Network of Excellence on
Engineering Secure Future Internet
Software Services and Systems

Network of Excellence

Deliverable D2.4

Second Release of the Method and Tool Evaluation

Project Number	:	256980
Project Title	:	NESSoS
Deliverable Type	:	Report

Deliverable Number	:	D2.4
Title of Deliverable	:	Second Release of the Method and Tool Evaluation
Nature of Deliverable	:	R
Dissemination Level	:	Public
Internal Version Number	:	3.0
Contractual Delivery Date	:	September 30 2013
Actual Delivery Date	:	October 31 2013
Contributing WPs	:	WP2
Editor(s)	:	Marianne Busch (LMU) Nora Koch (LMU)
Author(s)	:	Alexander van den Berghe (KUL) Marianne Busch (LMU) Nora Koch (LMU) Riccardo Scandariato (KUL) Le Minh Sang Tran (UNITN)
Reviewer(s)	:	Benoit Baudry (INRIA) Manuel Clavel (IMDEA) Marina Egea (ATOS) Marinella Petrocchi (CNR)

Abstract

This report describes the approach of the NESSoS project related to the evaluation and comparison of security features, vulnerabilities, methods, notations and tools in the area of software engineering for secure software and systems. The approach has been further developed since it was introduced in deliverable D2.1. It is based on the CBK of WP5, which focuses on a knowledge base of existing methods, notations and tools. Additionally, our approach, named SECEVAL*, addresses the collection, analysis and finer-grained representation of security-specific knowledge.*

Getting an overview and collecting data of existing security engineering methods, notations and tools is of major importance for security and software engineers, as it helps them to take decisions about solutions for upcoming tasks. Security engineers usually rely on their experience and sometimes are limited in their decisions to the development context and architecture. Nevertheless, more often than not they would like to have access to compact information about security mechanisms in order to find a more effective method, an appropriate notation, useful CASE tools for developing prototypes and systems, and further tools needed in the Software Development Life Cycle (SDLC).

Thus, the objective is to further develop our strategy sketched in D2.1 for evaluating and comparing methods, notations and tools; this time mainly focusing on evaluation. Our approach SECEVAL *provides an ontology, which comprises: (1) a security context model describing security features, methods, notations and tools; (2) a data collection model, which records how data is collected when researchers or practitioners search for an answer to a question they have; (3) a data analysis model specifying how analysis, using previously collected data, is done.*

SECEVAL *was influenced by our prior works, an evaluation of approaches for secure software design and an empirical validation of risk-based methods.*

Thereby, we provide an evaluation approach targeting security engineers who may also have research questions in mind which cannot be answered directly by the CBK and who want to use a sound, extensible model to specify their research actions as well as research results. Consequently, our approach is designed to ease the process of doing research in the area of security, no matter if the research question aims at scientific or engineering issues.

Keyword List

Description of Security Mechanisms, Evaluation of Security Mechanisms, Security Methods, Security Tools, Security Notations, Security Properties, Vulnerabilities, Threats

Document History

Version	Type of Change	Author(s)
0.1	Table of Contents	Marianne Busch (LMU)
0.2	Questionnaire as appendix and first description of model	Marianne Busch (LMU)
0.3	SecEval description of security context	Marianne Busch (LMU)
0.4	SecEval description of data collection	Marianne Busch (LMU)
0.5	SecEval description of data analysis	Marianne Busch (LMU)
0.6	Chapters about guided interview	Marianne Busch (LMU)
0.7	Abstract and conclusion	Marianne Busch (LMU)
0.8	Introduction and spell-checking and improvements of the models	Marianne Busch (LMU)
0.9	Improvements of the models	Marianne Busch, Nora Koch (LMU)
1.0	Proof reading, changes and comments	Nora Koch (LMU)
1.1	Background chapter	Marianne Busch (LMU)
1.2	Chapter about Empirical Validation of Security Methods	Le Minh Sang Tran (UNITN)
1.3	Literature review on secure software design	Alexander van den Berghe, Riccardo Scandariato (KUL)
1.4	Chapter about case study	Marianne Busch (LMU)
2.0	Improvements after internal reviews	Marianne Busch and Nora Koch(LMU)
3.0	Final version	Marianne Busch and Nora Koch(LMU)

Table of Contents

List of Acronyms

CASE Computer-Aided Software Engineering

CBK Common Body of Knowledge

CPU Central Processing Unit

GUI Graphical User Interface

KO Knowledge Object

NESSoS Network of Excellence on Engineering Secure Future Internet Software Services and Systems

OCL Object Constraint Language

OWASP Open Web Application Security Project

SDE Service Development Environment

SDLC Software Development Life Cycle

SLR Systematic Literature Review

SWRL Semantic Web Rule Language

UML Unified Modeling Language

URL Uniform Resource Locator

WP Work Package

XSS Cross-site scripting

List of Figures

Executive Summary

This report describes the approach of the NESSoS project related to the evaluation and comparison of security features, vulnerabilities, methods, notations and tools in the area of software engineering for secure software and systems. The approach has been further developed since it was introduced in deliverable D2.1. It is based on the CBK of WP5, which focuses on a knowledge base of existing methods, notations and tools, which are summarized by the term Knowledge Object (KO). Additionally, our approach, named SECEVAL, addresses the collection, analysis and finer-grained representation of security-specific knowledge.

Getting an overview and collecting data of existing security engineering methods, notations and tools is of major importance for security and software engineers, as it helps them to take decisions about solutions for upcoming tasks. Security engineers usually rely on their experience and sometimes are limited in their decisions to the development context and architecture. Nevertheless, more often than not they would like to have access to compact information about security mechanisms in order to find a more effective method, an appropriate notation, useful CASE tools for developing prototypes and systems, and further tools needed in the SDLC.

Thus, the objective is to further develop our strategy sketched in D2.1 for evaluating and comparing methods, notations and tools; this time mainly focusing on evaluation. Our approach SECEVAL provides an ontology, which comprises: (1) a security context model describing security features, methods, notations and tools; (2) a data collection model, which records how data is collected when researchers or practitioners do research to answer a question they have; (3) a data analysis model specifying how analysis, using previously collected data, is done.

SECEVAL was influenced by our prior works, an evaluation of approaches for secure software design and an empirical validation of risk-based methods.

Thereby, we provide an evaluation approach targeting security engineers who also have research questions in mind which cannot be answered directly by the CBK and who want to use a sound, extensible model to specify their research actions as well as research results. Consequently, our approach is designed to ease the process of doing research in the area of security, no matter if the research question aims at scientific or engineering issues.

With our evaluation approach, called SECEVAL, we do not claim to provide a one-fits-all ontology for IT-security (which would horribly overload any model), but introduce an extensible basis. SECEVAL defines an ontology which is represented using graphical models. It comprises:

- A security context model describing security properties, vulnerabilities and threats as well as methods, notations and tools.
- A data collection model, which records how data is collected when a researchers or practitioners do research to answer a question.
- A data analysis model specifying, how analysis using previously collected data, is done.

To verify our approach, we used SECEVAL for a case study in the area of security testing of web applications.

Prior to our work on SECEVAL, we performed two domain-specific evaluations of KOs: an evaluation of approaches for secure software design and an empirical validation of risk-based methods. The former is a study which provides an overview of the current state of the art regarding secure software design. The latter study compares a textual method and a graphical method for identifying threats and possible mitigations. Both works, beyond others, influenced the construction of SECEVAL.

After a short motivation, the deliverable starts describing the background of our evaluation approach, especially the connection to the CBK in chapter 2. Chapter 3 presents our evaluation approach called SECEVAL and a guided review with a questionnaire we used to improve it. Chapter 4 provides an example of instantiation of our model in form of a case study about methods and tools from the area of security testing. Chapter 5 introduces two examples of domain-specific KO evaluations. Finally, chapter 6 summarizes and sketches future steps and chapter 7 lists the NESSoS publications that are relevant for this work.

Note that the Service Development Environment (SDE) and its 20 integrated tools are not discussed in this deliverable. For further information about the SDE, please refer to deliverables D1.4 [2] and D2.3 [11].

1 Introduction

Software and security engineers constantly make decisions about which technology should be used in the different phases of the Software Development Life Cycle (SDLC). Therefore, a cost-benefit analysis and a subsequent selection of appropriate Knowledge Objects (KOs) (i.e., methods, tools and notations) for a specific task, play an important role. More often than not, there is no time to investigate on alternatives to well-known ones used so far. Most of the questions which arise are not entirely new, but useful scraps of knowledge are distributed in papers, books or the web, or just exist in the head of colleagues working at another department. What makes it even worse is that engineers often have to start from the very beginning by defining the process of evaluating KOs as well as the structure of the results from scratch, i.e. without having a template for their domain.

To ease the tasks of recording own results and of getting an overview of existing KOs, we introduced an ontology in deliverable D2.1 that works hand in hand with the Common Body of Knowledge (CBK) [13] which was implemented as a semantic Wiki within the scope of WP5. In the conclusion of deliverable D2.1 we stated that "it became apparent that comparisons can be drawn best for enumeration types" and that we wanted to further develop the approach. As time passed, we gained further experience in working with the CBK which raised three questions:

- How could our ontology be improved, so that security-related features become first-class citizens?
- How can we use our ontology for finding answers to research questions beyond comparing methods, notations and tools?
- How is the process of data collection and data analysis specified, to make sure that emerging research results are comprehensible and valid?

Evaluation in the area of cybersecurity does mean for us, e.g., to find out which authentication-related threats can or cannot be mitigated by a method, for which tool-support is implemented. Furthermore, it is advantageous to know which tools work together and can be used in a row and which notations are supported as tool inputs. Up to now, these kinds of questions require researchers to document their approaches and results in a self-made way. Consequently, other researchers, who want to build on those results, have first to understand many different schemas documenting research processes and their results. This is not only annoying and time-consuming, but also error-prone, as misunderstandings easily occur.

With our evaluation approach, called SECEVAL, we do not claim to provide a one-fits-all ontology for IT-security (which would horribly overload any model), but introduce an extensible basis. SECEVAL defines an ontology which is represented using graphical models. It comprises:

- A security context model describing security properties, vulnerabilities and threats as well as methods, notations and tools.
- A data collection model, which records how data is collected when researchers or practitioners do research to answer a question.
- A data analysis model specifying, how analysis using previously collected data, is done.

A simplified example of the process of using SECEVAL for evaluation is depicted in Figure 1.1. Research questions initiate the process of data collection, where sources (as papers, websites, ...) are gathered. These sources are then analyzed, which means to extract information, possibly process it and record it using SECEVAL's security context model.

To verify our approach, we used SECEVAL for a case study in the area of security testing of web applications.

Prior to our work on SECEVAL, we performed two domain-specific evaluations of Knowledge Objects (KOs): an evaluation of approaches for secure software design and an empirical validation of risk-based methods. The former is a study which provides an overview of the current state of the art regarding secure software design. The latter study compares a textual method and a graphical method for identifying threats and possible mitigations. Both works, beyond others, influenced the construction of SECEVAL.

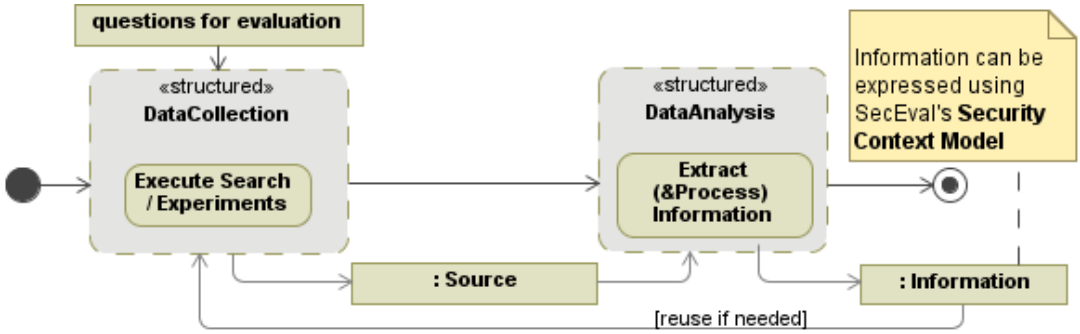


Figure 1.1: Overview of SECEVAL's Evaluation Process

Background/Foreground. The background for this deliverable was described in deliverable D2.1 and D5.4. More about the background of this work can be found in chapter 2.

Relations to other work packages. As written above, we introduced an ontology in deliverable D2.1 which works hand in hand with the CBK. The CBK was developed in WP5 and contains information about 163 security engineering methods, notations and tools, used and developed within the NESSoS project as well as from outside the project.

Structure of the deliverable. Chapter 2 describes the background of our evaluation approach, especially the connection to the CBK. Chapter 3 presents our evaluation approach called SECEVAL and a guided review with a questionnaire we used to improve it. Chapter 4 provides an example of instantiation of our model in form of a case study about methods and tools from the area of security testing. Chapter 5 introduces two examples of domain-specific KO evaluations. Finally, chapter 6 summarizes and sketches future steps and chapter 7 lists the NESSoS publications that are relevant for this work.

2 Background

In this chapter, we introduce related work in the area of evaluation of methods, notations, tools, vulnerabilities, threats or security features. Evaluation approaches are often tailored to the needs of a specific area. We start by introducing general approaches and continue with those which are security-specific.

2.1 General Evaluation Approaches

Kitchenham et al. [39] specify so called "Systematic Literature Reviews" (SLRs) in software engineering. The aim is to answer research questions by systematically searching and extracting knowledge of existing literature. The systematic literature review is executed in three main steps: first, the review is planned, then it is conducted and finally results are reported (this process is depicted in deliverable D5.2 [5, Fig. 3.2]). Our approach, SECEVAL, is based on their work. We focus on the use of arbitrary resources, as source code or experiments which are carried out to answer a research question. Furthermore, we specify a basic graphical structure for results (context model) which is security-specific. However, SECEVAL can also be used for methods, notations and tools which are not especially related to security features or it can be extended to also cover other areas of interest. In contrast to Kitchenham's approach, the process we define is generic and thus allows us to refine the way of searching while being in the phase of collecting data. Refining the search means to collect data during several phases, e.g., to perform a breadth-search before a concrete plan for a subsequent depth-search is made.

SIQinU (Strategy for understanding and Improving Quality in Use) [4] is a framework for evaluating the quality of a product version which can then be improved. It uses the conceptual framework **C-INCAMI** (Contextual-Information Need, Concept model, Attribute, Metric and Indicator), which specifies concepts and relationships for measurement and evaluation. SIQinU defines a strategy using UML activity diagrams whereas C-INCAMI is specified by a UML class diagram. C-INCAMI consists of six modules which are modeled as UML packages:

- measurement and evaluation project definition
- nonfunctional requirements specification
- context specification
- measurement design and implementation
- evaluation design and implementation
- analysis and recommendation specification

For example, in the package "evaluation..." they specify a decision criterion which defines thresholds which are used to determine the need for action (i.e., improvement of quality in order to fulfill requirements). Although C-INCAMI is used for the domain of quality evaluation and not for the domain of security Knowledge Objects (KOs), we recognized several properties we wanted to take care of in our own approach. UML class diagrams seem to be a sound way to represent concepts and relationships between them. Separation of concerns can easily be implemented by using several packages; however the overall size of the model should not get too big. We wondered why we found no paper that is depicting all six C-INCAMI modules. This reduces the advantage of graphical models, which usually is to quickly provide an overview. Besides, names for classes and their attributes have to be chosen with caution in order to avoid permanent lookups in the specification.

Moody [51] proposes an evaluation approach which is based on experiments. Practitioners use methods and afterwards answer questions about perceived ease of use, perceived usefulness and intention to use. Further details and how Moody's approach can be integrated in SECEVAL are discussed in subsection 3.3.2.

The **CBK** (Common Body of Knowledge) [6] defines an ontology for software engineers to describe Knowledge Objects (KOs), which are methods, techniques, notations, tools or standards. Techniques are methods which do not specify activities for applying the method. Note that those activities do not necessarily have to be supported by a tool. We use the CBK as a starting point for SECEVAL's ontology,

although in our context model we do not focus on standards and we aggregate the concepts of technique and method, as an instance model immediately shows whether actions (in our case called `Steps`) are defined.

The CBK is implemented as a semantic Wiki [13] and serves as a knowledge base containing all relevant information about existing KOs. At the moment it is primarily used and improved by security- and software engineers who are related to NESSoS. Further information about the improvement process of the CBK can be found in D5.3 [7, 6]. Unlike the CBK, SECEVAL is not implemented yet.

In deliverable D2.1 [10] we defined relations between KOs and exemplarily compared methods, notations and tools in a tabular way. SECEVAL is built on our experiences with the CBK and deliverable D2.1, which lead to the three questions we asked in the previous chapter.

2.2 Security-specific Evaluation Approaches

We also have a look at existing evaluation approaches which are security-specific. To the best of our knowledge, no framework exists which is as comprehensive as SECEVAL.

Frameworks which are more specific often consider concrete systems for their evaluation. An example is the **OWASP Risk Rating Methodology** [22], where the risk for a concrete application or system is estimated. As usual, estimation is based on factors for estimating likelihood and impact. Factors are made up by questions with predefined answers, which are rated from 0 to 9. The overall severity of the risk is then calculated by multiplying ratings of likelihood and impact.

For SECEVAL, we added vulnerability-dependent features of the OWASP model, as e.g., the difficulty of detecting or exploiting a vulnerability. Features that are related to a concrete system and the rating of a possible attack are introduced as an extension of SECEVAL in subsection 3.3.2.

The i* [66] metamodel is the basis of a vulnerability-centric requirements engineering framework introduced in [20]. The extended, **vulnerability-centric i* metamodel** aims at analyzing security attacks, countermeasures, and requirements based on vulnerabilities. The metamodel is represented using UML class models. Instances of this metamodel use different i* notations which are not based on UML. Main elements of the metamodel are: vulnerability, attack (which can exploit a vulnerability; executed by an actor), effect and security impact (e.g., on a resource).

Another approach which is focused on vulnerabilities is described by Wang et al. [70] They come up with a concept model which is less detailed than the i* metamodel. Their aim is not to describe reality by using graphical models, but to create a knowledge base which can then be queried using a language for the semantic web, called SWRL.

Moyano et al. [54] provide a **Conceptual Framework for Trust Models** which is also represented using UML. As trust is an abstract concept, which emerges between communication partners, we do not consider it in SECEVAL. However, strategies how to build a network of trust can be modeled in SECEVAL using the concept of a “method”.

Generally, we prefer to represent evaluation structures in a graphical way, as it allows to quickly get an overview of what is important for an evaluation. Additionally, a description should be provided so that details can be looked up. For example, the C-INCAMI model uses UML to specify concepts as metrics, indicators or decision criteria.

When it comes to recording evaluation results, we favor approaches that provide a basic structure, as e.g., the CBK does. We do not think that this structure has to be graphical at any cost, as text- or table-based versions also have their advantages, especially for huge amount of data. However, we provide a graphical structure, because relation between several elements can be recognized at a first glance. In case graphical instance models get bigger during the evaluation process, we suggest to use them only for the main results in order to give an overview. More detailed information can then be provided using tables or texts which are linked from the model. The best reusability of data seems to be given when the structure is implemented in a semantic Wiki, as shown by the CBK. This allows that researchers can add their knowledge in a more or less structured way.

3 Evaluation Approach in the Security Domain: SECEVAL

Our aim is to provide an approach for the evaluation of methods, notations and tools for the engineering of secure software systems. Evaluation should also be possible for security properties, vulnerabilities and threats. In this chapter we first introduce our conceptual evaluation framework SECEVAL [12] which can be used to collect and describe security-related data and metrics to analyze it. Second, we describe how we gathered expert knowledge to verify the soundness of our approach. For the graphical representation of concepts and relationships we selected the UML notation, as we think it fits our needs best.

3.1 Requirements

We start eliciting the requirements of such a framework, i.e. which stakeholders are involved, which concepts play a role in secure software and evaluation of methods, tools and notations, and how those concepts are related. Therefore, the first step is to name the common **stakeholders** for secure software: security engineers (i.e. security designers and developers), normal users of the software and attackers. In some cases, users can attack software without knowing it, e.g., when they have a virus installed. For us, those users are also classified as attackers, as well as developers which are e.g. trying to smuggle malicious code in the software. Figure 3.1 depicts stakeholders and use cases in a UML Use Case diagram.

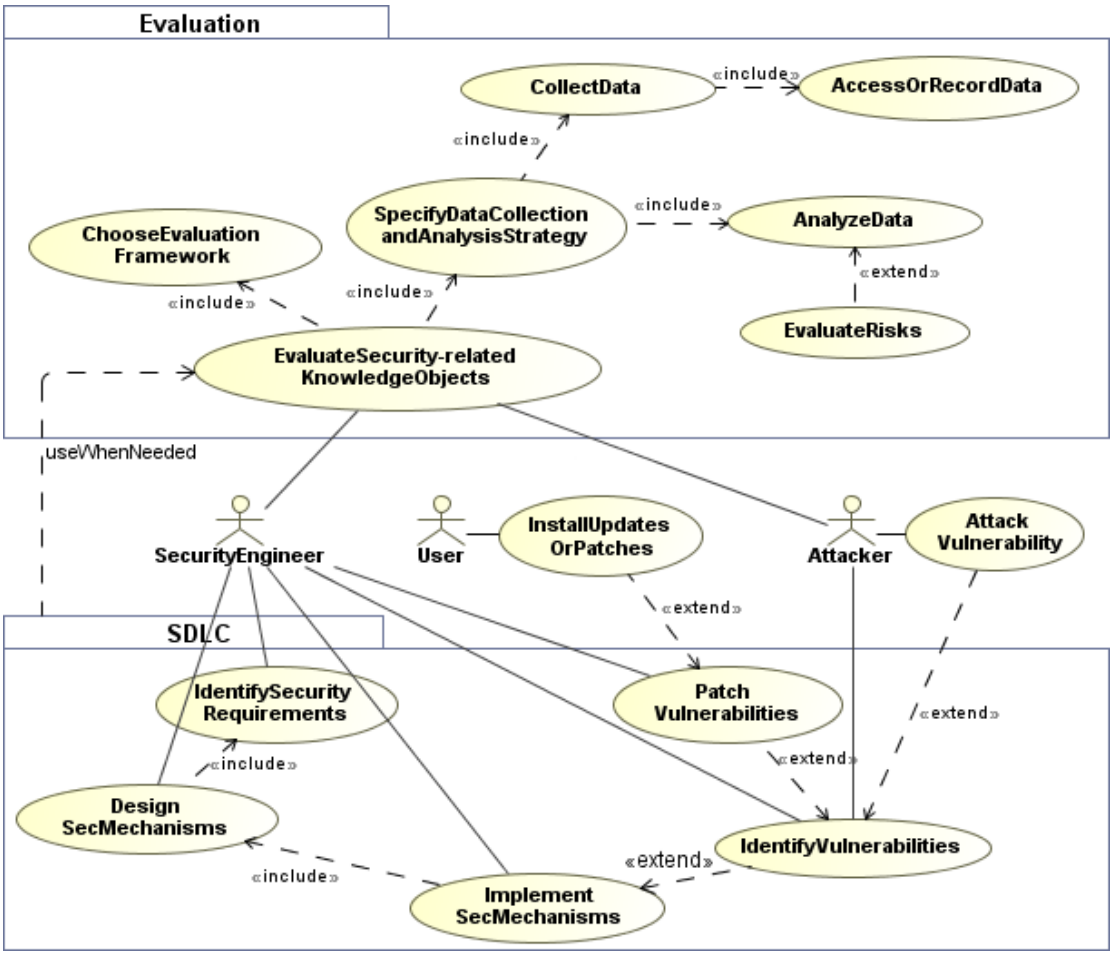


Figure 3.1: Stakeholders and Use Cases

The `Evaluation` package at the top contains all use cases related to evaluation, whereas `SDLC` at the bottom of the diagram refers to security-related tasks within the Software Development Life Cycle (SDLC). From time to time, all tasks within the SDLC require evaluating possibilities, as for example, which tool should be used for gathering security requirements or for designing secure web applications.

The «include» dependencies show the order of use cases in the SDLC: implementing secure software requires having a design, and having a design implies that requirements were elicitated beforehand. Both, the attacker and the security engineer can identify vulnerabilities, whereas the former usually attacks it and the latter tries to patch it, which is modeled using an «extend» dependency. Those patches can then be installed by users (which also might happen using an automatic update function). While designing software, security requirements need to be identified by the designer or developer.

For security experts it is helpful to know common security methods and tools that support this method. An umbrella term for methods, tools or notations and linked security properties, vulnerabilities and threats is Knowledge Object (KO). Gathering knowledge usually starts with selecting an evaluation framework, or by (re)inventing one. The framework should record the strategy for collecting and analyzing data in a structured way and support the strategy with suitable means.

