specific aspects, and allows for the specification of design and implementation aspects using related but different model elements.

Similarly, the Reuse strategy is applied with regard to different aspects of the design of SecEML. Reuse strives for the integration and application of existing entities instead of redundant specifications and implementations. Exemplary, the application of the Reuse strategy introduces means to access existing models and their elements instead of redundant redefinitions. Furthermore, it opts for the integration of established languages and implementations if possible (e.g., the application of OCL for the specification of constraints, derived attributes and analysis statements), and provides means to use partitioned models serialized using multiple resources (instead of one monolithic one) that might be reused in different projects.

The application of model-driven techniques and frameworks addresses the definition of SecEML and the provision of respective tooling. Specifying SecEML in a way that allows for the application of model-driven techniques and frameworks as well as applying those frameworks render short provisioning and feed back cycles optimizing the language possible. Furthermore,

the application of model-driven techniques and frameworks aligns SecEML with the growing group of DSMLs using similar techniques and frameworks and eases integration and reuse.

The decision on the means to relate SecEML models with business process models is largely restricted by the requirement to allow for a coexistence of SecEML models with other models (requirement 7): From the four strategies discussed in literature (cf. section 2.3.3.3) only entity reference and model weaving do not require changes to existing models or their tooling. As the entity reference strategy does not require the provision of an additional language (the weaving language) and respective tooling, we decided to apply it for SecEML.

Several requirements are met by implementation decisions more than design strategies. Nevertheless, the design strategies named in this section contribute to some of the requirements: The Direct Representation strategy contributes to the Coverage requirement (req. 1) since all relevant concepts for SecEPM will be directly represented in SecEML. Modularization addresses the Independence of Notations and Runtime Capabilities requirements (reqs. 4 and 5) as specializations can be separated from generic aspects of the models, e.g., to model the capabilities of a BPMS. The Reuse strategy accounts for the Integration in Tool Chains and Coexistence with Existing Models requirements (reqs. 6 and 7) as existing languages and their tooling are reused. The application of Model-driven Techniques addresses the Technical Assistance and the Integration in Tool Chains requirements (reqs. 3 and 6) since it provides means to effectively provide tooling to work with SecEML. The relation between the Coexistence with Existing Models requirement (req. 7) and the use of entity references has been discussed already.

This section presented our major design considerations for SecEML: the application of the Direct Representation, the Modularization, the Reuse, and the Model-driven Techniques strategies as well as the selection of entity references instead of weaving languages for the integration of SecEML models with models using other notations. The following section describes SecEML and demonstrates the application of these strategies.

5.4. Description

SecEML is a textual DSML to capture core work products of SecEPM. It is specified using a combination of languages: Ecore, OCL, Xtext, and natural language. Ecore is an implementation of the Essential Meta Object Facility (EMOF) [Obj14a] standard in the context of the EMF[2] that is commonly used for the specification of metamodels. In the case of SecEML, Ecore is used for the specification of the basic metamodel. Rules for the well-formedness of SecEML constructs that could not be expressed using Ecore directly have been specified using OCL and integrated as constraints in an extended Ecore metamodel. For the specification of the concrete syntax and the mapping of abstract onto concrete syntax Xtext[3] has been used, a framework for the development of DSMLs as well as a language defined for this purpose resembling the Extended Backus-Naur Form (EBNF). Semantics of well-formed SecEML constructs is provided in natural language (applying the definitions provided in section 2.4.1).

[2] http://www.eclipse.org/modeling/emf/
[3] http://www.eclipse.org/Xtext/

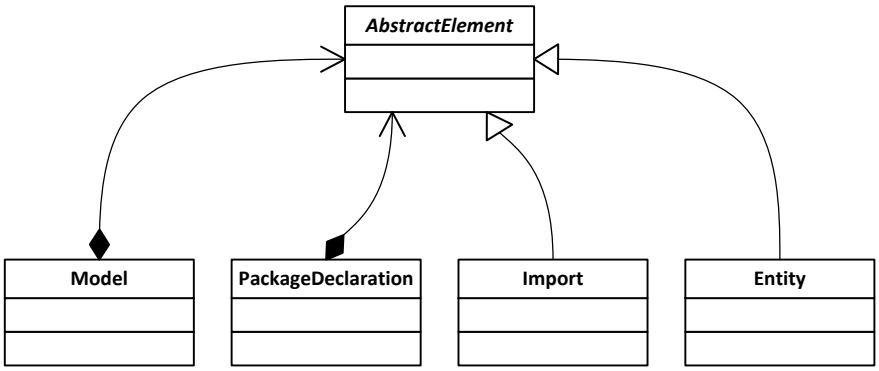

Figure 5.4.1.: Basic entities of SecEML (UML class diagram)

```
model replan
    language SecEML "1.0"

package Basic ()

package ThreatCatalog (
    import Basic.*
    // elements in package Basic might be referenced without FQN
)
```

Listing 5.1: Application of basic entities (SecEML notation)

The presentation of SecEML is based on a description of its metamodel. The description focuses on Ecore classes as main elements of the metamodel. In order to distinguish these Ecore classes from similar named constructs in SecEML, we will use the term entity (of SecEML) for them in the following. UML class diagrams showing parts of the metamodel are used in order to provide an overview of the entities and their relations. Examples applying the concrete syntax accompany the presentation.

5.4.1. Structure

SecEML artifacts are organized as models that contain all definitions. Figure 5.4.1 presents basic entities to organize SecEML models. The `Model` entity serves as container for the elements of a model. Every model has a unique name and provides optionally the number of the applied SecEML version to allow for transparent future improvements of SecEML.

Every element of a model that might be referenced by other elements has to provide a unique name as an identifier. As models might grow, SecEML supports hierarchical name spaces that are introduced defining packages applying the `PackageDeclaration` entity. The fully qualified name (FQN) of an elements consists of the concatenated identifiers (unique names) of the nesting packages and the identifier of the element using the "." as separator. The `Import` entity allows to import packages into other packages and to access elements of imported packages using their identifier instead of their FQN. Listing 5.1 demonstrates the application of the basic entities in SecEML.

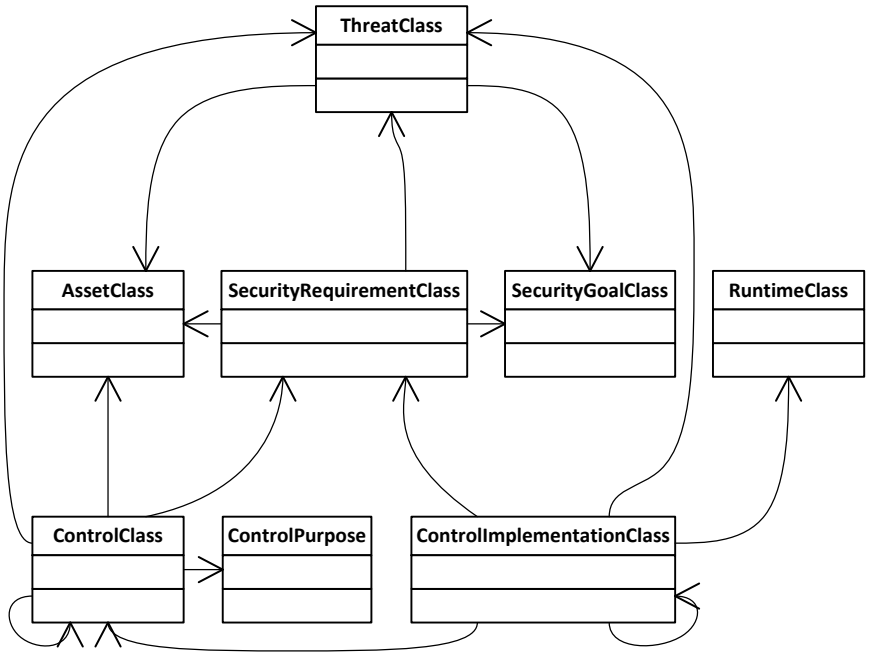

Figure 5.4.2.: Classification entities in SecEML (UML class diagram)

5.4.2. Classification

SecEPM strives to leverage the know-how of security experts and to support security non-professionals in executing many of its activities without extensive support. To align SecEML with this aim, general concepts and definitions that require security expertise are specified separately applying dedicated entities. Classification is used as a shared concept for those general concepts and definitions: Security experts, technical experst, and domain experts provide a comprehensive, interrelated set of classifications. These classifications are used (e.g., by a security nonprofessional) in order to document security needs and selected controls in the Security Analysis and Design Model. Therefore, the necessary skill set to specify these models is reduced. Furthermore, the classifications can be used to validate these models.

Figure 5.4.2 presents the main entities of SecEML and important relations in order to specify classifications. Generally, relations between entities in the metamodel are unidirectional and not mandatory (multiplicity [0..1] or [0..*]). This allows for the serialization of incomplete models as well as simpler dependency structures in order to support modularization and reuse (cf. section 5.3). A brief example demonstrating the application of the entities is depicted in listing 5.2 on the facing page. The example omits references and expressions with regard to elements external to the respective security model, those will be introduced in section 5.4.5.

The entity `AssetClass` allows for the distinction between different classes of assets. Basically, SecEPM distinguishes business assets and modeled assets, but the specification of further asset classes decomposing these distinction further—e.g., modeled assets into data at rest, data in transit, and process—might be applied as well. Asset classes might specify a

```
assetClass ASC_Business ()
assetClass ASC_Process ()
assetClass ASC_Data ()

securityGoalClass SGC_Integrity (
    description "The property of safeguarding the accuracy and
        completeness of assets"
)

threatClass THC_TamperingExecutionSequence (
    securityGoalClasses SGC_Integrity
    assetClasses ASC_Process
)

securityRequirementClass SRC_CheckExecutionSequence (
    securityGoalClasses SGC_Integrity
    assetClasses ASC_Process
    threatClasses THC_TamperingExecutionSequence
)

controlPurpose COP_Detection ()

controlClass COC_LogProcessExecution (
    purposes COP_Detection
    securityRequirementClasses SRC_CheckExecutionSequence
    introduces ASC_Data
)

runtimeClass Activiti_57

controlImplementationClass CIC_LogExecution (
    runtimeClass Activiti_57
    controlClasses COC_LogProcessExecution
    properties "logLevel"
)
```

Listing 5.2: Providing classifications (SecEML notation)

name for entities (i.e., an Ecore class name) which assets instantiating this asset class might reference in external models.

Security goal classes are specified using the corresponding `SecurityGoalClass` entity. SecEPM addresses initially the security goal classes integrity, confidentiality, availability, and non-repudiation as they have been defined before (cf. section 2.4.1), but might also include additional or deviating definitions. In the example listing 5.2, a security goal class “Integrity” is defined in accordance with definition 10 applying the `SecurityGoalClass` entity.

The Threat Catalog contains threats classes that might apply to (the) electronic business process (-es) in the scope of the project or the organization. These are specified applying the `ThreatClass` entity. Threat classes reference one or more security goal class and asset class that they endanger. Furthermore, they might provide three expressions: First, to specify matching patterns for the business process model, second, expressions to identify assets that—without mitigation—might get affected by this threat, and third, an expression to evaluate impacted elements of the business process model. Exemplary, a threat class “Tampering Execution Sequence” endangers security goals of the class “Integrity” and modeled assets (or, if this granularity has been chosen, process assets).

Security requirements refine security goals and state the intent to counter one or more threats. The `SecurityRequirementClass` entity allows for the classification of security requirements in order to provide common building blocks. They reference the classes of security goals that get refined as well as classes of assets and threats addressed. With the help of security requirement classes it is possible to connect related security goal classes, asset classes, and threat classes and to state one requirement refining several security goals. Security requirement classes might specify a matching expression that identifies elements from the business process model that are constrained by the requirement. Security requirement classes might be aligned utilizing security pattern catalogs (e.g., [BH04]) or existing requirement catalogs (e.g., [Int11a] as we demonstrate in section 6.3) In the example listing 5.2 on the previous page, tampering might endanger the integrity of process assets that requires checks of the execution sequence.

The Control Catalog specifies mitigation possibilities for threat classes specified in the Threat Catalog. The entity `ControlClass` references security requirement classes a control class might (partially) meet and threat classes it mitigates. Additionally, control classes specify their purpose referencing elements conforming to the `ControlPurpose` entity. Following the respective standard, these purposes might encompass prevention, deterrence, limitation, detection, correction, recovery, monitoring, and awareness [Int04]. In order to capture dependencies between controls, control classes might reference other control classes that they depend upon. Controls might also introduce new assets to a system. Therefore, a control class might reference the respective asset classes. Furthermore, control classes might specify a matching expression in order to identify elements from the business process model that are addressed by controls of this class.

In order to support the transformation of SecEML models into test and deployment artifacts, the `RuntimeClass` and the `ControlImplementationClass` entity provide means to specify the capabilities of the execution environment. A runtime class supplies merely a name and description of a runtime component. Control implementation classes reference the control classes they (partially) implement and optionally the subset of security requirement classes, threat classes, and purposes specified by the control classes they support. Similar to control classes, control implementation classes might reference necessary control implementation classes. Additionally, control implementation classes specify a list of properties necessary for the transformation of the actual control implementations into test and deployment artifacts.

5.4.3. Rating

In order to facilitate a repeatable rating of security goals (and control implementations), SecEML provides several entities to define a domain and organization specific context for this rating. Basically, SecEML allows for the specification of criteria that differentiate together a number of ratings and their structuring using damage scenarios. Furthermore, several aggregation strategies are supported in order to address dependencies between security goals in the course of the rating. Figure 5.4.3 on the facing page depicts core entities and

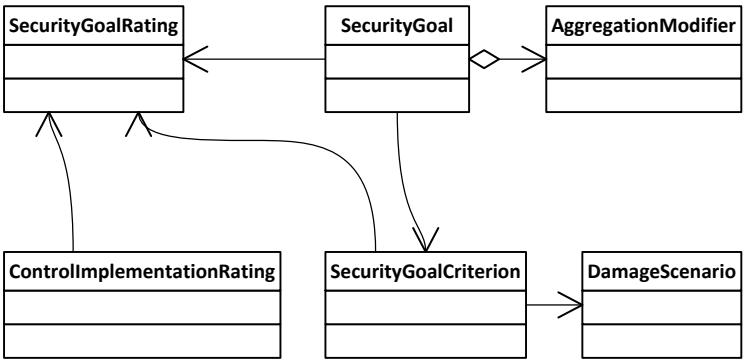

Figure 5.4.3.: Prioritization entities in SecEML (UML class diagram)

```
securityGoalRating Normal ( ordinal 1 )
securityGoalRating High ( ordinal 2 )
securityGoalRating VeryHigh ( ordinal 3 )

scenario ViolationOfLaws (
    title "Violations of laws, regulations, or contracts"
)

criterion VoL_MinorViolation (
    title "Violation with minor consequences"
    description "Violations of regulations and laws with minor
       consequences"
    rating Normal
    scenarios ViolationOfLaws
)
```

Listing 5.3: Providing ratings and corresponding criteria (SecEML notation)

their relation for the assessment and rating of security goals and control implementations. Listing 5.3 demonstrates their application.

The entity `SecurityGoalRating` allows for the definition of an ordinal scale for the rating of security goals. Every rating specifies an unique identifier and an ordinal number in order to reference, compare, and sort ratings. Example listing 5.3 defines three ratings from “Normal” to “Very High”.

Criteria to provide guidance for the consistent application of the ratings are defined with the help of the `SecurityGoalCritierion` entity. A criterion provides a description in natural language that details circumstances suggesting the application of the referenced security goal rating. The Security Analyst and Domain Expert have to ensure that criteria for different ratings are clear without ambiguity.

To organize and structure rating criteria, damage scenarios might be defined. A damage scenario describes a situation that potentially decreases the value of business assets and therefore provides a context for the application of rating criteria. If damage scenarios are defined (applying the `DamageScenario` entity), every scenario should provide rating criteria that address all ratings that have been defined.

The entity `AggregationModifier` allows for the differentiation of aggregation strategies for dependent security goals. A dependent security goal a is a security goal for supportive assets, i.e., assets that are supporting other assets (cf. section 4.5.2.5). At the moment, all strategies proposed by the IT-BPM are implemented in SecEML [Bun08]: First, consideration of dependencies might be disregarded at all. Second—if dependencies are considered—one of maximum, cumulation, and distribution strategies might be selected with maximum strategy as default. The maximum strategy considers ratings from all security goals that the security goal depends upon and sets the maximum rating for the security goal. The cumulation strategy sets the rating resulting from the maximum strategy but increased to the next higher level (if possible) for the security goal. The cumulation strategy is applied if the asset of the security goal in question supports comparatively many assets and therefore a violation of the security goal damages potentially multiple assets. Similarly, the distribution strategy sets the rating resulting from the maximum strategy but decreased to the next lower level (if possible) for the security goal. It is applied if the violation of the security goal does not (immediately) damage assets supported by the asset of the security goal.

5.4.4. Analysis and Design

While preceding entities of SecEML are mostly used by the Security Analyst and the Domain Expert, entities presented in this section address the Business Process Engineer. They provide means to document the Security Analysis and the Security Design Model. These models rely largely on the specifications provided in the Threat and Control Catalog as well as in the Runtime Capability Model utilizing those entities. Figure 5.4.4 on the facing page shows core entities for analysis and design as well as several important relations. Listing 5.4 on page 114 demonstrates the application of those entities. This demonstration does not include means to relate the security models with a business process model. Those are introduced in section 5.4.5.

The `Asset` entity is used to model assets. An asset instantiates an asset class, provides descriptions in natural language and references supported assets. The resulting dependency graph of assets must be acyclic. Additionally, an asset might reference elements external to the security model that represent this asset in the business process model. Exemplary, three assets are defined in listing 5.4 on page 114: a business asset for the POD, a process asset for the Replan Process of the logistics provider supporting the business asset, and an asset for the logging data introduced and generated by a control (cf. section 4.5.2).

Security goals are specified applying the `SecurityGoal` entity. A security goal instantiates a security goal class for one asset. It provides a rating, relevant criteria that substantiate the rating, and optionally a specification of the aggregation strategy for this security goal. In listing 5.4 only one integrity goal for the process asset is specified that is rated as “Normal” because violations of the security goal cause only minor violations of relevant contracts.

Actual threats are modeled utilizing the `Threat` entity. Threats are classified instantiating threat classes and reference security goals that are affected by the threat. Threats might provide either a set of elements from the business process model that are affected by the threat or an expression that yields a set of such elements. The resulting set of external elements must

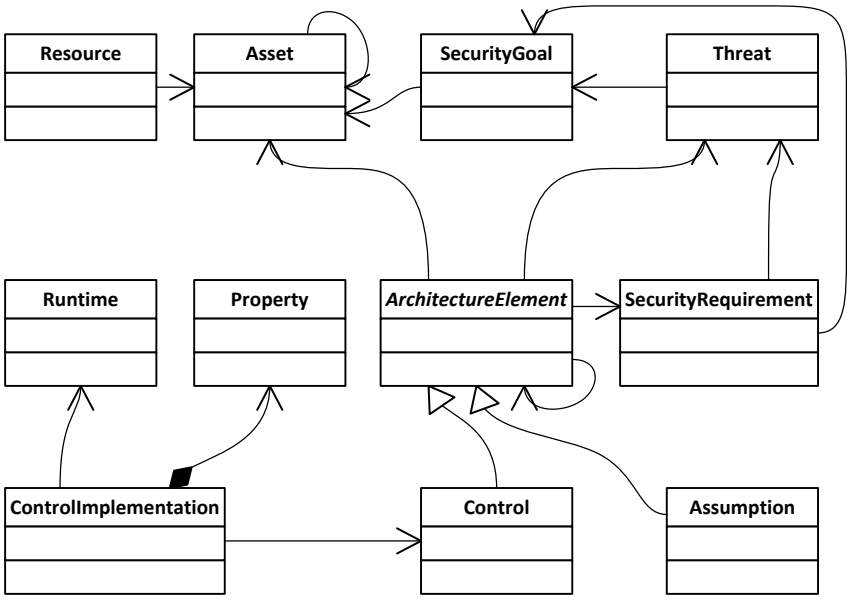

Figure 5.4.4.: Core analysis and design entities in SecEML (UML class diagram)

be a subset of the elements retrieved by the matching pattern of the threat class. Listing 5.4 instantiates the threat of tampering with the execution sequence of the Replan Process endangering the integrity goal for the corresponding asset.

A security requirement is specified applying the `SecurityRequirement` entity. A security requirement instantiates a security requirement class and references the security goals it refines as well as the threats it addresses. The security goals and the threats must match the respective classes specified in the security requirement class. Listing 5.4 specifies one security requirement for the logging of the Replan Process refining the integrity security goal for the Replan Process and addressing the tampering threat.

The `Control` entity allows for the specification of controls for the electronic business process. Controls instantiate control classes and reference security requirements that are (partially) met as well as threats that are mitigated by a control. Furthermore, assets that are introduced by a control are referenced as well as other controls that this control depends upon. The resulting dependency graph of the controls must be acyclic. All referenced elements must match the respective classes specified in the control class. Similar to threats, controls might provide either a set of elements from the business process model that are addressed by the control or an expression that yields a set of such elements. The resulting set of external elements must be a subset of the elements retrieved by the matching pattern of the control class. Exemplary, a control for the logging of the execution sequence of the Replan Process instantiates the logging control class, references the respective requirement, and introduces log data as new data asset.

Control implementations are specified using the `ControlImplementation` entity. They reference one or more control implementation classes they instantiate, the runtime environment they are implemented in, controls that they implement, and a set of properties provided

```
asset ASS_POD : ASC_Business ()
asset ASS_Replan : ASC_Process (
    supports ASS_POD
)
asset ASS_LogData : ASC_Data (
    // introduced by control, property "supports" not set
)

goal SGO_IntReplan : SGC_Integrity (
    asset ASS_Replan
    rating SGR_Normal
    criteria RCR_VoL_MinorViolation
)

threat THR_TamperReplan : THC_TamperingExecutionSequence (
    goals SGO_IntReplan
)

securityRequirement SRE_LogReplan : SRC_CheckExecutionSequence (
    goals SGO_IntReplan
    threats THR_TamperReplan
)

control CON_LogReplan : COC_LogProcessExecution (
    securityRequirements SRE_LogReplan
    introduces ASS_LogData
)

runtime RUN_Activiti : RUC_Activiti_57

controlImplementation CIM_LogExecution : CIC_LogExecution (
    runtime RUN_Activiti
    controls CON_LogReplan
    properties ( "loglevel" = "FINE" )
)
```

Listing 5.4: Specifying security requirements and controls (SecEML notation)

as name-value-pairs. The classes of all referenced elements must be elements of the superset of classes specified in the referenced control implementation classes. Listing 5.4 specifies a logger for the Activiti runtime implementing the logging control.

5.4.5. Relating SecEML and Business Process Models

The specification, documentation, and analysis of relations between the Security Analysis and Design Model and the Business Process Model is an important aspect of SecEPM. Several strategies for the integration of security aspects in model-driven engineering approaches have been discussed in section 2.3.3.3 and corresponding requirements and design decisions have been documented in sections 5.2 and 5.3. In order to apply the entity reference strategy to integrate SecEML and other existing models, several SecEML entities provide attributes that either store a reference to elements from the Business Process Model or an OCL expression that is evaluated on elements of the SecEML as well as the Business Process Model.

This section introduces the specification of relations between SecEML and Business Process Model elements and presents two examples. Listing 5.5 on the next page presents an excerpt of the Replan Process that has been modeled in an adapted textual syntax called MockBPMN (MBPMN) resembling BPMN without XML syntax elements for better comprehensibility (a representation using the BPMN exchange format is provided in the appendix, cf. listing B.1 on page 161). Only the collaboration between forwarder and logistics provider and the first activity of the process from the logistics provider is specified omitting further details. Listing 5.6 on the next page shows extended SecEML elements detailing references on elements of the Replan Process and related OCL expressions. Implementation aspects for the management of relations between SecEML and elements of the Business Process Model are covered in section 5.5.

Multiple relations between elements of SecEML models and those of the Business Process Model exist. An obvious example is the asset concept: SecEPM identifies and documents modeled assets, i.e., assets that are directly represented in the Business Process Model. SecEML explicates this relation capturing a reference to the respective element from the Business Process Model as an attribute of the `Asset` entity. Generally, the ID of the element of the Business Process Model is taken as reference but other approaches for the identification of model elements supported by EMF are valid as well. The type of the elements that might be referenced by an asset are specified by its asset class using the `type` attribute of the `AssetClass` entity. Other examples for direct element references supported by SecEML are the `entities` attribute of the `Threat, SecurityRequirement` and `Control` entity.

The definition of explicit references between elements of SecEML and those of the Business Process Model can be cumbersome if many elements have to be referenced or is even not feasible if not individual instances but more general patterns have to be specified. Therefore, SecEML supports also the definition of expressions using OCL. Exemplary, it might ease the identification of candidate threats if a threat class provides applicability conditions not only in natural language but also as evaluable expression supporting adequate tooling. In order to do so, the `ThreatClass` entity provides the possibility to specify a matching expression and an asset expression using the `matchExp` and the `assetExp` attribute. The matching expression returns all elements from the Business Process Model that will be affected by an

```
collaboration (
    participants
        participant ( P1_Logistics_Provider ),
        participant ( P2_Forwarder )
    messageFlows MF1, MF2
)

process P1_Logistics_Provider (
    laneSets laneSet (
        lanes L12_LogisticsSystem, L11_Dispatcher
    )
    flowElements  E11, T11_CheckStatus, S11
)

event E11 ()
task T11_CheckStatus ()
sequenceFlow S11 ( source E11, target T11_CheckStatus )
```

Listing 5.5: Excerpt of the Replan Process (MBPMN notation)

```
assetClass ASC_Process (
    type "mbpmn::Process"
)

asset Ass3_Replan : ASC_Process (
    supports ASS_POD
    entity P1_Logistics_Provider
)

threatClass THC_TamperingExecutionSequence (
    securityGoalClasses SGC_Integrity
    assetClasses ASC_Process
    matchExp "assets->collect(flowElements.oclIsTypeOf(mbpmn::
        SequenceFlow))"
    assetExp "mbpmn::Process.allInstances()->select(flowElements->
        exists(fe | fe.oclIsTypeOf(mbpmn::SequenceFlow)))"
)
```

Listing 5.6: Relations between security and business process models (SecEML notation)

successful attack of that threat class. The asset expression aligns the threat class with asset classes returning all elements of the types specified by the respective asset classes that will be impacted if elements returned by the matching expression are manipulated. Both attributes take an OCL expression that is evaluated in the context of the respective element it has been defined. Similar expressions are supported for the `SecurityRequirementClass` and the `ControlClass` entities.

