The present work was submitted to the Research Group Software Construction

of the Faculty of Mathematics, Computer Science, and Natural Sciences

Master Thesis

Designing a view-based approach to model security in software architectures

presented by

Sebastian Jörn Geiss

Aachen, June 26, 2023

Examiner

Prof. Dr. rer. nat. Horst Lichter

Prof. Dr. rer. nat. Bernhard Rumpe

Supervisor

Alex Sabau, M.Sc.

Abstract

As software systems become increasingly prevalent in our digitalized world, the security of these systems becomes a critical concern. Cybercriminals continually exploit vulnerabilities in software, making it essential for software developers to consider the security of the software system while planning. Since existing approaches focus on identifying vulnerabilities from an attacker's perspective, there is a research gap in modeling security from a defender's perspective.

This thesis addresses this gap by contributing a modeling language called the *Fortress Modeling Language (FML)*. FML uses security dimensions to model the defender's perspective. A security dimension is a part of a software architecture view that refers to defined security aspects, like for example authentication. Further, it consists of multiple security concerns, which are to be covered by the security dimension, and provides the link between the security concerns, requirements, and corresponding security design concepts.

The concept of security dimensions is based on the concept of "security views" defined by Sinkovec [Sin22].

This thesis presents the FML metamodel and an example for realization as well as an evaluation of FML. For the evaluation, experts from industry and research were surveyed about FML in a semi-structured expert interview to investigate whether FML is suitable as a modeling language for the security of a software system.

The results of the evaluation show that despite some criticism, the experts have considered the language to be overall suitable. The discussion of the results indicates that FML offers a promising first draft for modeling the security of software systems. Therefore, FML can serve as a foundation for upcoming security modeling languages. It can be used as a starting point for future improvements and shows a new way to model the security of system architectures.

Contents

List of Tables

List of Figures

1. Introduction

In an increasingly digitalized world, software systems have become an integral part of our daily lives. We use them in various fields such as communications, banking, healthcare, transportation, and many more. However, as our dependence on software increases, so does the threat to the security of these systems. Cybercriminals use advanced techniques to exploit vulnerabilities in software systems and gain unauthorized access to confidential information or cause damage.

However, software developers are not helpless against these attacks. There are software design principles that provide fundamental guidelines for software development [KK21]. These principles aim to improve the quality, maintainability, extensibility, and reusability of the code. In terms of software security, software design principles aim to avoid security vulnerabilities and enhance resilience against potential attacks.

An example of such a design principle is *zero trust networking.*" This security concept revolves around the notion that no network perimeter should be automatically trusted, even if it resides within the corporate network, and every action should be authenticated. By implementing this principle, software systems can establish a stronger security foundation.

However, there I perceive a research gap when it comes to modeling security in system architecture. Existing approaches often consider security from the perspective of an attacker, such as in threat modeling. This technique is used to identify potential threats and weaknesses in a system or application by analyzing its architecture, components, and interactions. The results of the analysis can then inform the development of corresponding countermeasures [BJE16].

Nevertheless, this technique does not assist in the actual design of the software. Software architects and software testers view the system from a defender's perspective. The defender perspective should focus on understanding the design decisions made to secure the software system and examines which design principles have been implemented at various stages.

Returning to the example of *zero trust networking*, from a defender's perspective, one can assess whether the design principle has been effectively implemented. Such a perspective could be valuable in the design of software systems. Additionally, when testing software architectures, the defender's perspective aids in evaluating where software design principles may not have been adequately implemented.

By adopting a defender's perspective and considering the implementation of design principles, software developers can enhance the security posture of software systems. This approach shifts the focus from solely identifying vulnerabilities and threats to actively incorporating security measures into the software design process. It empowers software developers to proactively defend against cyber threats and bolster the overall security of the systems they create.