Collecting data means to access data, e.g., in papers and to record data, e.g., by conducting empiric studies or executing benchmarks. Analyzing data includes comparing and transforming data as well as the evaluation of risks. An example for a small evaluation is the question during the implementation which library for authentication should be used. A more elaborated evaluation could be the evaluation of risks for a concrete software system which is important in all SDLC phases.

3.2 Ontology

We first give an overview of SECEVAL, before going into detail about context model, data collection and data analysis model in the following subsections.

SECEVAL provides a structure for data as well as a structure to perform a data analysis on the collected data. The aim of analyzing data is to convert facts (e.g., tool A can do this) into context-related knowledge (e.g., tools A and B support the same method, however tool B fulfills our requirements better than A does). Security engineers or attackers thus can choose our evaluation framework SECEVAL in order to evaluate KOs.

We depict the concepts and their relationships as a UML model, so that we can instantiate concrete KOs.The use cases from our requirements analysis were a starting point to identify relevant concepts related to security for using and evaluating methods, notations and tools in the software engineering process. We grouped these concepts in three packages: **Security Context**, **Data Collection** and **Data Analysis**. Figure 3.2 shows our ontology represented as a UML class diagram.

The **Security Context** package is used to specify the object-oriented data structure we use for describing methods and related security features, notations and tools. Within the **Security Context** package we represent a security feature as a class element. The classes `Method`, `Notation` and `Tool` inherit general attributes, as e.g., a problem description and whether or not it is based on standards, from the abstract class `Mechanism`. Additionally, a tool can support methods and a notation can be used for several methods. In this document we also use the upper-case term “Mechanism” when referring to a method, a notation or a tool.

Having a well-defined data structure which is extensible is one thing, collecting concrete data the other. The package **Data Collection** contains the search process as well as research questions which should be investigated.

Research questions define what is inside and outside the scope of research. Before specifying the research questions, a basic understanding of the context is needed, therefore a dependency, stereotyped «use», points from `ResearchQuestion` to the package `SecurityContext`. An example question would be “I want to plan how authorization is implemented in my web application for online banking. Which methods exist to support me in this task and which tools can support a secure implementation of authorization issues?”. Queries can be defined accordingly. They are used to find matching sources containing data which might help to answer the research questions. Research questions are questions that ask for results from research and define as precisely as possible what should be in the scope (and what not). Please note that for us the term “research question” does not have to refer to scientific research questions.

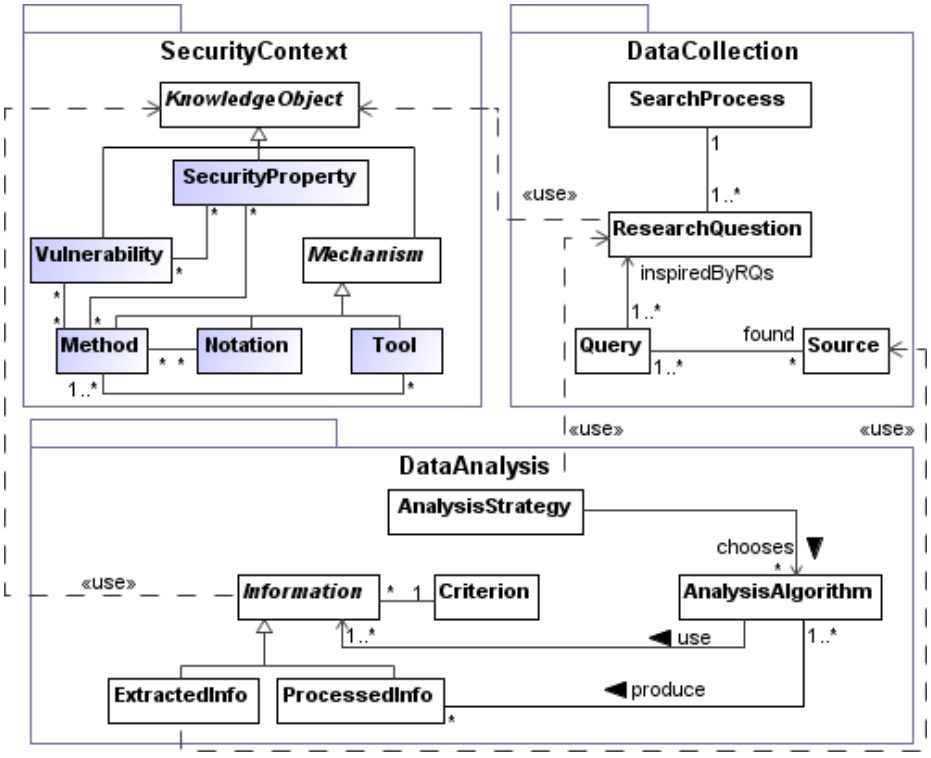


Figure 3.2: Model Overview

After collecting data, the source data may consist of, e.g., some papers, several websites or code. The process of extracting information from the data is called **Data Analysis**. An analysis strategy defines criteria which are important for answering the research question, like costs or preconditions. These criteria are used to classify information which is extracted from the sources. Information can also be processed, for example one could calculate annual costs for tools. The instructions used for those calculations are referred to as `AnalysisAlgorithm` in Figure 3.2. Each piece of information (processed or not) can relate to classes of the security context model, e.g., a tool can be described using information from the website of the tool. Although, information does not have to be related to a context model's class, as we might want to express aggregated values as for example the average costs of testing tools on the market.

In practice this implicates that the basic ingredients for an evaluation are:

- a research question to answer,
- search terms as a basis for searching resources, as online data bases or terms for asking experts,
- promising sources after finishing a search process,
- an analysis strategy that describe which sorts of information should be extracted and how it can be grouped by categories and eventually how they should be processed by an analysis algorithm,
- a set of information which can describe properties of methods, notations and tools (cf. security context) with a criterion attached to each information which provides, e.g., a metric (which could read "all costs should be given in Euros" or "percentages should be represented by $--$, $-$, 0, $+$ and $++$" using a step size of 20%),
- data which is not assigned to Mechanism(s) or should be categorized additionally can also be expressed using the modeling elements `Information` and `Criteria`.

In the following, going into more detail describing the three aspects of our conceptual ontology: security context, data collection and data analysis.

3.2.1 Security Context

Figure 3.3 shows the data structure of the Security Context package. The aim of this package is to provide a sound structure to be able to classify security-related methods, notations and tools together with security properties, vulnerabilities and threats. We introduce an abstract class `Mechanism` from which the classes `Method`, `Notation` and `Tool` inherit the common attributes such as `goals`, `costs`, `basedOnStandards`, etc. Once Mechanisms are described using this description model, it should be easy to get an overview of existing security-related methods, tools and notations for a certain area. Furthermore, the package should serve as a flexible basis and starting point for an evaluation, which means that it can be adopted to fit the needs of the researcher to examine a concrete research question.

In Figure 3.3, for convenience enumerations' texts are grey and the background of classes which can directly be instantiated is colored. All attributes and roles are typed; however the types are not shown in the figures due to brevity. Many types are obvious to be boolean (can.., has.., is..) as binary answers (`true`, `false`) can quickly give an overview of the main characteristics of a Mechanism. For a concrete implementation we suggest to use the values `true`, `false` and `undefined` to make clear whether or not an attribute is set. Generally, enumerations should be extensible when it comes to an implementation of our model. The CBK implements this behavior by providing a suggestion field for additional enumeration item. If the same enumeration item is suggested many times, it will become part of the default enumeration. The full MagicDraw 17.0[1] model can be downloaded from the Web.[2]

Explanations of the security context packages' elements are given one by one in the following.

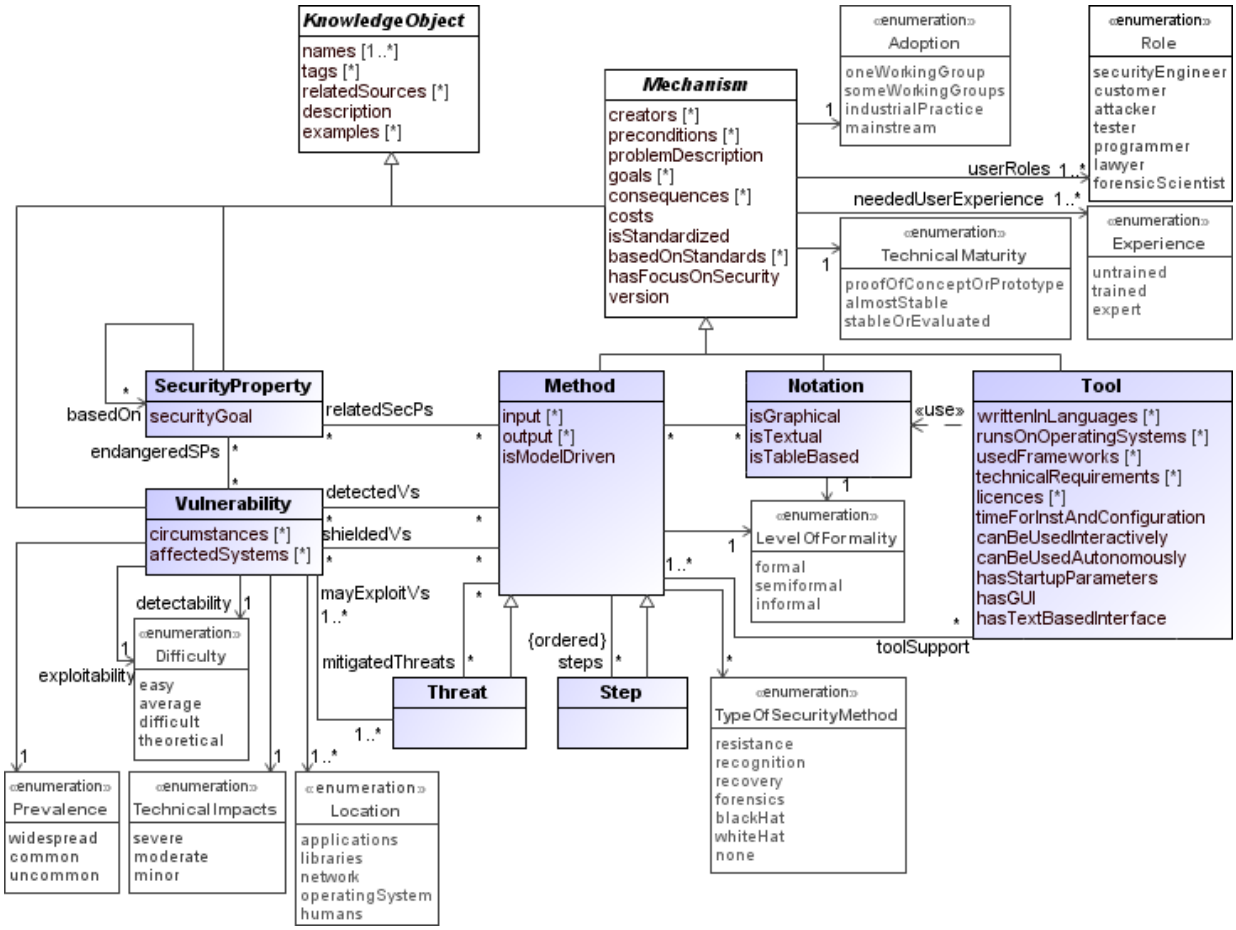


Figure 3.3: Security Context

As mentioned above, a MECHANISM is an abstract notation for a method, notation or tool. It can be

[1] MagicDraw. http://magicdraw.com/

[2] SecEval Project. http://www.pst.ifi.lmu.de/~busch/SecEval/SecEval.mdzip

described by a problem statement, by the goals it strives for, by its costs and by the consequences it implies. Mechanisms can be based on standards or be standardized themselves. They can have arbitrary many creators, as companies, inventors or developers. Before applying a Mechanism, the preconditions that are necessary to use it have to be fulfilled. Furthermore, an estimation regarding technical maturity and adoption in practice should be given. We also like to generally express whether or not the mechanism has a special focus on security, because in practice many mechanisms can also be used for security purposes, but do not directly focus them. Several levels of usability can be stated according to the experience a user needs to use a mechanism, e.g., a certain mechanism should best be used by trained users and experts. The kind of user can be specified by selecting roles which are proposed to use the Mechanism.

A METHOD has some general attributes, which are used to describe the method at a high level of abstraction, as input, output and if it is model-driven. For more extensive methods, each step of the method can also be described in detail, if necessary. A method or step can be supported by notations or tools.

An advantage of our model is that it can easily be extended. If someone needs a more detailed approach, further elements can be added. For example the steps to be taken while using certain method can be described by more attributes, like a step number, or a boolean which indicates if the step can be repeated iteratively, etc. Further details about methods are provided in the description on page 22.

For a NOTATION, we consider characteristics such as whether the notation is graphical, textual or based on a tabular representation. We also added a level of formality, which ranges from informal to formal.

For us, the description of a TOOL comprises the information of languages it is written in, of operating systems it supports, of frameworks it uses and of technical requirements, which have to be fulfilled in order to use it. The tool (or its parts) are released under certain licenses. Additionally, the needed time for installation and configuration can be provided. Booleans are used to describe if the tool can be used interactively or autonomously, if it has start parameters, a GUI or a text-based user interface. Further details about tools are provided in the description on page 25.

As seen before, a tool supports a certain method. However, we have not yet defined the quality of this support. Does the tool fully support the method? Does it partially support the method and which parts are not supported?

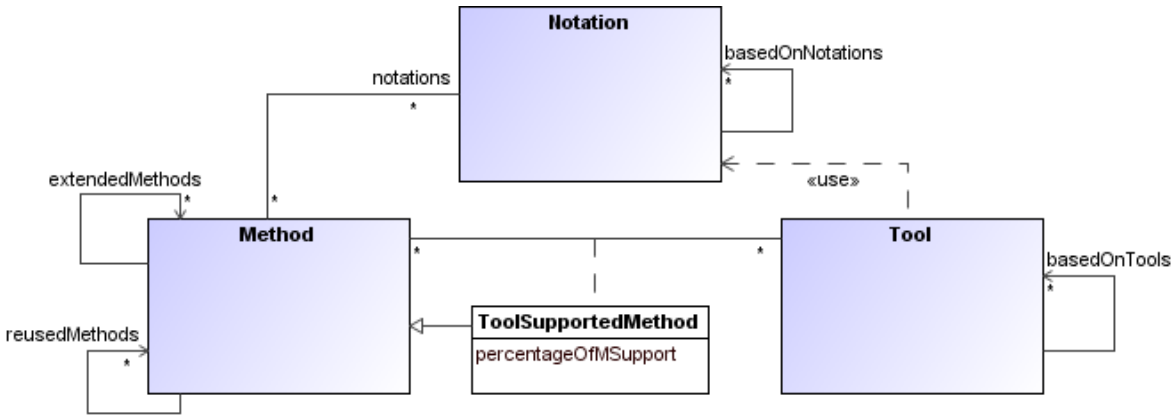


Figure 3.4: Security Context: Connections between Tools, Notations and Methods

We add this information using the association class `ToolSupportedMethod`, as depicted in Figure 3.4 with a dotted line. The association class itself is inherited from the class `Method`, thus can redefine attributes. For instance, a design tool can partly support a model-driven method (e.g., by facilitating the modeling process), although it cannot generate artifacts. In this case, DesignM.canGenerateArtifacts would be set to false (cf. p. 22).

A method can extends on other methods, which means it might also change them. In case the role `extendedMethods` should be further specified, we recommend adding an association class which inherits from the class `Method` (similar to the association between method and tool). In this way, it can be exactly described if and how the original methods are modified. It is also possible that other methods are used without changes (role `usesMethods`).

A tool can be based on other tools, which is especially the case when libraries are used or when plugins are written. Notations can also be based on other notations, for example many context-specific extensions for UML exist.

In addition, we adopted the abstract KNOWLEDGEOBJECT (KO) which was used in previous work to record most information of elements which are described. For SECEVAL, we applied separation of concerns so that only very general descriptions remain as attributes in a KO, which can be applied to all elements. Therefore, the class `KnowledgeObject` has names, tags and related sources, which could be any kinds of sources, as publications or URLs. A description and examples allow to quickly learn about a specific KO.

Security properties, vulnerabilities and mechanisms are kinds of KOs. The relationship "`X` is a kind of `KnowledgeObject`" is modeled by a UML inheritance.

We consider a SECURITY PROPERTY to represent basic concepts of security. A security property can be, e.g., authorization, authentication, integrity, etc. The attribute `SecurityGoal` denotes a string and describes the goal of the property, for instance authentication "is the act of confirming the truth of an attribute of a datum or entity"[3]. Security properties can also be based on other security properties, as authorization is based on authentication.

A VULNERABILITY is "a weakness which allows an attacker to reduce a system's information assurance"[4]. Thus, it endangers security properties. Examples are XSS, SQL Injection, Buffer Overflows, etc. Methods can detect such vulnerabilities or shield them from being exploited by a threat. Every vulnerability is located at least in one location (which is modeled as a UML enumeration). Furthermore, we include the categorization scheme from OWASP TOP 10 [23] (which is adapted from the OWASP Risk Rating Methodology [22]) using exploitability, prevalence, detectability and impact levels. The difficulty `theoretical` means that it is practically impossible to detect or exploit a vulnerability.

A THREAT is a "possible danger"[5] that may exploit vulnerabilities. We treat a threat as a kind of method which is vicious. At least one vulnerability has to be affected, otherwise a threat is not malicious (and the other way around), which is denoted by the multiplicity [1..*]. Additionally, threats can be mitigated by other methods.

Methods in the Context of the SDLC

In Figure 3.6 resp. 3.5, the classes `Tool` and `Method` are refined according to their usage in the SDLC, because the attributes which are used to describe a method or tool are related to the SDLC phases it covers. The phases of the SDLC are those we have chosen to classify tools and methods in D2.1 [10]: requirements, design, implementation, testing, assurance, risk & cost, service composition and deployment. We added the phase "runtime" as a category to distinguish methods and tools that operate at the runtime of a system without being plain testing tools. Examples are attack tools. Notations were not associated with phases of the SDLC as they are located in the same phase than the method they support and as far as we know, no phase-related attributes are needed to describe features of notations.

Figure 3.5 again depicts our class `Method`. The abstract class `MAreasOfDev` is a wildcard for detailed information about the method. As mentioned above, we group attributes according to the phases of the SDLC. This means that a method can support several development phases. For example a method can be used to design a secure application and the method also describes how to build an implementation accordingly. In this case, the attributes of the classes `DesignM` and `ImplementationM` would be used to describe the method.

The meaning of the attributes is described in the following:

RequirementsM.comprisesElicitation is set to true if a method is used for requirement elicitation.

RequirementsM.comprisesAnalysis is set to true if a method comprises the analysis of requirements.

RequirementsM.comprisesSpecification is set to true if a requirements specification is part of a method.

RequirementsM.comprisesManagement is set to true if the management of requirements is defined by a method.

RequirementsM.levelOfDetail describes how detailed the method's requirements features are. As this is a subjective value, we recommend adding a short explanation.

[3] Wikipedia – Authentication. https://en.wikipedia.org/wiki/Authentication
[4] Wikipedia – Vulnerability. https://en.wikipedia.org/wiki/Vulnerability_(computing)
[5] Wikipedia – Threat. https://en.wikipedia.org/wiki/Threat_(computer)

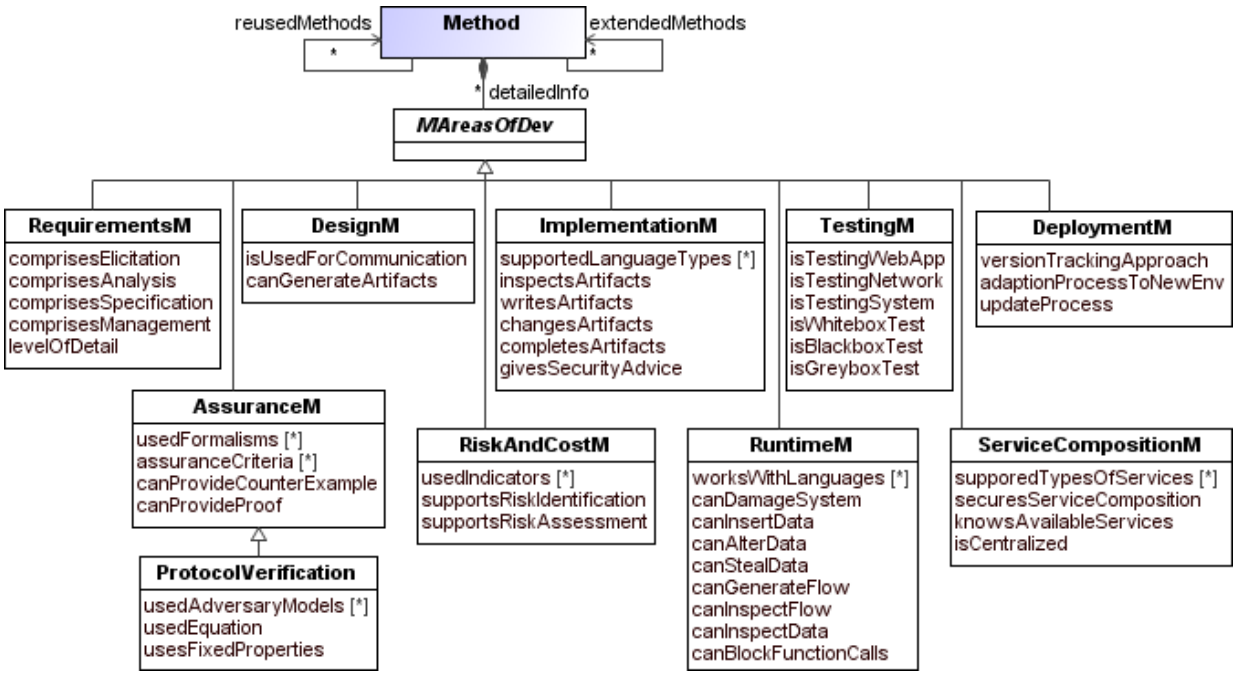


Figure 3.5: Security Context: Details of Methods

DesignM.isUsedForCommunication is set to true if a design method is also used for communication purposes, as e.g., UML.

DesignM.canGenerateArtifacts is set to true if the method specifies the generation of artifacts (code, configuration files, . . .) out of the design of an application.

ImplementationM.supportedLanguageTypes the types of languages which are supported by a method. An example would be: “functional programming languages” are supported by a method which auto-completes fragments of code.

ImplementationM.inspectsArtifacts is true if the method relies on read access to artifacts, as e.g., program code.

ImplementationM.writesArtifacts is true if the method writes artifacts, as in the above-mentioned example of code completion.

ImplementationM.changesArtifacts is true if the methods changes existing artifacts.

ImplementationM.completesArtifacts is true if the method completes already existing artifacts. An example is the auto completion used in development environments.

ImplementationM.givesSecurityAdvice is true if security-related hints are provided during the implementation.

TestingM.isTestingWebApp is true if a method describes how to test web applications.

TestingM.isTestingNetwork (same with network-related testing)

TestingM.isTestingSystem (same with system-related testing; the system refers to a soft- or hardware system)

TestingM.isWhiteboxTest is true if the method is based on white-box tests, which means that the tester has full access to all internal details of a system.

TestingM.isBlackboxTest is true if the method is based on black-box tests, which means that code and other internal details cannot be accessed by a tester.

TestingM.isGreyboxTest is true if the method uses tests that are a mixture of white-box and black-box tests, which means that the tester can access some internal details of the system, but not all.

DeploymentM.versionTrackingApproach describes the approach which is used for tracking available versions of a piece of software.

DeploymentM.adaptionProcessToNewEnv describes the process which is specified for adapting software to a changed environment, e.g., if hardware is changed.

DeploymentM.updateProcess describes the process for updating software.

AssuranceM.usedFormalisms lists the formal methods that are used by an assurance-related method. This is a shortcut in case we do not want to model those formalisms as a method on its own.

AssuranceM.assuranceCriteria gives the criteria the assurance process relies on.

AssuranceM.canProvideCounterExample. In case formal methods are applied, it often makes sense to provide a counter example. The attribute is true in case the method can give at least one.

AssuranceM.canProvideProof is true if a formal proof for a certain criterion can be given.

ProtocolVerification.usedAdversaryModels. Protocol verification is an example how to go into more detail without changing the core of SECEVAL: e.g., the used adversary models can be specified with this attribute.

ProtocolVerification.usedEquation describes the equation which is used.

ProtocolVerification.usesFixedProperties is true if fixed properties are used for the protocol verification.

RiskAndCostM.usedIndicators names the indicators used to analyze security-related risks and costs.

RiskAndCostM.supportsRiskIdentification is true if the method gives advice how to identify risks.

RiskAndCostM.supportsRiskAssessment is true if the method supports risk assessment approaches.

RuntimeM.worksWithLanguages lists programming languages on which the method can be applied. Usually, methods which define techniques related to the runtime behavior of the targeted system are rather language-specific.

RuntimeM.canDamageSystem is true if the method describes how to damage a running system. This applies especially for attack-related methods.

RuntimeM.canInsertData is true if the method allows data to be inserted into the running system so that the system uses it.