Listing 5.6 demonstrates this approach for the threat class `THC_TamperingExecution-Sequence`. It encompasses threats tampering with the execution sequence of the electronic business process. This sequence is specified using the `SequenceFlow` entity in BPMN. The asset class for this threat is `ASC_Process` that specifies the BPMN `Process` entity as related concept for modeled assets. Therefore, the asset expression specified in `assetExp` returns all elements of the Business Process Model of the type `Process` that reference `SequenceFlow` elements (the element `P1_Logistics_Provider` in listing 5.5). In turn, the matching expression specified in `matchExp` returns all `SequenceFlow` elements from the result set of the asset expression stored in the derived attribute `assets` (the element `S11` in the listing).

It should be noted that SecEML does not prescribe a specific approach to formulate these expressions nor does it prescribe the specification of specific threat classes etc. It merely provides means to support participants applying the SecEPM, e.g., the activity Model Threats and especially the tasks Identify Candidate Threats and Select Relevant Threats. Using flexible OCL expressions to explicate relations between models addresses the requirements Technical Assistance, Independence of Notation, Integration in Tool Chains, and Coexistence with Existing Models. It applies the Direct Representation design strategy as related elements are explicitly modeled as well as the Reuse and Model-driven Techniques strategies as it reuses existing notations and respective tooling that are commonly applied in the context of the model-driven tool chains.

5.4.6. Concrete Syntax

Examples using the concrete syntax for SecEML have been introduced in the preceding sections. This section briefly covers the concrete syntax in general and rationale for some design decisions.

A major design decision is concerned with the choice of a graphical or textual concrete syntax. Most business process modeling notations provide a graphical syntax (although almost every notation provides a textual one as well). Therefore, it might have been a natural choice to develop a graphical syntax for SecEML. Actually, in the course of the development of SecEML several approaches for a textual and graphical concrete syntax have been made adopting design rules and principles provided by multiple authors [Pet95, MC96, SR01, KKP+09]. In the end, textual representation has been favored over graphical representation for a number of reasons (cf. also [GKR+07, EFL12]):

- Tool independence and tool-chain integration: Text does not require specific platforms, environments, and tooling for reading, modifying, searching, and comparing. Furthermore, existing tooling can be used for textual models as they are omnipresent in the development chain. Especially, most systems for version control are text-based and have strong limitations with regard to the calculation of differences and merging of conflicting versions for

other representations. This rationale addresses the requirement for the Integration in Tool Chains and the Reuse design strategy.

- Formatting speed and quality: The creation of clearly laid-out graphical models is a time-consuming process that is only poorly supported by layout algorithms. On the other hand, text formatting is much easier and standard algorithms render good results. The requirements Accessible Concrete Syntax and Technical Assistance as well as the Reuse design strategy are concerns for this rationale.

- Language integration and composition: Languages might integrate other languages in order to reuse existing solutions and tooling (e.g., OCL in the case of SecEML). Seamless integration of two different graphical representations or mixtures of textual and graphical languages are often much harder to achieve than the integration of two textual languages. Furthermore, tooling that supports mixed language models with textual syntax are easier to develop, especially if they use a common development framework like EMF. This rationale (also) addresses the requirements Accessible Concrete Syntax and Technical Assistance as well as the Reuse and the Model-driven Techniques design strategy.

- Short development cycles: Although we used MDE frameworks for the development of modeling tools for both graphical and textual representations, only in the case of textual representations a short development cycle between changes in the metamodel, the concrete syntax, their mapping, and the provision of adapted tooling could be maintained. This might be (partly) caused by the frameworks themselves or their inappropriate application; as short development cycles allow for fast adaption of change this rationale accounted as well for the decision in favor of a textual concrete syntax for SecEML.

The textual syntax itself is based upon the Human-usable Textual Notation (HUTN) [Obj04] in order to provide a common appearance and a smooth learning curve. We reduced the number of syntax elements (e.g., the colon to separate attribute name and value and the quotation marks for identifiers) and simplified some tokens for international application (e.g., replacing curly brackets with standard ones). The definition of the grammar is provided in EBNF in appendix A.

5.5. Implementation

The preceding sections detailed the purpose, requirements, and major design decisions with regard to SecEML as well as an description of the language itself. This section provides a short presentation of the implementation of tooling to support the application of SecEML in the security engineering process. An alternative version of the tools presented in this section has been developed recently [Rup13]. An implementation of automated transformations of SecEML models into implementation artifacts has been demonstrated for three different BPMS elsewhere and will not be covered in this section [Ham13].

Two requirements for SecEML deal with tool support for SecEML: Technical Assistance and Integration in Tool Chains (cf. requirements 3 and 6). In order to meet the Technical Assistance requirement and to demonstrate the feasibility of our approach we developed

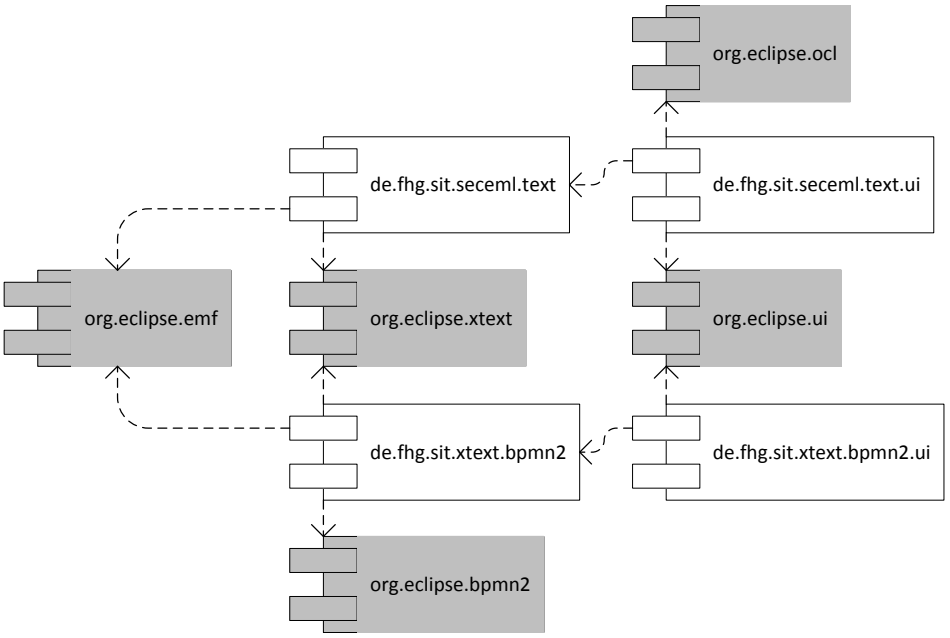

Figure 5.5.1.: SecEML editor plug-ins overview (UML component diagram)

an editor for SecEML that supports creation and validation of SecEML models. Analysis of SecEML is supported by existing tooling that we cover as well in this section. Similarly, we meet the requirement Integration in Tool Chains with the help of existing tooling addressing requirements management, project management, change management and other disciplines in the development life cycle.

For the development of our SecEML editor and the integration of related tooling we chose the Eclipse platform as foundation. The Eclipse platform has strong support in the domain of BPM as well in the industrial as in the academic world. Many BPMS vendors take the Eclipse platform as strategic choice for their front-end applications (e.g., major vendors like IBM, Software AG, Red Hat but also open source solutions like Activiti, Bonita, and others). Likewise, the Eclipse platform hosts with EMF a very well accepted framework in the domain of model-driven engineering that we used for the development of our SecEML editor. To facilitate the development of the editor we applied Xtext, a framework for the development of DSMLs on top of EMF. These choices also account for the Model-driven Techniques design strategy that we discussed in section 5.3.

The architecture of our SecEML editor is largely determined by the choice of the Eclipse platform and the MDE frameworks. Figure 5.5.1 depicts a simplified overview of the Eclipse plug-ins of our tooling to support the application of SecEML. Plug-ins with gray background are provided by Eclipse (and represent merely plug-in-families):

- de.fhg.sit.seceml.text: This plug-in contains the Ecore metamodel of SecEML, a Java implementation of the abstract syntax tree (AST), and a parser based on ANTLR.

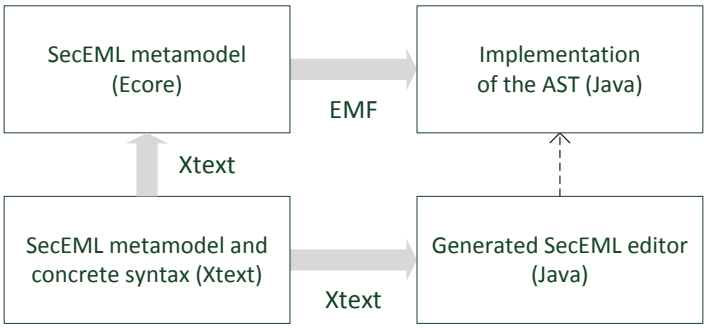

Figure 5.5.2.: Initial transformation process for the SecEML editor

- de.fhg.sit.seceml.text.ui: The SecEML editor supporting advanced IDE features like syntax highlighting, content assist, and model validation is implemented with this plug-in.
- de.fhg.sit.xtext.bpmn2 and de.fhg.sit.xtext.bpmn2.ui: Adapter plug-ins to allow for seamless access of BPMN business process models from the SecEML editor (or other, Xtext-based editors).

Automation is one of the main aims applying model-driven techniques. In the case of SecEML, automation gained by the application of MDE allowed for short feedback cycles within the development of the language and its tooling. For the implementation of the SecEML editor, technically we started with an integrated description of metamodel, concrete syntax, and their mapping using the Xtext notation (collapsing the first two steps of the language engineering process from Kleppe, cf. section 5.3). Figure 5.5.2 depicts the transformation process for the resulting SecEML editor in this phase. Applying the Xtext framework, the Ecore representation of the SecEML metamodel, an ANTLR-based parser, and the SecEML editor are generated (generation is depicted using bold arrows, the responsible framework is annotated). The embedded EMF framework generates a Java implementation of the AST that is than utilized by the editor (dependencies are depicted using dashed arrows).

After several iterations optimizing metamodel and concrete syntax using example models and feedback from users of the SecEML editor, we adapted the transformation process in order to streamline the metamodel, to add constraints and derived attributes to the metamodel, and to support further validations of SecEML models (cf. figure 5.5.3). The (initially generated) Ecore representation of the SecEML metamodel is now manually optimized and augmented with annotations specifying constraints and derived attributes using OCL. Similar to the previous transformation process, EMF is used to generate the Java implementation of the AST (including the augmentations). With the help of the Xtext framework parser and editor implementations are generated. Additional editor features are implemented enhancing the generated editor using Java and OCL. Although two manual augmentations are applied within the adapted transformation process (one to the metamodel and one to the generated editor), still changes to the metamodel as well as to the concrete syntax (mapping) could be applied and the transformations could be executed without touching manually provided parts.

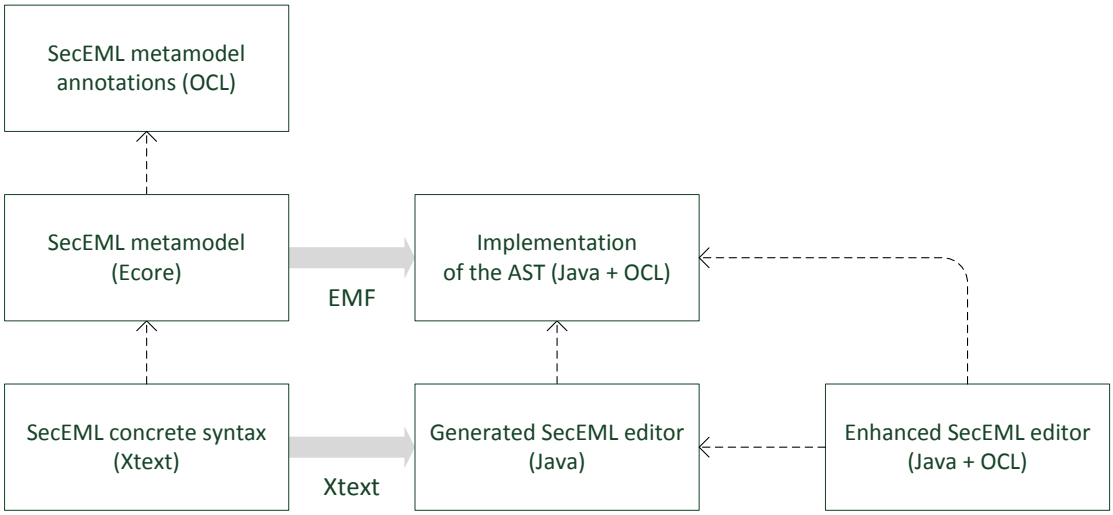

Figure 5.5.3.: Adapted transformation process for the SecEML editor

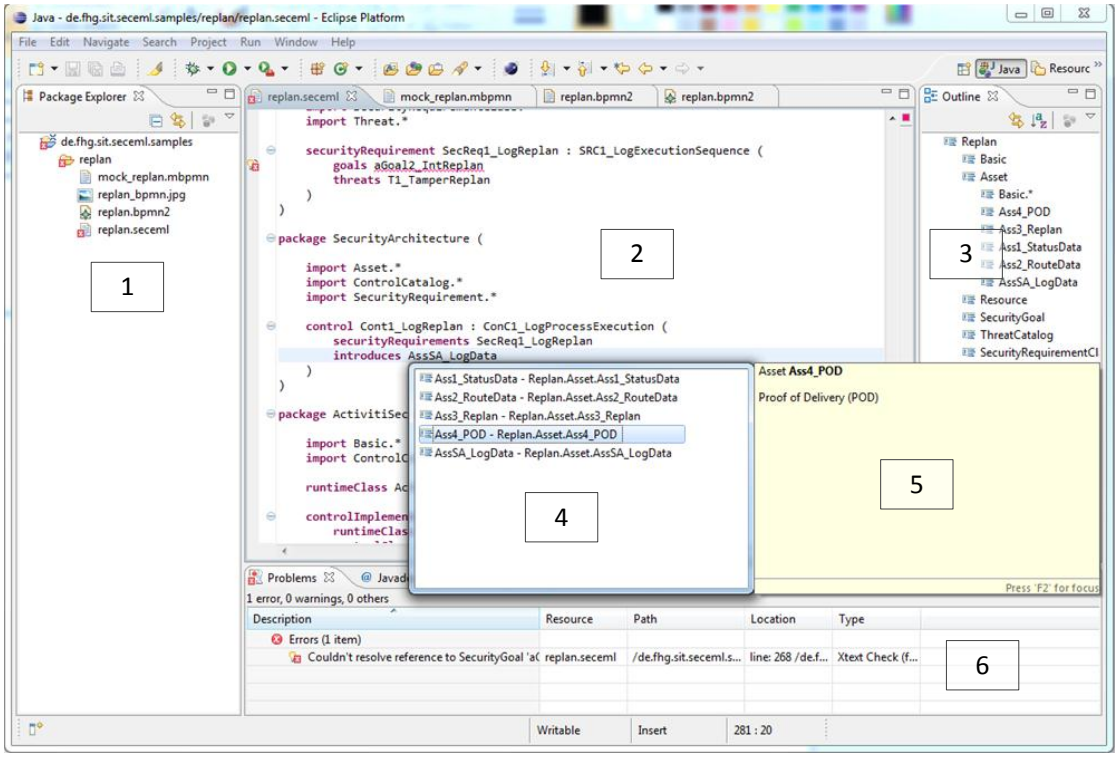

Figure 5.5.4.: Using the SecEML editor

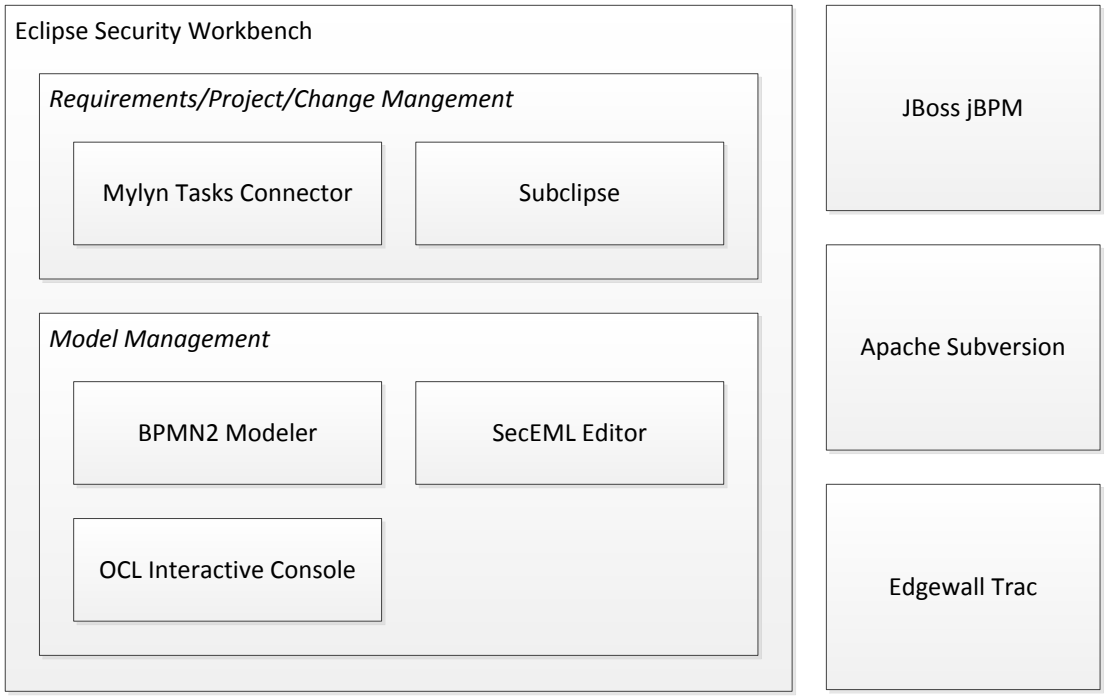

Figure 5.5.5.: Elements of the integrated security workbench and external components

Figure 5.5.4 displays a screen shot of the enhanced SecEML editor. The package explorer (1) shows the resources of the project. The editor (2) allows for the creation, manipulation, and validation of SecEML models. An outline (3) provides simplified access to the elements of the SecEML model representing elements in a tree-like structure. Quick fixes (4) are provided by the editor offering changes to the SecEML model in order to remove errors from the model. In-model documentation of model elements (5) are shown to the user in order to support editing and validation. Validation warnings and errors are displayed in the problems view (6) allowing for an easy overview and fast navigation.

As the SecEML editor uses the Eclipse platform as foundation, applies well accepted frameworks for the implementation, and is using a textual concrete syntax, it is easy to integrate SecEML in existing tool chains, especially if they already apply Eclipse-based tooling. In order to demonstrate the feasibility of the integration we provide an integrated workbench covering BPMN as well as SecEML model management (creation, update, validation, analysis), requirements and project management (feature/user story/task creation, access, update), and change management (issue creation, access, update as well as artifact versioning) utilizing the SecEML editor and existing plugins for the Eclipse platform. Figure 5.5.5 presents the components used for the security workbench. For model management, the BPMN2 Modeler[4] has been chosen that integrates smoothly with the SecEML editor allowing for navigation from SecEML to referenced elements of the BPMN models and the joint analysis evaluating OCL expressions

[4] `http://eclipse.org/bpmn2-modeler/`

to the model instances utilizing the OCL Interactive Console[5]. For requirements, project, and change management the Mylyn Tasks Connector[6] and the Subclipse plugin[7] provide access to the project management as well as requirements and issue tracking system Edgewall Trac[8] and the version control system Apache Subversion[9]. The BPMS JBoss jBPM[10] integrates well with the BPMN2 Modeler. Generally, the participants of a development project work with the same tool set. Nevertheless, depending on the respective role, components are used more intensively (e.g., the OCL Interactive Console is used for analysis and therefore more relevant to the Security Analyst than the Process Model Engineer). Additional components might be needed that are provided by the BPMS in question. It should be highlighted, that the choice of the components is not fixed and will be determined by the actual environment of an project for the development of secure electronic business processes.

5.6. Summary

We presented SecEML in this chapter. It is our proposal for a DSML to specify work products of SecEPM. It provides a basis for the validation of work products as well as for their transformation into deployment artifacts. We pinpointed the two faced relation of SecEML with the ideas of MDE: First, SecEML has its role as a DSML to minimize the semantic gap. Second, tooling supporting the application of SecEML is developed applying model-driven techniques and frameworks.

We confined the scope of SecEML on capturing the Security Analysis Model and the Security Design Model as core artifacts of SecEPM in order to support model-based security engineering of electronic business processes fostering a shared understanding and to allow for an iterative and incremental execution of SecEPM activities and tasks. Formal analysis of certain properties of electronic business processes as well as the validation of runtime policies have been considered outside the scope of SecEML.

Seven key requirements have been identified based on this scoping (cf. section 5.2): coverage of all relevant concepts and relations (1), provision of an accessible concrete syntax (2), technical assistance for the creation, validation, and analysis of SecEML models (3), independence with regard to notations used for the specification of electronic business models or specific runtime capabilities of BPMSs (4, 5), integration in existing tool chains for the development of electronic business processes (6), and coexistence with existing models (7). Criteria for the evaluation of security modeling approaches from literature have been used to validate these key requirements.

In order to develop SecEML systematically, we applied the language engineering approach from Kleppe that we sketched briefly in section 5.3. This section also covers four main design strategies we applied to meet the key requirements: the Direct Representation strategy following one of the main ideas of MDE (1), the Modularization strategy in order to provide sparsely

[5] http://www.eclipse.org/modeling/mdt/?project=ocl
[6] http://www.eclipse.org/mylyn/
[7] http://subclipse.tigris.org/
[8] http://trac.edgewall.org/
[9] http://subversion.apache.org/
[10] http://www.jboss.org/jbpm

dependent compartments comprising SecEML (2), the Reuse strategy aiming at integration of existing languages as well as modeled elements instead of redundant specifications and implementations (3), and the application of Model-driven Techniques for the specification of SecEML and the development of appropriate tooling (4).

We presented SecEML highlighting selected parts of its metamodel and corresponding listings demonstrating the application of the selected entities and relations (cf. section 5.4). We covered the structure of SecEML models, the specification of classifications, definition and application of ratings, modeling of analysis and design elements, and relations between SecEML models and the Business Process Model. We completed the presentation with rationale for the textual concrete syntax specified in appendix A.

The implementation of our tooling for the application of SecEML was presented in section 5.5. We introduced the transformation process for the initial implementation of our SecEML editor based on EMF and Xtext as well as its adaption in order to add further (OCL) constraints and derived attributes to the metamodel and features to the SecEML editor. The feasibility of the integration of the SecEML editor in existing tool chains was demonstrated integrating several existing Eclipse plugins for requirements, project, change, and model management that all work well in conjunction with SecEML resources and the SecEML editor.

6. Exemplary Study

6.1. Introduction

The preceding chapters introduced the constituent parts of our framework for model-based security engineering of electronic business processes: the process model SecEPM, our DSML SecEML, and tooling for the application of SecEPM as well as SecEML. This chapter provides a joint application of our framework presenting an exemplary study. It addresses the last research question and documents the demonstration and evaluation activities of our research approach (cf. section 1.3):

3. What observations do we get applying the framework developed in this thesis?

In order to systematize our observations we introduce a set of analysis criteria derived from our requirements for SecEPM and SecEML respectively (section 6.2). The Replan Process introduced in chapter 3 provides the background for our exemplary study (section 6.3). Further experiences gained from applications of (parts of) the framework are reported in section 6.4. Section 6.5 applies the analysis criteria and compares the framework developed in this thesis with existing approaches. Section 6.6 reviews application and comparison and discusses our findings. A summary of the main results in section 6.7 concludes this chapter.

6.2. Analysis Criteria

We derive our criteria for the analysis of the exemplary study and the comparison of the framework developed in this thesis with existing approaches from requirements specified for SecEPM and SecEML respectively.

We elicited these requirements considering key requirements that have been explicated for matching security engineering process models as well as requirements addressing the major issues detailed in the problem statement. Therefore, they are well aligned with the objective of this thesis. Furthermore, they provide a foundation to differentiate the framework developed in this thesis from existing approaches and the progress achieved with our contributions.

The derivation of the analysis criteria is straightforward for most of the requirements for SecEPM (cf. section 4.2). We denominate resulting criteria with the identifier of the corresponding requirement to foster clarity. The following paragraphs comment only possible ambiguities or decisions taken in the course of the derivation. An overview of the resulting criteria is provided in table 6.1.

In order to distinguish the Coverage criterion stemming from SecEPM requirements and the Coverage criterion stemming from SecEML requirements we name them Activity Coverage

Requirement	Criterion
Structure	Activities, roles, work products, and their relations are documented
(Activity) Coverage	Activities are provided for the assessment of security goals, elicitation of security requirements, threat modeling and prioritization, and control design as well as configuration
Separation of Problem and Solution Domain	Activities and work products are separated with regard to the problem and solution domain
Traceability	Tracing of dependencies between security goals, threats, requirements, and controls is supported
Integrability	Integration into development process models is considered and demonstrated
Independence from Development-time Technology	No specific technology or tooling is required to apply the process model
Independence from Runtime Technology	Different business process engines and runtime environments can be supported
Restricted Skill Sets	Participation of security nonprofessionals is considered and a respective role is responsible at least for one activity

Table 6.1.: Analysis criteria derived from SecEPM requirements

and Concept Coverage respectively. Activity Coverage defines the necessary scope of activities of the security engineering process model. The Traceability criterion focuses on dependencies and checks, whether traceability of dependencies of relevant entities is considered by the process model in question. In case of Integrability, the respective criterion is shaped to check explicit consideration and demonstration of the integration of the security engineering process model in question into at least one existing development process model. Similarly, the Restricted Skill Set criterion checks explicit consideration of security nonprofessionals as participants and the assignment of at minimum one activity of the security engineering process model to a role designated for security nonprofessionals.

Likewise, derivations from requirements for SecEML are mostly self-explanatory (cf. section 5.2). An overview of the resulting criteria is provided in table 6.2. The Concept Coverage criterion is straightened in comparison to the respective requirement in so far as it checks whether relevant concepts and relations are directly represented in the DSML. Exemplary, a security requirement modeled using the DSML in question needs to be directly identifiable as such and uses different syntax elements than a security control to fulfill this criterion. The Coexistence with Existing Models criterion checks two aspects of the respective DSML: First, whether a (digital) business process model does not need to be changed in order to apply the DSML. Generally, most languages applying the language extension or the dialect definition strategy will not fulfill this criterion as they require augmentation of existing models (cf. section 2.3.3.3). The second aspect is the possibility to link elements from the business process model and the security model, e.g., providing references.