Sinkovec created a first approach on how security could be modeled from the defender's perspective by using "security views". A "security view" should provide the link between related security concerns and requirements and corresponding security design concepts. Sinkovec considers a partition into different "security views" as necessary to cope with the complexity of the security-relevant design. Sinkovec defines the "security view" to be supposed to be created during the design and engineering of a software architecture. Further a "security view" should capture certain security aspects of such a system, e.g., authentication [Sin22].

This creates the need for a modeling language that models the security of software systems on the basis of the "security views".

1.1. Contribution

This thesis aims to fill the research gap and propose a first draft of a modeling language for security in system architectures. In this thesis, I build on the concept of "security views" according to Sinkovec and extend them under the name security dimensions.

These security dimensions serve as the basis for a modeling language called the *Fortress Modeling Language (FML)*. FML models the security of already designed software systems from the perspective of a security software tester. For this purpose FML offers a possibility to model the security dimensions in order to model the different security aspects. Therefore, FML can be used to model typical security design principles for the different security dimensions.

To test the validity of FML, I also contribute an initial evaluation. For this purpose, I conduct a semi-structured expert interview. This should show to what extent FML is suitable for modeling the security of software systems.

1.2. Research questions

The following research questions arise from the research gap of a missing modeling language for security in system architectures:

RQ1 *How can a metamodel for modeling security dimensions of software architecture views look like?*

RQ2 *Which modeling elements can be used to model an Authentication, Authorization, Secure Communication, Secure Storage, Secure Build and Deployment, and Monitoring Dimension?*

1.3. Structure

The thesis is structured as follows: In chapter 2, I introduce definitions and terms for modeling software and system architectures as well as the security terms which will further be used in this thesis. As this thesis builds on the thesis from Sinkovec about a software engineering perspective of security [Sin22], I describe the background of Sinkovec's thesis which this thesis is built on in chapter 3 and define the term security dimension. In chapter 4 I describe which information the creation of the FML is based on. For this purpose, I describe what information is missing and how this information was obtained. In chapter 5, I tackle the research questions by describing the metamodel of FML. Chapter 6 describes an example realization of FML. Chapter 7, describes the method and results of the evaluation. The evaluation results are discussed in chapter 8. In chapter 9, I distinguish FML from comparable languages. Finally, I summarize the thesis in chapter 10 and offer an outlook on future extensions.

2. Foundations

In this chapter, I introduce definitions and terms for modeling software and system architectures as well as the security terms which will further be used in this thesis. For this purpose, I first clarify basic definitions of concepts from software engineering in section 2.1. Subsequently, I clarify definitions used in this thesis for concepts and terms from the area of security in section 2.2.

2.1. Software- and system architecture definitions

Since there are many different definitions in the field of software engineering, I clarify in this section the basic definitions used in this thesis.

In this thesis FML is developed, which models the security of software systems. Since there are different definitions, it is important to build a common definition of what a system is in the context of this work. For this, I use definition 1 from the IEEE standard glossary of software engineering terminology.

Definition: 1: Software system

A software system is a collection of components organized to accomplish a specific function or set of functions. [Ele90]

A software system has a structure, the system architecture. There are multiple definitions for the term system architecture. [BCK03; Ele90]. In this work, I will further use the definition by Bass, Clements, and Kazman, defined in definition 2.

Definition: 2: System architecture

The system architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [BCK03]

Therefore a system architecture is more than just a list of software components with their corresponding relations. It includes a structure that is a product of different design decisions which were made by a system architect. If two system architects get the same problem to solve, they probably design different system architectures.

According to definition 2, software architectures consist of software elements. These software elements are e.g. software components.

Definition: 3: Software component

Is an architectural entity that (1) encapsulates a subset of the system‘s functionality, (2) restricts access to that subset via an explicitly defined interface, (3) has explicitly defined dependencies on its required execution context.

Taylor, Medvidović, and Dashofy describes a software component as an architectural entity that (1) encapsulates a subset of the system‘s functionality and/or data, (2) restricts access to that subset via an explicitly defined interface, (3) has explicitly defined dependencies on its required execution context [TMD09]. For the creation of FML, the importance of data security was pointed out. Therefore data related software elements like databases are separated from common software components in this work. This leads to definition 3.