RuntimeM.canAlterData is true if the method comprises changing data.

RuntimeM.canStealData is true if the method defines how to steal data.

RuntimeM.canInspectData is true if the method assumes that inspecting data is possible. This does not include that data can also be transferred outside of the system, which would be stealing.

RuntimeM.canInspectFlow is true if the flow of data can be inspected.

RuntimeM.canGenerateFlow is true if a data flow chart can be generated.

RuntimeM.canBlockFunctionCalls is true if the method specifies how to block function calls.

ServiceCompositionM.supporedTypesOfServices names services which are eligible to be composed.

ServiceCompositionM.securesServiceComposition is true if the method describes how to secure a service composition.

ServiceCompositionM.knowsAvailableServices is true if the concept is to know available services.

ServiceCompositionM.isCentralized is true if the method anticipates a centralized instance for being applied.

Please notice that the value “true” for attributes, as e.g., “RiskAndCostM.supportsRiskIdentification” can also mean that the risk identification strategy is “foreign”; i.e., the method does not support it itself, but the method comprises another method which supports risk identification. (Reminder: comprising another method is modeled using the role Method.reusedMethods or Method.extendedMethods.)

When implementing the security context model in a similar way than the CBK, it might be helpful to add axioms to our ontology. Axioms state assertions which have to be true. In our case, we could think about rules to describe dependencies between attributes, as e.g., that `isWhiteboxTest` and `isGreyboxTest` cannot be `true` at the same time, or that `extendedMethods` should not point to the same method. In UML those axioms are usually expressed using the Object Constraint Language (OCL) [56] statements. However, we want to keep our model as simple as possible, especially because OCL statements quickly get cryptic and overload the model.

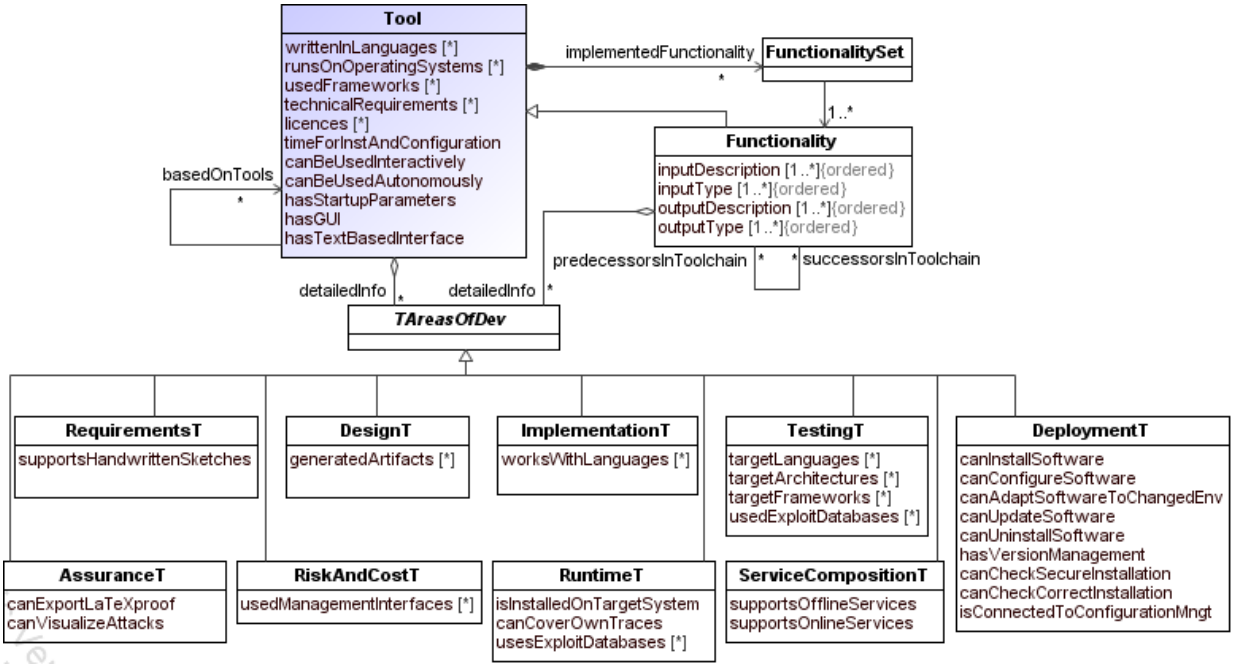


Figure 3.6: Security Context: Details of Tools

Tools in the Context of the SDLC

Figure 3.6 shows the class `Tool` and gives additional information about features of a tool. Usually, a tool can execute more than one functionality; very often they are grouped by purpose. For functionalities input and output can be specified and they can be used to build tool chains, using the output from one tool as input for another tool, as implemented in a tool workbench, namely the SDE. For further information about the SDE see deliverable D2.3 [11].

Similar to methods, tools and their functionalities can be described according to the areas of development they cover:

RequirementsT.supportsHandwrittenSketches is true if handwritten sketches can be managed with the tool.

DesignT.generatedArtifacts lists artifacts (like code) that can be generated by the tool.

ImplementationT.worksWithLanguages names languages which are supported by the tool. Examples are tools which perform auto-completion on the code. The related method might be called “auto-completion on imperative languages” and a tool could support Java.

TestingT.targetLanguages lists languages the target system can be written in so that the testing-related tool can be applied.

TestingT.targetArchitectures describes how the system architecture has to look like in order to test it with the tool.

TestingT.targetFrameworks names target frameworks that are supported by the tool.

TestingT.usedExploitDatabases refers to databases containing up-to-date exploits.

DeploymentT.canInstallSoftware is true if tool is able to install software.

DeploymentT.canConfigureSoftware is true if tool is able to configure software.

DeploymentT.canAdaptSoftwareToChangedEnv is true if the tool can adapt software to a changed environment, e.g., a changed hardware environment or a changed server configuration.

DeploymentT.canUpdateSoftware is true if the tool can update software.

DeploymentT.canUninstallSoftware is true if the tool can uninstall software.

DeploymentT.hasVersionManagement is true if the tool can track versions of software, which means it manages which versions of different software can be used together.

DeploymentT.canCheckSecureInstallation is true if the tool can make sure that software was installed correctly so that security requirements are fulfilled.

DeploymentT.canCheckCorrectInstallation is true if the tool can make sure that software was installed correctly.

DeploymentT.isConnectedToConfigurationMngt is true if the tool is connected to or implements a configuration management system.

AssuranceT.canExportLaTeXproof is true if a proof is part of the assurance-related method and it can be exported as LaTeX formulae.

AssuranceT.canVisualizeAttacks is true if attacks can be visualized using graphical models.

RiskAndCostT.usedManagementInterfaces lists management interfaces used by the tool, e.g., an interface to an Enterprise Resource Planning (ERP) system.

RuntimeT.isInstalledOnTargetSystem is true if the tool is installed on the same system where the target software is installed which is accessed at runtime.

RuntimeT.canCoverOwnTraces is true if the tool is able to cover the own traces, as e.g., logfile entries or additional files that were downloaded. This usually applies to tools that are used for attacks.

RuntimeT.usedExploitDatabases lists databases containing known exploits.

ServiceCompositionT.supportsOfflineServices is true if the tool supports services which run offline, i.e., are not running at a server.

ServiceCompositionT.supportsOnlineServices is true if the tool supports services which run online, i.e., are running at a server (which is usually not the host where the service composition tool is installed).

As shown, the security context model can easily be extended or even mapped to another domain by adding and removing elements. In fact, even non-security tools, as e.g., Eclipse[6] can be described without changing the model. In this case, neither `SecurityProperty` nor `Vulnerability` is instantiated (which is possible due to the multiplicity "$*$") and the role `typeOfSecurityMethod` should be set to `none`.

Two extensions of the context model, one for concrete risk evaluation and one for method evaluation including testing participant's feedback are detailed in subsection 3.3.2.

3.2.2 Data Collection

High-quality data is the basis for an evaluation, as the best analysis strategy cannot make up for low-quality data. Our aim is to create a schema which describes properties that have to be defined before starting collecting data. This is particularly needed, if the data collection has to be systematic as usual for scientific approaches (cf. use cases `SpecifyDataCollection` and `CollectData` section 3.2). Therefore, we base our approach on Kitchenham's systematic literature review [39]. However, we do not restrict our evaluation approach to reviewing literature, as we also include information about tools which cannot always be found in papers, but on websites and information which is extracted from experience, e.g., data gathered by benchmarks.

In order to collect data, it is common to define a search process (c.f. Figure 3.7) which specifies several steps called process phases. Each phase may follow another approach, e.g., the search can be automated or not, or it can be a depth-first or a breadth-first search. Depth-first means, that the aim of a search is to extract a lot of detail information about a relatively small topic, whereas a breadth-first search is good to get an overview of a broader topic. We noticed that Kitchenham et al. do not use more than one search phase. This might be the case because they prefer to not be biased by findings from a previous phase. The advantage of more than one search phase is that resources can be added which were considered interesting in a previous phase, such as further contributions from conferences or from authors which seemed promising.

Similar to Kitchenham's literature review, research questions are used to define the corner stones and the goals of the search. Queries can be derived from the research questions. They are then used and refined in the phases of the search process. As different search engines support different types of queries, concrete queries are specific for each resource, e.g., Google Scholar[7]. Thus, concrete queries document what was searched for in a certain resource so that results remain as repeatable and comprehensible as possible. At the same time, concrete queries belong to one query which generally defines what was searched for. Search expressions can contain keywords as well as special characters like "-" to exclude

[6] Eclipse. http://eclipse.org/

[7] Google Scholar. http://scholar.google.com/

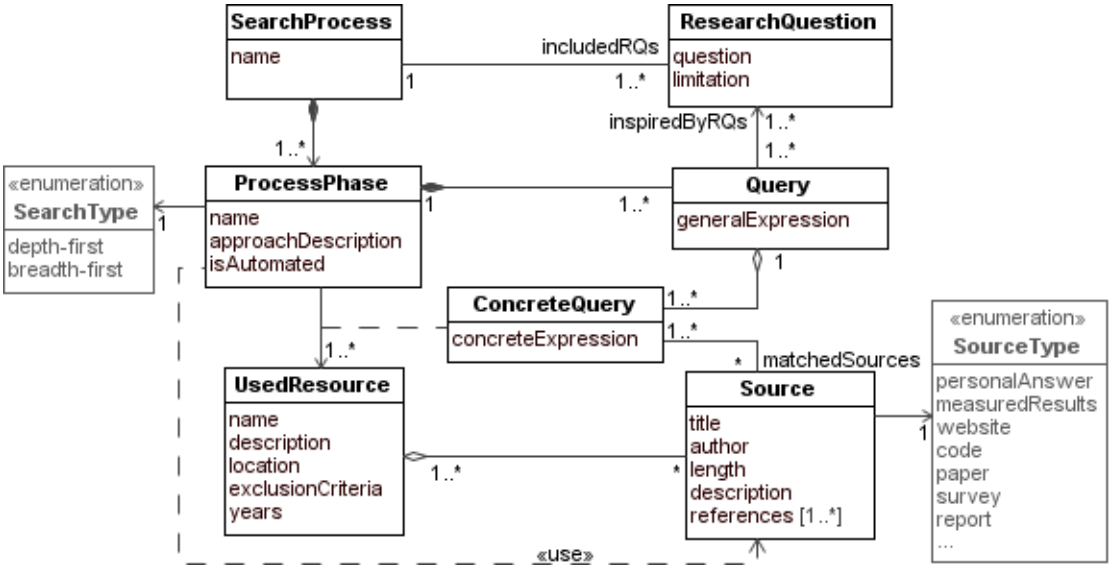


Figure 3.7: Data Collection

a keyword, "~" to search for similar keywords or quotes or group words that have to appear exactly in the given order. Queries can also refer to questions which are used as a basis for experiments (cf. chapter 4).

It is important to choose resources that will serve as data sources for the evaluation. It is worth to notice that resources can not only be scientific papers but also the web or measured data, code repositories or persons. For contacting a person, the concrete query expression is used as a question, which is asked. When measuring data or providing a "proof of concept", the source's exclusion criteria have to be empty or specified in a way that the result is not distorted. The use of an association class for `ConcreteQuery` (depicted by a dashed line) denotes that for each pair of `ProcessPhase` and `UsedResource`, the class `ConcreteQuery` is instantiated. The concrete search expression is derived from a general search expression.

For *example* the general search expression could be "recent approaches in Security Engineering" and we want to ask Google Scholar and a popular researcher. For Google Scholar we could use ""Security Engineering" 2012..2013" as a concrete search expression and the concrete expression for asking a researcher could read: "I'm interested in Security Engineering. Which recent approaches in Security Engineering do you know?".

If a concrete query matches sources, as papers, websites or personal answers, we classify the source at least by author and description (as an abstract) and provide information about the type of source and at least one reference where to find it.

Process phases can use the matching sources to get an idea how to define more appropriate queries, which is especially useful for depth-first search phases.

Figure 3.8 depicts the process of working with SECEVAL as a UML activity diagram. It details Figure 1.1, which was shown in the introduction. Used and produced artifacts can be found on the right. Notice that this process has to be adapted (and usually simplified) for each evaluation. Writing down the exact process might not always be necessary, as many activities can be executed in parallel or in any order, which is indicated by the black, horizontal bars. Artifacts used by the analysis process will be explained in the following.

3.2.3 Data Analysis

We do not only want to collect data in order to know a brief description, but we want to analyze it in order to answer research questions. This is why data analysis is applied. According to Kitchenham, the procedure how to collect as well as analyze data belongs to the "review protocol" and has always to be specified in the first place.

Figure 3.9 depicts what is needed for analyzing data. First, we have to specify which type of strategy

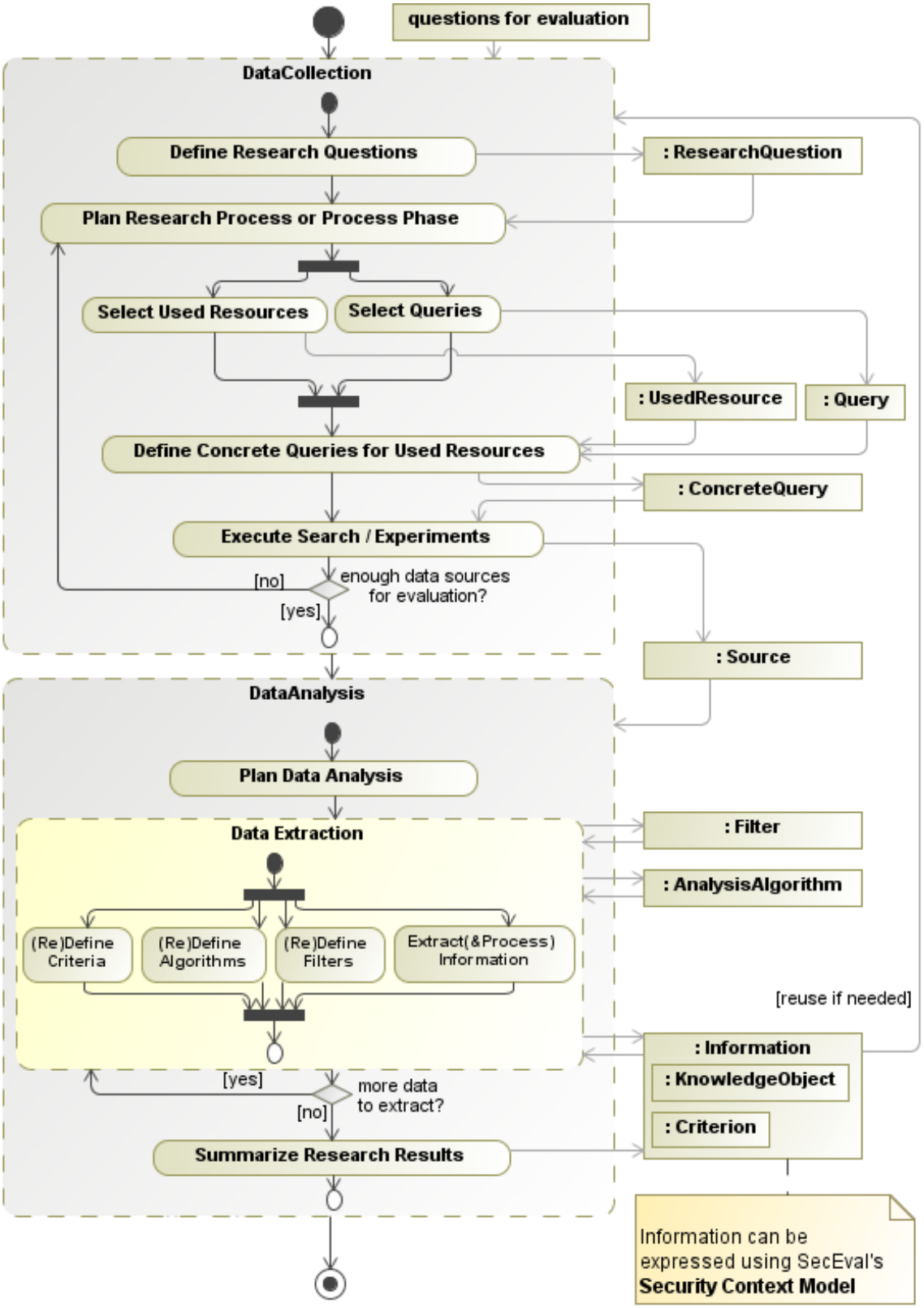


Figure 3.8: SECEVAL's Evaluation Process

we want to use. Do we only use quantitative analysis or do we mainly use qualitative analysis? Accordingly, one can later refer to Kitchenham's checklists for quantitative and qualitative studies [39, table 5 and 6] to ensure the quality of the own answers to the research questions.

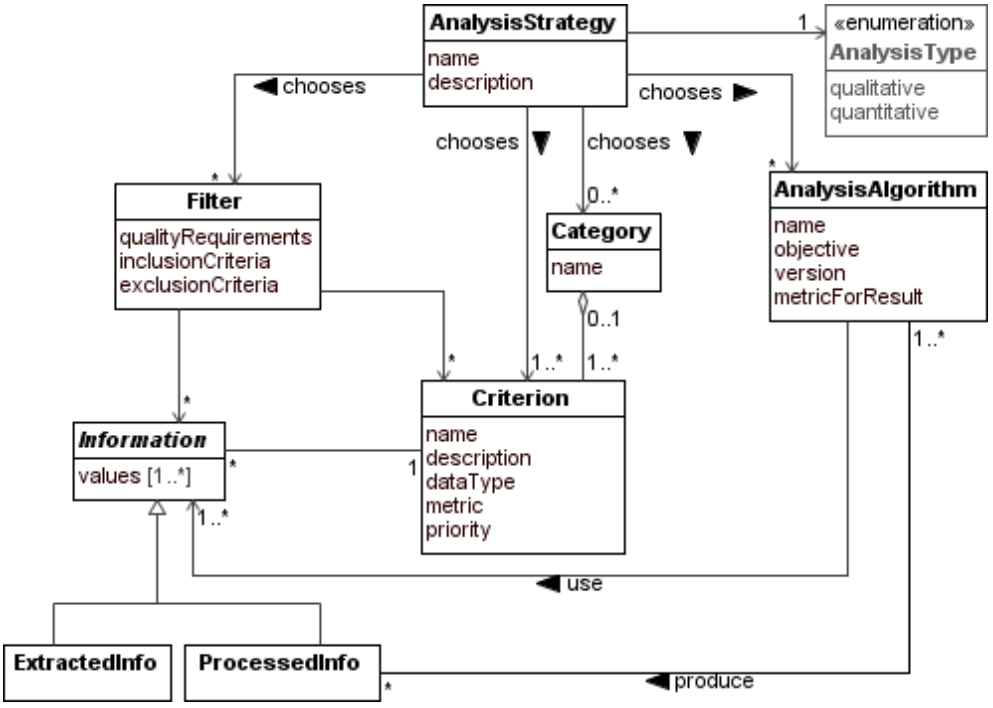


Figure 3.9: Data Analysis

The analysis strategy determines the used categories & criteria, analysis algorithms and filters according to the research question. Criteria can be grouped by categories. Whether or not a criterion is grouped by categories depends on the kind of information which should be associated to the criterion: information with a single value, as "costs" do not have to be grouped, whereas it makes sense to group boolean information if a testing method is testing web applications, networks or systems to a category called "type of testing target".

A CRITERION gives more information about data values as it defines the data type (string, list of booleans, ..) and the metric (milliseconds, ..). Additionally, the priority can be defined which is useful when Mechanisms should be compared. We suggest prioritizing using integer values from 0 (unimportant) to 10 (very important).

Information can be extracted from the sources which were found in the data collection phase (see «use» dependency starting from `ExtractedInfo` in Figure 3.2), or they can be processed using an ANALYSIS ALGORITHM. This algorithm does not have to be executable on a computer. The analysis strategy defines which algorithm is used and makes sure that the result of the algorithm fits to a criterion regarding meaning and metric. The algorithm may, of course, be implemented by tool in the sense of a `Tool` of the security context model. However, we do not want to overload the model at this point.

The relations we presented in deliverable D2.1 [10, section 3.2] can be seen as instances of analysis algorithms. For instance, the relation `IsCompatible_NxN_ToolIO` expresses that "two notations are compatible if there exists a tool chain that can transform the first given notation into the second one" [10, p. 22]. In this case, the algorithm might contain the depth-first search for a tool-chain consisting of tools where the output of one tool serves as input for the second one. We experienced that automating this algorithm is challenging, because in- and output of tools are usually described by strings and are influenced by many factors like the tool's functions which are used.

Besides, a FILTER can be defined to disqualify results according to certain criteria as costs or quality. This filter is finer grained than the `exclusionCriteria` used in the data collection (class `UsedResource`), which only can use obvious criteria, as e.g., the language the source is written in. In addition to this, the

filter for data analysis accesses information as well as criteria and thus can exclude, e.g., Mechanisms from the evaluation which do not meet a high-priority requirement.

A valid question is how information, criteria and the security context model fit together. This was shown in Figure 3.2: information can be stored in an instance of our security context model. Consequently, the attributes `name` and `dataType` of `Criterion` can be left blank when information is stored in an instance of our model, as attributes have a name and are typed. However, these attributes are needed when describing information which is not directly related to an instance of a knowledge object.

In summary it can be said that neither the collection of data nor the data analysis are security specific and thus can be applied equally to other domains within computer science. An example how we collected and analyzed data in the area of security testing can be found in chapter 4.

3.3 Gathering Expert Knowledge

Coming up with a broad evaluation ontology for security KOs is challenging, because many different areas of expertise are needed. Fortunately, NESSoS (associated) partners encompass the broad area of secure software development, e.g.:

- Security Requirements Engineering
- Model-driven Security
- Trust and Reputation
- Verification
- Security Monitoring
- Vulnerability Testing
- Risk and Cost
- Privacy requirements engineering
- Access Control
- Security Protocols
- Empirical methods for security
- Secure Booting
- Javascript Sandboxing

The easiest way to tap into each other's know-how was to come up with a basic structure of an evaluation approach and to discuss it. The basic structure we used was based on our experience from deliverable D2.1 [10] and inspired by the model of the CBK [6, Figure 2.1].

3.3.1 Guided Interview

A Guided Interview is "a one-on-one directed conversation with an individual that uses a pre-determined, consistent set of questions but allows for follow-up questions and variation in question wording and order."[8]

We used this kind of interview in a slightly modified way: first we explained our basic model (especially the basic Security Context Model) at the NESSoS plenary meeting in Malaga. Second, we handed out a description and a questionnaire, which can be found in Appendix A. Third, we were around while (associated) partners were filling in the questionnaire, answered questions, explained the model in further detail and discussed feedback which went beyond the questionnaire.

The discussions were very fruitful and they helped us, as well as the detailed answers to the questionnaire. We like to thank again all 14 (associated) partners who contributed with their ideas to the improvement of our evaluation approach.

[8] Education dictionary. http://www.mondofacto.com/facts/dictionary?guided+interview

3.3.2 Results and Extensions

The resulting evaluation approach SECEVAL is described in the previous section (3.2). Some changes due to the guided interviews are discussed in the following:

- "Maturity Level could be split into TechnicalMaturity and Adoption." This is a good point, because there are some Mechanisms which are technical mature, but not very adopted in practice.
- Denote if there is an "interactive or batch" mode for tools. We implemented this using the attributes `canBeUsedInteractively` and `canBeUsedAutonomously`.
- Threats can be mitigated by another method. For example the method of a firewall can be used to mitigate the threat by an attacker from outside.
- "Can it be represented that a tool is based on another tool?" We first thought it might be enough to base a method upon another method, but it turned out that tools are often based on other tools which are supporting the same method. We therefore added an association with the role `basedOnTools`.
- To make it possible to describe functionalities of a tool, two classes, called `Functionality` and `FunctionalitySet` were introduced.
- The class `ProtocolVerification` was added. First, because protocol verification is a common task regarding security. Second, it demonstrates how to easily extend the context model using inheritance.
- The difference between methods and tools sometimes did not become clear immediately. Therefore, we moved all attributes that describe a method and also can describe a tool to the heirs of `MAreasOfDev` and added the possibility that a tool can partially support a method and thus redefine attributes, which is depicted in Figure 3.4.
- It was suggested to extend the enumeration `Location` of vulnerability so that operating systems and humans (i.e., human behavior) can be specified as the location of a vulnerability. We liked this suggestion, because in this way we can also model non-technical vulnerabilities and tag them accordingly.