Requirement	Criterion
(Concept) Coverage	Concepts and relations defined and required by the process model are directly represented in the DSML
Accessible Concrete Syntax	Familiar appearance of the concrete syntax also for security nonprofessionals
Technical Assistance	Tools to create, validate, and analyze models applying the DSML are provided
Independence of Notation	DSML can be applied regardless of the notation of the business process model
Independence of Runtime Capabilities	The DSML does not require specific runtime capabilities of BPMSs
Integration in Tool Chains	Standard tooling can be reused to apply the DSML in existing tool chains
Coexistence with Existing Models	Existing business process models can be used without changes and elements of business process models using common notations can be linked to elements of models using the DSML

Table 6.2.: Analysis criteria derived from SecEML requirements

	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Process	Spoof User	Tamper Execution Sequence	Repudiate User Interaction	Disclose Process Information	Denial of Process Execution	Elevation of Privileges
Message	Spoof Participant	Tamper Message	Repudiate Message Exchange	Disclose Message	Denial of Message Exchange	
Data		Tamper Data		Disclose Data		

Table 6.3.: Mapping of STRIDE threat classes onto BPMN entities for modeled assets

6.3. The Replan Process

The Replan Process has been introduced as running example in chapter 3. In order to protect the interests of the participants in the originating research project and to ease reproducibility, we removed confidential details from the original business process model and adapted the solution to run in an open source environment for this exemplary study. Technical details that do not contribute to the security of the Replan Process are omitted.

Proceeding and results are presented following the structure of SecEPM representing a logical order of activities, not necessarily the actual order. For keeping the focus on SecEPM, the results are presented independently from an actual development process. Notwithstanding, interesting deviations are noted. The presentation is accompanied by tabular overviews depicting important relations. These relations are taken from the respective work product extracts that are provided in appendix B.

Security Requirement Class	Asset Class	Security Goal Class	Threat Class	Common Criteria SFR
User Authentication	Process	Confidentiality, Integrity, Non-Repudiation	Spoof User, Repudiate User Interaction	FIA_UAU.2
Message Confidentiality	Message	Confidentiality	Disclose Message, Spoof Participant	FDP_UCT.1
Message Integrity	Message	Integrity	Tamper Message, Spoof Participant	DFP_UIT.1
Audit Generation	Process	Integrity, Non-Repudiation	Tamper Execution Sequence	FAU_GEN.1
Data Integrity	Data	Integrity, Non-Repudiation	Tamper Data	FAU_STG.1, FPT_ITI.1
Data Confidentiality	Data	Confidentiality	Disclose Data	FAU_ITC.1
User Access Control	Process	Confidentiality, Integrity, Non-Repudiation	Spoof User, Disclose Process Information, Elevation of Privileges, Repudiate User Interaction	FDP_ACC.1, FDP_ITC.1
Secret Quality	Process, Message	Confidentiality, Integrity, Non-Repudiation	Spoof User, Spoof Participant, Repudiate User Interaction, Repudiate Message Exchange	FIA_SOS.1
Minimum Quotas	Process, Message	Availability	Denial of Process Execution, Denial of Message Exchange	FRU_RSA.2

Table 6.4.: Mapping of asset classes, security goal classes, and threat classes onto security requirement classes and related CC SFRs

6.3.1. Setup Process

As SecEPM separates tasks that need substantial input from security professionals from those that do not need that input, the activity Setup Process plays an important role in SecEPM and consumes significant effort at its initial execution in a new environment.

For the Replan Process, basic definitions are taken from ISO 13335-1 and the IT-BPM [Int04, Bun08]. The security goal classes “Confidentiality”, “Integrity”, “Availability”, and “Non-repudiation” are defined corresponding to [Int04]. The IT-BPM has been chosen as method to rate security goals for the Replan Process. Therefore, the ratings “Normal”, “High”, and “Very High” are defined accordingly. Furthermore, six damage scenarios are specified covering “Violation of laws”, “Impairment of the right to informational self-determination”, “Physical injury”, “Impaired ability to perform tasks at hand”, “Negative internal or external effects”, and “Financial consequences”. In order to rate security goals systematically, criteria to assess security goals are defined for all combinations of damage scenarios and security goal ratings. To support the identification of modeled assets, three asset classes are defined encompassing

Control Class	Security Requirement Class	Purpose	Introduced Asset Class	Dependency
User Authentication	User Authentication	Prevention	Data	Secret Quality
User Access Control	User Access Control	Prevention	Data	User Authentication
Physical Access Control	Data Confidentiality, Data Integrity	Prevention		
Channel Protection	Message Confidentiality, Message Integrity	Detection, Prevention	Data	
Message Encryption	Message Confidentiality	Prevention	Data	Secret Quality
Message Authentication	Message Integrity	Detection	Data	
Audit Generation	Audit Generation	Detection, Monitoring	Data	
Data Encryption	Data Confidentiality	Prevention	Data	Secret Quality
Data Integrity Protection	Data Integrity	Detection	Data	
Secret Quality	Secret Quality	Deterrence		
Minimum Quotas	Minimum Quotas	Limitation		

Table 6.5.: Control classes and their security goal classes, purposes, introduced asset classes, and dependencies

"Process" assets (covering sequences of flow nodes, assigned to the BPMN entity Process), "Message" assets (covering data in transit, assigned to the BPMN entity Message), and "Data" assets (covering data at rest, assigned to the BPMN entity DataObject). As control purposes "Prevention", "Deterrence", "Limitation", "Detection", "Correction", "Recovery", and "Monitoring" are defined. Rating of control implementations is restricted to "Weak" and "Strong", restricting weak control implementations to security requirements that refine only security goals rated as "Normal".

For threat modeling, the STRIDE method is selected and adopted to electronic business processes. Therefore, the six threat classes from STRIDE are mapped onto the three modeled assets resulting in 13 threat classes. Table 6.3 on page 127 displays the mapping and the resulting threat classes. For every threat class, OCL expressions are provided to identify candidate assets in the Business Process Model. Therefore, evaluation of the expressions and comparison of the result set with the modeled assets of a Security Analysis Model provide a basis for the identification of missing considerations with regards to threats.

In order to support the refinement of security requirements from security goals and threats for a given Business Process Model, security requirement classes are defined that map asset classes, security goals classes, and threat classes. The grouping of so called SFRs from Common Criteria (cf. [Int11a]) served as a basis for this mapping and allows for systematic extension of the defined classes as well as an alignment of a Security Analysis Model based on these

Control Implementation Class	Control Class	Dependency	Rating
Log Execution	Audit Generation		Weak
Key Store	Data Encryption, Data Integrity Protection		Weak
Basic Http Authentication	User Authentication		Weak
Transport Layer Security	Channel Protection, User Authentication	Key Store	Strong
User Access Control	User Access Control	Basic Http Authentication	Weak

Table 6.6.: The runtime capabilities of Activiti

security requirement classes with analysis artifacts conforming to this standard. Table 6.4 depicts the resulting mapping.

Control classes for the Control Catalog have been defined in correspondence with the security requirement classes. Control classes provide structure for the solution space. Their definition anticipates actual control implementations from targeted BPMSs in a generalized manner but has to consider possibilities to model necessary assumptions as well. Table 6.5 on the previous page depicts the defined control classes, their security requirement classes, purposes, introduced assets, and dependencies with other control classes. The Control Catalog is likely to be changed within a development project and the respective task from the Setup Process activity will be executed several times.

Activiti, the selected BPMS, provides only few security mechanisms that have been captured in the Runtime Capability Model. The control class implementations are shown in table 6.6. Comparing tables 6.5 and 6.6, it can be seen, that capabilities of a BPMS do not necessarily meet the specification of their control classes (e.g., Activiti does not support the checking and enforcement of strong secrets and the control implementation class “Basic Http Authentication” is not backed by a respective control implementation class to ensure the quality of secrets). These mismatches should be identified and reported to the user by appropriate tooling. Exemplary, the SecEML editor reports a warning offering the insertion of a reference to a matching control implementation or the creation of a corresponding control implementation class as quick fix.

The last task of the Setup Process is the provision of guiding artifacts. The guidance on the rating of security goals adapting IT-BPM (cf. section 4.6.2) is selected. Additionally, references to the STRIDE method as well as for Activiti are provided as background material. The documentation of the tailored security engineering process is published using the EPF Composer (cf. section 4.7.1 and figure 4.7.4).

6.3.2. Identify Assets

Until now, only preparatory tasks have been executed tailoring the security engineering process. Most of these tasks need strong participation of the Security Analyst or are assigned to that role. The following activities and tasks are mainly assigned to the Business Process Engineer and interactions with the Security Analyst are minimized in order to leverage security nonprofessionals and reduce the skill set necessary.

Analysis of assets and requirements takes the viewpoint of the logistics provider as the logistics provider is the driving force for the specification and enactment of the Replan Process. The activity Identify Assets results with the POD as business asset that captures the purpose of the Replan Process and is not represented in the Business Process Model. Two modeled assets are identified comprising the core Replan Process of the logistics provider (`ASS_ReplanProcess`) and the messages exchanged with the forwarder (`ASS_Messages`). The forwarder process itself, the wide area network (WAN) transmitting the messages, the dispatcher, and the logistics system are modeled as resources. The following dependencies are captured for those assets and resources: The business asset POD is supported by the modeled asset Replan Process that is in turn supported by the message asset. The resources forwarder process, logistics system, and WAN support the message asset as participants or media for the message exchange. The resources logistics system and dispatcher support the asset Replan Process. Table 6.7 on the next page displays the assets in the respective column (including those assets that have been introduced by controls in following activities).

The SecEML editor supports this activity by, e.g., identifying candidate (or probably missed) modeled assets, linking elements from the Security Analysis Model with those from the Business Process Model, and validating dependencies between assets and resources.

6.3.3. Assess Security Goals

Executing the activity Assess Security Goals generates four security goals for each modeled asset (one for each security goal class) that are rated “Normal” with regard to the security goal classes confidentiality and availability and “High” with regard to integrity and non-repudiation: The Replan Process does not handle highly classified data (confidentiality) and supports the timely delivery of shipments but does not hinder them (availability). Nevertheless, the Replan Process potentially re-routes shipments which might induce delays and substantial penalties in case of manipulations of the process or messages (integrity and non-repudiation). As dependencies between security goals those between assets are considered as relevant in order to identify necessary adjustments for the security goal ratings using the default maximum principle. Hence, no manual adjustment is necessary. Table 6.7 on the following page displays the security goals in the respective column.

The SecEML editor supports this activity by, e.g., identifying (potentially) missing security goals and calculating ratings from criteria assigned and dependencies between the security goals.

6.3.4. Model Threats

In order to model threats for the security goals documented for the Replan Process, candidate threats for the threat classes specified in the Threat Catalog are identified first. All threat candidates that endanger a security goal are then selected as relevant threats and added to the Security Analysis Model. A dedicated rating for the threats is not documented as it can be derived from the rating of the affected security goals and corresponds to the maximum rating from all affected security goals.

Security Requirement	Asset	Security Goal	Threat
User Authentication	Replan Process	Replan Confidentiality, Integrity, and Non-Repudiation	Spoof Dispatcher, Repudiate Route Selection
Message Integrity	Messages	Message Integrity	Tamper Messages, Spoof Participant
Message Confidentiality	Messages	Message Confidentiality	Disclose Message, Spoof Participant
Audit Generation	Replan Process	Replan Integrity and Non-Repudiation	Tamper Replan Process
Data Integrity	Audit Trail, Credentials, Privileges Keys	Audit Trail, Credential, and Keys Integrity	Tamper Data
Data Confidentiality	Audit Trail, Credentials, Privileges, Keys	Audit Trail, Credential, and Keys Confidentiality	Disclose Data
User Task Access Control	Replan Process	Replan Confidentiality, Integrity, and Non-Repudiation	Spoof Dispatcher, Disclose Route Data, Repudiate Route Selection
Minimum Task Resources	Replan Process, Messages	Replan and Messages Availability	Impede Message Transmission and Replan Execution

Table 6.7.: Security requirements for the Replan Process

For the Replan Process, eleven threats are initially documented and supplemented with additional threats for assets that are introduced by controls later in the security engineering process (cf. table 6.7 that displays the security requirements for the Replan Process referencing the respective threats).

The execution of the activity Model Threats is supported by the SecEML editor identifying candidate threats and possibly affected security goals.

6.3.5. Elicit Security Requirements

The elicitation of security requirements yields six initial requirements and two complementing requirements addressing assets introduced due to the selection of controls later in the process. Table 6.7 depicts the resulting set of security requirements for the Replan Process.

The identification, selection, and consolidation of security requirements is substantially based on the security requirement classes specified in the Threat Catalog. Identification of candidate requirements can be reduced to the instantiation of applicable security requirement classes. Consolidation is facilitated by the subsumption of requirements of the same security requirement class. Similarly, validation of the security requirements is supported by the basic definitions as well as the Threat Catalog. For all elements of the Security Analysis Model it can be checked what decisions have been taken in order to elicit the requirements compared to alternative possibilities within the scope of the general definitions (e.g., that the forwarder

process has not been considered as an asset or that no requirement with regard to the quality of the secrets has been specified). Furthermore, all security requirements can be traced to threats, security goals, and assets (and vice versa). Therefore, threats, assets, and security goals that have not been covered can be identified easily.

The application of SecEML and the SecEML editor provides substantial support for the execution of the tasks of the Elicit Security Requirements activity: Identification of candidate requirements, consolidation of requirements, and validation of completeness and consistency of the requirements are technically supported.

6.3.6. Design Controls

The design of the controls to meet the specified security requirements comprises a simple control set (anticipating the restricted control implementations provided by the selected BPMS). Users are authenticated and access to user tasks is mediated. Integrity and confidentiality of messages are protected using a secure channel for message exchange. Audit trails are generated in order to monitor the Replan Process execution and detect deviations from the specified execution sequence. Protection of data assets rests on the assumption that access to the data is restricted physically.

The controls introduce a couple of assets into the Replan Process that need some consideration as well. Therefore, the respective activities from the security engineering process are repeated for an incremental update of the Security Analysis Model. Table 6.8 displays the controls specified for the Replan Process.

Similar to requirements elicitation, execution of necessary tasks for the design of proper controls are substantially based on the Control Catalog: Identification, selection, and detailing of controls use control classes and those classes indicate the introduction of new assets alongside. Even more, checking of the coverage of all security requirements profits from tracing of controls to requirements and threats including affected elements of the Business Process Model.

The SecEML editor provides feedback on candidate controls, covered requirements etc. Exemplary, no control and assumption has been provided for the requirement to enforce the provision of minimum resources for the execution of tasks of the Replan Process as the selected BPMS and its environment do not support such mechanisms. Therefore, it can be easily deduced from the Security Analysis and Design Model (e.g., using the SecEML editor) that the impediment of the execution of the Replan Process as well as impediment of message transmission are not mitigated and the security goals Replan Process and message availability are not backed by proper controls.

6.3.7. Map Controls

The mapping of controls onto control implementations provided by Activiti mainly applies the control implementation classes provided with the Runtime Capability Model: Candidate control implementations are instantiated, effective control implementations are selected, and necessary configuration parameters are provided. If control implementations are provided for

Control / *Assumption*	Requirement	Introduced Asset	Dependency	Implementation
User Authentication	User Authentication	Credentials		Basic HTTP Authentication
User Task Access Control	User Access Control	Privileges	User Authentication	User Access Control
Message Protection	Message Confidentiality and Integrity	Keys		TLS
Audit Generation	Audit Generation	Audit Trail		Log Execution
Physical Access Control	Data Confidentiality and Integrity			

Table 6.8.: Controls, assumptions and control implementations for the Replan Process

all controls specified in the Security Design Model, the coverage check is successful (cf. table 6.8).

Additionally, the application of SecEML allows for further validation: Instantiation of the relations specified by the control (implementation) classes can be checked as well, e.g., in order to detect a missing instantiation of the dependency from the transport layer security (TLS) control implementation to a key store. The SecEML editor supports these validations as well as the generation of proposals for the instantiation of candidate control implementations.

6.3.8. Generate Control Artifacts and Test Cases

Generation of control artifacts and test cases is largely dependent on the environment, the embedding development process, and the development project. For larger and technically homogenous projects, setup and maintenance of transformation chains bridging several steps might be efficient, smaller or more heterogeneous environments and projects might apply only few automated transformation steps. As the implementation of transformation chains is not at the heart of this thesis, the generation of deployment artifacts (e.g., adaptation of the file "logging.properties" for the logging control implementation) and test artifacts will not further covered in this section. An example for the generation of respective artifacts is documented in a master's thesis that has been written in the context our efforts [Ham13].

6.4. Application Experiences

This section reports on exemplary experiences applying (parts of) the framework in different environments. These experiences do not resemble full case studies and are not backed by formal surveys. Therefore, the experiences are sketched only briefly. The intention of this section is to illustrate the application of the framework in different settings and to report on early feedback of users. Further applications within similar environments and with comparable results have been omitted as they do not yield further insights.

The Equipment Surveillance Process Within the course of the already mentioned research project ADiWa (cf. section 3.2), further electronic business processes have been analyzed and re-engineered to allow for better adaptability, service quality, and new service offerings. One example is the Equipment Surveillance process from an international provider of power and automation technologies. The Equipment Surveillance process identifies necessary and optional services for the customers of the automation technologies provider. Within the research project the electronic business process has been re-engineered to switch from a rigid service delivery to a flexible on-demand service delivery.

An early version of SecEPM has been employed in order to identify assets, assess security goals, elicit security requirements, and to select and configure necessary controls for the Equipment Surveillance Process. After preparation of the security engineering process, the respective activities have been executed by security nonprofessionals (employees of the automation technologies provider) independently as well as within workshops accompanied by security experts.

The coverage of activities met the needs of the project. Separation of Security Analysis Model and Security Design Model, traces between the elements, and independence from runtime technology proved helpful in the course of the project, since implications of alternative solutions (mainly due to changes in the business requirements or technical constraints) could be compared and discussed on a common basis. The security nonprofessionals have been able to execute the activities although further guiding artifacts might have been helpful to facilitate adoption. A formal integration into a development process model did not take place. SecEML has not been applied for the Equipment Surveillance Process.

The Account Opening Process Within the course of a master's thesis to develop alternative SecEML tooling, the Account Opening Process has been modeled and analyzed [Rup13]. The Account Opening Process offers customers and prospects the possibility to open a banking account using the website of the bank. Figure 6.4.1 depicts the business process model.

For the master's thesis an early version of SecEPM has been used in order to setup the security engineering process, identify assets, assess security goals, elicit security requirements, and to select controls. SecEML has been applied to capture and validate SecEPM work products. The activities have been executed by the student which has a background as security consultant for financial institutions.

Adoption and application of SecEPM as well as SecEML proved simple and straightforward. The activities covered all necessary aspects of the analysis. Separation of problem and solution domain, traceability of dependencies, independence from development-time and runtime technology facilitated adoption and application. The structure of SecEPM allowed for a simple integration of existing threat and control catalogs. The background of the student hindered the evaluation of the Restricted Skill Sets criteria in this case. SecEML provided all necessary concepts and relations to capture the work products. Concrete syntax, independence of notation and runtime capabilities proved helpful for the analysis. Technical assistance, integration in tool chains, and coexistence with existing models have not been evaluated since new tooling has been developed within the master's thesis.

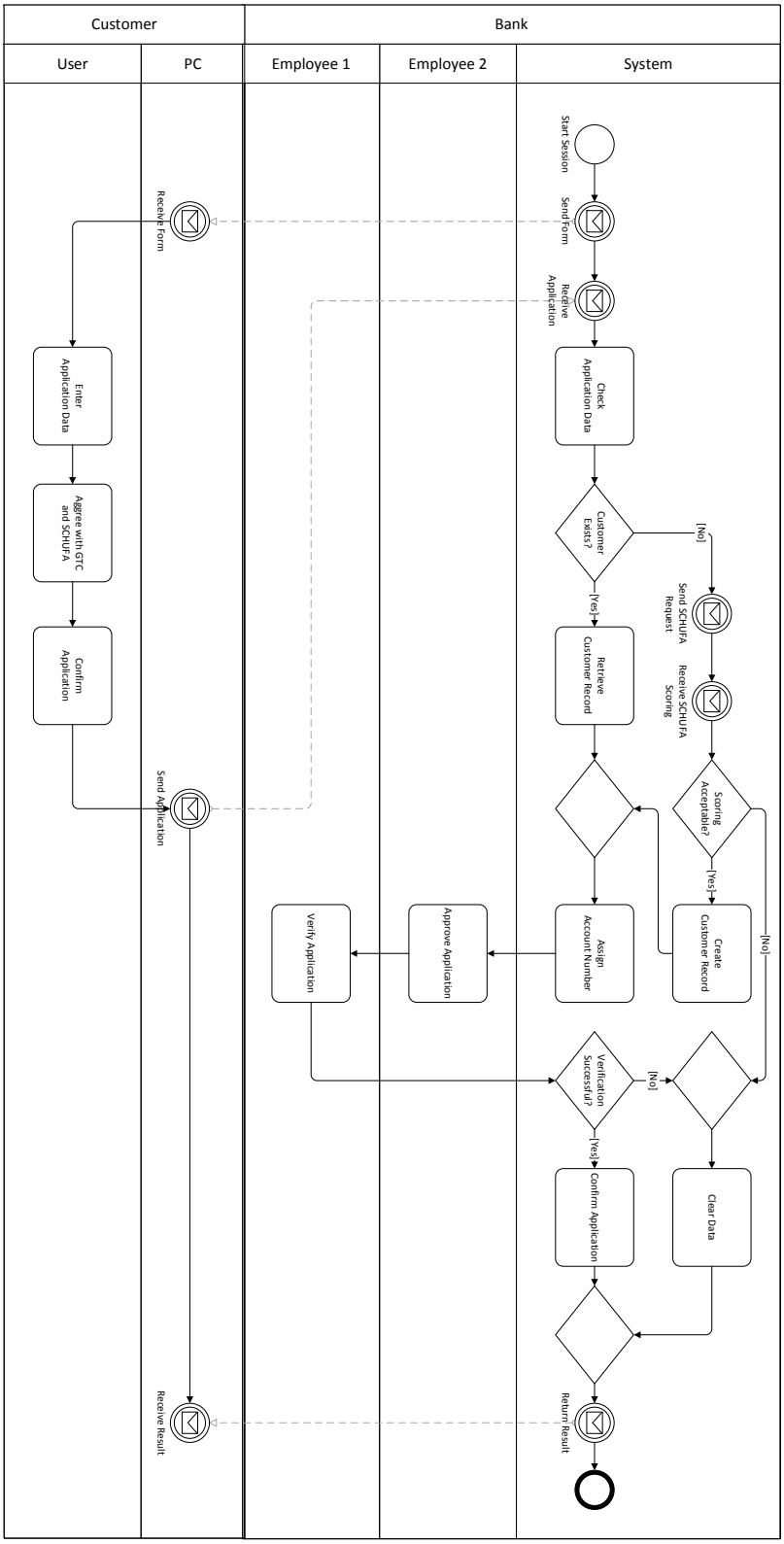

Figure 6.4.1.: The Account Opening Process [Rup13, p. 25] (BPMN process diagram)

The Partner Communication Service In a consulting project for a small and medium-sized enterprise (SME) a new communication service has been integrated into the business process to exchange production data with business partners of the SME. The business process receives status information about pre-products from business partners and sends corresponding information back to the business partners. [Eic11a]

Within the course of this project an early version of SecEML has been used to model the present security posture of the business process. Analogous, SecEML has been used to model, validate, and analyze different solutions for the integration of the new communication service. SecEML has been applied together with the security officer of the SME, a project manager, and several domain experts (for IT infrastructure, operations, and the production application). The security officer as well as domain experts for the IT infrastructure used SecEML regularly as end users. SecEPM has not been applied in the project.

The application of SecEML proved very helpful. Especially, the tooling to validate and analyze the models supported the identification of several weaknesses of the present business process. Similarly, tooling and the independence of runtime capabilities facilitated the discussion of alternative solutions. SecEML covered all necessary concepts for the analysis. Direct representation, independence of notation, and coexistence with existing models contributed to the quick acceptance of SecEML by the end users. The possibility to reuse standard tooling with SecEML allowed for an easy integration into the existing tool chain (e.g., versioning and comparison). On the downside, the application of SecEML has been questioned for larger projects as the textual approach does not foster a fine grained access control for respective models. Also, the appearance of the security workbench attracts people with background as software developer more. For long-term maintenance of SecEML models the stakeholders voted for an additional form-based user interface.

6.5. Comparison of Approaches

This section reviews our results comparing the framework developed in this thesis with existing approaches for security engineering of electronic business processes (cf. sections 2.5.1 and 2.5.3). The application of our framework provides necessary background for this review (cf. sections 6.3 and 6.4). The comparison is systematized applying the criteria developed in section 6.2.

Figure 6.5.1 depicts an overview of the comparison. Filled circles denote fulfillment of an criterion, striped circles indicate partial fulfillment, and empty circles implicate an unfulfilled criterion. If no circle is displayed, a criterion is not applicable with regard to the respective approach. The figure displays criteria for process models as well as for DSMLs. Captions in italic and a lighter grey background indicate criteria that are related to DSMLs.

6.5.1. Comparison of the Process Models

Structure All approaches document activities, roles, work products, and their relations. Not every approach explicates its structure very clearly. Especially, the relations between work

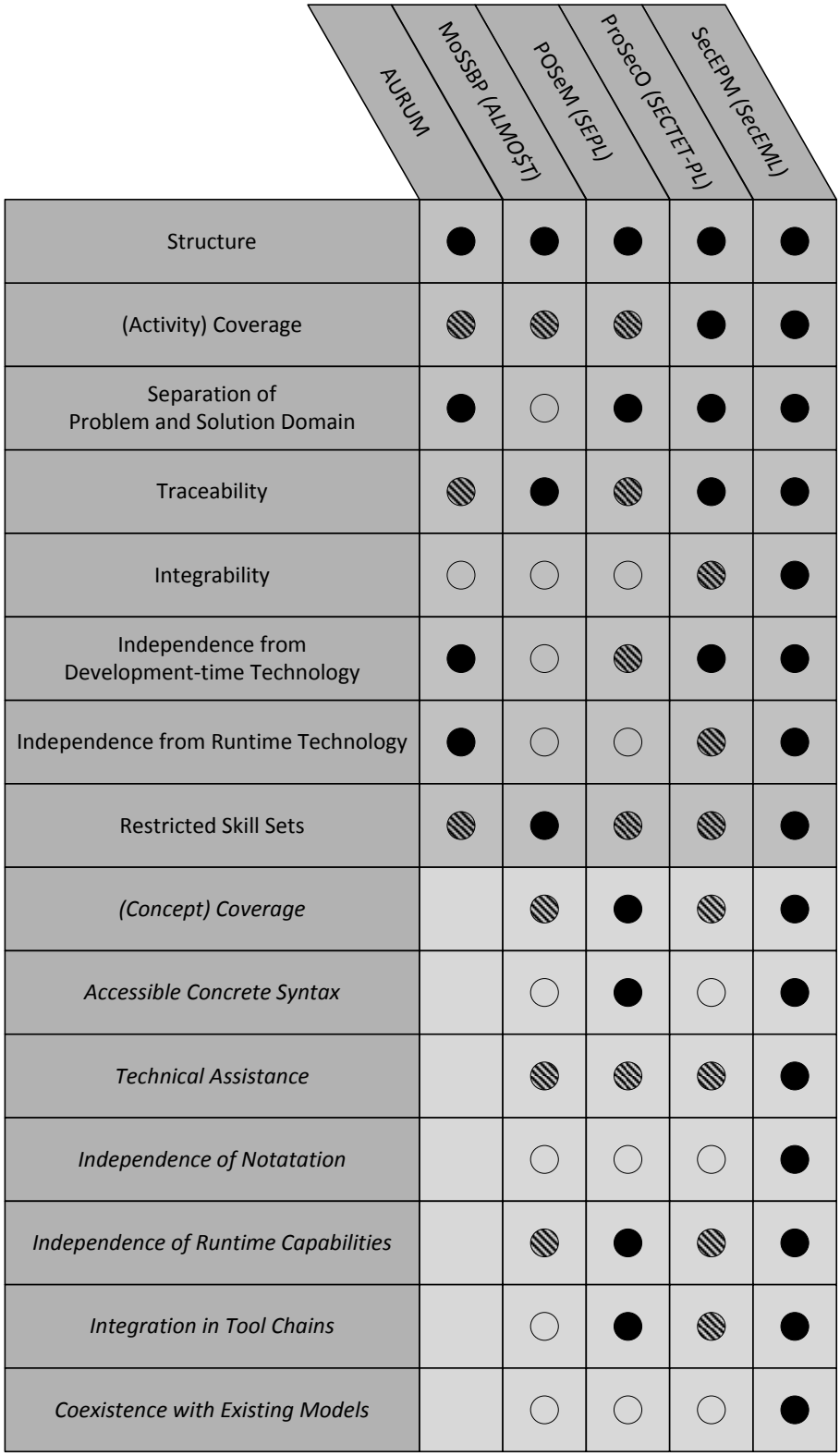

Figure 6.5.1.: Comparison of approaches for security engineering in the domain of BPM

products and roles are sometimes provided only indirectly (e.g., MoSSBP). Nevertheless, all approaches qualify as process model.