Definition: 4: View

A view expresses the architecture of the system from the perspective of one or more stakeholders to address specific concerns, using the conventions established by its viewpoint. [612]

The view of a system architecture refers to a specific perspective or representation of a system. This helps to break down and understand the system’s structure, behavior, or functionality from a particular angle. This particular angle is defined by the viewpoint.

Definition: 5: Viewpoint

A viewpoint is a set of conventions for constructing, interpreting, using, and analyzing one type of architecture view. [612]

In system architectures, a viewpoint refers to a specific perspective or frame of reference from which a system is analyzed, described, or understood. It represents a particular way of looking at the system and focuses on specific concerns or interests of a particular stakeholder or group of stakeholders.

Views are used to analyze, design, document, and communicate aspects of a complex system to different stakeholders, such as architects, developers, testers, and users. These stakeholders have a specific area of interest or focus within the system they formulate concerns.

Definition: 6: Concern

A concern is an interest in a system relevant to one or more of its stakeholders. [612]

A concern represents a particular aspect or issue. Concerns often represent cross-cutting themes that can impact multiple components of the system.

As views are an individual perspective of the system, the architectural description is the complete documentation or representation that encompasses these views.

Definition: 7: Architectural description

The architectural description is a model, document, product, or other artifact to communicate and record a system's architecture. An architectural description conveys a set of views each of which depicts the system by describing domain concerns. [Ell+96]

An architectural description, in the context of system architecture, refers to a comprehensive and structured representation of the system's architecture. It captures the essential design decisions, components, relationships, and behaviors of a system in a way that facilitates understanding, communication, and analysis among stakeholders.

The architectural description provides a blueprint or documentation of the system's structure, organization, and behavior, enabling stakeholders to gain insights into how the system is designed and how its various parts interact with each other. It serves as a common reference for architects, developers, testers, project managers, and other stakeholders involved in the system's lifecycle.

2.2. Security definitions

To avoid misinterpretation, I define some basic terms of security in this section, which will be used in the following of this thesis.

In the further course of the thesis, the terms authenticity and authorization will be used in connection with FML. Authentication and authorization are two fundamental concepts in access control and play an important role in protecting systems and resources.

In this context, I will follow the definitions of ISO/IEC 25010:2011:

Definition: 8: Authenticity

The goal of authenticity refers to the guarantee of the authenticity and trustworthiness of information, identities and entities. [250]

Authentication typically involves the use of credentials such as username and password, biometrics (fingerprint, iris scan, etc.), two-factor authentication, or other methods. The goal of authentication is to ensure that only authorized users or entities gain access to a system or resource.

Definition: 9: Authorization

The goal of authorization refers to the successful management of access and usage rights of a system. [250]

The authorization ensures that an authenticated entity can perform only those actions for which it is authorized. This is usually done by assigning permissions or roles that grant users certain privileges. For example, an authorized user may be granted permission to read, write, or delete files, while another user may be granted read-only privileges.

Furthermore, in the context of authorization, the access control models Attribute-Based Access Control (ABAC) and Role-Based Access Control (RBAC) are mentioned in the course of the thesis.

Definition: 10: Role-based access control (RBAC)

RBAC is a model for access control which uses roles to decouple the users from their privileges. The model includes five data types: [Fer+01]

Users (USERS) Users are defined as a person or software agents.

Roles (ROLES) A role is a function or a job in an organization. A role combines different privileges which allow the execution of an operation on a set of protected objects or resources. Roles are assigned to users with a user assignment. Roles can inherit from other roles, which means, that they include all permissions from the role they inherited from.

Objects (OBS) Objects are defined as resources that are protected by a security mechanism. These could be for example system resources.

Operations (OPS) Operations are defined as an action on an object which can be done via a system entity.

Permission (PRMS) A permission stands for the authorization to operate on a set of objects.