In order to keep the model as simple as possible, we decided not to implement all suggestions for changes, especially when they were at a high level of detail, which could be added when adapting our security context model to a concrete scenario. An example is the description of detailed asset, risk and attacker models which of course needs more attributes to describe complex relations. Furthermore, the basis of SECEVAL is used for general information which should not be too specific. When evaluating risks, the risks are usually evaluated for a concrete system. We modeled an extension to show how SECEVAL can be enhanced using OWASP's Risk Rating Methodology [22]. Figure 3.10 depicts the extended model whereby added connections use thick line linkings.

The class `Threat`, known from the basic context model, inherits its features to a concrete `Attack`. The severity of the risk (which is an attribute of `Threat`) can be calculated by likelihood multiplied with impact. The likelihood is derived from the factors which describe the vulnerabilities and the threat agents, whereas the impact is determined by the concrete technical and business-related impact. Therefore, each enumerations' literal is mapped to a likelihood rating from 0 to 9. For more information the interested reader is referred to [22].

Another point we omitted is, to detail the costs for using a Mechanism, as the most convenient level of detail has to be defined individually for a set of research questions: sometimes, the overall costs should be recorded, sometimes it is necessary to split the costs into cost to train personal, yearly license costs for using proprietary tools and so on.

A comment on the question why a researcher would not use the proposed evaluation framework was: "I would go for an experimental approach. I would run a set of comparative experiments where the participants apply the methods on a real case. To compare the methods I would use the method evaluation model proposed by Moody. [51]. In this model you have several constructs like actual efficiency, actual effectiveness, perceived ease of use, perceived usefulness, intention to use, actual usage." We welcome

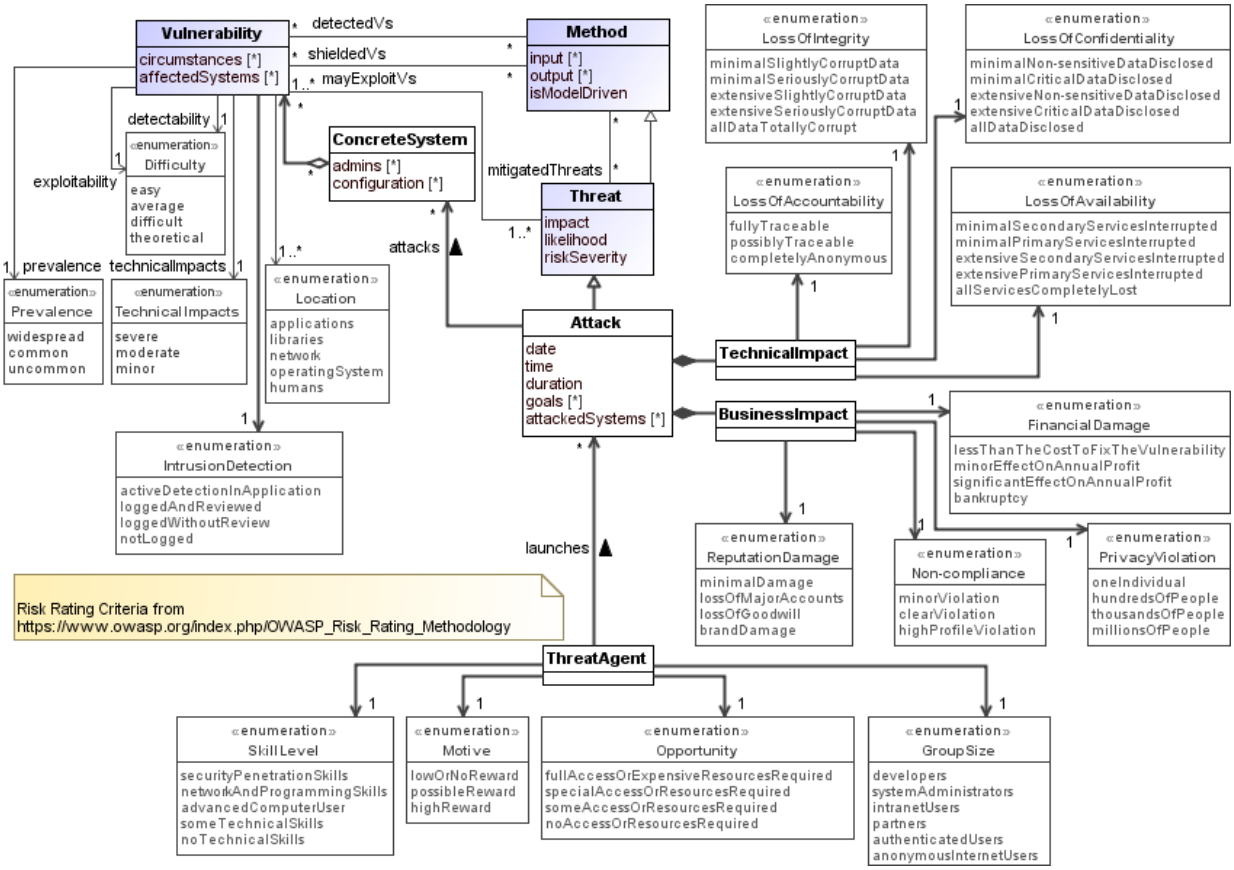


Figure 3.10: Inclusion of basic risk evaluation approach

the use of experimental approaches and SECEVAL is easy to extend, as shown in Figure 3.11: we introduce a `Test` class that is connected to at least one method and vice versa. The test uses the method on at least one example and is executed by `TestingParticipants`. Each participant assesses the method using Moody's evaluation criteria:

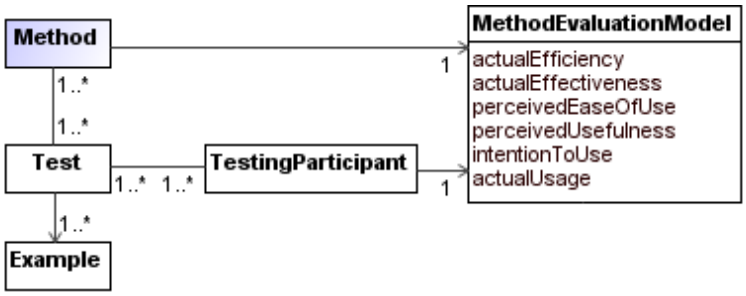


Figure 3.11: Method extension using Moody's method evaluation approach

- "Actual Efficiency: the effort required to apply a method.
- Actual Effectiveness: the degree to which a method achieves its objectives.
- Perceived Ease of Use: the degree to which a person believes that using a particular method would be free of effort.
- Perceived Usefulness: the degree to which a person believes that a particular method will be effective in achieving its intended objectives.

- Intention to Use: the extent to which a person intends to use a particular method.
- Actual Usage: the extent to which a method is used in practice" [51, page 5].

Usually, the average value of the participants' results is used as final evaluation result for the method under test.

Some misunderstandings occurred due to the brevity of the description that comes with the questionnaire, e.g., it was not clear to some partners that more than one phase of the SDLC can apply to a method or tool, using our context model. Furthermore, it was not pointed out explicitly in the questionnaire that a notation does not need attributes that are related to phases of the SDLC. These misunderstandings showed that it was good to be available personally at Malaga while the questionnaire was filled in.

As a proof of concept, we additionally use a case study in chapter 4.

4 Case Study on Security Testing of Web Applications using SECEVAL

Web applications came especially under fire in recent years, as they are available, and thus attackable, 24 hours a day from all regions over the world. With 27% of breaches within hacking, web applications of larger companies are a worthwhile target for hackers [69, p. 35].

An approach to harden web applications is to test for security flaws. Methods as “penetration testing” or “vulnerability scanning” are supported by many tools. Some tools are commercial, but also open-source and trial versions exist. In this chapter, we use our SECEVAL approach to evaluate vulnerability scanners for web applications. An excerpt of the data was collected as part of two student theses [60, 44].

4.1 Data Collection

First, we have to define what we want to know and how we plan to collect data, as shown in Figure 4.1. It depicts instances of the classes we have already defined in Figure 3.7, e.g., instances of the class `ResearchQuestion` define our two research questions, a high-level and a concrete one. We used identical background colors for instances of the same class and omitted `name` attributes in case an instance name (e.g., `p3`) is given in the header.

Research question `q1` is very general. Examples for methods which were selected in the first process phase and where more information was gathered in the second process phase are: vulnerability scanning, penetration testing, fuzzing and the classification into black- grey- and white-box testing. Examples for tools are WSFuzzer, X-Create and WS-Taxi, just to mention a few. As we already added most of the found methods and tools to the CBK [13], we focus on `q2` in this chapter.

Query instances could record the terms we used while searching the Web. We defined queries in the beginning of our research, but for such a broad question, as `q1`, we soon stopped recording concrete queries, as the overhead got too high in relation to the advantage of being able to reconstruct the search procedure at a later date. For recording our information sources, we used a table which contains the name of a method / tool and URLs from which information should be extracted in the data analysis phase.

Research question `q2` is a typical question which could be asked by security engineers working in a company. The “sources” (i.e., tools) we selected for analysis were (cf. [44]):

a) Acunetix Web Vulnerability Scanner[1]

b) Mavituna Security - Netsparker[2]

c) Burp Scanner[3]

d) Wapiti[4]

e) Arachni[5]

f) Nessus[6]

g) Nexpose[7]

h) Nikto[8]

[1] Acunetix. http://www.acunetix.com/
[2] Netsparker. https://www.mavitunasecurity.com/netsparker/
[3] Burp Scanner. http://portswigger.net/burp/scanner.html
[4] Wapiti. http://www.ict-romulus.eu/web/wapiti
[5] Arachni. http://www.arachni-scanner.com/
[6] Nessus. http://www.tenable.com/de/products/nessus
[7] Nexpose. https://www.rapid7.com/products/nexpose/
[8] Nikto. http://www.cirt.net/Nikto2

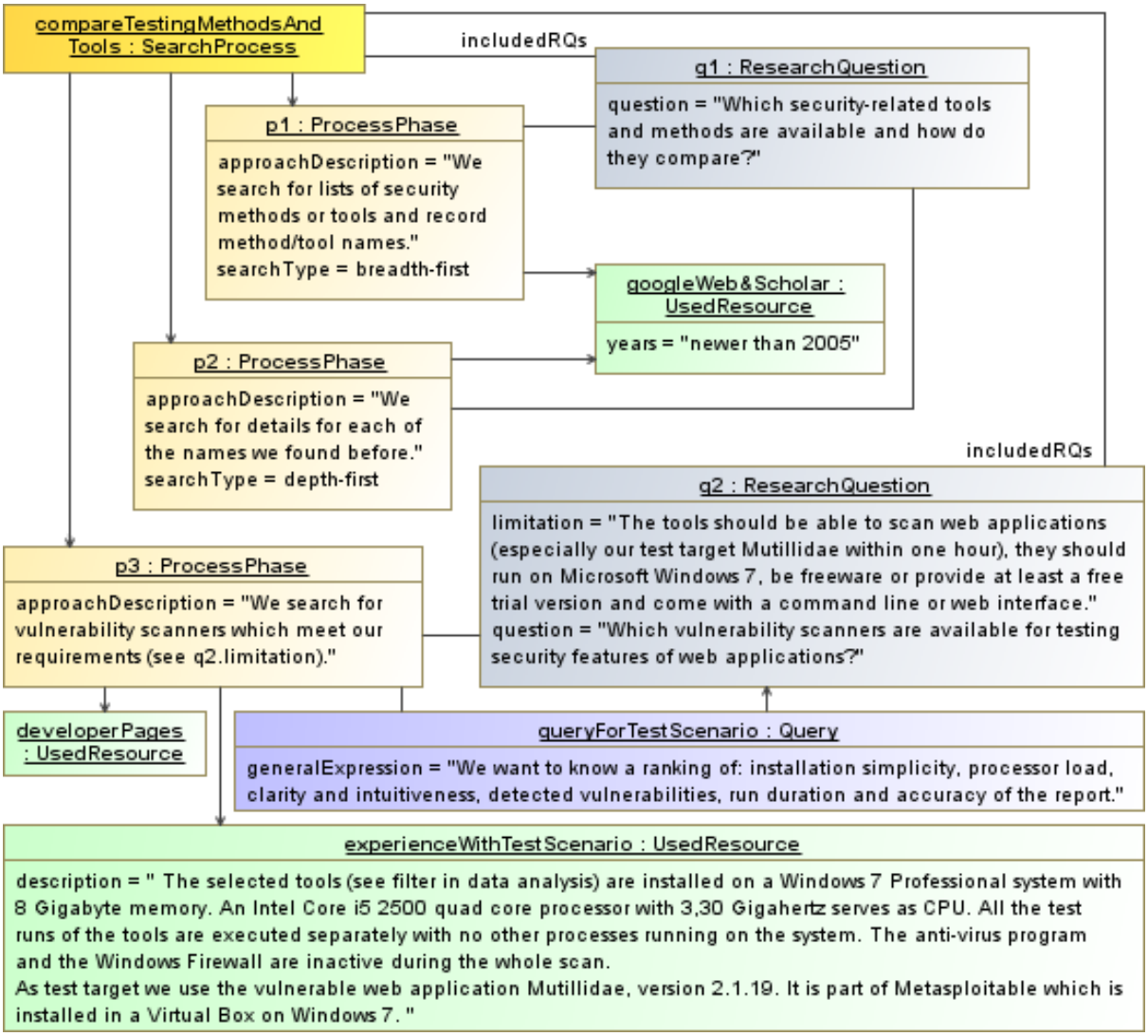


Figure 4.1: Case Study: Data Collection

The instance `experienceWithTestScenario` (cf. Figure 3.7) describes how the data collection is performed by testing the vulnerability scanners. Please note that SECEVAL does not impose the completion of the data collection phase before the data is analyzed. This means that the tests were partly executed on tools which were later classified as inappropriate. This becomes clear when we think of how evaluation works in practice: sometimes we have to collect a bunch of data before we observe information which, e.g., leads to the exclusion of a tool from the result set. Ideally, the already collected data is then also contributed to a knowledge base like the CBK (or a future implementation of SECEVAL's context model).

4.2 Data Analysis

For analyzing collected data we define an analysis strategy and select a filter which enforces the requirements (`limitations`) defined for question `q2`. Figure 4.2 depicts instances of the data analysis model we defined in Figure 3.9. The background colors of the instances are used to quickly recognize the type of an instance.

Before we go into detail about particular results of our experiments, we first take a look at the overall result regarding our research question `q2`. Figure 4.2 thus depicts an instance of the class `ProcessedInfo`,

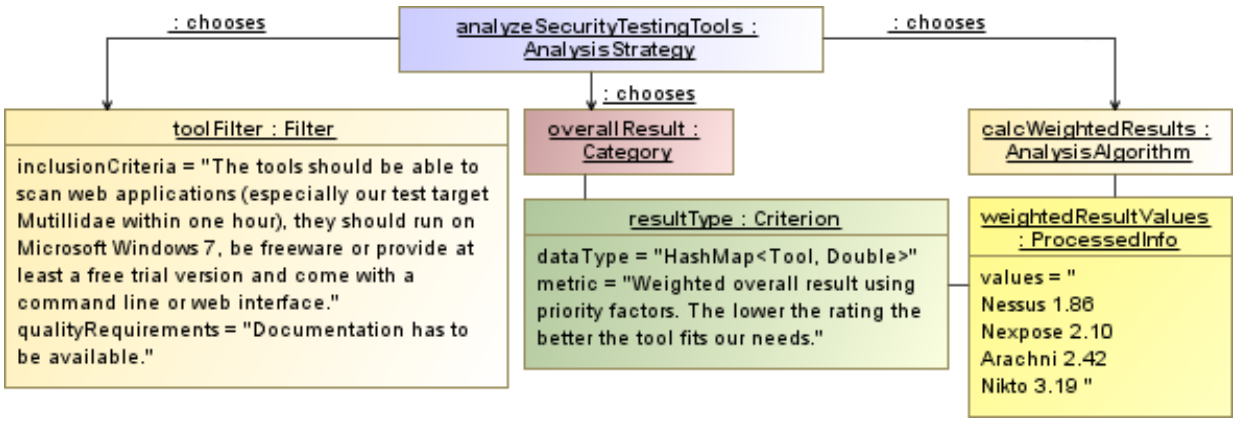


Figure 4.2: Case Study: Data Analysis – Results

which is called `weightedResultValues`. Only four tools passed our filter: Arachni and Nikto, which provide command-line interfaces and Nessus and Nexpose, which also use web interfaces. From our list of tools from above, the trial of $a)$ only allows to scan predefined sites. Tools $b)$ and $c)$ do not support a command line or web interface in the versions that are free. A run of tool $d)$ on our test target Multidae[9] took six hours.

Apart from information we gathered online, we experimented with the tools that passed the filter, in order to obtain data for our tool evaluation (q2). Additionally to the instance `queryForTestScenario`, we describe what we test in more detail in the following (from [44]):

- **Installation simplicity**
  Do any problems occur during the installation?
- **Costs**
  How much do the tools cost? Is it a one-time payment or an annual license?
- **Processor load**
  How high is the CPU load while running the tool?
- **Clarity and intuitiveness**
  Is the tool easy to understand, clearly structured and user-friendly?
- **Run duration**
  How long does a scan take?
- **Quality of the report**
  How detailed is the report of the scan? Which information does it contain?
- **Detected vulnerabilities**
  How many vulnerabilities does the tool detect on our test environment?

As we can see in Figure 4.2, an algorithm is involved, which calculates results according to a rating. The rating is depicted in Figure 4.3. Lower factors of a criterions' `priority` denote that we consider the criterion less important. The data used for the rankings result from our test and is not further processed, therefore we do not need an algorithm (that is why we use instances of the class `extractedInfo`).

Additionally, we show how intermediate values of our tests could be described in our data analysis model in Figure 4.4, as e.g., the costs of the tools or the operating systems it runs on. Concrete instances of `Information` classes are not depicted; the interested reader is referred to [44].

To sum up, Table 4.1 gives an overview of our results. It contains the ranking of the tools as well as average[10] and weighted[11] results.

[9] NOWASP (Mutillidae). http://sourceforge.net/projects/mutillidae/
[10] AVG: average
[11] WAVG: weighted average according to ratings

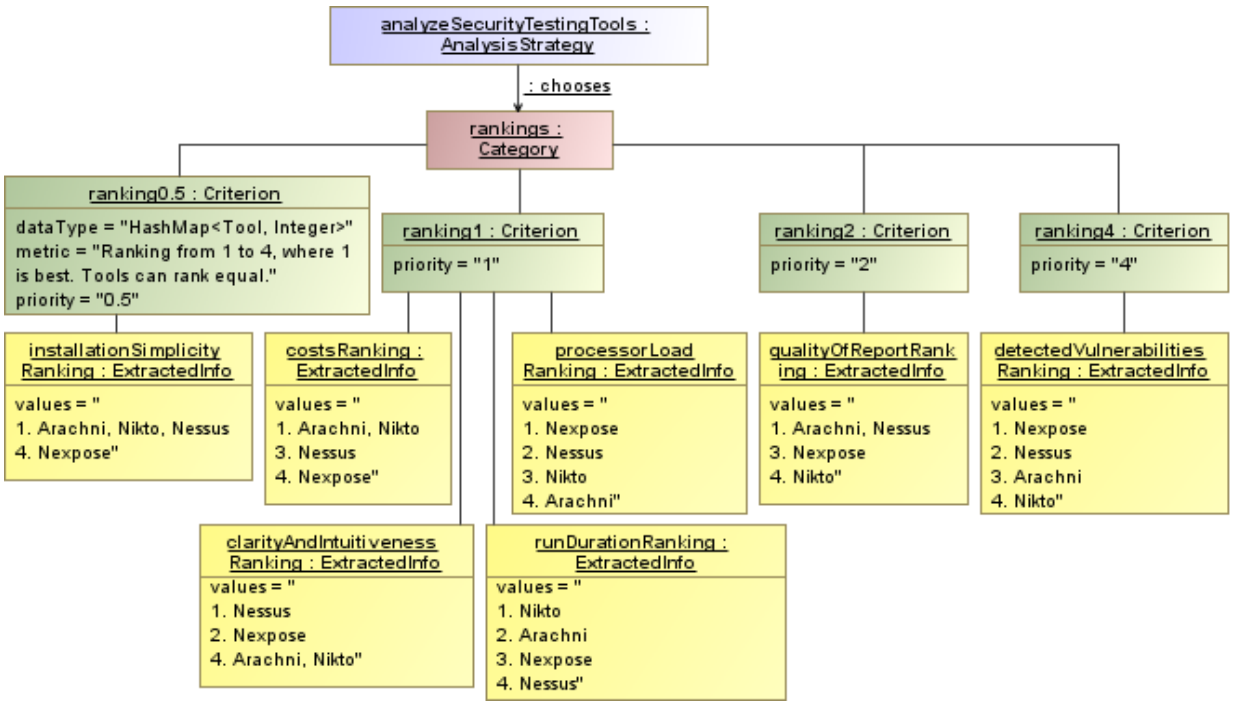


Figure 4.3: Case Study: Data Analysis – Ratings

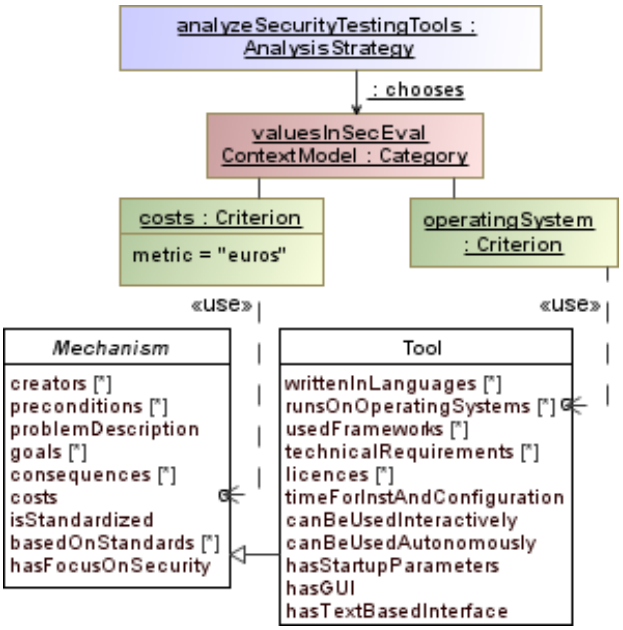


Figure 4.4: Case Study: Data Analysis – Values

Tool	Inst.	Costs	CPU	Clarity	Time	Vuln.	Report	AVG[10]	WAVG[11]
Nessus	1	2	2	1	4	1	2	1,86	1,86
Arachni	1	1	4	4	2	1	3	2,29	2,42
Nexpose	4	4	1	2	3	3	1	2,57	2,10
Nikto	1	1	3	4	1	4	4	2,57	3,19

Table 4.1: Case Study: Final Tool Ranking (adapted from [44])

4.3 Security Context Model

To allow security engineers to easily access the data we collected, we added entries for Nessus, Arachni, Nexpose and Nikto to the CBK [13]. Besides, we integrated all four tools into the NESSoS tool workbench, called SDE [11]. If they are executed from within the SDE, URL and port of the web application under test have to be provided by the user. To try out the vulnerability scanners it is possible to use Multidae as a target, as we did above. Multidae comes with Metasploitable[12], an intentionally vulnerable Linux virtual machine. Therefore, the default configuration for the integrated SDE tools point to a local Multidae instance, but can be changed at any time.

However, the CBK does not provide fine-grained categories for entering security-specific information. As SECEVAL's context model is more detailed, we modeled the context of vulnerability scanning of web applications and two of the tested tools: Nessus and Nikto. Figure 4.5 shows an instance diagram of the context model, which we have already depicted in Figure 3.3.

Connections between vulnerability and security property instances are not depicted, but stored in the model, which can be downloaded from the Web.[13] The three vulnerabilities that are modeled are the top 3 from OWASP's top 10 project 2013. [23] Of course, the shown method is not the only one which is supported by the tools, as e.g., Nessus also supports vulnerability scans on networks and not only on web applications. This would be the main advantage of an implementation of SECEVAL, similar to the CBK: connections to elements (as methods) which exist, but are not important for the current research, could be added without much effort so that future researchers can build upon them.