Activity Coverage Not all approaches fulfill the *Activity Coverage* criterion. AURUM and MoSSBP do not differentiate the assessment of security goals and the elicitation of security requirements. MoSSBP and POSeM do not include threat modeling but assume this step as prerequisite to the security engineering process. Control configuration is considered by POSeM but not explicated. Only ProSecO and SecEPM fulfill the Activity Coverage criterion completely.

The consequences of non-fulfillment of the criterion might be demonstrated by the following examples using the analysis of the Replan Process as background (cf. section 6.3): Differences between the security goal Replan Confidentiality and a corresponding security requirement User Access Control are not analyzed with AURUM or MoSSBP, underlying decisions are not explicated. Similarly, application of MoSSBP and POSeM does not reveal different threats like Spoof Dispatcher and Repudiate Route Selection that might raise the same security requirement User Access Control. In such cases, security analysis remains incomplete and development of alternative security designs becomes difficult.

Separation of Problem and Solution Domain Consequently, the problem and solution domain is not separated in MoSSBP: Security goals, security requirements, and threats are not consistently differentiated. Therefore, security analysis is intertwined with security design. The other approaches do separate the problem and solution domain. Nevertheless, POSeM and AURUM provide only a partial security analysis as threat modeling is not supported (POSeM) or security goals and requirements are not separated (AURUM). As exemplified for the Activity Coverage criterion, rationale and comparison of alternative security designs becomes difficult with these approaches.

Traceability The *Traceability* criterion is fulfilled for MoSSBP, ProSecO, and SecEPM. AURUM and POSeM do not provide explicit means to trace dependencies between security goals, threats, security requirements etc. but both can be augmented to do so. Nevertheless, although MoSSBP and ProSecO support traceability, their respective DSMLs do provide only little (ALMO$T) or no explicit (SECTET-PL) tool supported traceability features.

Lack of support for traceability hinders effective impact analysis as well as comprehension of work products. Exemplary, exploiting the support for traceability from SecEPM (and SecEML) allows for a simple (tool supported) analysis of all security goals that are (partially) protected applying the control Basic HTTP Authentication: The control addresses the three security goals Replan Confidentiality, Integrity, and Non-Repudiation.

Integrability The integration of the security engineering approaches into development process models is considered by ProSecO and SecEPM. Nevertheless, ProSecO does only conceptually sketch the integration into a generic development process model. Only the framework developed in this thesis provides necessary means for an actual integration into different development process models.

Exemplary, a SecEPM plug-in for a CAME tool is provided with this thesis. The plug-in conforms to SPEM and supports leading products in the market (cf. section 2.3.2). Additionally, a systematic approach for such an integration has been explicated and an actual integration is demonstrated in section 4.7.2. Without such means for integration activities, work products, and roles remain unaligned and the decision about the execution sequence and the consideration of mutual influence of activities remains to the individual participant. Without such plug-in a documentation of the integrated process model has to bridge the different representations of the initial models manually.

Independence from Development-time Technology AURUM, ProSecO, and SecEPM are independent from development-time technology. POSeM applies a proprietary WPDL dialect as starting point for its analysis and utilizes respective tooling. Nevertheless, also without such tooling POSeM might be applied. MoSSBP requires proprietary graph rewriting tools in order to be applied. Therefore, AURUM, ProSecO, and SecEPM fulfill the *Independence from Development-time Technology* criterion, POSeM fulfills partly the criterion, and MoSSBP does not fulfill the criterion.

Independence from Runtime Technology Independent from runtime technology are only AURUM and SecEPM. Conceptually, ProSecO is similarly technology agnostic but assumes a specific MDA tool chain supporting specific implementation-related standards like WS-BPEL and XACML in order to leverage all benefits. POSeM applies its proprietary WPDL dialect that needs to be supported by the runtime environment in order to avoid extra effort. MoSSBP depends on specific runtime environments supporting the resulting secured business process models.

Thus, only AURUM and SecEPM might have been applied without either changes to the runtime environment of the Replan Process or limitations as well as extra effort with regard to the adaption of the business process model.

Restricted Skill Sets The *Restricted Skill Sets* criterion is fulfilled by SecEPM and MoSSBP. All other approaches consider the participation of security nonprofessionals but do not assign activities with a role designated for security nonprofessionals as responsible role. Therefore, they fulfill the criterion only partly.

In case of SecEPM, the Security Analyst provides the Process Model Configuration, the Threat and Control Catalog as well as the tailored process during the activity Setup Process. This enables the Business Process Engineer to execute for example the activities Identify Assets, Assess Security Goals, and Elicit Security Requirements as responsible role.

6.5.2. Comparison of the DSMLs

The criteria related to DSMLs do not apply to AURUM as it does not propose the application of a dedicated DSML. The authors propose a specific method and corresponding tooling to quantify costs and benefits of selected control sets. This method might be integrated in other approaches like SecEPM but this integration is not covered within this thesis.

Concept Coverage The *Concept Coverage* criterion is fulfilled by SecEML and SEPL. The concepts of MoSSBP and ALMO$T are not aligned with the respective process model, i.e., different terms are used in the process model and the DSML. SECTET-PL does not provide direct representation: Exemplary, security goals are not named as such and represented as equations in UML Notes. This misalignment of concepts or lack of direct representation widens the semantic gap and lifts the level of skills necessary to represent the results of the activities using the respective DSML.

Accessible Concrete Syntax Analogously, the *Accessible Concrete Syntax* criterion is not fulfilled for ALMO$T and SECTET-PL. Both DSMLs address security experts that are able to translate the security concepts in question into technical representations or property-related equations and interpret resulting work products. Therefore, it becomes very hard for security nonprofessionals to apply ALMO$T and SECTET-PL in the context of MoSSBP and ProSecO (at least without tooling that shields the user from the language). SEPL applies a proprietary WPDL dialect that is easy to use for BPM domain experts. Analogously, SecEML is based upon HUTN that allows for a common appearance and smooth learning curve.

Technical Assistance All approaches provide technical assistance to apply their respective DSML. Nevertheless, tools for the application of POSeM are only partly implemented (e.g., application of the specified rule sets is not supported so far), for MoSSBP technical assistance for the creation and maintenance of repository items is not supported, and ProSecO/SECTET applies generic UML tools that do not provide SECTET-PL syntax checking, validation and analysis without further implementation.

Therefore, the lack of technical assistance increases the prerequisites to apply these approaches and the effort needed. Especially for security nonprofessionals, technical assistance for the application of a DSML eases its adoption and the validity of resulting models.

Independence of Notation Only SecEML can be applied independently of the notation used for the business process model. POSeM uses a proprietary WPDL dialect, MoSSBP (ALMO$T) and ProSecO (SECTET-PL) apply specific UML dialects. Therefore, all three approaches require a transformation of existing business process models into compatible notations that might break existing tool chains or render the application of established tools impossible.

For example, existing transformations in MDA tool chains often do not maintain relations between UML Notes and the respective model elements. Application of SECTET-PL will become difficult in these environments.

Independence of Runtime Capabilities All DSMLs provide means to model the capabilities of the BPMS in question. A dedicated possibility to model the different capabilities of several BPMSs is only provided by SEPL and SecEML. SECTET-PL does not encapsulate this information but spreads it partly across the UML Notes used with SECTET-PL and partly across the transformations that generate the secured business process model. ALMO$T does not separate capabilities of different BPMSs and depends on specific capabilities as precondition.

Therefore, it becomes very hard to model and compare the application of different BPMSs (or the same BPMS in different configurations) for the same business process in order to select the optimal engine and to balance trade-offs. Even worse, it might not be possible to apply SECTET-PL or ALMO$T at all.

Integration in Tool Chains SEPL and SecEML do not require a dedicated infrastructure or conceptually deviating tooling in order to create, validate, transform, and manage respective models. Dedicated tooling provided for these DSMLs can be used in combination with standard tooling, e.g. for change and model management. SECTET-PL rests on UML that fits very well in existing tool chains. Unfortunately, SECTET-PL requires a specific application procedure that places several constraints on the tools applied (e.g., handling of UML Notes or transformation environment) that might hinder effective application of standard tooling. ALMO$T in the context of MoSSBP requires dedicated tooling for model management and application that will not integrate easily in existing tool chains.

Coexistence with Existing Models Only SecEML allows for an effective coexistence with existing business process models. SECTET-PL is applied within UML Notes attached to elements in existing business process models and therefore changes existing models. SEPL integrates security directly into the business process model, existing WPDL models have to be changed in order to capture the security aspects. The application of ALMO$T in the context of MoSSBP requires specific business process models that are changed within the execution of the activities from MoSSBP.

6.5.3. Aggregation

None of the existing approaches that we compared with our framework meets all requirements (cf. figure 6.5.1). Nevertheless, the approaches exhibit different strength with regard to our analysis criteria. In order to streamline the results from the preceding sections comparing the existing approaches we assign our analysis criteria to the main issues with regard to security in the domain of BPM. Figure 6.5.2 displays the primary assignments, i.e., which analysis criteria applies foremost to one of the main issues: security nonprofessionals deciding and implementing security (labeled “Security Nonprofessionals”), heterogeneity of business process engines (labeled “Heterogeneity of BPM Engines”), and business process environmental heterogeneity (labeled “Environmental Heterogeneity”). Filled circles denote the primary assignment. The assignments are based mainly on rationale provided in the corresponding sections of the requirement analysis for SecEPM and SecEML (cf. sections 4.2 and 5.2).

With regard to process models, the analysis criteria Activity Coverage, Traceability, and Restricted Skill Sets apply primary to Security Nonprofessionals. The analysis criteria Separation of Problem and Solution Domain as well as the Independence from Runtime Technology are primarily assigned to the Heterogeneity of BPM Engines. The analysis criteria Structure, Integrability, and Independence from Development-time Technology apply primarily to Environmental Heterogeneity. With regard to DSMLs, the analysis criteria Concept Coverage,

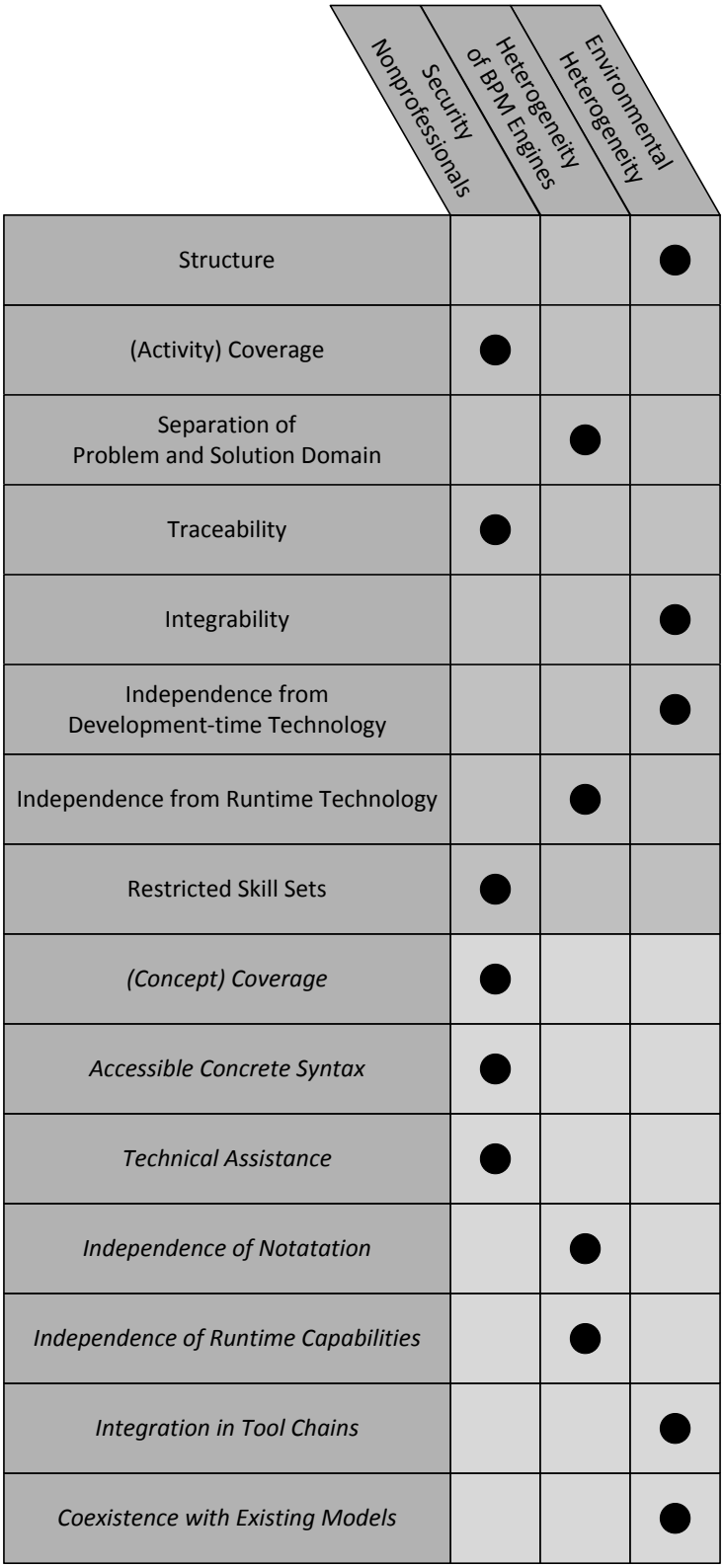

Figure 6.5.2.: Primary assignment of analysis criteria and main issues

	Security Nonprofessionals	Heterogeneity of BPM Engines	Environmental Heterogeneity	Sum
AURUM	0 (3)	2 (0)	2 (0)	4 (3)
MoSSBP	2 (1)	0 (0)	1 (0)	3 (1)
POSeM	0 (3)	1 (0)	1 (1)	2 (4)
ProSecO	2 (1)	1 (1)	2 (1)	5 (3)
SecEPM	3 (0)	2 (0)	3 (0)	8 (0)

Table 6.9.: Fulfillment of analysis criteria with regard to main issues by process model

	Security Nonprofessionals	Heterogeneity of BPM Engines	Environmental Heterogeneity	Sum
ALMO$T	0 (2)	0 (1)	0 (0)	0 (3)
SecEML	3 (0)	2 (0)	2 (0)	7 (0)
SECTET-PL	0 (2)	0 (1)	0 (1)	0 (4)
SEPL	2 (1)	1 (0)	1 (0)	4 (1)

Table 6.10.: Fulfillment of analysis criteria with regard to main issues by DSML

Accessible Concrete Syntax, and Technical Assistance are assigned primarily to Security Nonprofessionals. The analysis criteria Independence of Notation and Independence of Runtime Capabilities apply primarily to Heterogeneity of BPM Engines. Integration in Tool Chains and Coexistence with Existing Models are assigned primarily to Environmental Heterogeneity.

Table 6.9 displays the aggregation of fulfilled or (in brackets) partly fulfilled analysis criteria of the process models included in our comparison with regard to the main issues described earlier. The aggregation displays the following order: POSeM fulfills least analysis criteria, followed by MoSSBP and AURUM. Nevertheless, POSeM fulfills more criteria partly compared with MoSSBP. ProSecO is next in the list dominated by SecEPM. Within this total order, MoSSBP demonstrates its relative strength in the support of security nonprofessionals while AURUM and POSeM address heterogeneity of business process engines and heterogeneity of the environment better. ProSecO as well as SecEPM are (comparatively) well balanced with regard to all three aspects.

Table 6.10 displays a similar aggregation of fulfilled (or partly fulfilled) analysis criteria for the DSMLs. ALMO$T fulfills least analysis criteria, followed by SECTET-PL, SEPL, and SecEML. In line with the observation for the corresponding process model MoSSBP, ALMO$T demonstrates its relative strength to support security nonprofessionals. Unlike ProSecO, the respective DSML SECTET-PL fulfills rather few analysis criteria and remains weak in comparison to the other DSMLs as well as with respect to ProSecO. In contrast to the combination ProSecO and SECTET-PL, SEPL fulfills many analysis criteria and provides comparable good support also for security nonprofessionals—which is different for its process model POSeM. Our proposal SecEML fulfills all analysis criteria as well as our process model SecEPM.

6.6. Discussion

The joint application of our framework, the report on exemplary application experiences, and the systematic comparison based on our analysis criteria demonstrate the feasibility of the framework developed in this thesis and its advantages over other approaches. This discussion starts with an short interpretation of the comparison presented in the preceding section, details some specifics of our framework and closes with some general considerations.

The comparison of existing approaches with our framework revealed its advantages. As we designed SecEPM and SecEML applying requirements that have been used as basis for our analysis criteria, this result is not unexpected. Nevertheless, the comparison displayed the relative strength of the different approaches. Interestingly, one of the strongest process models (i.e., ProSecO) is accompanied by a rather weak DSML applying our analysis criteria. Similarly, one of the strongest DSMLs (i.e., SEPL) is accompanied by one of the weakest process model applying our analysis criteria. As process model and DSML are interrelated and often closely coupled, a replacement of a DSML and/or process model is not an easy task. Therefore, our framework demonstrates one unique strength providing a process model and DSML that both fulfill all of our analysis criteria uniformly.

But not only the systematic comparison of the approaches applying our analysis criteria demonstrates the advantages of our framework. The following paragraphs discuss specifics of our framework that have been demonstrated in the preceding sections and might be summarized using the terms adaptability, supportiveness, and maintainability.

The joint application displays multiple options to adapt our framework to heterogeneous environments and technical constraints in a structured and guided manner. One important aspect is the integration of different established methods into the security engineering process. Security goal classes, ratings, and damage scenarios including rating criteria stem from established methods and standards that have been incorporated into the Process Model Configuration. Exemplary, the adoption of the STRIDE method for threat modeling and the alignment of security requirement classes with SFRs from Common Criteria account for this. The mechanisms provided by Activiti have been represented as control implementation classes well. Equally, other methods and standards could have been used as a basis for these models which demonstrates the adaptability of the framework. These observations might be generalized: Separation of activities and guidance in SecEPM allows for a simple and transparent integration of additional methods and provision of supporting material. This differentiation is unique in comparison with the existing approaches.

Also, the integration of a preparatory activity Setup Process allows for an application of SecEPM in heterogeneous environments facing restricted skill sets with regard to security. A similar activity is only implicitly addressed by existing approaches, e.g., referring control or threat catalogs that need to be set up in advance. Thus, existing approaches do not take the separation of (preparatory) security-intense tasks and other tasks into focus compared with SecEPM.

The need to tailor the security engineering process for the respective organization has got not much attention in existing approaches. SecEPM explicitly introduces this need as an activity and provides modular process model content in order to fit to heterogeneous environments and to allow for an easy enrichment as well as reduction of the amount of

documentation provided. Similarly, the consideration of the need to integrate SecEPM into software development process models and its explication is unique with regard to existing approaches discussed. This allows to align security engineering with the existing development organization and does not offer an all-or-nothing alternative forcing unnecessary and costly changes to the organization.

Adaptability comes at some cost: The adaptability of our framework harbors the possibility to provoke inconsistencies and omissions. Exemplary, inconsistent guidelines might be lumped together, threats or their mitigation might be not considered. In order to limit possible inconsistencies and omissions, several means have been integrated into the framework. This encompasses the structuring and documentation of the elements of SecEPM including the relations and dependencies of the elements, the provision of a guideline on the integration of (further) methods and techniques, and the application of tooling for authoring and management of the security engineering process model. Similarly, the following means contribute with regard to the mitigation of costs of adaptability: SecEML specifying relevant elements of the work products and capturing important relations between them as well as tooling to support the creation, validation, and maintenance of consistent and analyzable work products. Nevertheless, without necessary skills available to setup the security engineering process and to validate its results the framework might not be applied successfully.

Reviewing the results of our study, supportiveness might be seen as another summarizing and discriminating property of our framework. The specification of the Process Model Configuration, Threat and Control Catalog, and the Runtime Capability Model using SecEML and the application of the SecEML editor provides comprehensive support to enable security nonprofessionals to create Security Analysis and Design Models.

Similarly, these models specified using SecEML and corresponding tooling provide means for security experts participating in a development project as Security Analyst or reviewing its results to understand the decisions taken in the security engineering process and to validate completeness and consistency of the resulting models. Furthermore, systematical analysis and comparison of design alternatives is supported applying existing tools from the EMF ecosystem, e.g., the Interactive OCL Console to evaluate OCL expressions in order to identify critical controls which failure endangers many or highly rated security goals. As well, the Process Model Engineer (and other roles like the Security Analyst) receive substantial support tailoring the security engineering process and integrating the process into the development process applying the SecEPM plug-in.

As a drawback, supportiveness of our framework for security nonprofessionals depends largely on non-trivial and potentially time-costly preparatory tasks mainly of the Security Analyst and—to some degree—other participating roles. Exemplary, the provision of the Threat and Control Catalog is critical: Only those threats will be considered for which corresponding threat classes have been specified in the Threat Catalog. Likewise, only those controls will be applied that are envisioned in the Control Catalog. Nevertheless, SecEPM does not only detail those necessary steps but respective artifacts specified using SecEML can be collected and shared intra- and inter-organizationally and therefore leverage existing solutions.

The means to allow for a simple maintenance of the security engineering process model as well as the work products resulting from its application in a development process are similar to those mitigating possible negative consequences from adaptability. Maintainability of the work

products is established by SecEML providing means to trace elements of the model as well as corresponding tooling checking the consistency of the work products specified using SecEML, allowing for simple navigation to referenced and referencing elements of the models, and evaluate derived attributes of elements based on their dependencies (e.g., security goal ratings). For SecEPM, modeling of the process model applying the well recognized standard SPEM and respective tooling as well as guidance for complementing SecEPM allow for maintainability.

As with adaptability, maintainability must be balanced with the ease of application of our framework. Creation, evaluation, and maintenance of dependencies between model elements (within work products or for the security engineering process model) requires effort that must not outweigh its benefits or introduce new challenges. The design of SecEPM considers this aspects decoupling activities, focusing on the description of dependencies between model elements with dedicated elements, and separating the different concerns with regard to roles, implementation, etc. (cf. sections 4.3 and 4.6).

Although SecEPM covers all activities identified in the requirements analysis (cf. section 4.2), it addresses only a restricted set of activities compared with approaches for the development of secure systems (cf. [Eur11]). This is mainly due to the confinement of the scope of this thesis in order to provide a comprehensive approach considering the major issues detailed in the problem statement (cf. section 1.2). Support for additional phases of the BPM life cycle would be interesting (e.g., Enactment, Analysis, or Administration). Also, inclusion of further methods such as the validation of security-related properties of the business process model itself or consideration of additional components and layers of the systems used for the enactment of electronic business processes might be valuable. Notwithstanding, SecEPM addresses comprehensively the major issues focused on in this thesis, and we are optimistic that it is flexible enough to incorporate many further aspects as well.

With regard to application experiences, our observations displays promising results (cf. section 6.4). Participants using (parts of) the framework have been able to understand and execute all activities and guidelines of SecEPM and to adopt SecEML quickly. Feedback of end users included the wish for further guidelines and a form-based user interface to support non-technical users better. These additions can be integrated seamlessly. The request for fine grained access control in the context of large scale work products using SecEML needs further investigation as it has not been considered in any of the approaches presented in section 2.5. Nevertheless, further case studies and empirical experiments will be necessary in order to complement the exemplary study of this thesis, to foster deeper insights, and to provide solid ground for an empirical evaluation of the framework. Unfortunately, empirical evaluations are generally rare in the security engineering domain and not available for the approaches discussed in this chapter.

6.7. Summary

This chapter provided a joint application of our framework presenting an exemplary study and documents the demonstration and evaluation activities of our research approach.

Eight analysis criteria from the requirements specified for SecEPM and seven additional analysis criteria from the requirements specified for SecEML have been derived and docu-

mented (cf. section 6.2). Few disambiguations have been made, e.g., in order to differentiate Activity and Concept Coverage. Similarly, several criteria have been adopted to pin down a clear and testable criterion, e.g., to specify that at least one activity is assigned to a role designated for security nonprofessionals in order to fulfill the Restricted Skill Sets criterion.

We applied our framework to the Replan Process in section 6.3. We documented the application of all activities, resulting work products (as summary tables as well as SecEML listings), accompanying considerations and rationale. Support for the execution of the activities provided by the the SecEML editor has been highlighted.

Exemplary experiences applying (parts of) the framework in different environments have been documented in section 6.4. Application of SecEPM for the Equipment Surveillance Process, application of SecEPM and SecEML for the Account Opening Process, and application of an early version of SecEML for the Partner Communication Service displayed promising results. Participants have been able to understand and execute all activities and guidelines of SecEPM and to adopt SecEML quickly. Possibilities for the improvement of the framework have been observed and documented.

A comparison of the framework developed in this thesis and existing approaches for security engineering of electronic business processes revealed advantages of our framework over the existing approaches (cf. section 6.5). Two criteria are not fulfilled for any of the other approaches investigated (Independence of Notation, Coexistence with Existing Models) or only partly fulfilled (Integrability, Technical Assistance). None of the other approaches fulfilled all criteria for either the process model or the DSML.