RBAC is an access protection model based on assigning roles to users. It was developed to simplify the management of permissions in large organizations. In comparison to RBAC, which is primarily based on roles and their assignment to users, the ABAC model uses attributes and attribute-based policies to control access.

Definition: 11: Attribute-Based Access Control (Abac)

ABAC is an access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes.

Attributes Attributes are characteristics that define certain aspects of the subject, object, environment conditions, and/or requested operations that are predefined and assigned by an authority. Attributes contain information specifying the class of information given by the attribute, a name, and a value

Subjects Subjects are active entities that cause the flow of information between objects or change the system state. They may be the user, the requester, or a mechanism acting on behalf of the user or requester. A subject can also be a non-personal entity, such as a system or process, rather than a human.

Subjects An object is a passive (in the context of the given request) information system-related entity (e.g., devices, files) containing or receiving information. For ABAC, attribute control can be more granular than at the object level.

Operation An operation is the execution of a function at the request of a subject upon an object. Operations include for example reading, writing, editing, and deleting.

[Hu+14]

In the ABAC model, access to resources is controlled based on a comprehensive assessment of attributes and policies. It allows flexible and fine-grained access control, as different attributes and conditions can be combined to create complex access rules. ABAC can also be dynamic, where access rules can change based on current attribute values or context.

Definition: 12: Security policy

Policies govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environmental conditions.[Hu+14]

Security policies are an important part of FML. In the context of access control, a policy refers, for example, to a set of rules, guidelines, or instructions that specify how access to resources is controlled in a system. A policy specifies which users or roles may have access to certain resources, what actions they can perform, and under what conditions access is allowed or denied. It serves as the basis for enforcing access control and maps the rules that are applied in the security system. In the further course of this thesis, the term policy will be used as a shorthand for the term security policy.

3. Background

This thesis builds on the thesis from Sinkovec about a software engineering view of security for microservice-based applications [Sin22]. In this chapter, I describe the background of Sinkovec's thesis in section 3.1. I then define the term security dimension, which is based on Sinkovec's thesis in section 3.2.

3.1. Software engineering view on security

In his thesis, Sincovec's first contribution is a catalog of security design concepts that apply to microservice-based applications. This catalog was created to structure the available measures that exist to secure such a software application from an architectural viewpoint and to provide a first approach toward a searchable and navigable knowledge database to be used by software architects. To accomplish this, he performed an adapted version of Kitchenham et al.'s Systematic Literature Review (SLR) [Sin22]. The resulting catalog of security design concepts is split into six catalogs:

- Security design principles (14)
- Security activities (10)
- Security tactics (20)
- Architectural security patterns (11)
- Security protocols (8)
- Intrusion detection/prevention system approaches

Further, Sinkovec assigned different properties to every item in this catalog, like the name and the publication, in which it was published. Some of the catalogs define special properties for example security design principles, the motivation behind the principle, or related principles. This led for example to the security design principle *zero trust networking.*

Name Zero Trust Networking

Security view Authentication, Authorization, Secure Communication

Summary All actions must be verified and all data transfers should be encrypted. There is no implicit trust between services, and trust must be evaluated continuously

Intent Always assume that the microservice network is already compromised

Related Principle Deny access by default, Principle of least privilege

Based on the insights from the conducted SLR as well as other related work, Sinkovec proposed a security metamodel for the description, modeling, and assessment of security in software architectures. The security metamodel should illustrate the most important concepts and relationships when modeling the security of such an architecture. The core of this metamodel defines the notion of a “security view”. A “security view” should provide the link between related security concerns and requirements and corresponding security design concepts. Sinkovec considers a partition into different “security views” as necessary to cope with the complexity of the security-relevant design. Sinkovec defines the “security view” to be supposed to be created during the design and engineering of a software architecture. Further a “security view” should capture certain security aspects of such a system, e.g., authentication.