We based our case study on the non-extended version of SECEVAL. For the tools' examples, we saw that it would be nice to extend the model to represent further attributes, as run duration or processor load. We recommend using additional classes for those extensions, e.g., a class to detail a test run. Our experience using SECEVAL and the UML CASE tool MagicDraw was that modeling is easy, but that the layout is not inviting to read the containing information. This is mainly because the order of instances attributes cannot be changed and because the font remains pixelated. Consequently, we are looking forward to a future implementation of SECEVAL as a kind of semantic wiki.

[12] Metasploitable. http://www.offensive-security.com/metasploit-unleashed/Metasploitable

[13] SecEval: Modeling Example (Testing). http://www.pst.ifi.lmu.de/~busch/SecEval/SecEval.mdzip

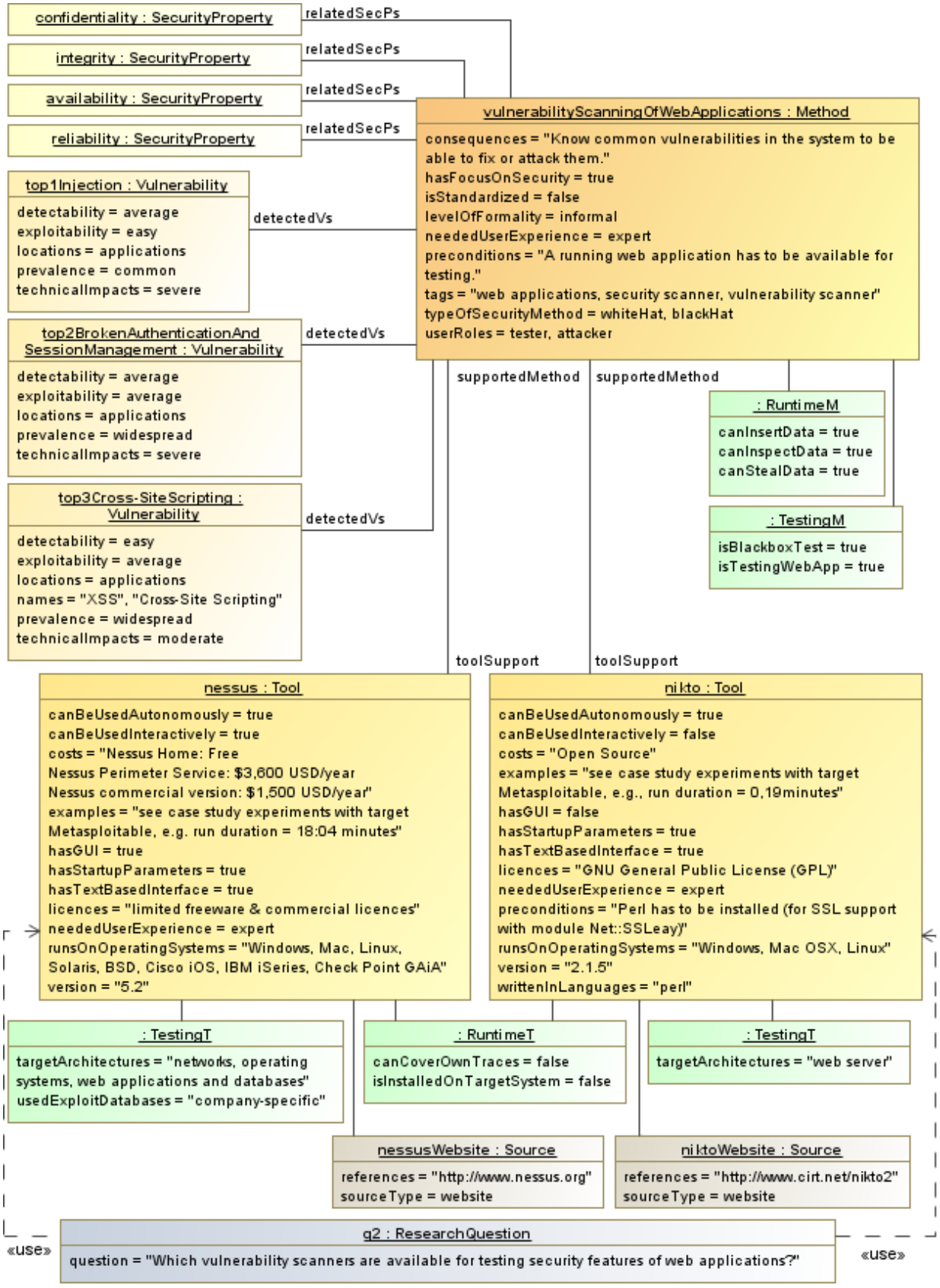


Figure 4.5: Case Study: Instances of Context Model (excerpt)

5 Domain-specific Evaluation of Security Mechanisms

Our evaluation framework SECEVAL is brand-new. The following two evaluations in the domains of security design [67] and risk [42, 43] are previous works. Therefore, they do not explicitly use our approach, but have influenced the construction of SECEVAL.

The first evaluation is a mapping study on secure software design. It aims at getting an overview of existing approaches to secure software design and at comparing their properties. A mapping study [58] is a kind of systematic literature review [39] where publications are categorized so that the coverage of the chosen research area can be depicted.

The second evaluation compares two risk-based methods: visual methods, represented by CORAS [45] and textual methods, represented by SREP [48]. The study is based on the systematic literature review by Kitchenham et al. and the experiment uses, beyond others, Moody's approach [51] for recording effectiveness, perceived ease of use, perceived usefulness and intention to use.

5.1 Mapping Study on Secure Software Design

The incorporation of security concerns in software design has become a popular research topic over the last decade. Numerous researchers have proposed a wide variety of approaches to secure software design. Unfortunately these approaches have been developed mostly independent from each other. This results in a complex tangle of different approaches.

In this section we report on a mapping study (a type of systematic literature review) we performed to disentangle the domain of secure software design. [67] The main goals of this study are first to provide an overview of the current state of the art. Second, identify gaps in current research, leading to interesting research opportunities.

5.1.1 Review method

Our study has been designed following the guidelines of Kitchenham and Charters [39]. The following paragraphs shortly describe the most important aspects of the study.

Research questions

In this mapping study we address the following research questions.

RQ1: What approaches to secure software design exist?

RQ2: What security properties are supported during software design?

RQ3: What type of support is provided for the security properties?

RQ3.1: Is a representation supported?

RQ3.2: Is an analysis supported?

RQ4: What evaluation is provided for the approaches?

Software design refers to both architectural and detailed design. Security properties are properties as, for example confidentiality and authentication. The full set of properties used in this study is shown later. A representation includes each explicit modeling, graphically or textually, of a security property. Analysis includes each verification of security properties against the security requirements of the system under design. An evaluation includes every application of an approach with the purpose of demonstrating and/or proving its use.

Selection criteria

There is a need to define which research will (not) be used to answer the above research questions. Research works are included if they satisfy the following inclusion criteria.

- The research work models security properties in software design or
- analyzes security properties in software design or
- models attacks or threats in software design or
- evaluates a proposal described in an included research work.

Research works are excluded if they satisfy the following exclusion criteria.

- The research work does not satisfy the inclusion criteria or
- is not published as a book or A-tier[1] publication or
- only mentions security as a general introductory term or
- is not available as a full version, only an extended abstract or presentation, or
- is a duplicate of an included research work or
- is superseded by an included research work or
- is published more then ten years ago.

We limited the included research works to books and A-tier publications because of the sheer number of research works available in this domain. Reading and analyzing all available research works was not considered feasible due to time and resource constraints.

In the case of duplicate research works only the most extensive or most recent, when they are all equally extensive, research work is included. We limited the scope to ten years since older research works either have been developed further, and are thus included by later research works, or have become too outdated for current technology.

Data extraction

The data extracted from the research works to answer the research questions can be divided in two main dimensions. First the security dimension classifies each research work over the security properties it supports and how they are supported (Figure 5.1).

The security properties are divided in two groups. On one hand the declarative properties which are the known CIAA security concerns. On the other hand the operational mechanisms are those mechanisms used to achieve the declarative properties.

The support for security concerns is divided in two aspects. A representation is constructive if it is actively used (e.g., used to generate implementation logic) further on in the development cycle. Otherwise it is a documentation representation. Analysis is precise if no expert security knowledge is needed to use it, otherwise it is imprecise. A white hat analysis proves a positive fact, for example a policy ensures confidentiality. A black hat analysis proves a negative fact, for example a system is not vulnerable to a certain attack.

Second the evaluation dimension classifies each evaluation discussed in the research works (Figure 5.2).

An industrial application is any evaluation in an industrial sized project. A researcher or student study is an evaluation in which researchers or students apply the approach to a test case. Toy examples are evaluations used to illustrate an approach.

Case studies are evaluations described in high level of detail, usually containing a thorough analysis of good and bad aspects, and a lesson learned element. Illustrations are evaluation described limited to the actual use of the approach, lacking any in-depth analysis. Mentioned evaluations lack any detail in their description.

[1] See the Excellence in Research for Australia (ERA) 2010 initiative (http://www.arc.gov.au/era/era_2010/archive/era_journal_list.htm) for more information on this ranking.

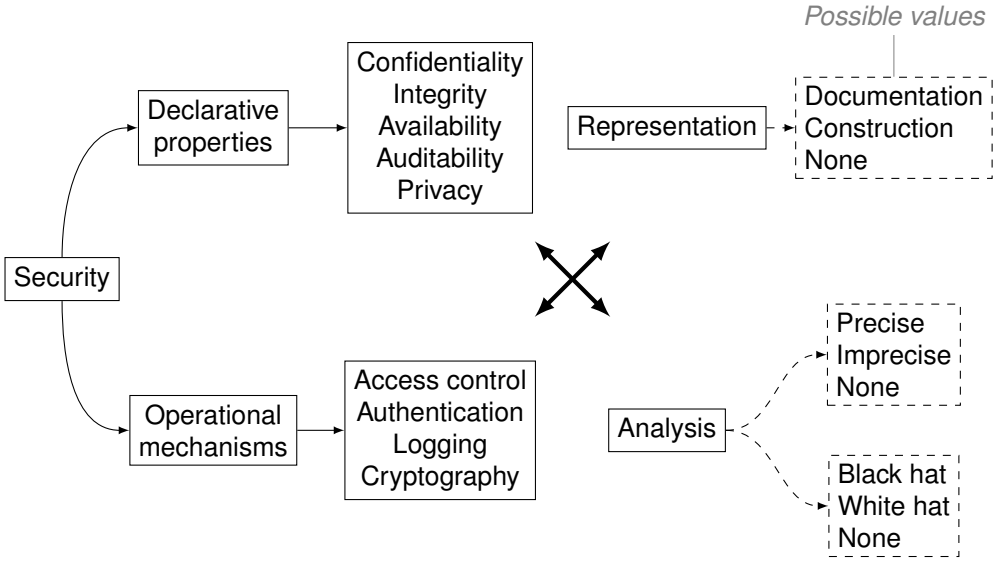


Figure 5.1: The security dimension classifies each research work over what and how security properties are supported.

5.1.2 Discussion of the research questions

This following paragraphs provide answers to the research question based on the data extracted according to the dimensions described above.

RQ1: What approaches to secure software design exist?

A total of 21 different approaches were discovered in the included research works. Table 5.1 shows an overview of the approaches and corresponding research works. Approaches for which the original authors did not specify a name were named after the main author combined with a suffix.

RQ2: What security properties are supported during software design?

Looking at what security properties are supported by each approach we see there is a strong focus on access control (Figure 5.3). Other security properties are considerably less supported and some are not supported at all.

A possible explanation for the popularity of access control the existence of well-known standards such as role-based access control (RBAC) to achieve this specific property. Such standards make it more straightforward to develop approaches supporting access control.

Looking at the number of security properties supported per each approach we see 12 out of 21 approaches, support only a single security property (Figure 5.4). Of these 12 approaches 10 support only access control. At most five security protocols are supported by an approach. This indicates a strong specialization among approaches.

RQ3: What type of support is provided for the security properties?

Most approaches limit their support to a representation of security properties. Only a minority (also) support analysis of their security properties (Figure 5.5).

Taking the type of representation into consideration shows that there is no significant difference in amount of support (Figure 5.6). Approaches do stick to one kind of representation for all supported properties, this is likely related to the fact the supported number of security properties is rather low.

Looking at the types of analysis the most significant is that most approaches support a precise analysis. A precise analysis requires no expert security knowledge making it considerably more practical.

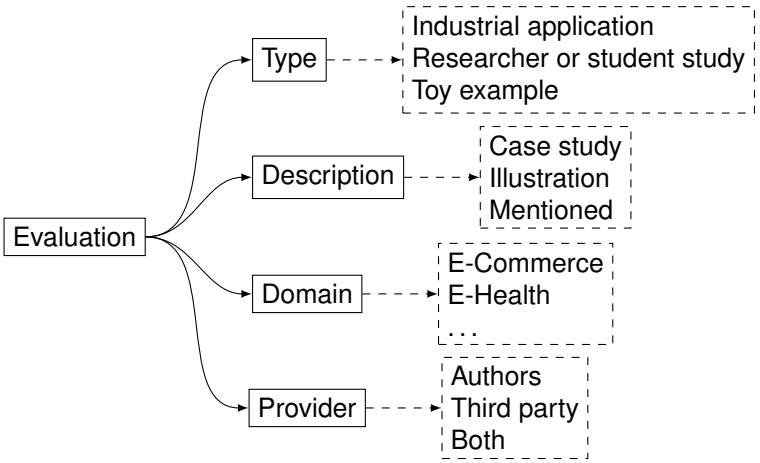


Figure 5.2: The evaluation dimension classifies each evaluation discussed in a research work over its type, level of description, domain and provider.

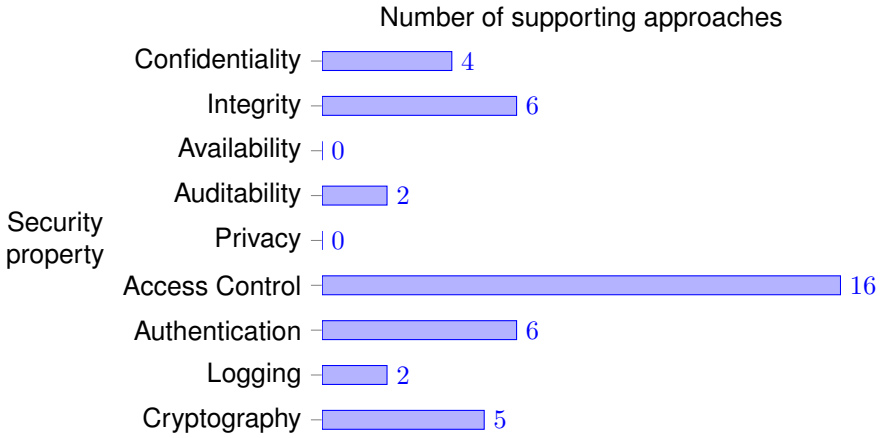


Figure 5.3: Access control is supported by considerably more approaches than other security properties.

RQ4: What evaluation is provided for the approaches?

The research works describe a total of 42 evaluations, most of which are illustrative toy examples (Figure 5.7). Only ten industrial applications have encountered, of which only 2 are described as case studies. Researcher or student studies are, despite the growing popularity of empirical research, still very rare.

Looking from the perspective of the approaches we can see that most approaches only provide toy examples as evaluation (Table 5.3). Only five approaches describe, in varying levels of detail, other types of evaluations. Furthermore there is one approach (Sohr-AC) that does not provide any evaluation. It should also be noted that seven out of ten industrial applications belong to a single approach (UMLsec).

Furthermore, all but three evaluations are provided by the authors of the approach. The three exceptions are industrial applications belonging to the same approach (UMLsec) and are provided by the authors in collaboration with an industrial partner.

5.1.3 Identified Gaps

Based on the above answers to the research questions a number of gaps can be identified. Each gap offers one or more research opportunities.

Approach	Research works
Abramov-AC	[1]
ADM-RBAC	[19]
Breu-WS	[29]
FDAF	[15, 16]
Georg-AO	[25, 24]
Giordano-AC	[26]
Gomaa-UML	[28]
Kim-AC	[38]
Koch-AC	[40]
Medina-DB	[21]
Nakamura-DB	[55, 59]
RBAC-WS-BPEL	[8]
SecureSOA	[49, 50]
SecureUML-MDS	[3]
SoaML4Security	[41]
Sohr-AC	[62]
UMLS-UMLsProfile	[30]
UMLsec	[34, 36, 35, 9, 37]
Vela-DB-XML	[68]
Wolter-BP	[72]
Xu-Petri	[73]

Table 5.1: Overview of the 21 approaches discovered in the literature.

Analysis	White hat	Black hat	Total
Precise	2	3	5
Imprecise	1	0	1
Total	3	3	6

Table 5.2: Most approaches provide a precises analysis, whereas an imprecise' analysis is not often encountered.

Security property coverage

The gap in security property coverage can be seen in two ways. First, a number of security properties are barely or not supported. Second, most approaches only support a limited number of security properties. Both views give rise to the fact that no single approach, or combination of approaches, can be used to incorporate all security properties in software design. This offers the research opportunity to extend existing approaches to more (unsupported) security properties or define more generally applicable (i.e., support for all or most security properties) approaches.

Analysis support

Only a minority of the approaches support analysis of their security properties. Therefore, designers often cannot verify whether their design meets the security requirements and must delay this verification until the implementation or deployment phases, where correcting errors is much more costly. Obviously, this is a research opportunity for extending approaches with analysis support or developing new analysis methods.

Evaluation

Most approaches are evaluated by the authors themselves using illustrative toy examples. Since toy examples are typically small and, to a certain degree, designed to fit well, they provide limited information

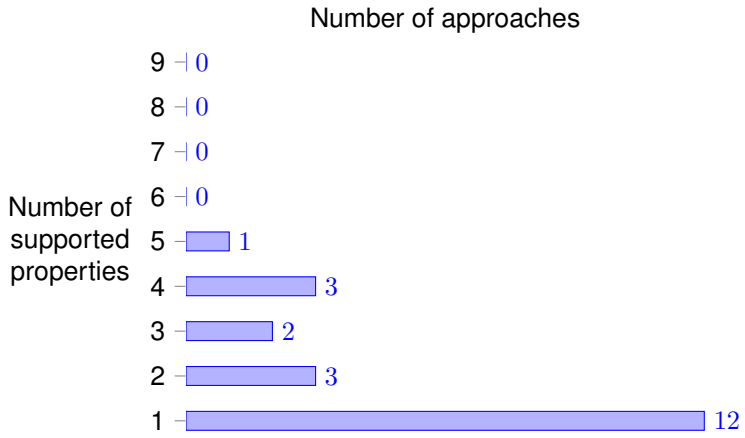


Figure 5.4: A majority of 12 approaches support only one security property.

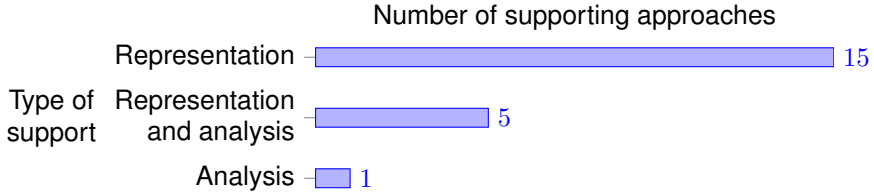


Figure 5.5: Most approaches only support analysis for their security properties.

on the practical usability of an approach.

Unfortunately performing an evaluation on an industrial application is often difficult. Empirical studies can fill this gap by providing more thorough evaluations.

5.1.4 Limitations of this study

The main limitation of this study is that only books and A-tier publications are considered. This leads to a substantial amount of research works left uncovered. This can result in possibly biased answers to the research questions.

Since we explicitly choose the top-level research works we believe the selection contains the more mature and advanced approaches available. The major gaps observed in this study (i.e., lack of a generally applicable approach and lack of thorough evaluation) are not likely to be solved by less mature or advanced approaches. Furthermore it is our observation, based on a partial analysis, that approaches described in the other research works tends to follow the same lines as those included.

5.2 Empirical Validation of Risk-based Methods

Several methods have been proposed to address security concerns during the early phases of the system development life cycle [48, 45, 53, 27, 18, 61]. However, there has been little empirical evaluation that shows how effective these methods are in practice. With few exceptions [57, 46, 52, 47], security methods are evaluated by the same researchers who have proposed them. As a consequence, security practitioners are not motivated to adopt new security methods, while researchers do not know how to improve their methods. To address this problem, there is thus the pressing need of conducting empirical evaluations to investigate which methods work better to identify threats and mitigations (i.e., security requirements for later phases) and why.

In this section, we report a controlled experiment that we conducted to compare two classes of risk-based methods [42, 43]: visual methods and textual methods. As instances of these classes of methods,

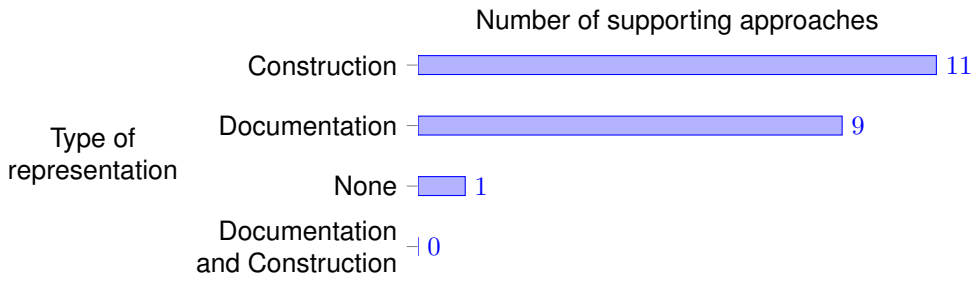


Figure 5.6: There is no significant difference is amount of support between representation types.

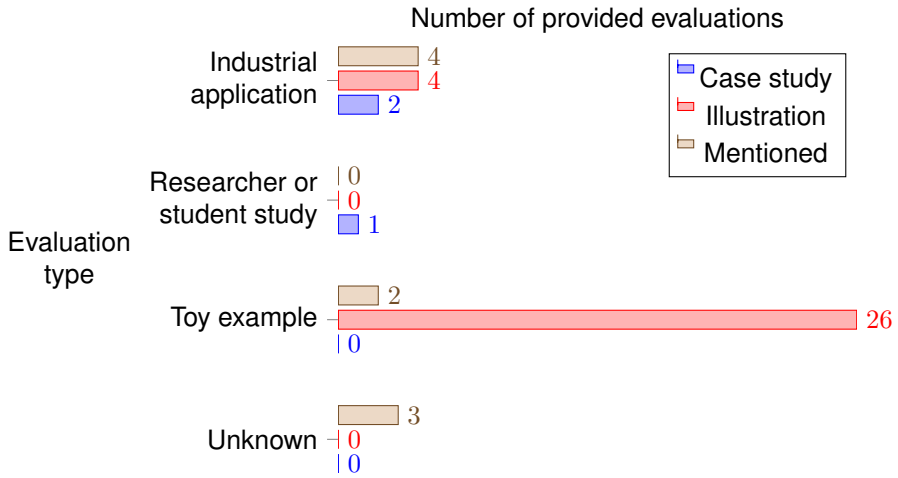


Figure 5.7: The large majority of evaluations found in literature are illustrative toy examples.

we have selected CORAS [45] and SREP [48]. CORAS is a *visual method* whose analysis is supported by a set of diagrams that represent assets, threats, risks and treatments. In contrast, SREP is a *textual method* whose artifacts are specified in natural language or in tabular form.

The goal of the experiment was to evaluate the *effectiveness* of visual and textual based methods, and the participants' *perception* of the methods, according to Moody [51]. Hence, the dependent variables were the *effectiveness* of the methods measured as number of threats and security requirements and the participants' *perceived ease of use*, *perceived usefulness* and *intention to use* of the two methods. The independent variable was the *method*. The experiment involved 28 participants: 16 students of the master in Computer Science and 12 students of the EIT ICT LAB master in Security and Privacy. They were divided into 16 groups using a randomized block design. Each group applied the two methods to identify threats and security requirements for different facets of a Smart Grid application scenario (ranging from security management to database security). The experiment was complemented with participants' interviews to gain insights on *why* the methods are effective or they are not.

The main findings are that the visual method yields to identify more threats than textual one, while the textual one is slightly better to identify security requirements. The difference in the number of threats identified with the two methods is statistically significant and participants' interviews suggest that this is due to the difference in the artifacts used to model threats. The visual method uses diagrams to represent threats while the textual method uses tables: diagrams help brainstorming on threats and thus yield participants to identify more threats. On the contrary, the difference in the number of security requirements identified with the two methods is not statistically significant. The textual method identified a slightly higher number of security requirements but this is not statistically significant. A possible explanation emerging from the interviews is that process supported by the textual method offers a systematic approach to identify security requirements. In addition, the visual method's overall perception and intention to use are higher than for the textual method.