Section 6.6 discussed the results from the preceding sections. The exemplary study demonstrated feasibility of the framework developed in this thesis and its advantages over other approaches. Especially, it proved to be uniquely adaptable providing nonetheless structure, guidance, and support to be applied by different participants. Similarly, the framework provides maintainable artifacts and results in maintainable work products that supports coping with the complexity of the framework and the security engineering of electronic business processes. Although the application experiences are promising, further case studies and empirical experiments will be necessary in order to complement the results of this exemplary study.

7. Conclusion

The preceding chapters presented development and application of our model-based security engineering framework in the domain of BPM. This chapter concludes the thesis and is structured as follows: We summarize our contributions in section 7.1. Findings with regard to our research questions reflecting our contributions are subsumed in section 7.2. The final section 7.3 provides directions on future work and research topics to further improve our framework.

7.1. Summary of Contributions

With this thesis we contribute our framework for model-based security engineering in the domain of BPM. This framework comprises three main components: the Security Engineering Process Model model (SecEPM), its accompanying Security Engineering Modeling Language (SecEML), and an integrated security workbench supporting the application of SecEPM and SecEML.

Alongside, we document our approach for the development of the framework and its components. This includes not only conceptual and methodical considerations but also technical aspects and proposals for an implementation.

Security Engineering Process Model (SecEPM)

We contribute our process model for security engineering that integrates security-related activities in the course of the development of secure electronic business processes. We elicit major requirements derived from matching security engineering process models and the issues detailed in the problem statement. Corresponding design strategies for our process model are explicated including specialization, separation of concerns, and decoupling.

We provide structure and key entities of SecEPM as well as their relationships comprising activities, work products, roles. Additionally, we depict guiding artifacts to supplement the aforementioned entities of the process model in a flexible and extendible way. We model SecEPM applying the internationally standardized metamodel SPEM and provide strategies as well as their application in order to integrate SecEPM into existing development process models.

Security Engineering Modeling Language (SecEML)

We complement SecEPM contributing our DSML to capture work products of SecEPM and to provide a basis for the validation and transformation of these models. We establish our DSML eliciting key requirements aligned with the process model and explicate corresponding design strategies encompassing direct representation, modularization, reuse, and application of MDE techniques. We document the metamodel of SecEML and the definition of its concrete syntax.

Workbench

As third part of our framework an integrated security workbench supporting the application of SecEPM and SecEML is provided. We document the application of a model-driven framework based on EMF and Xtext in order to implement a SecEML editor as well as the adaptation of the model-driven transformation chain in order to add further capabilities to the SecEML editor. We describe the integration of our SecEML editor into an integrated security workbench supporting model management, requirements management, project management and change management.

7.2. Findings

Important motives for the application of BPM and accompanying systems are flexibility and responsiveness: to allow for faster reactions to environmental and market changes as well as for proactive innovations of products and services, and thus finally, to become an adaptive enterprise. Although security of electronic business processes ought to be of high importance for every organization, only few proposals for the systematic development of secure electronic business processes have been published. We approached this topic focusing on three major issues with regard to security engineering in the domain of BPM: security nonprofessionals implementing security, heterogeneity of business process engines, and business process environmental heterogeneity. In order to develop a security engineering framework in the domain of BPM we detailed three questions to guide our research.

1. What requirements does a security engineering process model for electronic business processes place that copes with the prominence of security nonprofessionals, heterogeneous business process engines, and environmental heterogeneity in the domain of BPM and how could they be met?

We propose SecEPM as security engineering process model for electronic business processes suitable in the domain of BPM. It is systematically developed to address the issues detailed in the problem statement using existing proposals as foundation.

SecEPM leverages security nonprofessionals within the development process of secure electronic business processes. It details necessary activities from the initial identification of assets and the assessment of respective security goals to control design and their mapping onto capabilities of the runtime environment. It provides information on necessary work products and their content as well as guidance for the execution of the activities. Furthermore, it consolidates tasks that need security expertise as much as possible in a preparatory activity and allows security experts to understand and validate work products compiled by security nonprofessionals documenting relationships between assets, security goals, requirements, controls, and their configuration.

Heterogeneity of business process engines is considered by SecEPM separating runtime independent from runtime dependent aspects. This includes the specification of an activity to explicitly map generic controls onto runtime capabilities as well as the respective work products (especially, the Runtime Capability Model).

Likewise, SecEPM addresses environmental heterogeneity in the domain of BPM. It concentrates on core security-related activities, work products, roles, and guidance artifacts. These

method fragments and chunks are modeled using SPEM, an internationally accepted and standardized metamodel. Strategies for the integration of SecEPM with development process models and their application are detailed and backed by available tooling to support all necessary steps. Nevertheless, SecEPM is not bound to specific development-time technology hindering integration into existing tool chains.

2. What requirements does a DSML place that supports the elaboration of main work products of the developed security engineering process model and how could they be met?

SecEML is our proposal for a DSML that supports the elaboration of main work products of SecEPM and complements our framework for security engineering in the domain of BPM. It is systematically aligned with SecEPM and contributes to the research objective addressing analogously the issues identified in the problem statement.

SecEML and its respective tooling support security nonprofessionals elaborating the work products of SecEPM. SecEML covers necessary concepts and relations and provides an accessible textual syntax. The SecEML editor facilitates creation, validation, and analysis of work products.

An important aspect of SecEML is the ability to capture the configuration of chosen controls independently from generic aspects. Therefore, SecEML copes with the heterogeneity of business process engines as it provides a link between design and implementation that can be used within MDS tool chains.

Equally, SecEML and its corresponding tooling facilitates elaboration of main work products of SecEPM in heterogeneous environments. It does not require specific notations with regard to the specification of electronic business processes. The application of OCL statements within SecEML introduces powerful means to specify relations between modeled elements within work products as well as with the business process model. Furthermore, the application of a model-driven framework to develop the SecEML editor alleviates the adaptation of the tooling and the integration into an existing tool chain.

3. What observations do we get applying the framework developed in this thesis?

The prototypical implementation of the constituents of the framework and their joint application demonstrated the feasibility of our proposal. The application of analysis criteria derived from the requirements elicited in preceding sections shows significant advantages of the framework developed in this thesis over other approaches for security engineering in the domain of BPM. No other approach fulfills all analysis criteria and several criteria are only met by our approach.

The exemplary study demonstrates multiple options to adapt our framework to heterogeneous environments and technical constraints imposed by different BPMSs in a structured and guided manner. SecEPM is SPEM compliant, allows for integration of existing methods and is furthermore prepared for the integration into existing development processes. SecEML contributes similarly to adaptability providing configurable means for technical assistance and adaptation. Nevertheless, adaptability of our framework induces a complexity that must be managed in order to balance utility and tailoring effort.

Supportiveness might be seen as another summarizing and discriminating property of our framework. Our framework is supportive for participants as it provides flexible but detailed directions. It assists participants using the integrated workbench creating, validating, and analyzing work products. On the downside, in order to fully exploit the supportiveness of the framework, non-trivial and potentially time-costly preparatory tasks have to be executed.

The means to allow for a simple maintenance of the security engineering process model as well as the work products resulting from its application in a development process are similar to those mitigating possible negative consequences from adaptability. The maintainability of our framework is achieved providing means to manage, trace, and validate elements and relationships conceptually and technically for the process model and (instantiated) work products. As with flexibility, maintenance must be managed.

The reported application experiences display promising results. Participants using (parts of) the framework have been able to understand and execute all activities and guidelines of SecEPM and to adopt SecEML quickly. Further improvements have been proposed that might be realized in future development. Nevertheless, further case studies and empirical experiments will be necessary in order to complement the exemplary study of this thesis, to foster deeper insights, and to provide solid ground for an empirical evaluation of the framework.

7.3. Future Work

The development of our framework for security engineering of electronic business processes included a couple of decisions that might be challenged. Exemplary, the focus on the three issues depicted in the problem statement influenced largely objective, development, and finally the result of this thesis. We based our choice on objectives of BPM and important findings from scientific publications. Other authors might consider additional issues equally or even more pressing. Future work might be devoted to further analyze complementary issues with regard to security in the domain of BPM. Since foundations and procedure for the development of our framework has been documented in detail, we would expect that analysis and adaptation of the framework with regard to potential deviating issues might be significantly facilitated.

The exemplary study demonstrated application and benefits of our framework. As it has been noted in the findings, further case studies and empirical experiments will be necessary to support our results. Empirical studies analyzing the benefits of approaches for security engineering are generally rare. Thus, it seems to be very promising to not only support our results with respective studies but provide further insights in this research field based on our framework. Exemplary, we consider the cross-sectional comparison of the application of our framework within different environments (e.g., smaller and larger companies) but similar projects (in terms of technical complexity and project size) as an interesting approach. Complementary, it seems to be promising to measure the change in efficiency due to the application within many projects in the same environment as a longitudinal study.

As we restricted the scope of SecEPM for this thesis, it might be worthwhile to extend the scope of the process model. Further life cycle phases might be included into the process

model. Additional methods and development processes might be selected for analysis and integration. Likewise, the analysis of different domains and the transfer of results from this thesis into promising domains might be interesting.

Similarly, enhancements of the technical assistance for the application of the framework might further leverage its effect. Especially with regard to SecEML, alternative representations of work products might lower reservations concerning the use of SecEML, e.g., providing a form-based representation. The implementation of tool chains to automate the transformation of SecEML models into implementation and test artifacts might bolster the efficiency for (technically) similar projects. Additionally, the focus of the contributed security workbench might be broadened. As it is focused on the support for the actual security engineering process it might be enhanced to become part of a CAME in order to collect, harmonize, and provide work products like the Threat and Control Catalog or the Runtime Capability Model.

We might draw an analogy to confine the role of our framework: As security controls do not dissolve security problems but shift important aspects in order to keep them manageable, the application of our framework for security engineering of electronic business processes does not dissolve the hardships of security engineering in general. It provides a proposal to structure and setup the security engineering process in a flexible manner, explains and guides the execution of necessary tasks, and provides means to technically assist the security engineering process. Hence, it shifts important aspects in order to align security engineering with the prominence of security nonprofessionals, heterogeneous business process engines, and environmental heterogeneity in the domain of BPM.

A. SecEML Grammar

Grammar of SecEML provided using EBNF according to ISO 14977 [Int96][1].

Model → 'model' ID
['language' ID [STRING]]
['process' ObjectRef]
{AbstractElement} ;

AbstractElement → Entity | PackageDeclaration | Import ;

PackageDeclaration → 'package' QualifiedName '('
{AbstractElement}
')' ;

Import → 'import' QualifiedNameWithWildcard ;

Entity → Stakeholder | AssetClass | Asset | Resource | SecurityGoalClass |
SecurityGoalRating | DamageScenario | SecurityGoalCriterion |
SecurityGoal | Threat | ThreatClass | ControlPurpose |
SecurityRequirementClass | SecurityRequirement | ControlClass |
ArchitectureElement | Runtime | ControlImplementationClass |
ControlImplementationRating | ControlImplementation |
RuntimeClass ;

Stakeholder → 'stakeholder' ID '('
['title' STRING]
['description' STRING]
')' ;

AssetClass → 'assetClass' ID '('
['title' STRING]
['description' STRING]
['type' STRING]
')' ;

Asset → 'asset' ID ':' AssetClassRef '('
['title' STRING]
['description' STRING]
['stakeholder' StakeholderRef {',' StakeholderRef}]
['supports' AssetRef {',' AssetRef}]
['elements' ObjectRef {',' ObjectRef}]
')' ;

[1] We do not provide a minimal set of syntax rules to make the definition easier to understand.

Resource → 'resource' ID '('
['title' STRING]
['description' STRING]
['supports' AssetRef {',' AssetRef}]
['elements' ObjectRef {',' ObjectRef}]
')' ;

SecurityGoalClass → 'securityGoalClass' ID '('
['title' STRING]
['description' STRING]
')' ;

SecurityGoalRating → 'securityGoalRating' ID '('
['title' STRING]
['description' STRING]
'ordinal' INT
')' ;

DamageScenario → 'scenario' ID '('
['title' STRING]
['description' STRING]
')' ;

SecurityGoalCriterion → 'criterion' ID '('
['title' STRING]
['description' STRING]
'rating' SecurityGoalRatingRef
'scenarios' DamageScenarioRef {',' DamageScenarioRef}
')' ;

AccountingForDependencies → 'Dependencies' |'NoDependencies' ;

AggregationStrategy → 'Maximum' |'Cumulation' |'Distribution' ;

AggregationModifier → [AccountingForDependencies] AggregationStrategy ;

SecurityGoal → 'goal' ID ':' SecurityGoalClassRef '('
'asset' AssetRef
['stakeholder' StakeholderRef {',' StakeholderRef}]
['rating' SecurityGoalRatingRef]
['criteria' SecurityGoalCriterionRef {',' SecurityGoalCriterionRef}]
['aggregatedUsing' AggregationModifier]
['comment' STRING]
')' ;

ThreatClass → 'threatClass' ID '('
'securityGoalClasses' SecurityGoalClassRef {',' SecurityGoalClassRef}
'assetClasses' AssetClassRef {',' AssetClassRef}
['assetExp' STRING]

['matchExp' STRING]
['transformExp' STRING]
')' ;

Threat → 'threat' ID ':' ThreatClassRef '('
'goals' SecurityGoalRef {',' SecurityGoalRef}
(['entities' ObjectRef (',' ObjectRef)*) | ('entityExp'
STRING])
')' ;

SecurityRequirementClass → 'securityRequirementClass' ID '('
'securityGoalClasses' SecurityGoalClassRef {','
SecurityGoalClassRef}
'assetClasses' AssetClassRef {',' AssetClassRef}
'threatClasses' ThreatClassRef {',' ThreatClassRef}
['matchExp' STRING]
')' ;

SecurityRequirement → 'securityRequirement' ID ':'
SecurityRequirementClassRef '('
'goals' SecurityGoalRef {',' SecurityGoalRef}
'threats' ThreatRef {',' ThreatRef}
[('entities' ObjectRef {',' ObjectRef}) | ('entityExp'
STRING)]
')' ;

ControlPurpose → 'controlPurpose' ID ;

ControlClass → 'controlClass' ID '('
'securityRequirementClasses'
SecurityRequirementClassRef {',' SecurityRequirementClassRef}
['threatClasses' ThreatClassRef {',' ThreatClassRef}]
['purposes' ControlPurposeRef {',' ControlPurposeRef}]
['introduces' AssetClassRef {',' AssetClassRef}]
['reliesOn' ControlClassRef {',' ControlClassRef}]
['matchExp' STRING]
')' ;

ArchitectureElement → (Control | Assumption) ':' ControlClassRef '('
'securityRequirements' SecurityRequirementRef {','
SecurityRequirementRef}
['threats' ThreatRef {',' ThreatRef}]
['introduces' AssetRef {',' AssetRef}]
['reliesOn' ArchitectureElementRef {','
ArchitectureElementRef}]
[('entities' ObjectRef {',' ObjectRef}) | ('entityExp'
STRING)]
')' ;

```
Assumption → 'assumption' ID ;
Control → 'control' ID ;
RuntimeClass → 'runtimeClass' ID '('
    ['title' STRING]
    ['description' STRING]
    ')' ;
Runtime → 'runtime' ID ':' RuntimeClassRef ;
ControlImplementationRating → 'controlImplementationRating' ID '('
    'ordinal' INT
    ['ratingRestriction' SecurityGoalRatingRef {',' SecurityGoalRatingRef}]
    ')' ;
Property → '(' STRING '=' STRING ')' ;
ControlImplementationClass → 'controlImplementationClass' ID '('
    'runtimeClass' RuntimeClassRef
    'controlClasses' ControlClassRef {',' ControlClassRef}
    ['rating' ControlImplementationRatingRef]
    ['securityRequirementClasses' SecurityRequirementClassRef {',' SecurityRequirementClassRef}]
    ['threatClasses' ThreatClassRef {',' ThreatClassRef}]
    ['purposes' ControlPurposeRef {',' ControlPurposeRef}]
    ['reliesOn' ControlImplementationClassRef {',' ControlImplementationClassRef}]
    ['properties' STRING {',' STRING}]
    ['validationExp' STRING ]
    ')' ;
ControlImplementation → 'controlImplementation' ID ':'
    ControlImplementationClassRef {',' ControlImplementationClassRef}
    '(' 'runtime' RuntimeRef 'controls' ControlRef ['[' ControlRef)*
    {'properties' Property (',' Property}]
    ')' ;
StakeholderRef → QualifiedName ;
AssetClassRef → QualifiedName ;
AssetRef → QualifiedName ;
SecurityGoalClassRef → QualifiedName ;
SecurityGoalRatingRef → QualifiedName ;
DamageScenarioRef → QualifiedName ;
SecurityGoalCriterionRef → QualifiedName ;
SecurityGoalRef → QualifiedName ;
```

ThreatRef → QualifiedName ;
ThreatClassRef → QualifiedName ;
ControlPurposeRef → QualifiedName ;
SecurityRequirementClassRef → QualifiedName ;
SecurityRequirementRef → QualifiedName ;
ControlClassRef → QualifiedName ;
ArchitectureElementRef → QualifiedName ;
RuntimeRef → QualifiedName ;
ControlImplementationClassRef → QualifiedName ;
ControlImplementationRatingRef → QualifiedName ;
ControlImplementationRef → QualifiedName ;
RuntimeClassRef → QualifiedName ;
ObjectRef → QualifiedName ;
QualifiedNameWithWildcard → QualifiedName '.*' ? ;
QualifiedName → ID {'.' ID} ;
STRING → *?String like in Java?* ;
INT → *?Integer like in Java?* ;
ID → *?Identifier like in Java?* ;

B. Work Products from the Exemplary Study

B.1. Business Process Model

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL
       " xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:dc
       ="http://www.omg.org/spec/DD/20100524/DC" xmlns:di="http://www.omg
       .org/spec/DD/20100524/DI" id="ReplanDefinitions" name="
       ReplanDefinitions" targetNamespace="http://sample.bpmn2.org/bpmn2/
       sample/collaboration">
3    <bpmn2:collaboration id="ReplanCollaboration" name="
         ReplanCollaboration">
4      <bpmn2:participant id="P2_Forwarder" name="P2_Forwarder"
           processRef="P2_ForwarderProcess"/>
5      <bpmn2:participant id="P1_LogisticsProvider" name="
           P1_LogisticsProvider" processRef="P1_LogisticsProviderProcess"
           />
6      <bpmn2:messageFlow id="MF1" messageRef="M1_Status" name="MF1"
           sourceRef="T21_SendStatus" targetRef="E11"/>
7      <bpmn2:messageFlow id="MF2" messageRef="M2_Route" name="MF2"
           sourceRef="T15_SendRoute" targetRef="E23"/>
8    </bpmn2:collaboration>
9    <bpmn2:process id="P2_ForwarderProcess" name="P2_ForwarderProcess">
10     <bpmn2:laneSet id="LS2_ForwarderLaneSet" name="
           LS2_ForwarderLaneSet">
11       <bpmn2:lane id="L21_OBU" name="L21_OBU">
12         <bpmn2:flowNodeRef>E21</bpmn2:flowNodeRef>
13         <bpmn2:flowNodeRef>E22</bpmn2:flowNodeRef>
14         <bpmn2:flowNodeRef>T21_SendStatus</bpmn2:flowNodeRef>
15         <bpmn2:flowNodeRef>E23</bpmn2:flowNodeRef>
16         <bpmn2:flowNodeRef>E24</bpmn2:flowNodeRef>
17       </bpmn2:lane>
18       <bpmn2:lane id="L22_Driver" name="L22_Driver">
19         <bpmn2:flowNodeRef>T22_AcceptRoute</bpmn2:flowNodeRef>
20       </bpmn2:lane>
21     </bpmn2:laneSet>
22     <bpmn2:startEvent id="E21" name="E21">
23       <bpmn2:outgoing>SequenceFlow_1</bpmn2:outgoing>
24     </bpmn2:startEvent>
25     <bpmn2:sequenceFlow id="SequenceFlow_1" sourceRef="E21" targetRef
           ="T21_SendStatus"/>
26     <bpmn2:endEvent id="E22" name="E22">
27       <bpmn2:incoming>SequenceFlow_2</bpmn2:incoming>
28     </bpmn2:endEvent>
29     <bpmn2:task id="T21_SendStatus" name="T21_SendStatus">
30       <bpmn2:incoming>SequenceFlow_1</bpmn2:incoming>
31       <bpmn2:outgoing>SequenceFlow_2</bpmn2:outgoing>
32     </bpmn2:task>
33     <bpmn2:sequenceFlow id="SequenceFlow_2" sourceRef="T21_SendStatus
           " targetRef="E22"/>
34     <bpmn2:startEvent id="E23" name="E23">
```

```
35        <bpmn2:outgoing>SequenceFlow_12</bpmn2:outgoing>
36      </bpmn2:startEvent>
37      <bpmn2:sequenceFlow id="SequenceFlow_12" sourceRef="E23"
           targetRef="T22_AcceptRoute"/>
38      <bpmn2:endEvent id="E24" name="E24">
39        <bpmn2:incoming>SequenceFlow_14</bpmn2:incoming>
40      </bpmn2:endEvent>
41      <bpmn2:userTask id="T22_AcceptRoute" name="T22_AcceptRoute">
42        <bpmn2:incoming>SequenceFlow_12</bpmn2:incoming>
43        <bpmn2:outgoing>SequenceFlow_14</bpmn2:outgoing>
44      </bpmn2:userTask>
45      <bpmn2:sequenceFlow id="SequenceFlow_14" name="" sourceRef="
           T22_AcceptRoute" targetRef="E24"/>
46      <bpmn2:association id="Association_1" sourceRef="M1_Status"
           targetRef="MF1"/>
47      <bpmn2:association id="Association_3" sourceRef="M2_Route"
           targetRef="MF2"/>
48    </bpmn2:process>
49    <bpmn2:process id="P1_LogisticsProviderProcess" name="
         P1_LogisticsProviderProcess">
50      <bpmn2:laneSet id="LS1_LogisticsProviderLaneSet" name="
           LS1_LogisticsProviderLaneSet">
51        <bpmn2:lane id="L11_Dispatcher" name="L11_Dispatcher">
52          <bpmn2:flowNodeRef>T14_SelectRoute</bpmn2:flowNodeRef>
53        </bpmn2:lane>
54        <bpmn2:lane id="L12_LogisticsSystem" name="L12_LogisticsSystem"
             >
55          <bpmn2:flowNodeRef>E11</bpmn2:flowNodeRef>
56          <bpmn2:flowNodeRef>T11_CheckStatus</bpmn2:flowNodeRef>
57          <bpmn2:flowNodeRef>G12_StatusOK</bpmn2:flowNodeRef>
58          <bpmn2:flowNodeRef>T12_IdentifyShipmentsAtRisk</
               bpmn2:flowNodeRef>
59          <bpmn2:flowNodeRef>T13_CalculateRoutes</bpmn2:flowNodeRef>
60          <bpmn2:flowNodeRef>T15_SendRoute</bpmn2:flowNodeRef>
61          <bpmn2:flowNodeRef>E13</bpmn2:flowNodeRef>
62          <bpmn2:flowNodeRef>E12</bpmn2:flowNodeRef>
63        </bpmn2:lane>
64      </bpmn2:laneSet>
65      <bpmn2:startEvent id="E11" name="E11">
66        <bpmn2:outgoing>SequenceFlow_3</bpmn2:outgoing>
67      </bpmn2:startEvent>
68      <bpmn2:sequenceFlow id="SequenceFlow_3" sourceRef="E11" targetRef
           ="T11_CheckStatus"/>
69      <bpmn2:task id="T11_CheckStatus" name="T11_CheckStatus">
70        <bpmn2:incoming>SequenceFlow_3</bpmn2:incoming>
71        <bpmn2:outgoing>SequenceFlow_4</bpmn2:outgoing>
72      </bpmn2:task>
73      <bpmn2:sequenceFlow id="SequenceFlow_4" sourceRef="
           T11_CheckStatus" targetRef="G12_StatusOK"/>
74      <bpmn2:exclusiveGateway id="G12_StatusOK" name="G12_StatusOK"
           gatewayDirection="Diverging">
75        <bpmn2:incoming>SequenceFlow_4</bpmn2:incoming>
76        <bpmn2:outgoing>SequenceFlow_5</bpmn2:outgoing>
77        <bpmn2:outgoing>SequenceFlow_6</bpmn2:outgoing>
78      </bpmn2:exclusiveGateway>
79      <bpmn2:sequenceFlow id="SequenceFlow_5" sourceRef="G12_StatusOK"
           targetRef="E12"/>
80      <bpmn2:sequenceFlow id="SequenceFlow_6" sourceRef="G12_StatusOK"
           targetRef="T12_IdentifyShipmentsAtRisk"/>
81      <bpmn2:task id="T12_IdentifyShipmentsAtRisk" name="
           T12_IdentifyShipmentsAtRisk">
82        <bpmn2:incoming>SequenceFlow_6</bpmn2:incoming>
83        <bpmn2:outgoing>SequenceFlow_7</bpmn2:outgoing>
84      </bpmn2:task>
```

```
85      <bpmn2:sequenceFlow id="SequenceFlow_7" sourceRef="
            T12_IdentifyShipmentsAtRisk" targetRef="T13_CalculateRoutes"/>
86      <bpmn2:task id="T13_CalculateRoutes" name="T13_CalculateRoutes">
87        <bpmn2:incoming>SequenceFlow_7</bpmn2:incoming>
88        <bpmn2:outgoing>SequenceFlow_8</bpmn2:outgoing>
89      </bpmn2:task>
90      <bpmn2:sequenceFlow id="SequenceFlow_8" sourceRef="
            T13_CalculateRoutes" targetRef="T14_SelectRoute"/>
91      <bpmn2:task id="T15_SendRoute" name="T15_SendRoute">
92        <bpmn2:incoming>SequenceFlow_10</bpmn2:incoming>
93        <bpmn2:outgoing>SequenceFlow_11</bpmn2:outgoing>
94      </bpmn2:task>
95      <bpmn2:sequenceFlow id="SequenceFlow_11" sourceRef="T15_SendRoute
            " targetRef="E13"/>
96      <bpmn2:endEvent id="E13" name="E13">
97        <bpmn2:incoming>SequenceFlow_11</bpmn2:incoming>
98      </bpmn2:endEvent>
99      <bpmn2:userTask id="T14_SelectRoute" name="T14_SelectRoute">
100       <bpmn2:incoming>SequenceFlow_8</bpmn2:incoming>
101       <bpmn2:outgoing>SequenceFlow_10</bpmn2:outgoing>
102     </bpmn2:userTask>
103     <bpmn2:sequenceFlow id="SequenceFlow_10" sourceRef="
            T14_SelectRoute" targetRef="T15_SendRoute"/>
104     <bpmn2:endEvent id="E12" name="E12">
105       <bpmn2:incoming>SequenceFlow_5</bpmn2:incoming>
106     </bpmn2:endEvent>
107   </bpmn2:process>
108   <bpmn2:message id="M1_Status" itemRef="StatusMessageDefintion" name
          ="M1_Status"/>
109   <bpmn2:message id="M2_Route" itemRef="RouteMessageDefinition" name=
          "M2_Route"/>
110   <bpmn2:itemDefinition id="StatusMessageDefintion" structureRef="
          StatusMessageDefintion"/>
111   <bpmn2:itemDefinition id="RouteMessageDefinition" structureRef="
          RouteMessageDefinition"/>
112 </bpmn2:definitions>
```

Listing B.1: Extract from the Business Process Model (BPMN notation)

B.2. Process Model Configuration