3.2. Security dimension

In chapter 2 the term “view” is defined as an *expression of an architecture of the system from the perspective of one or more stakeholders to address specific concerns, using the conventions established by its viewpoint.* The term view in “security view” defined by Sinkovec, is more concerned with what is seen, the security of the software system, rather than the viewpoint from which the architecture is viewed. Therefore the term view does not quite fit in this context in my opinion and the term security dimension will be used for this concept in the further course of the work.

Definition: 13: Security dimension

A security dimension is a part of a view that refers to defined security aspects, like for example authorization. A security dimension provides the link between related security concerns and requirements and corresponding security design concepts and consists of multiple security concerns, which are to be covered by the security dimension.

In addition to the renaming, other changes were made to the concept. Sinkovec defined a secure build and deployment “view” where the build part relates to a process, but the deployment part relates to a structure. Since the simultaneous modeling of a process and a structure is difficult to implement, the secure build and deployment “view” was split into two dimensions, the secure build dimension, and the secure deployment dimension. In order to reduce the scope of FML to model the structure of systems, the secure build dimension is excluded from this work. This leads to a set of six security dimensions with their corresponding concerns, which will be referred to in the following work and are shown in table 3.1.

Dimension	Concerns
Authentication	SC-AN1: Which components can be trusted? SC-AN2: Which communication paths exist that need to be authenticated? SC-AN3: Which authentication method is used? SC-AN4: Which assets (secrets) are required to perform the authentication method? SC-AN5: Does the authentication method need to be secured? SC-AN6: Who validates the identity claim? SC-AN7: Who manages the identity information of components?
Authorization	SC-AZ1: Which authorization model is used? SC-AZ2: Which roles/attributes need to be considered? SC-AZ3: How are roles/attributes assigned to services/clients? SC-AZ4: Which permissions are assigned to each role? Which access rules exist and on which attributes do they depend? SC-AZ5: Who determines the permissions of the entity? SC-AZ6: Who validates whether the requested action is allowed?
Secure Communication	SC-SC1 Which communication paths exist in the system architecture? SC-SC2 What type of information is transported on these communication paths? SC-SC3 Which communication paths need to be secured? SC-SC4 How do communication paths need to be secured? SC-SC5 Which assets are required to secure the communication path?
Secure Storage	SC-SS1 Which storage assets exist? SC-SS2 Which storage assets are secured? SC-SS3 How are storage assets secured? SC-SS4 Which assets are required to secure the storage asset?
Secure Deployment	SC-SB2 Which application and container dependencies exist in each component? SC-SB3 Which vulnerabilities exist in those dependencies and how severe are they? SC-SB5 On which machines or deployment units are components deployed or installed? SC-SB6 How are components configured? SC-SB7 Which capabilities and actions do components have permissions for?
Secure Monitoring	SC-MT1 Which entities need to be monitored? SC-MT2 Which actions need to be traced? SC-MT3 How and by whom are traced actions evaluated? SC-MT4 How are irregular actions reported? SC-MT5 Which prevention measures are executed (if any) on an irregular action and by whom?

Table 3.1.: Overview of the security dimensions as defined in definition 13 and their concerns

4. Research Approach

In this chapter, I describe which information the creation of the FML is based on. Therefore, in section 4.1, I describe why the security dimensions alone are not sufficient as an information base and why additional information is needed. Then I describe in section 4.2 the ISTQB Security Tester Syllabus, a training document for security software testers, from which the additional information is taken. In section 4.3 I describe the Need Analysis, the method of how additional information are gained out of the ISTQB Security Tester Syllabus. In section 4.4 I present the results of the Need Analysis.

4.1. Information need

In the previous chapters I defined the terms view (definition 4) and security dimension (definition 13). The goal of FML is to be able to model the security dimensions of a system architecture. From the definition it can be derived, that if a security dimension is to be modeled, it must be associated with a view.