Approach	Industrial application			Researcher or student study			Toy Example		
	C	I	M	C	I	M	C	I	M
Abramov-AC	0	0	1	0	0	0	0	1	0
ADM-RBAC	0	0	0	1	0	0	0	1	0
Breu-WS	0	0	0	0	0	0	0	1	0
FDAF	0	0	0	0	0	0	0	2	0
Georg-AO	0	0	0	0	0	0	0	2	0
Giordano-AC	0	1	0	0	0	0	0	0	0
Gomaa-UML	0	0	0	0	0	0	0	1	0
Kim-AC	0	0	0	0	0	0	0	2	0
Koch-AC	0	0	0	0	0	0	0	1	0
Medina-DB	0	0	0	0	0	0	0	1	0
Nakamura-SOA	0	0	0	0	0	0	0	2	0
RBAC-WS-BPEL	0	0	0	0	0	0	0	1	0
SecureSOA	0	0	0	0	0	0	0	1	1
SecureUML-MDS	0	0	0	0	0	0	0	1	1
SoaML4Security	0	1	0	0	0	0	0	0	0
Sohr-AC	0	0	0	0	0	0	0	0	0
UMLS-UMLsProfile	0	0	0	0	0	0	0	1	0
UMLsec	2	2	3	0	0	0	0	4	1
Vela-DB-XML	0	0	0	0	0	0	0	1	0
Wolter-BP	0	0	0	0	0	0	0	1	0
Xu-Petri	0	0	0	0	0	0	0	1	0

C = Case study, I = Illustration, M = Mentioned

Table 5.3: Most of the approaches are limited to toy examples as evaluation. Seven out of ten industrial applications belong to the same approach.

5.2.1 Research method

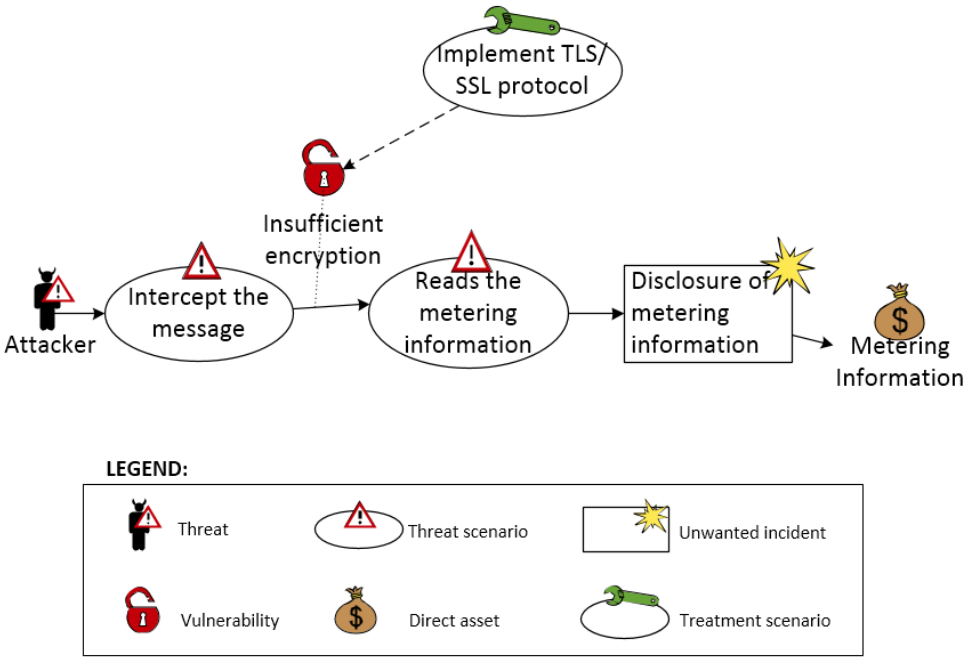
This section describes the design of the performed experiment, following the guidelines by Wohlin et al. [71].

Selection of methods

CORAS is a visual method which consists of three tightly integrated parts, namely, a method for risk analysis, a language for risk modeling, and a tool to support the risk analysis process. The risk analysis in CORAS is a structured and systematic process which use diagrams (see Figure 5.8(a)) to document the result of the execution of each step. The steps are based on the international standard ISO 31000 [31] for risk management: context establishment, risk analysis (that identifies assets, unwanted incidents, threats and vulnerabilities), and risk treatments.

The Security Requirements Engineering Process (SREP) is an asset-based and risk-driven method for the establishment of security requirements in the development of secure Information Systems. SREP supports a micro-process, consisting of nine steps: agree on definitions, identify critical assets, identify security objectives, identify threats and develop artifacts, risk assessment, elicit security requirements, categorize and prioritize security requirements, requirements inspection, and repository improvement. The result of the execution of each step of the process is represented using tables or natural language (see Figure 5.8(b)). SREP is compliant with international standards ISO/IEC 27002 [33] and ISO/IEC 15408[32] within the scope of requirements engineering and security management.

For additional details about CORAS and SREP we refer the reader to [45, Chap. 3] and [48]. Note that, in the rest of the paper, we denote with “security requirements” both the concepts “treatments” in CORAS and “security requirements” in SREP because they have the same semantic: they are both defined as a



(a) CORAS - Threat Diagram

Name of Misuse Case: Spoof of information		
ID 1		
Summary: the attacker gains access to the message exchange between the SM and SNN and disclose the secret exchange of information		
Probability: Frequent		
Preconditions: 1) The attacker have access to the communication channel between SM and SNN		
User Interactions	Misuser interactions	System Interaction
The SM sends the information about power consumption		
	The attacker reads the information	
		The SSN receives the information without knowing that someone have read the message
Postconditions: 1) The attacker knows personal information about the power consumption of the customer		

(b) SREP - Threat Specification using misuse cases

Figure 5.8: Examples of Visual (CORAS) and Textual (SREP) Methods' Artefacts.

means to reduce the risk level associated with a threat.

Research approach

The *goal* of the experiment was to evaluate and compare two types of risk-driven methods, namely, visual methods (CORAS) and textual methods (SREP) with respect to their *effectiveness* in identifying threats and security requirements, and the participants' *perception* of the two methods. Hence, visual and textual methods were the two treatments that we have considered in the experiment. We want to investigate the following research questions:

RQ1 *Is the effectiveness of the methods significantly different between the two type of methods?*

RQ2 *Does the effectiveness of the methods vary with the assigned tasks?*

RQ3 *Is the participants' preference of the method significantly different between the two type of methods?*

RQ4 *Is the participants' perceived ease of use of the method significantly different between the two type of methods?*

RQ5 *Is the participants' perceived usefulness of the method significantly different between the two type of methods?*

RQ6 *Is the participants' intention to use the method significantly different between the two type of methods?*

To answer the first two research questions we have measured *effectiveness* by counting the number of threats and the number of security requirements as the main outcomes of the methods' application (as done in [57, 65]). Research questions $RQ3 - RQ6$ have been answered by measuring perception-based variables *perceived usefulness* (PU), *perceived ease of use* (PEOU), *intention to use* (ITU) with a post-task questionnaire. In order to gain a better understanding of *why a method is effective* (or more effective than another) we also carried out individual interviews with the participants.

Hypotheses

We have translated research questions $RQ1 - RQ6$ into a list of null hypotheses to be statistically tested. Due to the lack of space we report here only the main alternative hypotheses to the null ones denoted as Hn_A where n specifies the research question to which the hypothesis is related and the index A specifies that is an alternative hypothesis.

$H1.1_A$ There will be a difference in the number of threats found with the visual method and with the textual method

$H1.2_A$ There will be a difference in the number of security requirements found with the visual method and with the textual method

$H2.1_A$ There will be a difference in the number of threats found with the visual and the textual method within each facet

$H2.2_A$ There will be a difference in the number of security requirements found with the visual and the textual method within each facet

$H3_A$ There will be a difference in the participants preference for the visual and the textual method

$H4_A$ There will be a difference in the participants perceived ease of use for the visual and the textual method

$H5_A$ There will be a difference in the participants perceived usefulness for the visual and the textual method

$H6_A$ There will be a difference in the participants intention to use for the visual and the textual method

Facet/Method	Visual	Textual
Mgmnt	6	10
App/DB	9	7
Net/Teleco	9	7
Mobile	8	8

Table 5.4: Experimental design

Hypotheses $H1.1_A$-$H1.2_A$ are related to $RQ1$ and suppose that there will be a difference in the effectiveness of the methods. $H2.1_A$-$H2.2_A$ assume a possible relation between the effectiveness of the methods and the facets on which the methods is applied (RQ2). Hypothesis $H3_A$ assumes there will be a difference in the participants' overall preference for the methods (RQ3). $H4_A$-$H6_A$ assume that the participants' perceived ease of use, perceived usefulness, and intention to use variables will differ for the two methods (RQ4-RQ6).

Experimental design

Participants for the experiments were recruited among master students enrolled in the Security Engineering course at the University of Trento. The participants had no previous knowledge of the methods under evaluation. A within-subject design where all participants apply both methods was chosen to ensure a sufficient number of observations to produce significant conclusions. In order to avoid learning effects, the participants had to identify threats and mitigations for different types of security facets of a Smart Grid application scenario. The Smart Grid is an electricity network that can integrate in a cost-efficient manner the behavior and actions of all users connected to it like generators, and consumers. They use information and communication technologies to optimize the transmission and distribution of electricity from suppliers to consumers.

The tasks differ in the security facets for which the groups had to identify threats and security requirements. The security facets included Security Management (Mgmnt), Application/Database Security (App/DB), Network/Telecommunication Security (Net/Teleco), and Mobile Security (Mobile). For example, in the App/DB facet, groups had to identify application and database security threats like cross-site scripting or aggregation attacks and propose mitigations.

The participants were divided into 16 groups so that each group would apply the visual method (CORAS) to exactly two facets and the textual method (SREP) to the remaining two facets. For each facet, the method to be applied by the groups was randomly determined. Table 5.4 shows for each facet the number of groups assigned to visual and textual methods.

Experimental Procedure

The experiment was performed during the Security Engineering course held at University of Trento from September 2012 to January 2013. The experiment was organized in three main phases:

- **Training**. Participants were given a tutorial on the Smart Grid application scenario and a tutorial on visual and textual methods of the duration of two hours each. The Smart Grid scenario focused on the gathering of metering information from the smart meters and their transmission to the utility services for billing purposes. Then, participants were administered a questionnaire to collect information about their background and their previous knowledge of other methods and they were divided into groups based on the experimental design.

- **Application**. Once trained on the Smart Grid scenario and the methods, the groups had to repeat the application of the methods on four different facets: Security Management, Application/Database Security, Network Security and Mobile Security. For each facet, the groups:

  - Attended a two hours lecture on the threats and possible mitigations specific for the facet but not concretely applied to the case study.

- Had one week to apply the assigned method to identify threats and security requirements specific for the facet.
- Gave a short presentation about the preliminary results of the method application and received feedback.
- Had one week to deliver an intermediate report to get feedback.

At the end of the course in mid-January 2013, each group submitted a final report documenting the application of the methods on the four facets.

- **Evaluation**. In this phase, the experimenters (the authors of this paper) assessed participants' final reports while the participants evaluated the method through questionnaires and interviews. First, each group gave a presentation summarizing their work in front of the experimenters and of the expert. The expert evaluated the quality of the threats and the mitigations proposed for the Smart Grid application scenario. Then, participants were administered the post-task questionnaire to be filled in online. Last, each participant was interviewed for half an hour by one of the experimenters to investigate which are the advantages and disadvantages of the methods.

The interview guide contained open questions about the overall opinion of the methods, their advantages and disadvantages, the difficulties encountered during the application of the methods and the main differences among them. The interview questions were the same for all the interviewees even though some specific questions were added for some of the participants when their answers to the questionnaire were contradictory.

The questionnaire was adapted from the questionnaire reported in [57] which was inspired to the Technology Acceptance Model (TAM) [17]. The questionnaire consisted of 22 questions which were formulated in an opposite statements (positive statement on the right and negative statement on the left) format with answers on a 5-point Likert scale. The questions were formulated as follows: Q1: Whether the method was easy or hard to use; Q2: The method made the security analysis easier or harder than an ad hoc approach; Q3: The method was easy or difficult to master; Q4: Intention to use the method to identify threats and security requirements in a future project course; Q5: The method is better in identifying threats and security requirements than using common sense; Q6: Intention to use the method to identify threats and security requirements in a future project at work; Q7: Confusion about how to apply the method to the problem; Q8: Whether the method made the search for threats and security requirements more or less systematic; Q9: Intention to use the method if suggested by someone at work; Q10: Whether the method would be easy or hard to remember; Q11: Whether the method makes more or less productive in identifying threats and security requirements; Q12: Intention to use the method in a discussion with a customer; Q13: Whether the process of the method is well or not well detailed; Q14-Q15: A catalog of threats and security requirements makes easier or harder the security analysis with the method; Q16-Q17: The method helps or not helps in brainstorming on the threats and the security requirements; Q18: Whether the tool is easy or hard to use (asked just for the visual method because it had tool support); Q19-Q22: Difficulties of facets. To avoid that the participants answered on "auto-pilot", some of the questions (e.g., Q2, Q10, Q13) were given with the most positive response on the left and the most negative on the right.

5.2.2 Reports' Analysis

Coding

To assess the effectiveness of visual and textual methods, the final reports delivered by the groups were coded by the authors of this paper to count the number of threats and security requirements. An expert on security of the Smart Grid was asked to assess the quality of the threats and security requirements. The level of quality was evaluated on a four item scale: *Unclear* (1), *Generic* (2), *Specific* (3) and *Valuable* (4).

Based on this scale, the groups who have got an assessment *Valuable* or *Specific* were classified as *good groups* because they have produced threats and security requirements of good quality. On the contrary, the groups who were assessed *Generic* or *Unclear* were considered as not so good (bad) groups.

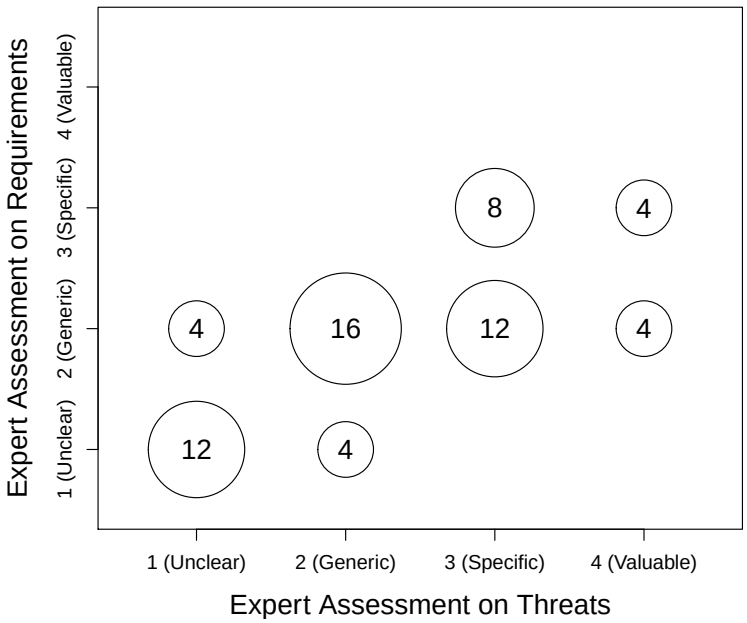


Figure 5.9: Expert assessment.

Figure 5.9 reports the expert assessment of all groups for all facets. In total we had 64 method applications because each of the 16 groups has applied one of the methods on the four facets. The number inside each bubble denotes the number of method applications which got a given expert's assessment for threats (reported on x-axis) and security requirements (reported on y-axis). There were 48 (75%) method applications that generated some clear threats (meaning threats evaluated generic, specific and valuable by the expert) while 28 (44%) method applications were specific to the scenario and appreciated by the expert. In contrast, the quality of produced security requirements was slightly lower than for threats: 48 (75%) method applications produced clear security requirements but more than half (36) were generic. In general, we can conclude that the overall quality of the outcomes of method applications was satisfactory.

Number of threats and security requirements

To test the effectiveness of visual and textual methods with respect to the number of identified threats and security requirements, we applied the ANOVA statistical test with a significance level α of 0.05. The ANOVA tables are not reported due to lack of space. Before the application of the test, we verified whether the dependent variables were normally distributed with Shapiro-Wilk test (returned *p-values* are 0.17, 0.68 for requirements and threats respectively). We also checked the homogeneity of variance with the Fligner-Killeen test (returned *p-values* are 0.45 for requirements, and 0.64 for threats). So we have no evidence to reject either assumptions.

We first analyzed the differences in the number of threats identified with visual and textual methods. As shown in Figure 5.10 (left), if we consider all groups, the visual method is more effective in identifying threats than the textual one. This result is also confirmed if we consider only the groups who have produced good quality threats as shown in Figure 5.10 (right). The ANOVA test shows that the effect of the applied methods on the number of identified threats is statistically significant for all groups ($F = 18.49,$ *p-value* $= 1.03 \cdot 10^{-4}$) and good groups ($F = 26.10,$ *p-value* $= 1.59 \cdot 10^{-4}$).

Similarly, Figure 5.11 represents the means of the number of security requirements identified with the visual and the textual method by all groups (left) and by good groups (right). The figure shows that both for all groups and for good groups, the textual method is slightly better than the visual method in identifying security requirements. However, the ANOVA test shows that the difference in the security requirements identified with the textual and visual method is not statistically significant for all groups ($F = 1.18,$ *p-value* $= 0.28$) and good groups ($F = 1.98,$ *p-value* $= 0.23$).

Figure 5.12 confirms the results shown in Figure 5.10 and Figure 5.11. The figure reports a scatter

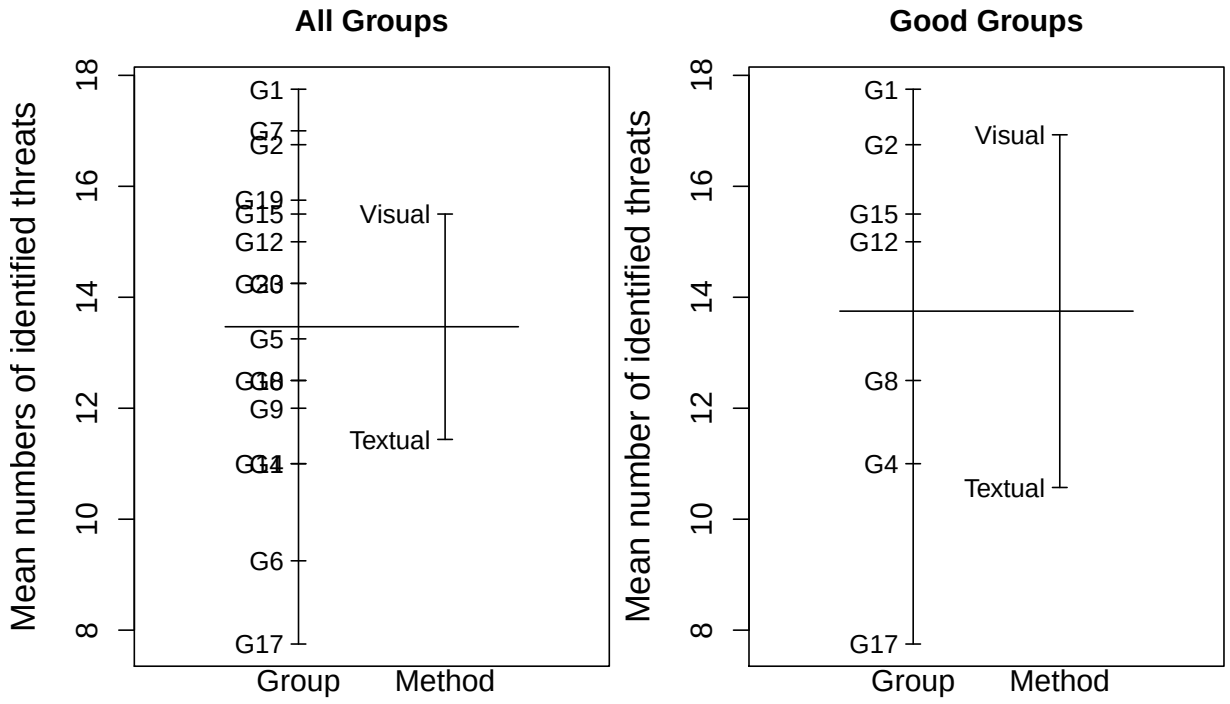


Figure 5.10: Means of identified threats in all groups (left) and good groups (right).

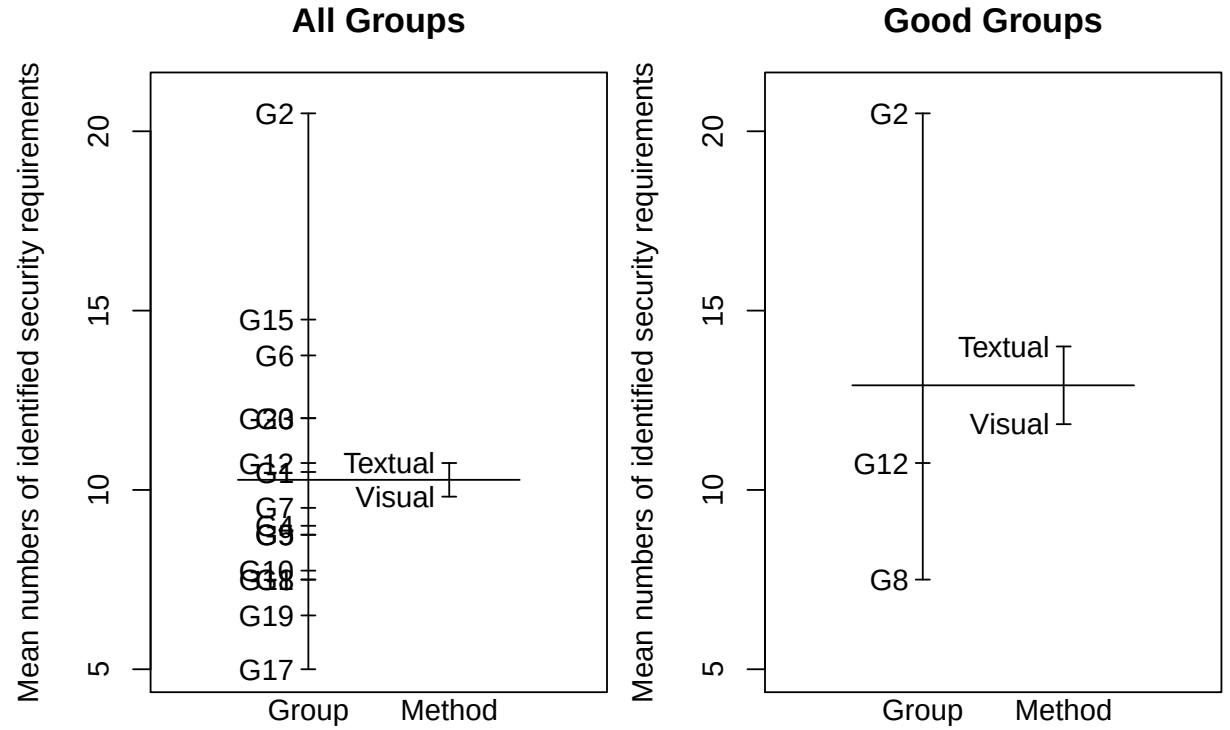


Figure 5.11: Means of identified security requirements in all groups (left) and good groups (right).

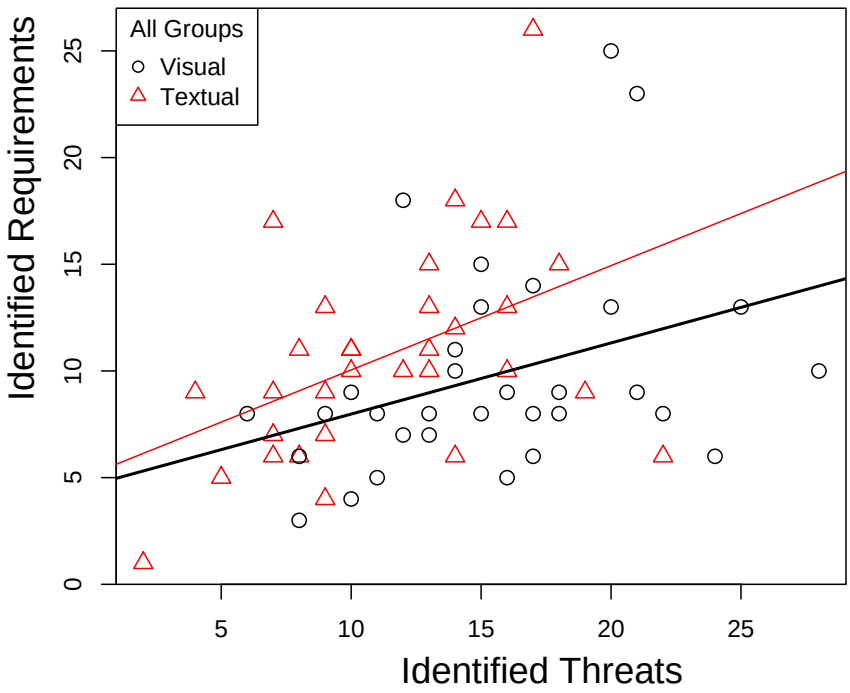


Figure 5.12: Scatter plot of identified threats and security requirements for the two methods.

view of the distribution of identified security requirements, and identified threats for the groups which have applied the visual method (circles) and the one which have applied the textual method (triangles). The groups which applied the visual method tend to identify more threats, but less security requirements than groups which applied the textual method. The linear regression models on security requirements and threats show that the textual method is slightly better than the visual one in terms of the number of identified security requirements given the number of identified threats, but with no statistical significance.