```
1  /**
2   * Process model configuration for Replan Process case study
3   * Author: Jörn Eichler
4   */
5  model MDL_Replan language SecEML "1.1" process ReplanDefinitions
6
7  /** Package for basic definitions: this package reflects the
       information security policy of the organization */
8  package PCK_Basic (
9
10     /** Definition of confidentiality (ISO 13335-1) */
11     securityGoalClass SGC_Confidentiality (
12         description "The property that information is not made
               available or disclosed to unauthorized individuals,
               entities, or processes"
13     )
14     /** Definition of availability (ISO 13335-1) */
15     securityGoalClass SGC_Availability (
16         description "The property of being accessible and usable upon
                demand by an authorized entity"
```

```
17      )
18      /** Definition of integrity (ISO 13335-1) */
19      securityGoalClass SGC_Integrity (
20          description "The property of safeguarding the accuracy and
                completeness of assets"
21      )
22      /** Definition of non-repudiation (ISO 13335-1) */
23      securityGoalClass SGC_NonRepudiation (
24          description "The ability to prove an action or event has
                taken place, so that this event or action cannot be
                repudiated later"
25      )
26
27      /** No rating for security goal applicable (in order to define a
            non-goal explicitly) */
28      securityGoalRating SGR_NotApplicable (
29          description "No loss or damage is to be expected"
30          ordinal 0
31      )
32      /** Definition of rating "normal" (IT-BPM) */
33      securityGoalRating SGR_Normal (
34          description "The impact of any loss or damage is limited and
                calculable."
35          ordinal 1
36      )
37      /** Definition of rating "high" (IT-BPM) */
38      securityGoalRating SGR_High (
39          description "The impact of any loss or damage may be
                considerable."
40          ordinal 2
41      )
42      /** Definition of rating "very high" (IT-BPM) */
43      securityGoalRating SGR_VeryHigh (
44          description "The impact of any loss or damage may be of
                catastrophic proportions which could threaten the very
                survival of the organization."
45          ordinal 3
46      )
47
48      /** Definition of damage scenario "violation of laws, regulations
            , or contracts (IT-BPM) */
49      scenario DSC_ViolationOfLaws (
50          title "Violations of laws, regulations, or contracts"
51      )
52      /** Definition of damage scenario "impairment of the right to
            informational self-determination" */
53      scenario DSC_InformationalSelfDetermination (
54          title "Impairment of the right to informational self-
                determination"
55      )
56      /** Definition of damage scenario "physical injury" */
57      scenario DSC_PhysicalInjury (
58          title "Physical injury"
59      )
60      /** Definition of damage scenario "impaired ability to perform
            tasks at hand" */
61      scenario DSC_ImpairedPerformance (
62          title "Impaired ability to perform tasks at hand"
63      )
64      /** Definition of damage scenario "negative internal or external
            effects" */
65      scenario DSC_NegativeEffects (
66          title "Negative internal or external effects"
67      )
```

```
68      /** Definition of damage scenario "financial consequences" */
69      scenario DSC_FinancialConsequences (
70          title "Financial consequences"
71      )
72
73      /** Sole criterion for security goal rating "not applicable" */
74      criterion SCR_NoHarm (
75          title "No danger or harm considerable"
76          description "No security goal with this class existing: no
                damage or harm considerable"
77          rating SGR_NotApplicable
78          scenarios DSC_ViolationOfLaws,
                DSC_InformationalSelfDetermination, DSC_PhysicalInjury,
79               DSC_ImpairedPerformance, DSC_NegativeEffects,
                     DSC_FinancialConsequences
80      )
81      /** Criterion for violations of regulations and laws with minor
            consequences */
82      criterion SCR_VoL_MinorViolation (
83          title "Violation with minor consequences"
84          description "Violations of regulations and laws with minor
                consequences"
85          rating SGR_Normal
86          scenarios DSC_ViolationOfLaws
87      )
88      /** Criterion for minor breaches of contract which result in at
            most minor contractual penalties */
89      criterion SCR_VoL_MinorPenalties (
90          title "Violation with minor penalties"
91          description "Minor breaches of contract which result in at
                most minor contractual penalties"
92          rating SGR_Normal
93          scenarios DSC_ViolationOfLaws
94      )
95      /** Criterion for violations of regulations and laws with
            substantial consequences */
96      criterion SCR_VoL_SubstantialViolation (
97          title "Violations with substantial consequences"
98          description "Violations of regulations and laws with
                substantial consequences"
99          rating SGR_High
100         scenarios DSC_ViolationOfLaws
101     )
102     /** Criterion for major breaches of contract with high
            contractual penalties */
103     criterion SCR_VoL_SubstantialPenalties (
104         title "Violations with substantial consequences"
105         description "Major breaches of contract with high contractual
                penalties"
106         rating SGR_High
107         scenarios DSC_ViolationOfLaws
108     )
109     /** Criterion for fundamental violations of regulations and laws
            */
110     criterion SCR_VoL_FundamentalViolation (
111         title "Fundamental violations"
112         description "Fundamental violations of regulations and laws"
113         rating SGR_VeryHigh
114         scenarios DSC_ViolationOfLaws
115     )
116     // [additional criteria omitted]
117
118     /** Business asset that is not explicitly modeled in the business
            process model */
```

```
119     assetClass ASC_BusinessAsset (
120         description "Asset class for assets that are not explicitly
                modeled in the business process model"
121     )
122     /** Asset class for sequences of flow nodes (including tasks,
            events etc.) */
123     assetClass ASC_ProcessAsset (
124         description "Asset class for sequences of flow nodes (
                including tasks, events etc.)"
125         type "bpmn::Process"
126     )
127     /** Asset class for data in transit (messages) */
128     assetClass ASC_MessageAsset (
129         description "Asset class for data in transit (messages)"
130         type "bpmn::Message"
131     )
132     /** Asset class for data at rest */
133     assetClass ASC_DataAsset (
134         description "Asset class for data at rest"
135         type "bpmn::DataObject"
136     )
137
138     /** Prevention */
139     controlPurpose Prevention
140     /** Deterrence  */
141     controlPurpose Deterrence
142     /** Limitation */
143     controlPurpose Limitation
144     /** Detection */
145     controlPurpose Detection
146     /** Correction */
147     controlPurpose Correction
148     /** Recovery */
149     controlPurpose Recovery
150     /** Monitoring */
151     controlPurpose Monitoring
152
153     /** Rating for control implementations that provide only basic
            protection */
154     controlImplementationRating CIR_Weak ( ordinal 1
            ratingRestriction SGR_NotApplicable, SGR_Normal )
155     /** Rating for control implementations that provide strong
            protection */
156     controlImplementationRating CIR_Strong ( ordinal 2 )
157 )
```

Listing B.2: Extract from the Process Model Configuration (SecEML notation)

B.3. Threat Catalog

```
1  /**
2   * Threat catalog for Replan Process case study
3   * Author: Jörn Eichler
4   */
5  model MDL_Replan language SecEML "1.1" process ReplanDefinitions
6
7  /** Package for the threat classes: here STRIDE adapted for BPMN
       process models */
8  package PCK_ThreatCatalog (
9
10     import MDL_Replan.PCK_Basic.*
```

```
11
12      /** Generic threat class to cover arbitrary instances */
13      threatClass THC_GenericThreat (
14          securityGoalClasses SGC_Integrity, SGC_Availability,
                SGC_Confidentiality, SGC_NonRepudiation
15          assetClasses ASC_BusinessAsset, ASC_DataAsset,
                ASC_MessageAsset, ASC_ProcessAsset
16      )
17      /** Tamper the execution sequence of processes */
18      threatClass THC_TamperExecutionSequence (
19          securityGoalClasses SGC_Integrity
20          assetClasses ASC_ProcessAsset
21          // retrieve candidate elements for modeled assets: all
                Process entities that contain sequence flow elements
22          assetExp "bpmn::Process.allInstances()->select(flowElements->
                exists(fe | fe.oclIsTypeOf(bpmn::SequenceFlow)))"
23          // retrieve affected elements: all SequenceFlow elements of a
                 candidate modeled asset
24          matchExp "assetElements->collect(flowElements.oclIsTypeOf(
                bpmn::SequenceFlow))"
25      )
26      /** Tamper messages */
27      threatClass THC_TamperMessage (
28          securityGoalClasses SGC_Integrity
29          assetClasses ASC_MessageAsset
30          // retrieve affected elements: candidate modeled assets
31          matchExp "assetElements"
32      )
33      /** Tamper data at rest */
34      threatClass THC_TamperData (
35          securityGoalClasses SGC_Integrity
36          assetClasses ASC_DataAsset
37          // retrieve affected elements: candidate modeled assets
38          matchExp "assetElements"
39      )
40      /** Spoof participants */
41      threatClass THC_SpoofParticipant (
42          securityGoalClasses SGC_Confidentiality, SGC_Integrity,
                SGC_NonRepudiation
43          assetClasses ASC_MessageAsset
44      )
45      /** Spoof users within a process */
46      threatClass THC_SpoofUser (
47          securityGoalClasses SGC_Confidentiality, SGC_Integrity,
                SGC_NonRepudiation
48          assetClasses ASC_ProcessAsset
49      )
50      /** Repudiate messages */
51      threatClass THC_RepudiateMessage (
52          securityGoalClasses SGC_NonRepudiation
53          assetClasses ASC_MessageAsset
54          // retrieve affected elements: candidate modeled assets
55          matchExp "assetElements"
56      )
57      /** Repudiate interaction with a process instance as a user */
58      threatClass THC_RepudiateUserInteraction (
59          securityGoalClasses SGC_NonRepudiation
60          assetClasses ASC_ProcessAsset
61      )
62      /** Disclose confidential message content */
63      threatClass THC_DiscloseMessage (
64          securityGoalClasses SGC_Confidentiality
65          assetClasses ASC_MessageAsset
66          // retrieve affected elements: candidate modeled assets
67          matchExp "assetElements"
```

```
68      )
69      /** Disclose confidential process information from user tasks */
70      threatClass THC_DiscloseProcessInformation (
71          securityGoalClasses SGC_Confidentiality
72          assetClasses ASC_ProcessAsset
73      )
74      /** Disclose confidential data at rest */
75      threatClass THC_DiscloseData (
76          securityGoalClasses SGC_Confidentiality
77          assetClasses ASC_DataAsset
78      )
79      /** Denial or impede process execution */
80      threatClass THC_DenialOfProcessExecution (
81          securityGoalClasses SGC_Availability
82          assetClasses ASC_ProcessAsset
83          // retrieve candidate elements for modeled assets: all
                Process elements
84          assetExp "bpmn::Process.allInstances()"
85          // retrieve affected elements: candidate modeled assets
86          matchExp "assetElements"
87      )
88      /** Denial or impede of message exchange */
89      threatClass THC_DenialOfMessageExchange (
90          securityGoalClasses SGC_Availability
91          assetClasses ASC_MessageAsset
92      )
93      /** Elevate privileges within process execution */
94      threatClass THC_ElevatePrivileges (
95          securityGoalClasses SGC_Availability, SGC_Confidentiality,
                SGC_Integrity, SGC_NonRepudiation
96          assetClasses ASC_ProcessAsset
97          // retrieve candidate elements for modeled assets: all
                Process elements
98          assetExp "bpmn::Process.allInstances()"
99          // retrieve affected elements: candidate modeled assets
100         matchExp "assetElements"
101     )
102 )
103
104 /** Package for the requirement classes: here Common Criteria is used
        as classification schema */
105 package PCK_RequirementClasses (
106
107     import MDL_Replan.PCK_Basic.*
108     import MDL_Replan.PCK_ThreatCatalog.*
109
110     /** The system shall authenticate users before any interaction (
            CC FIA_UAU.2) */
111     securityRequirementClass SRC_UserAuthentication (
112         securityGoalClasses SGC_Confidentiality, SGC_Integrity,
                SGC_NonRepudiation
113         assetClasses ASC_ProcessAsset
114         threatClasses THC_SpoofUser, THC_RepudiateUserInteraction
115     )
116     /** The system shall enforce message transmission or reception in
            a manner protected from
117      * unauthorized disclosure (CC FDP_UCT.1)
118      */
119     securityRequirementClass SRC_MessageConfidentiality (
120         securityGoalClasses SGC_Confidentiality
121         assetClasses ASC_MessageAsset
122         threatClasses THC_DiscloseMessage, THC_SpoofParticipant
123     )
124     /** The system shall enforce message transmission or reception in
            a manner protected from
```

```
125         * modification, deletion, insertion, or replay errors (CC
                FDP_UIT.1)
126         */
127       securityRequirementClass SRC_MessageIntegrity (
128           securityGoalClasses SGC_Integrity
129           assetClasses ASC_MessageAsset
130           threatClasses THC_TamperMessage, THC_SpoofParticipant
131       )
132       /** The system shall generate an audit record (CC FAU_GEN.1) */
133       securityRequirementClass SRC_AuditGeneration (
134           securityGoalClasses SGC_Integrity, SGC_NonRepudiation
135           assetClasses ASC_ProcessAsset
136           threatClasses THC_TamperExecutionSequence
137       )
138       /** The system shall be able to prevent unauthorized
              modifications to stored data (CC FAU_STG.1, FPT_ITI.1) */
139       securityRequirementClass SRC_DataIntegrity (
140           securityGoalClasses SGC_Integrity, SGC_NonRepudiation
141           assetClasses ASC_DataAsset
142           threatClasses THC_TamperData
143       )
144       /** The system shall be able to prevent unauthorized disclosure
              of stored data (CC FAU_ITC.1) */
145       securityRequirementClass SRC_DataConfidentiality (
146           securityGoalClasses SGC_Confidentiality
147           assetClasses ASC_DataAsset
148           threatClasses THC_DiscloseData
149       )
150       /** The system shall enforce access control of users to user
              tasks (CC FDP_ACC.1, FDP_ACF.1) */
151       securityRequirementClass SRC_UserAccessControl (
152           securityGoalClasses SGC_Integrity, SGC_Confidentiality,
                  SGC_NonRepudiation
153           assetClasses ASC_ProcessAsset
154           threatClasses THC_SpoofUser, THC_DiscloseProcessInformation,
                  THC_ElevatePrivileges, THC_RepudiateUserInteraction
155       )
156       /** The system shall verify that secrets meet defined quality
              metrics (CC FIA_SOS.1) */
157       securityRequirementClass SRC_SecretQuality (
158           securityGoalClasses SGC_Integrity, SGC_Confidentiality,
                  SGC_NonRepudiation
159           assetClasses ASC_ProcessAsset, ASC_MessageAsset
160           threatClasses THC_SpoofUser, THC_SpoofParticipant
161       )
162       /** The system shall ensure the provision of minimum quantity of
              CPU, memory, and disk allocation
163        * for each task to be executed or message to be processed within
               a given time frame (CC FRU_RSA.2)
164        */
165       securityRequirementClass SRC_MinimumQuotas (
166           securityGoalClasses SGC_Availability
167           assetClasses ASC_ProcessAsset, ASC_MessageAsset
168           threatClasses THC_DenialOfMessageExchange,
                  THC_DenialOfProcessExecution
169       )
170   )
```

Listing B.3: Extract from the Threat Catalog (SecEML notation)

B.4. Control Catalog

```
/**
 * Control catalog for Replan Process case study
 * Author: Jörn Eichler
 */
model MDL_Replan language SecEML "1.1" process ReplanDefinitions

/** Package for the control classes */
package PCK_ControlClasses (

    import MDL_Replan.PCK_Basic.*
    import MDL_Replan.PCK_ThreatCatalog.*
    import MDL_Replan.PCK_RequirementClasses.*

    /** User authentication */
    controlClass COC_UserAuthentication (
        securityRequirementClasses SRC_UserAuthentication
        purposes Prevention
        // User credentials are introduced by control
        introduces ASC_DataAsset
    )
    /** User access control */
    controlClass COC_UserAccessControl (
        securityRequirementClasses SRC_UserAccessControl
        purposes Prevention
        // User privileges are introduced by control
        introduces ASC_DataAsset
        reliesOn COC_UserAuthentication
    )
    /** Access to physical resources are restricted */
    controlClass COC_PhysicalAccessControl (
        securityRequirementClasses SRC_DataConfidentiality,
            SRC_DataIntegrity
        purposes Prevention
    )
    /** Message encryption */
    controlClass COC_MessageEncryption (
        securityRequirementClasses SRC_MessageConfidentiality
        purposes Prevention
        // Keys are introduced by control
        introduces ASC_DataAsset
    )
    /** Channel protection */
    controlClass COC_ChannelProtection (
        securityRequirementClasses SRC_MessageConfidentiality,
            SRC_MessageIntegrity
        purposes Detection, Prevention
        // Keys are introduced by control
        introduces ASC_DataAsset
    )
    /** Message authentication */
    controlClass COC_MessageAuthentication (
        securityRequirementClasses SRC_MessageIntegrity
        purposes Detection
        // Keys are introduced by control
        introduces ASC_DataAsset
    )
    /** Audit generation */
    controlClass COC_AuditGeneration (
        securityRequirementClasses SRC_AuditGeneration
        purposes Detection, Monitoring
        // Audit trail is introduced by control
        introduces ASC_DataAsset
    )
    /** Encryption of data at rest */
```

```
63      controlClass COC_DataEncryption (
64          securityRequirementClasses SRC_DataConfidentiality
65          purposes Prevention
66          // Keys are introduced by control
67          introduces ASC_DataAsset
68      )
69      /** Integrity protection of data at rest */
70      controlClass COC_DataIntegrityProtection (
71          securityRequirementClasses SRC_DataIntegrity
72          purposes Detection
73          // Keys are introduced by control
74          introduces ASC_DataAsset
75      )
76      /** Verification of secret quality */
77      controlClass COC_SecretQuality (
78          securityRequirementClasses SRC_SecretQuality
79          purposes Deterrence
80      )
81      /** Provision of minimum quotas */
82      controlClass COC_MinimumQuotas (
83          securityRequirementClasses SRC_MinimumQuotas
84          purposes Limitation
85      )
86  )
```

Listing B.4: Extract from the Control Catalog (SecEML notation)

B.5. Runtime Capability Model

```
1   /**
2    * Runtime capabilities of Activiti for Replan Process case study
3    * Author: Jörn Eichler
4    */
5   model MDL_Replan language SecEML "1.1" process ReplanDefinitions
6
7   /** Package for basic definitions: this package reflects the
        information security policy of the organization */
8   package PCK_Activiti (
9
10      import MDL_Replan.PCK_Basic.*
11      import MDL_Replan.PCK_ThreatCatalog.*
12      import MDL_Replan.PCK_RequirementClasses.*
13      import MDL_Replan.PCK_ControlClasses.*
14
15      runtimeClass Activiti_57 ()
16
17      /** Logging of all task executions (start, stop, user) */
18      controlImplementationClass CIC_LogExecution (
19          runtimeClass Activiti_57
20          controlClasses COC_AuditGeneration
21          rating CIR_Weak
22          properties "logLevel"
23      )
24      /** Protected storage of keys */
25      controlImplementationClass CIC_Keystore (
26          runtimeClass Activiti_57
27          controlClasses COC_DataEncryption,
                COC_DataIntegrityProtection
28          rating CIR_Weak
29          // allowed properties :- keystore: JKS, PKCS11 or PKCS12,
                tool: keytool, openSSL or MS key-manager
```

```
30          properties "keystoreType", "keystoreLocation", "tool"
31      )
32      /** Authentication for remote users */
33      controlImplementationClass CIC_BasicHttpAuthentication (
34          runtimeClass Activiti_57
35          controlClasses COC_UserAuthentication
36          rating CIR_Weak
37      )
38      /** Channel protection applying TLS */
39      controlImplementationClass CIC_TLS (
40          runtimeClass Activiti_57
41          controlClasses COC_ChannelProtection, COC_UserAuthentication
42          rating CIR_Strong
43          reliesOn CIC_Keystore
44          properties "clientAuthentication", "serverAuthentication"
45      )
46      /** Build-in Activiti access control for user tasks */
47      controlImplementationClass CIC_UserAccessControl (
48          runtimeClass Activiti_57
49          controlClasses COC_UserAccessControl
50          rating CIR_Weak
51      )
52  )
```

Listing B.5: Extract from the Runtime Capability Model (SecEML notation)

B.6. Security Analysis Model

```
1   /**
2    * Security Analysis Model for Replan Process case study
3    * Author: Jörn Eichler
4    */
5   model MDL_Replan language SecEML "1.1" process ReplanDefinitions
6
7   /** Definition of assets and resources */
8   package PCK_Assets (
9
10      import MDL_Replan.PCK_Basic.*
11      import ReplanDefinitions.*
12
13      /** The logistics provider is the view point of the analysis */
14      stakeholder STA_LogisticsProvider (
15          title 'Logistics Provider'
16      )
17
18      /** Objective of Replan process: supporting the Proof of Delivery
            (POD) */
19      asset ASS_POD : ASC_BusinessAsset (
20          title "Proof of Delivery"
21      )
22
23      // modeled assets
24
25      /** Replan process (Pool that is operated by the logistics
            provider; perspective taken by the security analysis) */
26      asset ASS_ReplanProcess : ASC_ProcessAsset (
27          title "Replan Process"
28          supports ASS_POD
29          elements P1_LogisticsProviderProcess
30      )
31      /** Status data and routing data exchanged with forwarder */
```

```
32      asset ASS_Messages : ASC_MessageAsset (
33          title "Replan Messages"
34          supports ASS_ReplanProcess
35          elements M1_Status, M2_Route
36      )
37
38      // assets introduced by security decisions
39
40      /** Logging data */
41      asset ASS_AuditTrail : ASC_DataAsset (
42          title "Log Data"
43          // does not support assets explicitly as it is introduced by
                control
44          // does not link an process model element as it is not
                represented there
45      )
46      /** User credentials and privileges */
47      asset ASS_CredentialsPrivileges : ASC_DataAsset (
48          title "User Credentials and Privileges"
49          // does not support assets explicitly as it is introduced by
                control
50          // does not link an process model element as it is not
                represented there
51      )
52      /** Keys for secure channel establishment */
53      asset ASS_Keys : ASC_DataAsset (
54          title "Keys for Secure Message Exchange"
55          // does not support assets explicitly as it is introduced by
                control
56          // does not link an process model element as it is not
                represented there
57      )
58
59      /** Actual logistics system */
60      resource RES_LogisticsSystem (
61          supports ASS_Messages, ASS_ReplanProcess
62          elements P1_LogisticsProviderProcess.
                LS1_LogisticsProviderLaneSet.L12_LogisticsSystem
63      )
64      /** Dispatcher */
65      resource RES_Dispatcher (
66          supports ASS_ReplanProcess
67          elements P1_LogisticsProviderProcess.
                LS1_LogisticsProviderLaneSet.L11_Dispatcher
68      )
69      /** Medium for message exchange */
70      resource RES_GSM (
71          supports ASS_Messages
72          elements ReplanCollaboration.MF1, ReplanCollaboration.MF2
73      )
74      /** Forwarder as participant of the collaboration */
75      resource RES_Forwarder (
76          supports ASS_Messages
77          elements P2_ForwarderProcess
78      )
79  )
80
81  /** Package for security goals and their rating */
82  package PCK_Goals (
83
84      import MDL_Replan.PCK_Basic.*
85      import MDL_Replan.PCK_Assets.*
86
87      /** Integrity of Replan process */
88      goal SGO_ReplanIntegrity : SGC_Integrity (
```

```
89          asset ASS_ReplanProcess
90          stakeholder STA_LogisticsProvider
91          rating SGR_High
92          criteria SCR_VoL_SubstantialPenalties
93      )
94      /** Confidentiality of Replan process information */
95      goal SGO_ReplanConfidentiality : SGC_Confidentiality (
96          asset ASS_ReplanProcess
97          stakeholder STA_LogisticsProvider
98          rating SGR_Normal
99          criteria SCR_VoL_MinorViolation
100     )
101     /** Availability of Replan process */
102     goal SGO_ReplanAvailabilty : SGC_Availability (
103         asset ASS_ReplanProcess
104         stakeholder STA_LogisticsProvider
105         rating SGR_Normal
106         criteria SCR_VoL_MinorPenalties
107     )
108     /** Non-repudiation of Replan process */
109     goal SGO_ReplanNonRepudiation : SGC_NonRepudiation (
110         asset ASS_ReplanProcess
111         stakeholder STA_LogisticsProvider
112         rating SGR_High
113         criteria SCR_VoL_SubstantialPenalties
114     )
115     /** Confidentiality of message data */
116     goal SGO_MessageConfidentiality : SGC_Confidentiality (
117         asset ASS_Messages
118         // no individual rating, only derivation
119     )
120     /** Integrity of message */
121     goal SGO_MessageIntegrity : SGC_Integrity (
122         asset ASS_Messages
123         // no individual rating, only derivation
124     )
125     /** Availability of message / message exchange */
126     goal SGO_MessageAvailability : SGC_Availability (
127         asset ASS_Messages
128         // no individual rating, only derivation
129     )
130     /** Non-repudiation of message transmission */
131     goal SGO_MessageNonRepudiation : SGC_NonRepudiation (
132         asset ASS_Messages
133         // no individual rating, only derivation
134     )
135     /** Integrity of log data */
136     goal SGO_AuditTrailIntegrity : SGC_Integrity (
137         asset ASS_AuditTrail
138         // no individual rating, only derivation
139     )
140     /** Confidentiality of log data */
141     goal SGO_AuditTrailConfidentiality : SGC_Confidentiality (
142         asset ASS_AuditTrail
143         // no individual rating, only derivation
144     )
145     /** Integrity of credentials and privileges */
146     goal SGO_CredentialIntegrity : SGC_Integrity (
147         asset ASS_CredentialsPrivileges
148         // no individual rating, only derivation
149     )
150     /** Confidentiality of credentials and privileges */
151     goal SGO_CredentialConfidentiality : SGC_Confidentiality (
152         asset ASS_CredentialsPrivileges
153         // no individual rating, only derivation
```