The stakeholders of the view define which software components are relevant. This helps to define the scope of the view and therefore of the security dimensions and to ensure that only the relevant parts of the system are covered. The stakeholders also define the level of abstraction. Security design patterns can be used at different abstraction levels. A low level of abstraction can be applied, for example, from a programmer's point of view, when they refer to catching errors in the code to prevent the system from crashing. At a high level of abstraction, the security design patterns relate, for example, to the secure exchange of data when two large applications interact. Thus, a stakeholder is essential to model the security dimensions in a meaningful way to define which design principles should be shown. In addition, the stakeholder also defines the goal of the view and thus also of the security dimensions. In this way, the stakeholder defines the purpose for which the security dimensions are to be modeled and what information is to be read.

In order to be able to use FML as a modeling language for the security dimensions, I have to create an information base that supplements the security dimensions with additional information related to a stakeholder to create a view.

For the first version, which is developed in the context of this thesis, I first focus on one stakeholder. For this purpose, the security software tester is chosen. Therefore, FML is based on the security softwaretester view. The background of this decision is the availability of good information material in the form of training material. For this purpose, the ISTQB Security Tester Syllabus, which is further described in section 4.2, is used for the creation of FML. In a future version, FML should be suitable for use in views with different stakeholders.

4.2. ISTQB Security Tester Syllabus

The ISTQB Security Tester Certification is a certification for professionals who specialize in security testing. It is offered by the International Testing Qualifications Board (ISTQB)[1]. The certification focuses on planning, performing, and evaluating security tests from multiple perspectives, including risk, requirements, vulnerability, and human factors. Further security standards and testing tools are covered.

The Certified Tester Advanced Level Syllabus Security Tester is a document that outlines the topics and subtopics that are covered in the ISTQB Advanced Level Security Tester certification. It provides the knowledge which is needed to get certificated and is separated into nine chapters:

1. The basis of security testing
2. Security testing purposes, goals, and strategies
3. Security testing processes
4. Security testing throughout the software lifecycle
5. Testing security mechanisms
6. Human factors in security testing
7. Security test evaluation and reporting
8. Security testing tools
9. Standards and industry trends

The following text was taken from the ISTQB Security Tester Syllabus and will be used as an example for the Need Analysis, described in section 4.3:

> *When evaluating security controls, the auditor should look at a system from the perspective of an attacker and anticipate how people, process or technology could be exploited to gain unauthorized access to valuable assets. Management in organizations is often surprised that the security mechanisms they thought were secure, are not. [Ist]*

Figure 4.1.: Example of a work instruction in the ISTQB Security Tester Syllabus [Ist]

The first chapter covers the fundamentals of security testing, addressing topics such as security risks, information security policies, security procedures, and security auditing. Moving on to the second chapter, it provides the context of security testing by explaining its purpose, scope, and goals. This chapter also explores different approaches to testing.

[1] https://www.istqb.org/

The third chapter focuses on the processes involved in security testing, including planning, design, execution, and evaluation. It provides a comprehensive overview of each stage. In the fourth chapter, the testing processes are aligned with the various stages of software development. Starting from requirements and extending all the way to maintenance, each stage is discussed in detail. Chapter Five is dedicated to teaching about the testing of security mechanisms. This includes topics such as system hardening, encryption, intrusion detection, and malware scanning. Moving to chapter six, it delves into the human factors of security testing. This encompasses perspectives from attackers, user awareness, and the potential for social engineering. Chapter seven addresses the evaluation of security tests and how the results are presented. It provides guidance on assessing the effectiveness of security measures. The last two chapters cover the tools used for security testing and provide insights on understanding and implementing security standards.

Figure 4.1 shows an excerpt from the ISTQB Security Tester Syllabus. This is used in the following chapter as an example for the need analysis. In the example you can see that a security tester should look at the authorization to identify possible vulnerabilities.

4.3. Method

The previous section describes why the security dimensions need to be supplemented with additional information about a corresponding view to be an information base for FML. In this section, I describe the Need Analysis, after which the additional information is procured. In the Need Analysis a document, which is a training document for a stakeholder, is analyzed systematically. Thereby important needs and information for the work of the stakeholder are extracted, to be able to represent these later in a view. The process is based on the assumption that a teacher, who teaches a student, knows best about these needs.