We have also investigated the differences in the number of threats and security requirements identified with the visual and the textual method within each facet. The boxplots in Figure 5.13 (left) show that the distribution of the visual method is always above the distribution of textual method. This means that using the visual method produces more threats than using the textual method in all four facets. This difference is less marked for the facet Net/Teleco (facet 3) but it is not statistically significant. If we consider only the facets Mgmnt, App/DB, and Mobile, the difference in the number of threats identified with the visual and the textual method is statistically significant because the ANOVA test returned a p-value $2.78 \cdot 10^{-3}$ ($F = 9.95$) which is less than 0.05. If we consider all facets, the difference is also statistically significant because the p-value returned by the ANOVA test is equal to $1.79 \cdot 10^{-3}$ ($F = 10.66$). Thus, we can conclude that across all facets the visual method is globally better than the textual method in identifying threats.

Figure 5.13 (right) reports the number of security requirements identified with the visual and the textual method within each facet. The boxplots show that textual method is slightly better than the visual one in identifying security requirements in the first three facets. In particular, in facet Net/Teleco the difference is higher than in the facets Mgmnt and App/DB. However, the ANOVA test given the facet Net/Teleco returned $F = 3.37,$ *p-value* $= 0.09$, which means visual and textual methods distributions are different but the difference is not statistically significant. In the last facet Mobile, the situation is inverted and the visual method is better than the textual one in identifying security requirements. Given all facets, the ANOVA test returned $F = 0.57,$ *p-value* $= 0.45$ which means the difference in the number of security requirements is not statistically significant.

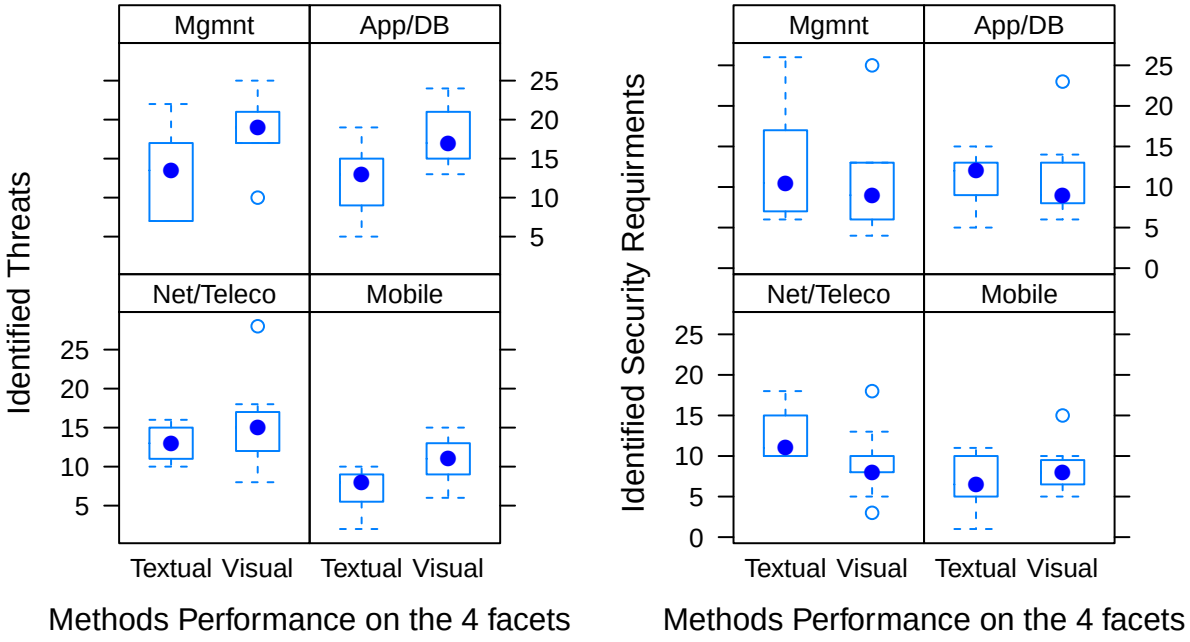


Figure 5.13: The distribution of identified threats (left) and security requirements (right) within each facet.

5.2.3 Questionnaire Analysis

We have analyzed the responses to the post-task questionnaire to determine if there is a difference in the participants' perception of visual and textual methods. When reporting the results all answers have been realigned to 5 being the best. As the responses were paired, in general not normally distributed, and our samples had ties, we have used the exact Wilcoxon signed-ranks test with Wilcoxon method for handling ties [14]. We set the significance level α to 0.05. As mentioned, for some of the questions (e.g., Q2 or Q10), we had to invert the order of negative and positive responses so that for all questions all negative responses were on the right and positive responses were on the left.

The results are summarized and compared in Table 5.5. For each question, the table reports to which perception variable the question refers to (PEOU, PU, ITU), the mean of the answers by all and by good participants (the one who were part of groups that produced good quality threats and security requirements based on expert's assessment), and the level of statistical significance based on the p-value returned by the Wilcoxon test. The level of statistical significance is specified by • (p<0.1), or * (* p<0.05, ** p<0.01, *** p<0.001). The table also reports the average responses for each perception variable and for all questions related to perception (Q1-Q12).

The results show that for some aspects the difference in the perception of visual (CORAS) and textual method (SREP) is statistically significant (p<0.05) or has minimum 10% significance level:

Q1 All participants prefer visual method over the textual one for easy of use but the difference in perception is not statistically significant. Instead, for good participants the difference is statistically significant.

Q2 Visual method is better than textual approach with respect to making the security analysis easier than an ad hoc approach. All participants prefer visual method with statistical significance. This is also true for good participants but the difference in participants' perception is not statistically significant.

Q5 When considering finding threats and security requirements more quickly than using common sense

		All subjects			Good subjects		
Q	**Type**	**Mean**		**Z**	**Mean**		**Z**
		Textual	**Visual**		**Textual**	**Visual**	
1	PEOU	3.0	3.3	-0.9	2.8	3.8	-2.1 *
2	PU	3.1	3.6	-2.1 *	3.3	3.5	-0.6
3	PEOU	3.3	3.1	0.4	3.5	3.8	-0.7
4	ITU	3.0	3.1	-0.2	2.8	3.4	-1.5
5	PU	3.1	3.0	0.3	2.9	3.6	-2.3 *
6	ITU	2.9	2.9	0.0	2.5	3.0	-1.3
7	PEOU	3.0	3.1	-0.2	3.2	3.2	0.0
8	PU	3.6	3.5	0.4	3.8	3.8	0.0
9	ITU	3.2	3.4	-0.9	3.2	3.5	-1.3
10	PEOU	3.5	3.7	-0.5	3.8	3.9	0.2
11	PU	3.1	3.2	-0.4	2.8	3.2	-1.0
12	ITU	3.0	3.3	-0.8	2.9	3.3	-1.0
13	Control	3.8	3.8	-0.2	3.5	4.2	-1.7
14	Control	4.4	3.9	3.1 ***	4.5	3.8	2.3 *
15	Control	4.3	4.2	0.9	4.5	4.3	1.3
16	Control	3.0	3.6	-2.5 *	2.9	3.7	-1.8
17	Control	3.1	3.5	-1.7	3.2	3.8	-2.3 *
PEOU		3.2	3.3	-0.7	3.3	3.7	-1.7 •
PU		3.2	3.4	-1.0	3.2	3.5	-2.0 •
ITU		3.0	3.2	-1.0	2.8	3.3	-2.5 *
Total		3.2	3.3	-1.6 •	3.1	3.5	-3.5 ***

• - p-value <0.1, * - p-value <0.05, ** - p-value <0.01, *** - p-value <0.001

Table 5.5: Wilcoxon signed-ranks test of responses

the results are not clear. The results show a small preference for textual method by all participants which is not statistically significant. In contrast, good participants show a statistically significant preference for visual method.

Q14 Both all participants and good participants with statistical significance believe that a catalog of threats would be more needed by textual method than the visual one.

Q16 Visual method is better than textual one with respect to helping brainstorming on the threats. This holds across all participants and the difference of preference is statistically significant. This is also true for good participants but with no statistical significance.

Q17 Visual method is better than the textual one with respect to helping brainstorming on the security requirements. This holds across all participants but it is not statistically significant. This is also true for good participants and it is statistically significant.

PEOU Visual method is better than the textual method with respect to overall PEOU across all participants but the preference is not statistically significant. Good participants show a small preference for visual method with 10% significance level.

PU Visual method is better than the textual one with respect to overall PU across all participants but the preference is not statistically significant. Good participants show a small preference for visual method with 10% significance level.

ITU Visual method is better than the textual one with respect to overall ITU across all participants but the preference is not statistically significant. Good participants show a statistically significant preference for visual method.

The average responses to Q1-Q12 show a small preference for visual method by all participants with 10% significance level. For good participants, instead the preference for visual method is statistically significant.

5.2.4 Interviews' Analysis

For a better understanding of which features influence visual and textual methods effectiveness, we complemented our experiment by interviewing each participant for half an hour.

Table 5.6: Frequency of reported aspects of the methods

Advantages	**Visual**	**Textual**	**Total**
Clear process	12	16	28
Help in brainstorming threats	21	15	36
Help in brainstorming security requirements	12	24	36
Easy to use and remember	17	11	28
Help to understand interdependencies	6	7	13
Support visual summary	24	0	24
No time consuming	0	4	4
Total	92	77	
Disadvantages			
No clear process	2	11	14
Do not support interdependencies	2	3	5
No help in brainstorming threats	3	3	6
No help in brainstorming security requirements	9	1	10
Primitive tool	20	0	20
No support visual summary	1	6	7
Visual summary does not scale	10	0	10
Too time consuming	11	9	20
No easy to use and remember	0	2	2
Total	58	35	
Improvements			
Have security resource repository	0	5	5
Have visual summary	0	2	2
Support automatic risk level computation	1	0	1
Support diagram creation	1	0	1
Total	2	7	

The interviews were analyzed with a content analysis technique called coding [63]. The analysis consists of the following steps: *1*) we transcribed and analyzed to identify recurring themes, which serve as the basis to build categories that explain why visual and textual methods work in practice or not; *2*) we identified a set of recurring participants' statements in the interviews and we classified them in *advantages*, *disadvantages* and *improvements* of the methods; *3*) for each group of statements, we coded and classified them into iteratively emerging categories; *4*) we counted the frequency of statements in each category as an indication of their relative importance. Table 5.6 presents the categories and the frequency of statements in each category made by the participants.

The main advantage of visual method that participants indicated is that it provides a visual summary of the results of the security analysis (89%). Indeed, the diagrams give an overview of the assets and the possible threats scenarios and treatments. A typical statement made by the participants referring to this advantage was: "*Diagrams are useful. You have an overview of the possible threat scenarios and you can find links among the scenarios*". Another noteworthy advantage of visual method reported by the 82% of the participants was that it helps brainstorming on the threats. As the participants indicated, diagrams play a key role in helping to brainstorm on threats: "*Yes it helped to identify which are the threats. In CORAS method everything is visualized. The diagrams helped brainstorming on threats.*" The next advantage refers to perceived ease of use. The 60% of the participants reported that visual method

Table 5.7: Results of hypothesis testing

$H1.1_A$	Difference in the number of threats found with visual and with textual method	YES (More threats were found with visual method than with textual method)
$H1.2_A$	Difference in the number of security requirements found with visual and with textual method	NO (Slightly more security requirements were found with textual method than with the visual one but the difference is not statistically significant)
$H2.1_A$	Difference in the number of threats found with visual and with textual method within each facet	YES (For each facet more threats were found with visual than with textual method)
$H2.2_A$	Difference in the number of security requirements found with visual and with textual method within each facet	NO (For each facet slightly more security requirements were found with textual than with visual method but the difference is not statistically significant)
$H3_A$	Difference in the participants preference for visual and textual method	YES (Overall visual method is preferred to the textual one)
$H4_A$	Difference in the participants perceived ease of use for visual and textual method	MAY BE (Visual method is perceived as easier to use than textual approach with 10% significance level)
$H5_A$	Difference in the participants perceived usefulness for visual and textual method	MAY BE (Visual method is perceived as more useful than the textual method with 10% significance level)
$H6_A$	Difference in the participants intention to use for visual and textual method	YES (Participants intend to use the visual method more than the textual one)

is a "*good methodology, not difficult to use. It is much clear to understand the security case there*".

The main advantage of textual method according to the 96% participants was that the method helps in identifying security requirements. Typical statements in this category were: "*SREP helped in brainstorming. The steps were pretty much defined. Step by step helped to discover more*" and "*SREP helped in brainstorming. The order of the steps helped to identify security requirements*". The second advantage of textual method is that it has a clear process to follow (60%): "*Well defined steps. Clear process to follow.*" is an example of typical statement made by the participants for this category.

With respect to methods' disadvantages and improvements, the statements were fewer than the ones about advantages. The most indicated disadvantage of visual method was that visual notation does not scale well for complex scenarios. Typical statements in this category were: "*The diagrams are not scalable when there are too many links*" and "*For big systems the diagrams would be very large. Even with the support of the computer it would be difficult to see them*. In addition, 75% of the participants complained about the tool. The major problems reported were the tool bad memory usage that makes the tool too slow and the modeling feature of the tool that does not provide automatic support for the generation of the diagrams (e.g., generating a treatment diagram from a threat diagram). Examples of typical statements for this category were: "*The tool is not difficult to use but it is very slow. It is impossible to copy a diagram from a type of diagram to another. Objects have no references between the diagrams. Changes on an object in a diagram are not reflected on the same object in other diagrams.*" and "*The tool takes too much to arrange things. Drawing assets and threats is not easy. When the diagrams are too large, the tool occupies too much memory*". Instead, textual method has two main drawbacks. First, it is unclear how to perform some of the steps of the textual method process: risk assessment, requirements inspection and repository improvement. Second, the use of tables to represent threats makes it difficult to show the link among assets, threats and security requirements, and thus to give a summary of the results of the security analysis. As reported by the participants "*It is not easy to represent what you think because there are a lot of tables. If you are project manager and you want to show the results of the security analysis to your boss it is difficult because you use tables*".

5.2.5 Discussion

In this section we present the main findings regarding each of the research questions and possible explanations for the findings. A summary of the findings is shown in Table 5.7.

Methods' effectiveness

As shown in the previous sections, visual method is more effective in identifying threats than textual method. This result is also confirmed if we consider the *number of threats* identified with visual and textual methods across the task assigned to the groups. Since the difference in the number of threats identified with the two methods is statistically significant, we can accept the alternative hypotheses $H1.1_A$ and $H2.1_A$ of difference between the number of threats identified with the two methods. Instead, with respect to *number of security requirements*, textual method is slightly more effective than the visual one in identifying security requirements but the difference is not statistically significant across all groups and tasks. The alternative hypotheses $H1.2_A$ and $H2.2_A$ of difference in the number of security requirements can therefore be rejected.

Methods' perception

Participants' *overall preference* is higher for visual than for textual method. Among all the groups the difference has 10% significance level, while for the participants who were part of groups who produced good quality threats and security requirements, the difference in the overall preference is statistically significant. The conclusion is that the alternative hypothesis $H3_A$ of difference in the overall preference of the two methods is upheld. Similarly, for all participants there is no statistically significant difference in perceived *ease of use* and *usefulness*, while for "good" participants the difference has a 10% significance level. For this reason, there is no evidence that the null hypotheses $H4_0$ of no difference in the perceived ease of use and $H5_0$ of no difference in perceived usefulness do not hold. Thus, the alternative hypotheses $H4_A$ and $H5_A$ cannot be rejected or accepted. With respect to *intention to use*, "good" participants intend to use more visual than textual method and the difference in participants' perception is statistically significant. The alternative hypothesis $H6_A$ of difference in the intention to use for the two methods can thus be accepted.

Qualitative Explanation

The different number of threats and security requirements identified with visual and textual methods can be likely explained by the differences between the two methods indicated by the participants during the interviews. Diagrams in visual method help brainstorming on the threats because they give an overview of the possible threats (who initiate the threats), the threat scenarios (possible attacks) and the assets, while the identification of threats in textual method is not facilitated by the use of tables because it is difficult to keep the link between assets and threats. As suggested by the answers to question $Q14$ in the post-task questionnaire, the identification of threats in textual method could be made easier if a catalog of common threats was available. In addition, during the interviews some of the participants indicated that a visual representation for threats would be better than a tabular one.

Textual method is slightly more effective in eliciting security requirements than visual approach because the order of steps in textual method process guides the analyst in the identification of security requirements, while the same it seems not to hold for the visual method's process.

5.2.6 Threats to Validity

We discuss the four main types of threats to validity [71] in what follows.

Conclusion validity Conclusion validity is concerned with issues that affect the ability to draw the correct conclusion about the relations between the treatment and the outcome of the experiment. There are three main threats to conclusion validity relevant for our experiment:

- *Low statistical power*. An important threat to validity is related to the sample size that must be big enough to come to correct conclusions. We conducted a post-hoc power analysis for the ANOVA test and Wilcoxon signed-rank test (with G*Power 3 tool[2]) for participants from good groups. For Wilcoxon signed-rank test, we obtained a power (1-β) equal to 0.86 setting as parameter the effect size ES = 0.71, the total sample size N = 24, and α = 0.05. For the ANOVA test, we have instead a power of 0.89 with 32 observations for each method and between variance at least 16 observations are needed to have an effect size of 2 like in our experiment. We thus have enough observations to conclude that our results on the relation between the methods applied and their performance in terms of number of threats and security requirements and with responses to the post-task questionnaire are correct.

- *Violated assumptions of statistical tests*. Before running the ANOVA and Wilcoxon signed rank tests we have checked with Shapiro-Wilk and Flinger-Killeen tests that their assumptions are not violated. For example, before applying ANOVA to test the effect that the method has on the number of assets, threats, and security requirements, we have checked the assumption about the homogeneity of the variance using the Flinger-Killeen test which in all three cases reported a p-value greater than 0.05. Thus, we are sure that the assumptions of statistical tests were not violated and that our results are correct.

- *Heterogeneity of subjects*. If groups in the sample are too heterogeneous, the variation due to individual differences may be larger than due to treatment. We have reduced this threat by running the experiment with master students who had similar knowledge and background.

Internal validity Internal validity is concerned with issues that may falsely indicate a causal relationship between the treatment and the outcome, although there is none.

- *Participants' background*. The familiarity of the participants with the methods evaluated during the experiment is a threat to internal validity. At the beginning of the experiment, we have administered a questionnaire to check the background of the participants and their knowledge of security methods. The questionnaire has shown that all participants had a similar background and had no prior knowledge about visual and textual methods.

- *Bias in the tutorials*. Differences in the methods' performance may occur if a method is presented in a better way than the other. In our experiment we limit this threat by giving the same structure and the same duration to the tutorials on textual and visual methods.

- *Participants' behavior*. During the execution of the experiment, the subjects may react differently over time e.g., subjects may become bored or tired, or they may become more or less positive to one or another method. We notice that the performance of the participants in terms of number of threats and security requirements identified was almost the same for the first, second and third task, while on the last task the performance decreases because the participants got tired or did not put much effort. Yet, this phenomenon was common to both methods.

- *Bias in data analysis*. To avoid bias in the reports analysis, the coding of the participants' reports was conducted by the authors of the paper independently. In addition, the quality of the threats and security requirements identified by each group was assessed by an expert external to the experiment.

Construct validity Construct validity concerns generalizing the result of the experiment to the concept and theory behind the experiment. The main threat to construct validity in our experiment is the design of the research instruments: interviews and questionnaires. The questionnaire was designed following the Technology Acceptance Model with four questions for each of the independent variables we wanted to measure: *perceived usefulness*, *perceived ease of use*, *intention to use*. The interview guide included

[2]http://www.psycho.uni-duesseldorf.de/abteilungen/aap/gpower3/

questions concerning research questions $RQ3$ and methods' advantages and disadvantages. Three researchers independently have checked the questions included in the interview guide and in the questionnaire: therefore we are reasonably confident that our research instruments measured what we wanted to measure.

External validity External validity concerns the ability to generalize experiment results beyond the experiment settings. External validity is thus affected by the objects and the subjects chosen to conduct the experiment.

- *Use of students instead of practitioners*. Using students rather than practitioners as subjects is known as a major threat to external validity. However, Svahnberg et al. [64] recognized that students may work well as subjects in empirical studies in the requirements engineering area.
- *Realism of the application scenario and facets*. We reduce the threat to external validity by making the experimental environment as realistic as possible. In fact, as object of our experiment we have chosen a real industrial application scenario proposed by National Grid. Furthermore, the reports of participants have been evaluated by an expert from National Grid: the quality of both security requirements and threats identified is good enough for the study (see also §5.2.2).

6 Conclusion

We have presented a full evaluation approach, called SECEVAL, which can be used to do evaluation-based research in the area of secure software in a structured way.

SECEVAL is based on the work previously done in WP2 and WP5 (CBK) [13, 6], as well as on the structured literature review by Kitchenham et al. [39] and inspired by the C-INCAMI framework of Becker et al. [4], to name a few. It defines an ontology which is represented as UML model. SECEVAL specifies:

- an improved, flexible security context model for describing features of methods, notations and tools in the security area.
- a model that records the way how data is collected. It mainly comprises the research question, collection process, used resources and the queries which are used for finding sources which might be used to answer the question.
- an analysis model which defines the analysis strategy and the filters and algorithms it uses on the collected sources. Furthermore, the data structure for information is exactly specified, regardless of whether the data is to be stored in the security context model or not.

SECEVAL was improved using a guided interview and we additionally provided a case study about methods and tools from the area of security focusing on a research question about the selection of vulnerability scanner for web applications.

Furthermore, we reported on an evaluation of approaches for secure software design as well as an empirical validation of risk-based methods. The former is a mapping study [58] which provides an overview of the current state of the art regarding secure software design. The latter study compares a textual method (i.e. CORAS [45]) and a graphical method (i.e. SREP [48]), based on the approaches of Kitchenham et al. and Moody [51].

For both studies, we claim that a slightly extended version of SECEVAL (cf. extension for Moody's approach in subsection 3.3.2) could easily record the process of doing research as well as the result data. As this process is quite straight-forward, we omitted to draw SECEVAL instance diagrams. However, we strongly recommend to add the outcome to an implemented version of SECEVAL, as soon as it will be available.

Summarizing, SECEVAL provides a sound basis for evaluating research questions related to secure software engineering. This eases the process of doing research in the area of security no matter if the research question aims at scientific or engineering issues.

Further goals are to evaluate more examples like the testing case study presented in this deliverable. We already noticed that SECEVAL's security context model can easily be extended or even mapped to another domain by adding and removing elements. Further practical experience would be exciting at this point.

It would also be interesting to implement the context model in a more flexible version of the CBK, although it is challenging to strike a balance between complexity of the CBK's implementation and the effort such a flexible model would cause. For the implementation, we suggest experimenting with an attribute-based suggestion system: when creating a new element in the Wiki, it first is empty, but on the right hand side of the screen common attributes are shown which can be dragged onto the Wiki page in order to fill and arrange them. A useful feature is to be able to insert text from other web pages, e.g., from Wikipedia or vulnerability management systems, which are then correctly cited. Step-by-step wizards and good tag recommendation according to the tags selected so far and the information provided would ease this task. Additionally, recommendation includes that the system needs to explain rules inferred by SECEVAL, as e.g., that tools and notations are described by different attributes and that it is useful to describe tool and notation in a separate entry, also in case that a notation is only used by one tool yet. A focus should also be on the connection of several entries and on the possibility to add data which is not only associated with one entry (e.g., data which is semi-automatically extracted from existing papers). Adding metrics to an attribute or to a numeric value within a text-based attribute would round off the implementation.

So far, we also reported on newly integrated SDE tools in deliverables of WP2. This time, information about the SDE can be found in deliverable D1.4 [2]. Details will follow in deliverable D2.5.

7 NESSoS Third-Year Publications

1. A. van den Berghe, R. Scandariato, and W. Joosen. Towards a systematic literature review on secure software design. In *Proceedings of the Doctoral Symposium of the International Symposium on Engineering Secure Software and Systems (ESSoS-DS 2013)*. CEUR-WS, 2013.

2. K. Labunets, F. Massacci, F. Paci, and L. M. S. Tran. An experimental comparison of two risk-based security methods. *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2013.