```
154     )
155     /** Integrity of keys */
156     goal SGO_KeysIntegrity : SGC_Integrity (
157         asset ASS_Keys
158         // no individual rating, only derivation
159     )
160     /** Confidentiality of keys */
161     goal SGO_KeysConfidentiality : SGC_Confidentiality (
162         asset ASS_Keys
163         // no individual rating, only derivation
164     )
165 )
166
167 /** Package for relevant threats */
168 package PCK_Threats (
169
170     import MDL_Replan.PCK_Basic.*
171     import MDL_Replan.PCK_ThreatCatalog.*
172     import MDL_Replan.PCK_Goals.*
173
174     /** Tampering of the execution sequence of the Replan process */
175     threat THR_TamperReplanProcess : THC_TamperExecutionSequence (
176         goals SGO_ReplanIntegrity
177     )
178     /** Tampering of the messages */
179     threat THR_TamperMessages : THC_TamperMessage (
180         goals SGO_MessageIntegrity
181     )
182     /** Tamper of the log data */
183     threat THR_TamperData : THC_TamperData (
184         goals SGO_AuditTrailIntegrity, SGO_CredentialIntegrity,
              SGO_KeysIntegrity
185     )
186     /** Spoofing of the forwarder or the logistics provider */
187     threat THR_SpoofParticipant : THC_SpoofParticipant (
188         goals SGO_MessageConfidentiality, SGO_MessageIntegrity,
              SGO_MessageNonRepudiation
189     )
190     /** Spoofing of the dispatcher  */
191     threat THR_SpoofDispatcher : THC_SpoofUser (
192         goals SGO_ReplanConfidentiality, SGO_ReplanIntegrity,
              SGO_ReplanNonRepudiation
193     )
194     /** Repudiate message transmission */
195     threat THR_RepudiateMessages : THC_RepudiateMessage (
196         goals SGO_MessageNonRepudiation
197     )
198     /** Repudiate route selection by dispatcher */
199     threat THR_RepudiateRouteSelection : THC_RepudiateUserInteraction
          (
200         goals SGO_ReplanNonRepudiation
201     )
202     /** Disclose message data */
203     threat THR_DiscloseMessages : THC_DiscloseMessage (
204         goals SGO_MessageConfidentiality
205     )
206     /** Disclose log data */
207     threat THR_DiscloseData : THC_DiscloseData (
208         goals SGO_AuditTrailConfidentiality,
              SGO_CredentialConfidentiality, SGO_KeysConfidentiality
209     )
210     /** Disclose route data via user interaction */
211     threat THR_DiscloseRouteData : THC_DiscloseProcessInformation (
212         goals SGO_ReplanConfidentiality
213     )
```

```
214     /** Impede Replan Process execution */
215     threat THR_ImpedeReplanExecution : THC_DenialOfProcessExecution (
216         goals SGO_ReplanAvailabilty
217     )
218     /** Impede message Transmission */
219     threat THR_ImpedeMesssageTransmission :
            THC_DenialOfMessageExchange (
220         goals SGO_MessageAvailability
221     )
222     /** Elevate privileges in Replan Process */
223     threat THR_ElevatePrivilegesInReplan : THC_ElevatePrivileges (
224         goals SGO_ReplanAvailabilty, SGO_ReplanConfidentiality,
                SGO_ReplanIntegrity, SGO_ReplanNonRepudiation
225     )
226 )
227
228 /** Package for requirements */
229 package PCK_Requirements (
230
231     import MDL_Replan.PCK_Basic.*
232     import MDL_Replan.PCK_RequirementClasses.*
233     import MDL_Replan.PCK_Goals.*
234     import MDL_Replan.PCK_Threats.*
235
236     /** Authenticate users before any interaction */
237     securityRequirement SRE_UserAuthentication :
            SRC_UserAuthentication (
238         goals SGO_ReplanConfidentiality, SGO_ReplanIntegrity,
                SGO_ReplanNonRepudiation
239         threats THR_SpoofDispatcher, THR_RepudiateRouteSelection
240     )
241     /** Enforce message transmission without modification, deletion,
            insertion, or replay */
242     securityRequirement SRE_MessageIntegrity : SRC_MessageIntegrity (
243         goals SGO_MessageIntegrity
244         threats THR_TamperMessages, THR_SpoofParticipant
245     )
246     /** Enforce message transmission without unauthorized disclosure
            */
247     securityRequirement SRE_MessageConfidentiality :
            SRC_MessageConfidentiality (
248         goals SGO_MessageConfidentiality
249         threats THR_DiscloseMessages, THR_SpoofParticipant
250     )
251     /** Generate audit record */
252     securityRequirement SRE_AuditGeneration : SRC_AuditGeneration (
253         goals SGO_ReplanIntegrity, SGO_ReplanNonRepudiation
254         threats THR_TamperReplanProcess
255     )
256     /** Prevent unauthorized modification of log data */
257     securityRequirement SRE_DataIntegrity : SRC_DataIntegrity (
258         goals SGO_AuditTrailIntegrity, SGO_CredentialIntegrity,
                SGO_KeysIntegrity
259         threats THR_TamperData
260     )
261     /** Prevent unauthorized disclosure of log data */
262     securityRequirement SRE_DataConfidentiality :
            SRC_DataConfidentiality (
263         goals SGO_AuditTrailConfidentiality,
                SGO_CredentialConfidentiality, SGO_KeysConfidentiality
264         threats THR_DiscloseData
265     )
266     /** Enforce access control for user tasks */
267     securityRequirement SRE_UserTaskAccessControl :
            SRC_UserAccessControl (
```

```
268         goals SGO_ReplanConfidentiality, SGO_ReplanIntegrity,
                SGO_ReplanNonRepudiation
269         threats THR_SpoofDispatcher, THR_DiscloseRouteData,
                THR_RepudiateRouteSelection
270     )
271     /** Provide minimum resources for every task to be executed */
272     securityRequirement SRE_MinimumTaskResources : SRC_MinimumQuotas
            (
273         goals SGO_MessageAvailability, SGO_ReplanAvailabilty
274         threats THR_ImpedeMesssageTransmission,
                THR_ImpedeReplanExecution
275     )
276 )
```

Listing B.6: Extract from the Security Analysis Model (SecEML notation)

B.7. Security Design Model

```
1  /**
2   * Security Design Model for Replan Process case study
3   * Author: Jörn Eichler
4   */
5  model MDL_Replan language SecEML "1.1" process ReplanDefinitions
6
7  /** Definition of controls */
8  package PCK_Controls (
9
10     import MDL_Replan.PCK_ControlClasses.*
11     import MDL_Replan.PCK_Assets.*
12     import MDL_Replan.PCK_Requirements.*
13
14     /** Authentication for user tasks */
15     control CON_UserAuthentication : COC_UserAuthentication (
16         securityRequirements SRE_UserAuthentication
17         introduces ASS_CredentialsPrivileges
18     )
19     /** Access control for user tasks */
20     control CON_UserTaskAccessControl : COC_UserAccessControl (
21         securityRequirements SRE_UserTaskAccessControl
22         introduces ASS_CredentialsPrivileges
23         reliesOn CON_UserAuthentication
24     )
25     /** Channel protection for message exchange */
26     control CON_MessageProtection : COC_ChannelProtection (
27         securityRequirements SRE_MessageConfidentiality,
               SRE_MessageIntegrity
28         introduces ASS_Keys
29     )
30     /** Generate audit for process execution */
31     control CON_AuditGeneration : COC_AuditGeneration (
32         securityRequirements SRE_AuditGeneration
33         introduces ASS_AuditTrail
34     )
35
36     /** Data integrity and confidentiality provided by physical
           access control */
37     assumption ASU_PhysicalAccessControl : COC_PhysicalAccessControl
           (
38         securityRequirements SRE_DataConfidentiality,
               SRE_DataIntegrity
39     )
```

```
)

/** Package for control implementations in Activiti */
package PCK_ActivitiRuntime (

    import MDL_Replan.PCK_ControlClasses.*
    import MDL_Replan.PCK_Activiti.*
    import MDL_Replan.PCK_Controls.*

    runtime RUN_Activiti : Activiti_57

    /** Logging of the execution trace */
    controlImplementation CIM_LogExecution : CIC_LogExecution (
        runtime RUN_Activiti
        controls CON_AuditGeneration
        properties ( "logLevel" = "fine" )
    )
    /** Application of TLS to protect message exchange */
    controlImplementation CIM_TLS : CIC_TLS (
        runtime RUN_Activiti
        controls CON_MessageProtection
        properties
            ( "clientAuthentication" = "true "),
            ( "serverAuthentication" = "true" )
    )

    controlImplementation CIM_UserAccessControl :
       CIC_UserAccessControl (
        runtime RUN_Activiti
        controls CON_UserTaskAccessControl
    )

)
```

Listing B.7: Extract from the Security Design Model (SecEML notation)

List of Figures

List of Tables

List of Listings

Acronyms

ASM	Abstract State Machine
AST	abstract syntax tree
AUK	Activity Use Kind
AURUM	Automated Risk and Utility Management
BPM	business process management
BPMN	Business Process Modeling and Notation
BPMS	business process management system
BPSec	Business Process Security
BSI	Bundesamt für Sicherheit in der Informationstechnik
CAME	computer aided method engineering
CC	Common Criteria
CIM	computational independent model
DSL	domain-specific language
DSML	domain-specific modeling language
EBNF	Extended Backus-Naur Form
EJB	Enterprise Java Bean
EMF	Eclipse Modeling Framework
EMOF	Essential Meta Object Facility
EPC	Event-driven Process Chains
EPF	Eclipse Process Framework
EPFC	Eclipse Process Framework Composer
ESB	enterprise service bus
FMC	Fundamental Modeling Concepts
FML	Formal Methods and Modeling Language Framework
FQN	fully qualified name
HUTN	Human-usable Textual Notation

IATA	International Air Transport Association
ICT	information and communications technology
IDE	integrated development environment
IEEE	Institute of Electrical and Electronics Engineers
ISM	implementation specific model
ISO	International Organization for Standardization
IT	information technology
IT-BPM	IT Baseline Protection Methodology
M-BPSec	Method for Business Process Security
MBPMN	MockBPMN
MDA	Model-driven Architecture
MDE	model-driven engineering
MDS	model-driven security
ME	method engineering
MEL	Model Extension Language
MOF	Meta Object Facility
MoSS	Modeling Security Semantics
MoSSBP	Modeling Security Semantics of Business Processes
MS SDL	Microsoft Secure Development Lifecycle
NIST	National Institute of Standards and Technology
OBU	on-board unit
OCL	Object Constraint Language
OMG	Object Management Group
OpenUP	Open Unified Process
PIM	platform independent model
POD	proof of delivery
POSeM	Process Oriented Security Model
ProMeLa	Protocol Meta Language
PSM	platform specific model
RBAC	role based access control
ROPE	Risk-Oriented Process Evaluation
RUP	Rational Unified Process
SecEML	Security Engineering Modeling Language

SecEPM	Security Engineering Process Model
SEPL	Security Enhanced Process Language
SFR	security functional requirement
SLA	service level agreement
SME	small and medium-sized enterprise
SPEM	Software & System Process Engineering Model
STRIDE	Spoofing, Tempering, Repudiation, Information disclosure, Denial of service, Elevation of privileges
TLS	transport layer security
UML	Unified Modeling Language
VE	variability element
WAN	wide area network
WFM	workflow management
WfMC	Workflow Management Coalition
WPDL	Workflow Process Definition Language
WPDR	Work Product Definition Relationship
WS-BPEL	Web Services Business Process Execution Language
XACML	Extensible Access Control Markup Language
XPDL	XML Process Definition Language

Bibliography

[AB05] N. Arni-Bloch. Towards a CAME tools for situational method engineering. In *Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA 2005)*, pages 45–50, 2005.

[ACPP11] W. Arsac, L. Compagna, G. Pellegrino, and S. Ponta. Security validation of business processes via model-checking. In *Engineering Secure Software and Systems (ESSoS 2011)*, volume 6542 of *LNCS*, pages 29–42. Springer, 2011.

[AGMP12] A. Armando, E. Giunchiglia, M. Maratea, and S. E. Ponta. An action-based approach to the formal specification and automatic analysis of business processes under authorization constraints. *Journal of Computer and System Sciences*, 78(1):119–141, 2012.

[AHB07] M. Alam, M. Hafner, and R. Breu. Model-driven security engineering for trust management in SECTET. *Journal of Software*, 2(1):47–59, 2007.

[Ame04] American National Standards Institute. ANSI INCITS 359-2004: Role based access control, 2004.

[And08] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2nd edition, 2008.

[AS04] R. S. Aguilar-Savén. Business process modelling: Review and framework. *International Journal of Production Economics*, 90(2):129–149, 2004.

[AS08] S. Ardi and N. Shahmehri. Integrating a security plug-in with the OpenUP/Basic Development Process. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES 2008)*, pages 284–291. IEEE, 2008.

[AW04] M. Andrews and J. Whittaker. Computer security. *IEEE Security & Privacy*, 2(5):68–71, 2004.

[AW08] V. Atluri and J. Warner. Security for workflow systems. In *Handbook of Database Security*, pages 213–230. Springer, 2008.

[ÅWK+03] P. Ågerfalk, K. Wistrand, F. Karlsson, G. Börjesson, M. Elmberg, and K. Möller. Flexible processes and method configuration: Outline of a joint industry-academia research project. In *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003)*, volume 3, pages 185–190, 2003.

[Bé06] J. Bézivin. Model driven engineering: An emerging technical space. In *Generative and Transformational Techniques in Software Engineering*, volume 4143 of *LNCS*, pages 36–64. Springer, 2006.

[Bas93] R. Baskerville. Information systems security design methods: implications for information systems development. *ACM Computing Surveys*, 25(4):375–414, 1993.

[BB08] E. W. N. Bernroider and M. Bernroider. A comparative study of business process management tools based on open source software and a commercial reference. In *Proceedings of the 5th International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2008)*, 2008.

[BBDF+06] J. Bézivin, S. Bouzitouna, M. Del Fabro, M. Gervais, F. Jouault, D. Kolovos, I. Kurtev, and R. Paige. A canonical scheme for model composition. In *Proceedings of the 2nd European Conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA 2006)*, volume 4066 of *LNCS*, pages 346–360. Springer, 2006.

[BBH+03] R. Breu, K. Burger, M. Hafner, J. Jürjens, G. Popp, G. Wimmel, and V. Lotz. Key issues of a formally based process model for security engineering. In *Proceedings of the 16th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2003)*, 2003.

[BBI+04] G. Booch, A. Brown, S. Iyengar, J. Rumbaugh, and B. Selic. An MDA manifesto. *MDA Journal*, 5:2–9, 2004.

[BBT11] Y. Badr, F. Biennier, and S. Tata. The integration of corporate security strategies in collaborative business processes. *IEEE Transactions on Services Computing*, 4(3):243–254, 2011.

[BCDE09] D. Basin, M. Clavel, J. Doser, and M. Egea. Automated analysis of security-design models. *Information and Software Technology*, 51(5):815–831, 2009.

[BCE11] D. Basin, M. Clavel, and M. Egea. A decade of model-driven security. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies (SACMAT 2011)*, pages 1–10. ACM, 2011.

[BCH10] M. Brusó, K. Chatzikokolakis, and J. Hartog. Formal verification of privacy for RFID systems. In *23rd IEEE Computer Security Foundations Symposium (CSF 2010)*, pages 75–88. IEEE, 2010.

[BD09] B. Bruegge and A. H. Dutoit. *Object-Oriented Software Engineering*. Pearson, 3rd edition, 2009.

[BDL06] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From UML models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39–91, 2006.

[Bez03] K. Beznosov. eXtreme security engineering: On employing XP practices to achieve "good enough security" without defining it. In *Proceedings of the 1st ACM Workshop on Business Driven Security Engineering (BizSec 2003)*, 2003.

[BFH+10] M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu. Seamless model-based development: From isolated tools to integrated model engineering environments. *Proceedings of the IEEE*, 98(4):526–545, 2010.

[BH04] B. Blakley and C. Heath. Security design patterns. Technical Guide G031, Open Group, 2004.

[BH13] A. D. Brucker and I. Hang. Secure and compliant implementation of business process-driven systems. In *Business Process Management Workshops (BPM 2013)*, volume 132 of *LNBIP*, pages 662–674. Springer, 2013.

[BHIOW08] R. Breu, M. Hafner, F. Innerhofer-Oberperfler, and F. Wozak. Model-driven security engineering of service oriented systems. In *Information Systems and e-Business Technologies (UNISCON 2008)*, volume 5 of *LNBIP*, pages 59–71. Springer, 2008.

[Bis02] M. Bishop. *Computer security: Art and science*. Addison-Wesley, 2002.

[BL73] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report 2547, MITRE, 1973.

[BM11] J. Bau and J. Mitchell. Security modeling and analysis. *IEEE Security & Privacy*, 9(3):18–25, 2011.

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. Security in business process engineering. In *Business Process Management (BPM 2003)*, volume 2678 of *LNCS*, pages 1019–1019. Springer, 2003.

[Bri96] S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280, 1996.

[BSH98] S. Brinkkemper, M. Saeki, and F. Harmsen. Assembly techniques for method engineering. In *Advanced Information Systems Engineering (CAiSE 1998)*, volume 1413 of *LNCS*, pages 381–400. Springer, 1998.

[Bun08] Bundesamt für Sicherheit in der Informationstechnik. BSI-Standard 100-2: IT-Grundschutz methodology, 2008.

[Cap12] Capgemini. Global business process management report, 2012.

[CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *Computer Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, pages 359–364. Springer, 2002.

[CD11] M. Cheung and A. Dayley. Market share analysis: Operating system software, worldwide, 2010. Technical Report G00212436, Gartner, 2011.

[CdSBE08] M. Clavel, V. da Silva, C. Braga, and M. Egea. Model-driven security in practice: An industrial experience. In *Model Driven Architecture – Foundations and Applications (ECMDA-FA 2008)*, volume 5095 of *LNCS*, pages 326–337. Springer, 2008.

[Cha95] B. Chadha. A model driven methodology for business process engineering. In *Proceedings of the Computers in Engineering Conference*, pages 1165–1182. ASME, 1995.

[CKE+10] E. W. Cope, J. M. Kuster, D. Etzweiler, L. A. Deleris, and B. Ray. Incorporating risk into business process models. *IBM Journal of Research and Development*, 54(3):4:1–4:13, 2010.

[CPG05] H. Chivers, R. Paige, and X. Ge. Agile security using an incremental security architecture. In *Extreme Programming and Agile Processes in Software Engineering (XP 2005)*, volume 3556 of *LNCS*, pages 57–65. Springer, 2005.

[CZ08] J. Cabot and N. Zannone. Towards an Integrated Framework for Model-driven Security Engineering. In *Proceedings of the 1st Modeling Security Workshop (MODSEC 2008)*, volume 413. CEUR, 2008.

[CZM+11] I. Ciuciu, G. Zhao, J. Mülle, S. Stackelberg, C. Vasquez, T. Haberecht, R. Meersman, and K. Böhm. Semantic support for security-annotated business process models. In *Enterprise, Business-Process and Information Systems Modeling (BPMDS 2011)*, volume 81 of *LNBIP*, pages 284–298. Springer, 2011.

[Dav05] T. Davenport. The coming commoditization of processes. *Harvard Business Review*, 83(6):100–108, 2005.

[dBDG+03] F. den Braber, T. Dimitrakos, B. A. Gran, M. S. Lund, K. Stølen, and J. Ø. Aagedal. The CORAS methodology: model-based risk assessment using UML and UP. In *UML and the Unified Process*, pages 332–357. IRM Press, 2003.

[dBHL+07] F. den Braber, I. Hogganvik, M. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps – a guided tour to the CORAS method. *BT Technology Journal*, 25(1):101–117, 2007.

[Den76] D. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[Dhi11] D. Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy*, 9(4):41–47, 2011.

[DRGRdGP09] A. Delgado, F. Ruiz, I. García-Rodríguez de Guzmán, and M. Piattini. Towards a service-oriented and model-driven framework with business processes as

first-class citizens. In *Business Process, Services Computing and Intelligent Service Management (BPSC 2009)*, volume 147 of *LNI*. GI, 2009.

[DRMSR06] M. Diallo, J. Romero-Mariona, S. Sim, and D. Richardson. A comparative evaluation of three approaches to specifying security requirements. In *Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ 2006)*, 2006.

[DRR+02] T. Dimitrakos, B. Ritchie, D. Raptis, J. Ø. Aagedal, F. den Braber, K. Stølen, and S. H. Houmb. Integrating model-based security risk management into eBusiness systems development: The CORAS approach. In *Proceedings of the IFIP Conference on Towards The Knowledge Society: E-Commerce, E-Business, E-Government (I3E 2002)*, pages 159–175. Kluwer, 2002.

[DVG10] R. Dijkman and P. Van Gorp. BPMN 2.0 execution semantics formalized as graph rewrite rules. In *Business Process Modelling Notation (BPMN 2010)*, volume 67 of *LNBIP*, pages 16–30. Springer, 2010.

[Eck14] C. Eckert. *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. De Gruyter Oldenbourg, 9th edition, 2014.

[EFL12] J. Eichler, A. Fuchs, and N. Lincke. Supporting security engineering at design time with adequate tooling. In *Proceedings of the 15th IEEE International Conference on Computational Science and Engineering (CSE 2012)*, pages 194–201. IEEE, 2012.

[EFN09] A. Ekelhart, S. Fenz, and T. Neubauer. AURUM: A framework for supporting information security risk management. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS 2009)*, pages 1–10. IEEE, 2009.

[Eic10] J. Eichler. Sicherheitsmodellierung dynamischer Geschäftsprozesse. Deliverable D.G7.2 im Forschungsprojekt ADiWa (BMBF Förderkennzeichen 01IA08006), 2010.

[Eic11a] J. Eichler. Lightweight modeling and analysis of security concepts. In *Engineering Secure Software and Systems (ESSoS 2011)*, volume 6542 of *LNCS*, pages 128–141. Springer, 2011.

[Eic11b] J. Eichler. Modellgetriebener IT-Grundschutz: Erstellung und Analyse von IT-Sicherheitskonzeptionen in offenen Werkzeugketten. In *Sicher in die digitale Welt von morgen – Tagungsband zum 12. Deutschen IT-Sicherheitskongress*, pages 11–22. Bundesamt für Sicherheit in der Informationstechnik, 2011.

[Eic12a] J. Eichler. SecEPM: A security engineering process model for electronic business processes. In *Proceedings of the 9th IEEE International Conference on e-Business Engineering (ICEBE 2012)*, pages 206–213. IEEE, 2012.

[Eic12b] J. Eichler. Towards a security engineering process model for electronic business processes. In *Fast Abstracts & Student Forum Proceedings of the 9th European Dependable Computing Conference (EDCC 2012)*, number 1204.4428v1. CoRR, 2012.

[EM97] C. Eckert and D. Marek. Developing secure applications: A systematic approach. In *Proceedings of the IFIP TC11 13th International Conference on Information Security in Research and Business (SEC 1997)*, pages 267–279. Chapman & Hall, 1997.

[Eur11] European Network and Information Security Agency. Secure software engineering initiatives. Technical report, 2011.

[EW05] S. Evans and J. Wallner. Risk-based security engineering through the eyes of the adversary. In *Proceedings from the 6th Annual IEEE SMC Information Assurance Workshop (IAW 2005)*, pages 158–165. IEEE, 2005.

[EY07] G. Elahi and E. Yu. A goal oriented approach for modeling and analyzing security trade-offs. In *Conceptual Modeling (ER 2007)*, volume 4801 of *LNCS*, pages 375–390. Springer, 2007.

[EYZ10] G. Elahi, E. Yu, and N. Zannone. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requirements Engineering*, 15(1):41–62, 2010.

[FGH+10] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt. A comparison of security requirements engineering methods. *Requirements Engineering*, 15(1):7–40, 2010.

[FHS02] D. G. Firesmith and B. Henderson-Sellers. *The OPEN process framework: An introduction*. Addison-Wesley, 2002.

[FMS07] G. Frankova, F. Massacci, and M. Seguran. From early requirements analysis towards secure workflows. In *Trust Management (IFIP-TM 2007)*, volume 238 of *IFIP*, pages 407–410. Springer, 2007.

[For12] Forrester Research. The software security risk report. Technical Report 1-HMGX0Z, 2012.

[FR07] R. France and B. Rumpe. Model-driven development of complex software: A research roadmap. In *Proceedings of the Workshop on the Future of Software Engineering (FOSE 2007)*, pages 37–54. IEEE, 2007.

[FS11] M. Fleming and J. Silverstein. Worldwide business process management and middleware 2010 vendor shares. Technical Report 228317, IDC, 2011.

[FSG+11] G. Frankova, M. Séguran, F. Gilcher, S. Trabelsi, J. Dörflinger, and M. Aiello. Deriving business processes with service level agreements from early requirements. *Journal of Systems and Software*, 84(8):1351–1363, 2011.

[Gö11] M. Götz. BPM-Systeme im Vergleich, 2011. iTransparent.

[Gee10] D. Geer. Are companies actually using secure development life cycles? *Computer*, 43(6):12–16, 2010.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: Elements of reusable object-oriented design*. Addison-Wesley, 1995.

[GKR+07] H. Grönniger, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel. Text-based modeling. In *Proceedings of the 4th International Workshop on Software Language Engineering (ATEM 2007)*, 2007.

[Gur95] Y. Gurevich. *Specification and validation methods*, chapter Evolving algebras 1993: Lipari guide, pages 9–36. Oxford University Press, 1995.

[Gus08] B. Gustafsson. OpenUP – the best of two worlds. *Methods & Tools*, 16(1):21–32, 2008.

[Ham13] S. Hameed. Model-driven, secure configuration of runtime environments for electronic processes. Master's thesis, Fachhochschule Kiel, 2013.

[HB09] M. Hafner and R. Breu. *Security engineering for service-oriented architectures*. Springer, 2009.

[HBAN06] M. Hafner, R. Breu, B. Agreiter, and A. Nowak. SECTET: an extensible framework for the realization of secure inter-organizational workflows. *Internet Research*, 16(5):491–506, 2006.

[HH06] P. Herrmann and G. Herrmann. Security requirement analysis of business processes. *Electronic Commerce Research*, 6(3):305–335, 2006.

[HHJS11] D. Hatebur, M. Heisel, J. Jürjens, and H. Schmidt. Systematic development of UMLsec design models based on security requirements. In *Fundamental Approaches to Software Engineering (FASE 2011)*, volume 6603 of *LNCS*, pages 232–246. Springer, 2011.

[HHS07] D. Hatebur, M. Heisel, and H. Schmidt. A security engineering process based on patterns. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA 2007)*, pages 734–738. IEEE, 2007.

[HK01] P. Herrmann and H. Krumm. Object-oriented security analysis and modeling. In *Proceedings of the 9th International Conference on Telecommunication Systems – Modeling and Analysis (ICTSM 2001)*, pages 21–32. IFIP, 2001.

[HK03] P. Hung and K. Karlapalem. A secure workflow model. In *Proceedings of the Australasian Information Security Workshop (AISW 2003)*, pages 33–41. Australian Computer Society, 2003.

[HMPR04] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

[Hol97] G. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[HP99] G. Herrmann and G. Pernul. Viewing business-process security from different perspectives. *International Journal of Electronic Commerce*, 3(3):89–103, 1999.

[HS06] B. Henderson-Sellers. Method engineering: Theory and practice. In *Information Systems Technology and its Applications (ISTA 2006)*, volume 84 of *LNI*, pages 13–23. GI, 2006.

[HSR10] B. Henderson-Sellers and J. Ralyté. Situational method engineering: state-of-the-art review. *Journal of Universal Computer Science*, 16(3):424–478, 2010.

[Ins90] Institute of Electrical and Electronics Engineers. IEEE standard glossary of software engineering terminology (std. 610.12-1990), 1990.

[Ins97] Institute of Electrical and Electronics Engineers. IEEE standard for developing software life cycle processes (std. 1074-1997), 1997.

[Int96] International Organization for Standardization. ISO/IEC 14977: Information technology – syntactic metalanguage – extended BNF, 1996.

[Int04] International Organization for Standardization. ISO/IEC 13335-1: Information technology – security techniques – management of information and communications technology security – part 1: Concepts and models for information and communications technology security management, 2004.

[Int07] International Organization for Standardization. ISO/IEC 24744: Software engineering – metamodel for development methodologies, 2007.

[Int11a] International Organization for Standardization. ISO/IEC 15408-2: Information technology – security techniques – evaluation criteria for IT security – part 1: Security functional components, 2011.

[Int11b] International Organization for Standardization. ISO/IEC 27005: Information technology – security techniques – information security risk management, 2011.

[Int13a] International Organization for Standardization. ISO/IEC 27001: Information technology – security techniques – information security management systems – requirements, 2013.

[Int13b] International Organization for Standardization. ISO/IEC 27002: Information technology – security techniques – code of practice for information security controls, 2013.

[Int14] International Organization for Standardization. ISO/IEC 15408-1: Information technology – security techniques – evaluation criteria for IT security – part 1: Introduction and general model, 2014.

[IRRG09] M. Indulska, J. Recker, M. Rosemann, and P. Green. Business process modeling: Current issues and future challenges. In *Advanced Information Systems Engineering (CAiSE 2009)*, volume 5565 of *LNCS*, pages 501–514. Springer, 2009.

[Jü01] J. Jürjens. Developing secure systems with UMLsec – from business processes to implementation. In *Verlässliche IT-Systeme 2001: Sicherheit in komplexen IT-Infrastrukturen (VIS 2001)*, pages 151–161. Vieweg, 2001.

[Jü05] J. Jürjens. *Secure Systems Development with UML*. Springer, 2005.

[Jü09] J. Jürjens. Security and dependability engineering. In *Security and Dependability for Ambient Intelligence*, chapter 2, pages 21–36. Springer, 2009.

[JB06] S. Johnson and A. Brown. A model-driven development approach to creating service-oriented solutions. In *Service-Oriented Computing (ICSOC 2006)*, volume 4294 of *LNCS*, pages 624–636. Springer, 2006.

[JJ11] J. Jensen and M. Jaatun. Not ready for prime time: A survey on security in model driven development. *International Journal of Secure Software Engineering*, 2(4):49–61, 2011.

[JM05] K. Jayaram and A. Mathur. Software engineering for secure software – state of the art: A survey. Technical Report 2005-67, Department of Computer Science, Purdue University, USA, 2005.

[Joh10] R. G. Johnston. Being vulnerable to the threat of confusing threats with vulnerabilities. *The Journal of Physical Security*, 4(2):30–34, 2010.

[JT09] S. Jakoubi and S. Tjoa. A reference model for risk-aware business process management. In *Proceedings of the 4th International Conference on Risks and Security of Internet and Systems (CRiSIS 2009)*, pages 82–89. IEEE, 2009.

[JTGK10] S. Jakoubi, S. Tjoa, S. Goluch, and G. Kitzler. Risk-aware business process management – establishing the link between business and security. In *Complex Intelligent Systems and Their Applications*, volume 41 of *Springer Optimization and its Applications*, pages 109–135. Springer, 2010.

[JTGQ09] S. Jakoubi, S. Tjoa, G. Goluch, and G. Quirchmayr. A survey of scientific approaches considering the integration of security and risk aspects into business process management. In *Proceedings of the 20th International Workshop on*

Database and Expert Systems Application (DEXA 2009), pages 127–132. IEEE, 2009.

[JTQ07] S. Jakoubi, S. Tjoa, and G. Quirchmayr. ROPE: A methodology for enabling the risk-aware modelling and simulation of business processes. In *Proceedings of the 15th European Conference on Information Systems (ECIS 2007)*, 2007.

[KÅ04] F. Karlsson and P. J. Ågerfalk. Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, 46(9):619–633, 2004.

[KÅ09] F. Karlsson and P. Ågerfalk. Exploring agile values in method configuration. *European Journal of Information Systems*, 18(4):300–316, 2009.

[KDK00] S. Kokolakis, A. Demopoulos, and E. Kiountouzis. The use of business process modelling in information systems security analysis and design. *Information Management and Computer Security*, 8(2/3):107–115, 2000.

[Ken02] S. Kent. Model driven engineering. In *Integrated Formal Methods (IFM 2002)*, volume 2335 of *LNCS*, pages 286–298. Springer, 2002.

[KGT05] A. Knöpfel, B. Gröne, and P. Tabeling. *Fundamental modeling concepts*. Wiley, 2005.

[KH10] K. Khanmohammadi and S. Houmb. Business process-based information security risk assessment. In *Proceedings of the 4th International Conference on Network and System Security (NSS 2010)*, pages 199–206, 2010.

[KKP+09] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel. Design guidelines for domain specific languages. In *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM 2009)*, 2009.

[Kle08] A. Kleppe. *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Addison-Wesley, 2008.

[KLL09] R. Ko, S. Lee, and E. Lee. Business process management (BPM) standards: a survey. *Business Process Management Journal*, 15(5):744–791, 2009.

[Krc10] H. Krcmar. *Informationsmanagement*. Springer, 5th edition, 2010.

[KS06] F. Keblawi and D. Sullivan. Applying the common criteria in systems engineering. *IEEE Security & Privacy*, 4:50–55, 2006.

[KSS12] E. Kiesling, C. Strausss, and C. Stummer. A multi-objective decision support framework for simulation-based security control selection. In *Proceedings of the 7th International Conference on Availability, Reliability and Security (ARES 2012)*, pages 454–462. IEEE, 2012.

[Lan01] C. E. Landwehr. Computer security. *International Journal of Information Security*, 1:3–13, 2001.

[LLL02] Y. Lee, J. Lee, and Z. Lee. Integrating software lifecycle process standards with security engineering. *Computers & Security*, 21(4):345–355, 2002.

[LMV+05] J. Lopez, J. A. Montenegro, J. L. Vivas, E. Okamoto, and E. Dawson. Specification and design of advanced authentication and authorization services. *Computer Standards & Interfaces*, 27(5):467–478, 2005.

[LS07] R. Lu and S. Sadiq. A survey of comparative business process modeling approaches. In *Business Information Systems (BIS 2007)*, volume 4439 of *LNCS*, pages 82–94. Springer, 2007.

[LSS11] M. Lund, B. Solhaug, and K. Stølen. *Model-driven risk analysis: the CORAS approach*. Springer, 2011.

[MB11] J. Müller and K. Bohm. The architecture of a secure business-process-management system in service-oriented environments. In *Proceedings of the 9th IEEE European Conference on Web Services (ECOWS 2011)*, pages 49–56. IEEE, 2011.

[MBS12] G. Monakova, A. D. Brucker, and A. Schaad. Security and safety of assets in business processes. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012)*, pages 1667–1673. ACM, 2012.

[MBSFM10] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina. A systematic review of security requirements engineering. *Computer Standards & Interfaces*, 32(4):153–165, 2010.

[MC96] L. McIver and D. Conway. Seven deadly sins of introductory programming language design. In *Proceedings of the International Conference on Software Engineering: Education and Practice (SEEP 1996)*, pages 309–316. IEEE, 1996.

[McD07] N. McDonald. Model-driven security: Enabling a real-time, adaptive security infrastructure. Technical Report G00151498, Gartner, 2007.

[McG06] G. McGraw. *Software Security: Building Security In*. Addison-Wesley, 2006.

[MFMP07] D. Mellado, E. Fernández-Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244–253, 2007.

[MG07] H. Mouratidis and P. Giorgini. Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.

[MHN04] J. D. Moffett, C. B. Haley, and B. Nuseibeh. Core security requirements artefacts. Technical Report 23, Department of Computing, The Open University, 2004.

[MHS05] N. R. Mead, E. D. Hough, and T. R. Stehney. Security quality requirements engineering (SQUARE) methodology. Technical Report CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University, 2005.

[Mic12] Microsoft Corporation. Microsoft Security Development Lifecycle – SDL process guidance version 5.2. Technical report, 2012.

[MJF06] H. Mouratidis, J. Jürjens, and J. Fox. Towards a comprehensive framework for secure systems development. In *Advanced Information Systems Engineering (CAiSE 2006)*, volume 4001 of *LNCS*, pages 48–62. Springer, 2006.

[MM10] M. Menzel and C. Meinel. SecureSOA: Modelling security requirements for service-oriented architectures. In *Procedings of the IEEE International Conference on Services Computing (SCC 2010)*, pages 146–153. IEEE, 2010.

[MMRV03] A. Mana, J. A. Montenegro, C. Rudolph, and J. L. Vivas. A business process-driven approach to security engineering. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA 2003)*, pages 477–481. IEEE, 2003.

[MMS97] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murφ. In *IEEE Symposion on Security and Privacy (S&P 1997)*, pages 141–151. IEEE, 1997.

[Mol12] D.-C. Moldovan. Ausarbeitung und Untersuchung eines Vorgehensmodells für das Security Engineering. Master's thesis, Hochschule RheinMain, Wiesbaden, 2012.

[MRS09] N. Moebius, W. Reif, and K. Stenzel. Modeling security-critical applications with UML in the SecureMDD approach. *International Journal on Advances in Software*, 1(1):59–79, 2009.

[MS95] S. T. March and G. F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, 1995.

[MSK07] H. Mantel, H. Sudbrock, and T. Kraußer. Combining different proof techniques for verifying information flow security. In *Logic-Based Program Synthesis and Transformation (LOPSTR 2006)*, volume 4407 of *LNCS*, pages 94–110. Springer, 2007.

[MSR10] N. Moebius, K. Stenzel, and W. Reif. Formal verification of application-specific security properties in a model-driven approach. In *Engineering Secure Software and Systems (ESSoS 2010)*, volume 5965 of *LNCS*. Springer, 2010.

[MvSB11a] J. Mülle, S. von Stackelberg, and K. Böhm. Modelling and transforming security constraints in privacy-aware business processes. In *Procedings of the IEEE International Conference on Service Oriented Computing and Applications (SOCA 2011)*. IEEE, 2011.

[MvSB11b] J. Mülle, S. von Stackelberg, and K. Böhm. A security language for BPMN process models. Technical Report 2011, 9, Karlsruhe Institute of Technology, Faculty of Informatics, 2011.

[MW14] Merriam-Webster. Merriam-webster's collegiate dictionary, December 2014.

[Nat96] National Institute of Standards and Technology. An introduction to computer security – the NIST handbook. Special Publication 800-12, 1996.

[Nat11] National Institute of Standards and Technology. Glossary of key information security terms. Technical Report IR 7298, 2011.

[NEF08] T. Neubauer, A. Ekelhart, and S. Fenz. Interactive selection of ISO 27001 controls under multiple objectives. In *Proceedings of the IFIP TC 11 23rd International Information Security Conference (SEC 2008)*, volume 278 of *IFIP*, pages 477–492. Springer, 2008.

[Neu09] T. Neubauer. An empirical study about the status of business process management. *Business Process Management Journal*, 15(2):166–183, 2009.

[NH08a] T. Neubauer and J. Heurix. Defining secure business processes with respect to multiple objectives. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES 2008)*, pages 187–194. IEEE, 2008.

[NH08b] T. Neubauer and J. Heurix. Objective types for the valuation of secure business processes. In *Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)*, pages 231–236. IEEE, 2008.

[NJC90] J. F. Nunamaker Jr and M. Chen. Systems development in information systems research. In *Proceedings of the 23rd Annual Hawaii International Conference on System Sciences (HICSS 1990)*, pages 631–640. IEEE, 1990.

[NKB05] T. Neubauer, M. Klemen, and S. Biffl. Business process-based valuation of IT-security. In *Proceedings of the 7th International Workshop on Economics-Driven Software Engineering Research (EDSER 2005)*, pages 1–5. ACM, 2005.

[NKB06] T. Neubauer, M. Klemen, and S. Biffl. Secure business process management: A roadmap. In *Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES 2006)*. IEEE, 2006.

[NP10] T. Neubauer and M. Pehn. Workshop-based risk assessment for the definition of secure business processes. In *Proceedings of the International Conference on Information, Process, and Knowledge Management (eKNOW 2010)*, pages 74–79. IEEE, 2010.

[NR08] A. Niknafs and R. Ramsin. Computer-aided method engineering: An analysis of existing environments. In *Advanced Information Systems Engineering (CAiSE 2008)*, volume 5074 of *LNCS*, pages 525–540. Springer, 2008.

[Obj03] Object Management Group. MDA guide version 1.0.1, June 2003.

[Obj04] Object Management Group. Human-usable textual notation (HUTN) specification version 1.0, August 2004.

[Obj08] Object Management Group. Software & systems process engineering metamodel specification version 2.0, 2008.

[Obj11a] Object Management Group. Business process model and notation (BPMN) version 2.0, January 2011.

[Obj11b] Object Management Group. OMG unified modeling language (OMG UML) infrastructure version 2.4.1, 2011.

[Obj14a] Object Management Group. Meta object facility (MOF) core specification version 2.4.2, 2014.

[Obj14b] Object Management Group. OMG object constraint language (OCL) version 2.4, 2014.

[Org06] Organization for the Advancement of Structured Information Standards. Web service security: SOAP message security 1.1, 2006.

[Org07] Organization for the Advancement of Structured Information Standards. Web services business process execution language version 2.0, 2007.

[Org13] Organization for the Advancement of Structured Information Standards. Extensible access control markup language version 3.0, 2013.

[Pal07] N. Palmer. A survey on business process initiatives, 2007.

[PCBV10] S. Patig, V. Casanova-Brito, and B. Vögeli. IT requirements of business process management in practice – an empirical study. In *Business Process Management (BPM 2010)*, volume 6336 of *LNCS*, pages 13–28. Springer, 2010.

[Pet62] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *Proceedings of the IFIP Congress on Information Processing (IFIP 1962)*, pages 386–390, 1962.

[Pet95] M. Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6):33, 1995.

[PH07] C. E. d. B. Paes and C. M. Hirata. RUP extension for the development of secure systems. In *Proceedings of the 4th International Conference on Information Technology (CIT 2007)*, pages 643–652. IEEE, 2007.

[Pop05] G. Popp. *Methode zur Integration von Sicherheitsanforderungen in die Entwicklung zugriffssicherer Systeme*. PhD thesis, Technische Universität München, Januar 2005.

[PTRC07] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[PvdH07] M. Papazoglou and W. van den Heuvel. Business process development life cycle methodology. *Communications of the ACM*, 50(10):79–85, 2007.

[Rö03] S. Röhrig. *Using Process Models to Analyse IT Security Requirements*. PhD thesis, Universität Zürich, 2003.

[Ral04] J. Ralyté. Towards situational methods for information systems development: Engineering reusable method chunks. In *Proceedings of the International Conference on Information Systems Development (ISD 2004)*, pages 271–282, 2004.

[Ran00] K. Rannenberg. Multilateral security a concept and examples for balanced security. In *Proceedings of the Workshop on New Security Paradigms (NSPW 2000)*, pages 151–162. ACM, 2000.

[RdGFMP10] A. Rodríguez, I. G.-R. de Guzmán, E. Fernández-Medina, and M. Piattini. Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach. *Information and Software Technology*, 52(9):945–971, 2010.

[RDR03] J. Ralyté, R. Deneckère, and C. Rolland. Towards a generic model for situational method engineering. In *Advanced Information Systems Engineering (CAiSE 2003)*, volume 2681 of *LNCS*, pages 95–110. Springer, 2003.

[Red07] S. T. J. Redwine, editor. *Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software*. U.S. Department of Homeland Security, 2007.

[RFMP06] A. Rodríguez, E. Fernández-Medina, and M. Piattini. Towards a UML 2.0 extension for the modeling of security requirements in business processes. In *Trust and Privacy in Digital Business*, volume 4083 of *LNCS*, pages 51–61. Springer, 2006.

[RFMP07a] A. Rodriguez, E. Fernández-Medina, and M. Piattini. A BPMN extension for the modeling of security requirements in business processes. *IEICE Transactions on Information and Systems*, 90(4):745, 2007.

[RFMP07b] A. Rodríguez, E. Fernández-Medina, and M. Piattini. M-BPSec: a method for security requirement elicitation from a UML 2.0 business process specification. In *Advances in Conceptual Modeling: Foundations and Applications (FP-UML 2007)*, volume 4802 of *LNCS*, pages 106–115. Springer, 2007.

[RFMTP11] A. Rodríguez, E. Fernández-Medina, J. Trujillo, and M. Piattini. Secure business process model specification through a UML 2.0 activity diagram profile. *Decision Support Systems*, 51(3):446–465, 2011.

[RHP99] A. W. Röhm, G. Herrmann, and G. Pernul. A language for modeling secure business transactions. In *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC 1999)*, pages 22–31. IEEE, 1999.

[Ric11] C. Richardson. The ROI of BPM suites. Technical Report 60205, Forrester Research, Inc., 2011.

[Rit99] P. Rittgen. From process model to electronic business process. In *Proceedings of the European Conference on Information Systems (ECIS 1999)*, 1999.

[RK04] S. Röhrig and K. Knorr. Security analysis of electronic business processes. *Electronic Commerce Research*, 4(1):59–81, 2004.

[RMMG09] S. Rougemaille, F. Migeon, T. Millan, and M.-P. Gleizes. Methodology fragments definition in SPEM for designing adaptive methodology: A first step. In *Agent-Oriented Software Engineering IX (AOSE 2008)*, volume 5386 of *LNCS*, pages 74–85. Springer, 2009.

[RPM99] K. Rannenberg, A. Pfitzmann, and G. Müller. IT security and multilateral security. In *Multilateral Security in Communications – Technology, Infrastructure, Economy*, pages 21–29. Addison-Wesley, 1999.

[RR01] J. Ralyté and C. Rolland. An assembly process model for method engineering. In *Advanced Information Systems Engineering (CAiSE 2001)*, volume 2068 of *LNCS*, pages 267–283. Springer, 2001.

[RRIG09] J. Recker, M. Rosemann, M. Indulska, and P. Green. Business process modeling: a comparative analysis. *Journal of the Association for Information Systems*, 10(4):333–363, 2009.

[RSP+10] R. Ross, G. Stoneburner, E. Porter, G. Rogers, M. Swanson, R. Graubart, B. Hodge, A. Johnson, S. Katzke, G. Turner, K. Dempsey, and C. Enloe. Recommended security controls for federal information systems and organizations. Special Publication 800-53, National Institute of Standards and Technology, 2010.

[Rup13] C. Ruppricht. Entwicklung eines Werkzeugs für die Sicherheitsmodellierung elektronischer Prozesse. Master's thesis, Wilhelm Büchner Hochschule, Darmstadt, 2013.

[SB12] A. Schaad and M. Borozdin. TAM^2: automated threat analysis. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012)*, pages 1103–1108. ACM, 2012.

[SC09] S. Spiekermann and L. F. Cranor. Engineering privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2009.

[SCFY96] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[Sch99] B. Schneier. Modeling security threats. *Dr. Dobb's Journal*, December 1999.

[Sch02] A.-W. Scheer. *Vom Geschäftsprozess zum Anwendungssystem*. Springer, 4th edition, 2002.

[Sch03] M. Schumacher. *Security engineering with patterns: origins, theoretical model, and new applications*. Springer, 2003.

[Sch06] D. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2):25–31, 2006.

[SF03] H. Smith and P. Fingar. *Business process management: the third wave*. Meghan-Kiffer, 2003.

[SFBH+05] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, 2005.

[SH08] M. Siponen and J. Heikka. Do secure information system design methods provide adequate modeling support? *Information and Software Technology*, 50(9-10):1035–1053, 2008.

[Sha09] R. Sharp. Report: CC-based design of secure application systems. In *Engineering Secure Software and Systems (ESSoS 2009)*, volume 5429 of *LNCS*, pages 111–121. Springer, 2009.

[SHAAR12] Z. Shakeri Hossein Abad, A. Alipour, and R. Ramsin. Enhancing tool support for situational engineering of agile methodologies in Eclipse. In *Software Engineering Research, Management and Applications (SERA 2012)*, volume 430 of *SCI*, pages 141–152. Springer, 2012.

[SHF04] G. Stoneburner, C. Hayden, and A. Feringa. Engineering principles for information technology security (a baseline for achieving security). Special Publication 800-27, National Institute of Standards and Technology, 2004.

[Shi07] R. Shirey. Internet security glossary, version 2. Request for Comments 4949, IEFT, 2007.

[Sho14] A. Shostack. *Threat Modeling: Designing for Security*. Wiley, 2014.

[Sil06] B. Silver. The 2006 BPMS report: Understanding and evaluating BPM suites, 2006. BPM Institute.

[Sim96] H. A. Simon. *The sciences of the artificial*. MIT Press, 1996.

[Sip05] M. T. Siponen. Analysis of modern IS security development approaches: towards the next generation of social and adaptable ISS methods. *Information and Organization*, 15(4):339–375, 2005.

[SLS06] A. Schaad, V. Lotz, and K. Sohr. A model-checking approach to analysing organisational controls in a loan origination process. In *Proceedings of the 11th ACM Symposion on Access Control Models and Technologies (SACMAT 2006)*, pages 139–149. ACM, 2006.

[SR01] J. Steel and K. Raymond. Generating human-usable textual notations for information models. In *Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001)*, pages 250–261. IEEE, 2001.

[SS75] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[SS04] F. Swiderski and W. Snyder. *Threat modeling*. Microsoft, 2004.

[SSSW98] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner. Toward a secure system engineering methodolgy. In *Proceedings of the Workshop on New Security Paradigms (NSPW 1998)*, pages 2–10. ACM, 1998.

[Sun11] A. Sunyaev. *Health-Care Telematics in Germany: Design and Application of a Security Analysis Method*. Gabler, 2011.

[SVEH07] T. Stahl, M. Völter, S. Efftinge, and A. Haase. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. dpunkt, 2007.

[Tal08] P. Tallon. Inside the adaptive enterprise: an information technology capabilities perspective on business process agility. *Information Technology and Management*, 9(1):21–36, 2008.

[TJ08] S. Taubenberger and J. Jürjens. IT security risk analysis based on business process models enhanced with security requirements. In *Proceedings of the Workshop on Modeling Security (MODSEC 2008)*, volume 413. CEUR, 2008.

[TJG+11] S. Tjoa, S. Jakoubi, G. Goluch, G. Kitzler, S. Goluch, and G. Quirchmayr. A formal approach enabling risk-aware business process modeling and simulation. *IEEE Transactions on Services Computing*, 4(2):153–166, 2011.

[TML+09] C. Talhi, D. Mouheb, V. Lima, M. Debbabi, L. Wang, and M. Pourzandi. Usability of security specification approaches for UML design: A survey. *Journal of Object Technology*, 8(6):103–122, 2009.

[VdA04] W. Van der Aalst. Business process management demystified: A tutorial on models, systems and standards for workflow management. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 1–65. Springer, 2004.

[VdAtHW03] W. Van der Aalst, A. ter Hofstede, and M. Weske. Business process management: A survey. In *Business Process Management (BPM 2003)*, volume 2678 of *LNCS*, pages 1019–1019. Springer, 2003.

[VFMP05] R. Villarroel, E. Fernández-Medina, and M. Piattini. Secure information systems development – a survey and comparison. *Computers & Security*, 24(4):308–321, 2005.

[VGD11] P. Van Gorp and R. Dijkman. BPMN 2.0 execution semantics formalized as graph rewrite rules: extended version. Beta Working Paper 353, Technische Universiteit Eindhoven, 2011.

[VHF02] R. Vaughn, R. Hennig, and K. Fox. An empirical study of industrial security-engineering practices. *Journal of Systems and software*, 61(3):225–232, 2002.

[VM02] J. Viega and G. McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, 2002.

[VML03] J. L. Vivas, J. A. Montenegro, and J. López. Towards a business process-driven framework for security engineering with the UML. In *Information Security (ISC 2003)*, volume 2851 of *LNCS*, pages 381–395. Springer, 2003.

[vO06] D. von Oheimb. Formal methods in the security business: Exotic flowers thriving in an expanding niche. In *Formal Methods (FM 2006)*, volume 4085 of *LNCS*, pages 592–597. Springer, 2006.

[VWW02] M. Vetterling, G. Wimmel, and A. Wisspeintner. Secure systems development based on the common criteria: the PalME project. *ACM SIGSOFT Software Engineering Notes*, 27(6):138, 2002.

[Wes07] M. Weske. *Business process management: concepts, languages, architectures*. Springer, 2007.

[WGHS99] M. Weske, T. Goesmann, R. Holten, and R. Striemer. A reference model for workflow application development processes. *ACM SIGSOFT Software Engineering Notes*, 24(2):1–10, 1999.

[WH12] C. Wolf and P. Harmon. The state of business process management 2012, 2012. BPTrends Report.

[Whi01] J. J. Whitmore. A method for designing secure solutions. *IBM Systems Journal*, 40(3):747–768, 2001.

[Wim05] G. Wimmel. *Model-Based Development of Security-Critical Systems*. PhD thesis, Technische Universität München, 2005.

[WMM08] C. Wolter, M. Menzel, and C. Meinel. Modelling security goals in business processes. In *Modellierung 2008*, volume 127 of *LNI*, pages 197–212. GI, 2008.

[WMS+09] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. Model-driven business process security requirement specification. *Journal of Systems Architecture*, 55(4):211–223, 2009.

[Wol10] C. Wolter. *A Methodology for Model-Driven Process Security*. PhD thesis, Universität Potsdam, 2010.

[Wor08] Workflow Management Coalition. Process definition interface – XML process definition language, 2008.

[WSM07] C. Wolter, A. Schaad, and C. Meinel. Deriving XACML policies from business process models. In *Web Information Systems Engineering (WISE 2007) Workshops*, volume 4832 of *LNCS*, pages 142–153. Springer, 2007.

[WV11] K. Weldemariam and A. Villafiorita. Procedural security analysis: A methodological approach. *Journal of Systems and Software*, 84(7):1114–1129, 2011.

[WvP10] S. Winkler and J. von Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling*, 9(4):529–565, 2010.

[YB97] J. Yoder and J. Barcalow. Architectural patterns for enabling application security. In *Proceedings of Pattern Languages of Programs (PLoP 1997)*, 1997.