3. K. Labunets and F. Massacci. Empirical validation of security methods. In *Proceedings of the Doctoral Symposium of the International Symposium on Engineering Secure Software and Systems (ESSoS-DS 2013)*. CEUR-WS, 2013.

Bibliography

[1] J. Abramov, O. Anson, A. Sturm, and P. Shoval. Tool Support for Enforcing Security Policies on Databases. In S. Nurcan, editor, *IS Olympics: Information Systems in a Diverse World*, volume 107 of *Lecture Notes in Business Information Processing*, pages 126–141. Springer Berlin Heidelberg, 2012.

[2] C. Bartolini and A. Bertolino. NESSoS Deliverable D1.4 – NESSoS Joint Virtual Research Lab. 2013.

[3] D. Basin, J. Doser, and T. Lodderstedt. Model Driven security: From UML Models to Access Control Infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91, 2006.

[4] P. Becker, F. Papa, and L. Olsina. Enhancing the Conceptual Framework Capability for a Measurement and Evaluation Strategy. *4th International Workshop on Quality in Web Engineering* , (6360):1–12, 2013.

[5] K. Beckers, S. Eicker, M. Heisel, and W. S. (UDE). NESSoS Deliverable D5.2 – Identification of Research Gaps in the Common Body of Knowledge. 2012.

[6] K. Beckers, S. Eicker, M. Heisel, and W. S. (UDE). NESSoS Deliverable D5.3 – Assessment of the CBK also through open consultation. 2013.

[7] K. Beckers and M. Heisel. A usability evaluation of the nessos common body of knowledge. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES) - 2nd International Workshop on Security Ontologies and Taxonomies (SecOnT 2013)*. IEEE Computer Society, 2013. Accepted for Publication.

[8] E. Bertino, J. Crampton, and F. Paci. Access Control and Authorization Constraints for WS-BPEL. In *Web Services, 2006. ICWS '06. International Conference on*, pages 275–284, 2006.

[9] B. Best, J. Jürjens, and B. Nuseibeh. Model-based Security Engineering of Distributed Information Systems using UMLsec. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 581–590, Washington, DC, USA, 2007. IEEE Computer Society.

[10] M. Busch and N. Koch. NESSoS Deliverable D2.1 – First release of Method and Tool Evaluation. 2011.

[11] M. Busch and N. Koch. NESSoS Deliverable D2.3 – Second Release of the SDE for Security-Related Tools. 2012.

[12] M. Busch, N. Koch, and M. Wirsing. Seceval: An evaluation framework for engineering secure systems, 2014. submitted.

[13] CBK. Common Body of Knowledge. http://nessos-project.eu/cbk, 2013.

[14] W. J. Conover. On methods of handling ties in the wilcoxon signed-rank test. *Journal of the American Statistical Association*, 68(344):985–988, 1973.

[15] L. Dai and K. Cooper. Modeling and performance analysis for security aspects. *Science of Computer Programming*, 61(1):58–71, 2006. Special Issue on Quality system and software architectures.

[16] L. Dai and K. Cooper. Using FDAF to bridge the gap between enterprise and software architectures for security. *Science of Computer Programming*, 66(1):87–102, 2007. Special Issue on the 5th International Workshop on System/Software Architectures (IWSSA£06).

[17] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, pages 319–340, 1989.

[18] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.

[19] P. Diaz, I. Aedo, D. Sanz, and A. Malizia. A model-driven approach for the visual specification of Role-Based Access Control policies in web systems. In *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pages 203–210, 2008.

[20] 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.

[21] E. Fernández-Medina, J. Trujillo, R. Villarroel, and M. Piattini. Developing secure data warehouses with a UML extension. *Information Systems*, 32(6):826–856, 2007.

[22] O. Foundation. OWASP Risk Rating Methodology, 2013. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[23] O. Foundation. OWASP Top 10 – 2013, 2013. http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf.

[24] G. Georg, K. Anastasakis, B. Bordbar, S. Houmb, I. Ray, and M. Toahchoodee. Verification and Trade-Off Analysis of Security Properties in UML System Models. *Software Engineering, IEEE Transactions on*, 36(3):338–356, 2010.

[25] G. Georg, I. Ray, and R. France. Using Aspects to Design a Secure System. In *Proceedings of the Eighth International Conference on Engineering of Complex Computer Systems*, ICECCS '02, pages 117–, Washington, DC, USA, 2002. IEEE Computer Society.

[26] M. Giordano, G. Polese, G. Scanniello, and G. Tortora. A system for visual role-based policy modelling. *Journal of Visual Languages & Computing*, 21(1):41–64, 2010.

[27] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *In Proc. of the 13th IEEE International Conference on RE*, pages 167–176. IEEE, 2005.

[28] H. Gomaa and M. Eonsuk Shin. Modelling Complex Systems by Separating Application and Security Concerns. In *Engineering Complex Computer Systems, 2004. Proceedings. Ninth IEEE International Conference on*, pages 19–28, 2004.

[29] M. Hafner, M. Breu, R. Breu, and A. Nowak. Modelling Inter-organizational Workflow Security in a Peer-to-Peer Environment. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages –540, 2005.

[30] R. Heldal and F. Hultin. Bridging Model-Based and Language-Based Security. In E. Snekkenes and D. Gollmann, editors, *Computer Security - ESORICS 2003*, volume 2808 of *Lecture Notes in Computer Science*, pages 235–252. Springer Berlin / Heidelberg, 2003.

[31] International Organization for Standardization. *ISO 31000 Risk management – Principles and guidelines*, 2009.

[32] International Organization for Standardization and International Electrotechnical Commission. *ISO/IEC 15408 Information technology - Security techniques - Evaluation criteria for IT security*, 2005.

[33] International Organization for Standardization and International Electrotechnical Commission. *ISO/IEC 27002 Information technology - Security techniques - Code of practice for information security management*, 2005.

[34] J. Jürjens. *Secure Systems Development with UML*. Springer, 2004.

[35] J. Jürjens. Sound Methods and Effective Tools for Model-based Security Engineering with UML. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 322–331, New York, NY, USA, 2005. ACM.

[36] J. Jürjens, M. Lehrhuber, and G. Wimmel. Model-Based Design and Analysis of Permission-Based Security. In *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on*, pages 224–233, 2005.

[37] J. Jürjens, J. Schreck, and P. Bartmann. Model-based Security Analysis for Mobile Communications. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 683–692, New York, NY, USA, 2008. ACM.

[38] S. Kim, D.-K. Kim, L. Lu, S. Kim, and S. Park. A feature-based approach for modeling role-based access control systems. *Journal of Systems and Software*, 84(12):2035–2052, 2011.

[39] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[40] M. Koch, L. V. Mancini, and F. Parisi Presicce. A graph-based formalism for RBAC. *ACM Trans. Inf. Syst. Secur.*, 5(3):332–365, 2002.

[41] S. Kou, M. A. Babar, and A. Sangroya. Modeling Security for Service Oriented Applications. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 294–301, New York, NY, USA, 2010. ACM.

[42] K. Labunets and F. Massacci. Empirical validation of security methods. In *Proceedings of the Doctoral Symposium of the International Symposium on Engineering Secure Software and Systems (ESSoS-DS 2013)*. CEUR-WS, 2013.

[43] K. Labunets, F. Massacci, F. Paci, and L. M. S. Tran. An experimental comparison of two risk-based security methods. *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 2013.

[44] C. Lacek. In-depth comparison and integration of tools for testing security features of web applications, 2013. Bachelor Thesis.

[45] M. S. Lund, B. Solhaug, and K. Stølen. *Model-driven risk analysis: the CORAS approach*. Springer, 2011.

[46] Y. Martínez, C. Cachero, and S. Meliá. Mdd vs. traditional software development: A practitioner's subjective perspective. *Information and Software Technology*, 2012.

[47] F. Massacci and F. Paci. How to select a security requirements method? a comparative study with students and practitioners. In *Secure IT Systems*, pages 89–104. Springer, 2012.

[48] D. Mellado, E. Fernández-Medina, and M. Piattini. Applying a security requirements engineering process. In *Proc. of the 11th European Symposium on Research in Computer Security (ESORICS)*, pages 192–206. Springer, 2006.

[49] M. Menzel and C. Meinel. A Security Meta-model for Service-Oriented Architectures. In *Services Computing, 2009. SCC '09. IEEE International Conference on*, pages 251–259, 2009.

[50] M. Menzel and C. Meinel. SecureSOA Modelling Security Requirements for Service-Oriented Architectures. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 146–153, 2010.

[51] D. L. Moody. The method evaluation model: a theoretical model for validating information systems design methods. In C. U. Ciborra, R. Mercurio, M. de Marco, M. Martinez, and A. Carignani, editors, *ECIS*, pages 1327–1336, 2003.

[52] M. Morandini, A. Marchetto, and A. Perini. Requirements comprehension: A controlled experiment on conceptual modeling methods. In *In Proc. of the International Workshop on Empirical RE*, pages 53–60. IEEE, 2011.

[53] H. Mouratidis. Secure software systems engineering: The secure tropos approach. *Journal of Software*, 6(3):331–339, 2011.

[54] F. Moyano, C. Fernandez-Gago, and J. Lopez. A conceptual framework for trust models. In S. Fischer-Hübner, S. Katsikas, and G. Quirchmayr, editors, *9th International Conference on Trust, Privacy & Security in Digital Business (TrustBus 2012)*, volume 7449 of *Lectures Notes in Computer Science*, pages 93–104, Vienna, 2012. Springer Verlag, Springer Verlag.

[55] Y. Nakamura, M. Tatsubori, T. Imamura, and K. Ono. Model-driven security based on a Web services security architecture. In *Services Computing, 2005 IEEE International Conference on*, volume 1, pages 7–15, 2005.

[56] OMG. OCL 2.0. http://www.omg.org/spec/OCL/2.0/.

[57] A. L. Opdahl and G. Sindre. Experimental comparison of attack trees and misuse cases for security threat identification. *Information and Software Technology*, 51(5):916–932, 2009.

[58] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77. British Computer Society, 2008.

[59] F. Satoh, Y. Nakamura, and K. Ono. Adding Authentication to Model Driven Security. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, pages 585–594, Washington, DC, USA, 2006. IEEE Computer Society.

[60] S. Schreiner. Comparison of security-related tools and methods for testing software, 2013. Bachelor Thesis.

[61] G. Sindre and A. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.

[62] K. Sohr, G.-J. Ahn, M. Gogolla, and L. Migge. Specification and Validation of Authorisation Constraints Using UML and OCL. In S. Vimercati, P. Syverson, and D. Gollmann, editors, *Computer Security £ ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 64–79. Springer Berlin Heidelberg, 2005.

[63] A. L. Strauss and J. M. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 1998.

[64] M. Svahnberg, A. Aurum, and C. Wohlin. Using students as subjects-an empirical evaluation. In *Proceedings of the Second International Symposium on ESEM*, pages 288–290. ACM, IEEE, 2008.

[65] A. Teh, E. Baniassad, D. Van Rooy, and C. Boughton. Social psychology and software teams: Establishing task-effective group norms. *Software, IEEE*, 29(4):53–58, 2012.

[66] R. A. University. i* notation. http://istar.rwth-aachen.de/.

[67] A. van den Berghe, R. Scandariato, and W. Joosen. Towards a systematic literature review on secure software design. In *Proceedings of the Doctoral Symposium of the International Symposium on Engineering Secure Software and Systems (ESSoS-DS 2013)*. CEUR-WS, 2013.

[68] B. Vela, C. Blanco, E. Fernández Medina, and E. Marcos. A practical application of our MDD approach for modeling secure XML data warehouses. *Decision Support Systems*, 52(4):899–925, 2012.

[69] Verizon. Vector for hacking actions. *Data Breach Investigations Report*, 2013. http://www.verizonenterprise.com/resources/reports/es_data-breach-investigations-report-2013_en_xg.pdf.

[70] J. A. Wang and M. Guo. Security data mining in an ontology for vulnerability management. In *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS '09. International Joint Conference on*, pages 597–603, 2009.

[71] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in software engineering*. Springer, 2012.

[72] C. Wolter and A. Schaad. Modeling of Task-Based Authorization Constraints in BPMN. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 64–79. Springer Berlin / Heidelberg, 2007.

[73] D. Xu and K. Nygard. Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets. *Software Engineering, IEEE Transactions on*, 32(4):265–278, 2006.

A Appendix: Questionnaire

In the following, the questionnaire about our Security Context model is reprinted. It was handed out at the NESSoS plenary meeting in Malaga (29.5.-31.5.2013). In the original version of the questionnaire, there was space after each question so that the answers could be handwritten.

A.1 Security Engineering Method and Tool Evaluation

Our **aim** is to provide an approach for the evaluation of methods and tools for the engineering of secure software systems. In our approach we do not only distinguish methods and tools, but also notations. For an evaluation and comparison approach we need to define (1) the process of how to conduct a comparison and (2) the structure used to collect security-related data and metrics to analyze it. Therefore, we define a conceptual framework that comprises these three aspects: Security Context, Data Collection and Analysis. We depict the concepts and their relationships as a model (see Figure A.1 for an overview), so that we can instantiate concrete methods, tools and notations.

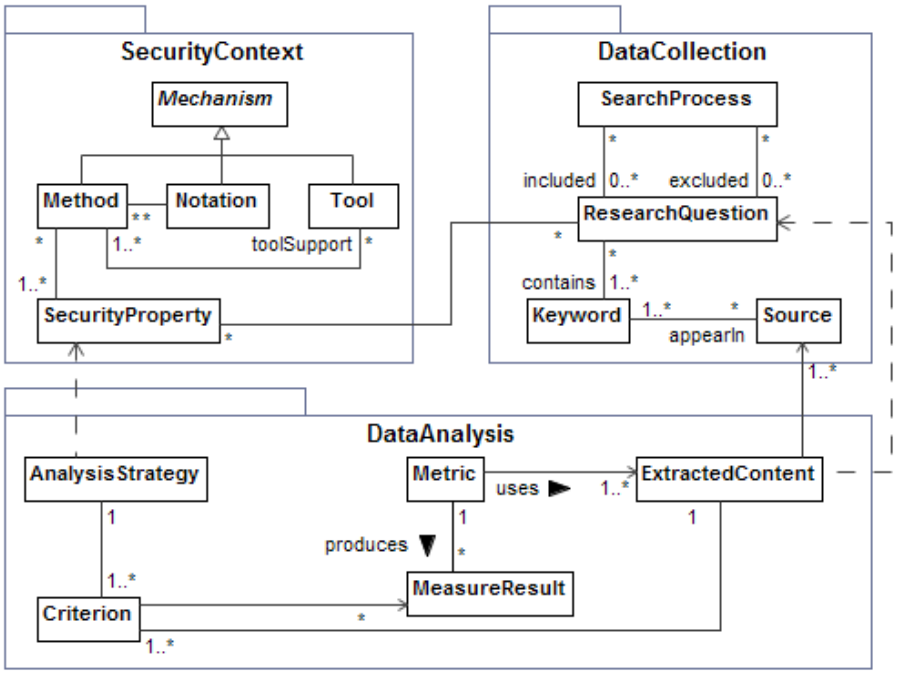


Figure A.1: Overview

Additionally, we will compare other approaches with our framework in order to further adapt or extend it with the objective to make it more general. The **basic structure** of our model of the Security Context is as depicted in Figure A.2.

The classes Method, Notation and Tool are depicted in the center. They inherit general attributes, as e.g., names and URLs, from the abstract class Mechanism (we are still looking for a better name to replace “mechanism”, suggestions are welcome!). A tool can support methods and a notation can be used for several methods.

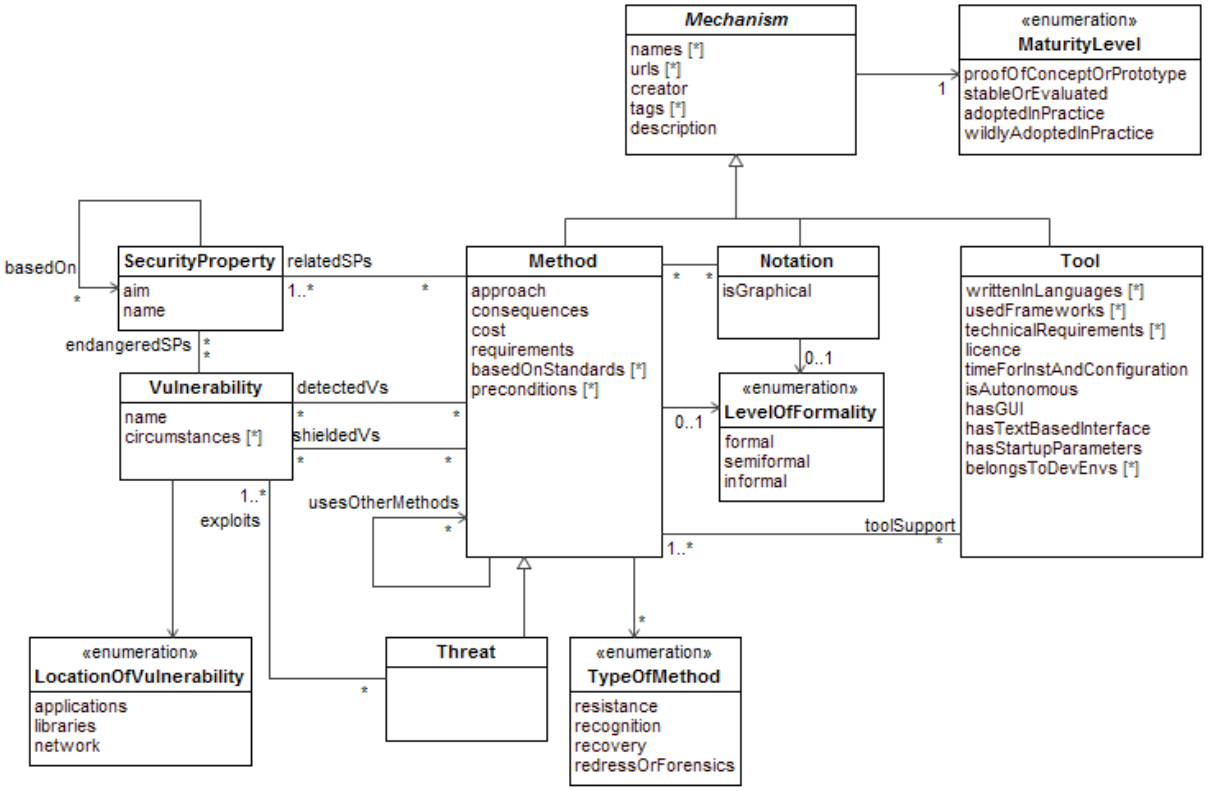


Figure A.2: Security Context

Security features are shown on the left hand side of Figure A.2.

- A SECURITY PROPERTY can be, e.g., authorization, authentication, integrity, etc. Several security properties can be enforced or attacked by a method.
- A VULNERABILITY can endanger security properties. Examples are XSS, SQL Injection, Buffer or Overflows, etc.
- A THREAT can exploit vulnerabilities. Threats are kind of methods which are vicious.

In Figure A.3 resp. A.4, the classes **Tool and Method** are refined according to their usage in the Software Development Life Cycle (SDLC).

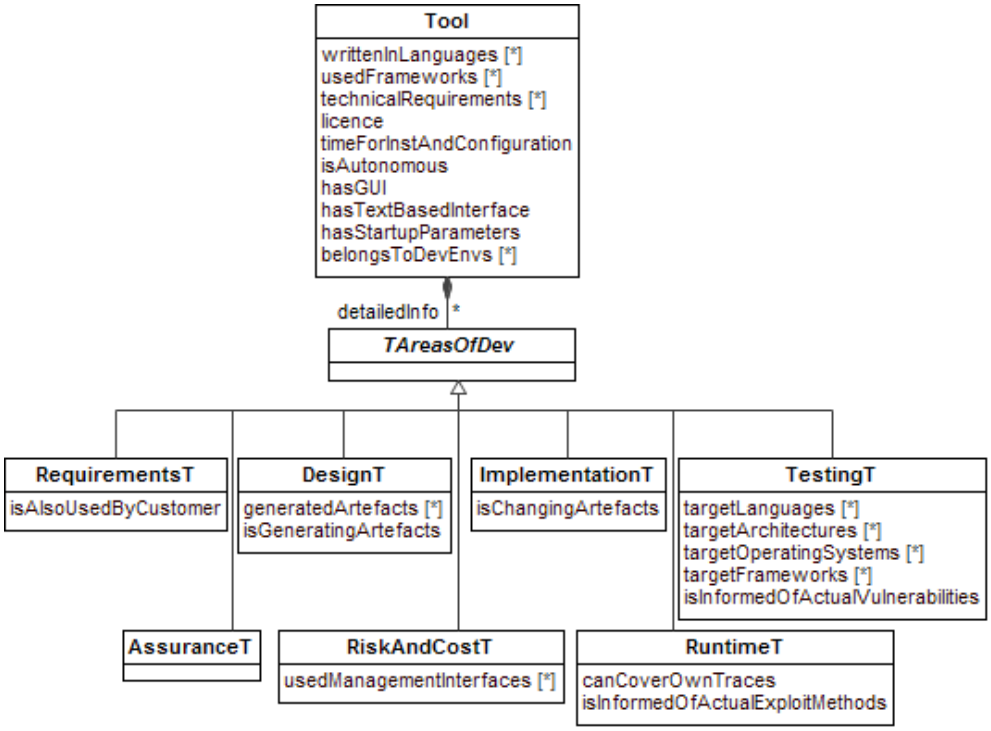


Figure A.3: Security Context: Details of Tools

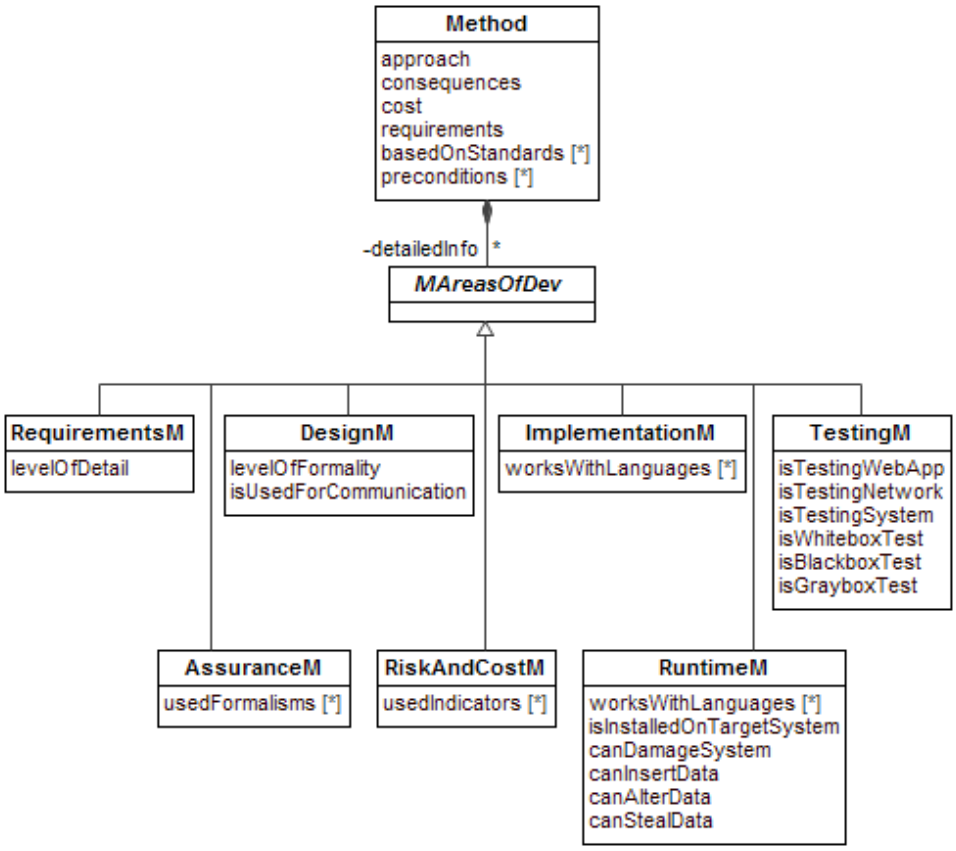


Figure A.4: Security Context: Details of Methods

A.2 Questions and Suggestions

1. Name:
   Partner:

2. Areas of security you are working in?

3. Can methods from your area be represented using our model?
   Provide examples of methods.

   Are concepts or relationships missing? Which?

4. Can tools from your area be represented using our model?
   Provide examples for tools.

   Are concepts or relationships missing? Which?

5. Can notations from your area be represented using our model?
   Provide examples of notations.

   Are concepts or relationships missing? Which?

7. Would you use our structure to evaluate tool, methods or notations in your area? If not, what would your approach look like?

   If yes, where do you see its strengths?

8. Can you suggest related work (esp. for managing tool and method portfolios for the area you are working in or general approaches to compare with our approach)?

9. General comments or improvements?