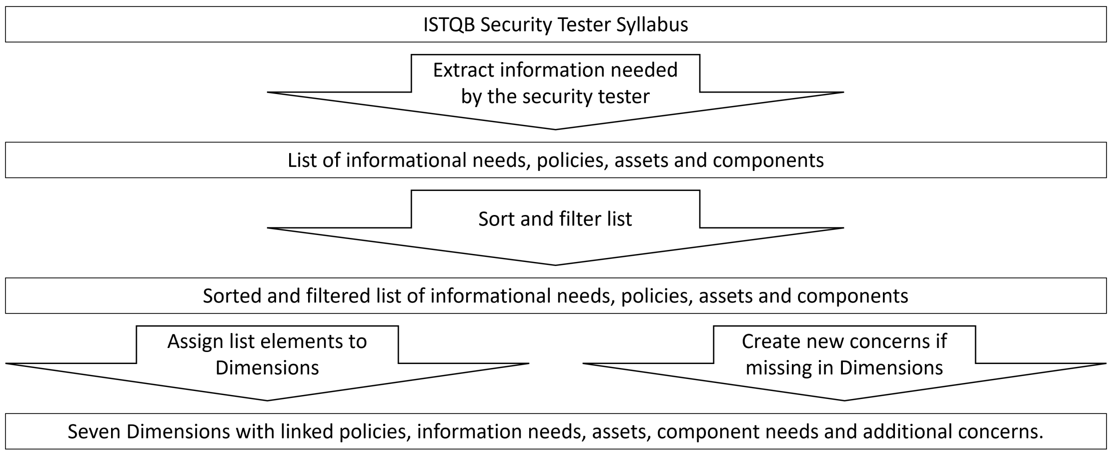

ISTQB Security Tester Syllabus
Extract information needed by the security tester
List of informational needs, policies, assets and components
Sort and filter list
Sorted and filtered list of informational needs, policies, assets and components
Assign list elements to Dimensions
Create new concerns if missing in Dimensions
Seven Dimensions with linked policies, information needs, assets, component needs and additional concerns.

Figure 4.2.: Overview over the information extraction process to enrich the security dimensions with the information needed by the security tester.

Figure 4.2 shows a graphical visualization of the Need Analysis. In the following the methodology is explained step by step. Figure 4.1 is used as an example to further clarify the process.

Step 1: Analysis

In the first step, the syllabus is analyzed chapter by chapter. All needs which are named in the ISTQB Security Tester Syllabus are noted in a list which is further called Information List. A piece of information is added to the list if one of the following criteria is met:

- The information directly describes a need for a security software tester
- The information directly describes an asset, software component, or concern that is important for security testing
- The information describes a policy

All information that meets the criteria is added to the Information List, even if they were already added during the analysis of another chapter. Thus, the Information List contains duplicates. If too many filters are already applied during the analysis process, information could be missed. Therefore, allowing duplicates prevents missing a piece of information due to similarities with a piece of information that is already in the Information List. The Information List should contain all the information from the ISTQB Security Tester Syllabus which are needed to extend the existing security dimensions but also contain unnecessary information which is not needed for the extension of the security dimensions.

In the example of figure 4.1 this means that the information “user of the system”, “technologies of the system” and “access rights” is extracted from the passage and add it to the Information List.

Step 2: Filtering

In the second step the Information List is ordered and elements are excluded based on the following list:

- Exclude information that relate to testing in general, e.g. test metrics
- Exclude information on organizational level based on figure 4.3
- Exclude information that is already covered by existing models e.g. threat model.

General testing information is excluded in step two because the dimensions are meant to model a software system to analyze the security testing. Testing information would be part of the modeling of the results of the security testing and not of the software system. Further also the organizational information based on figure 4.3 is excluded because they are out of scope for the modeling of a software system. Organizational information would fit in an enterprise model which is not in the scope of this thesis.

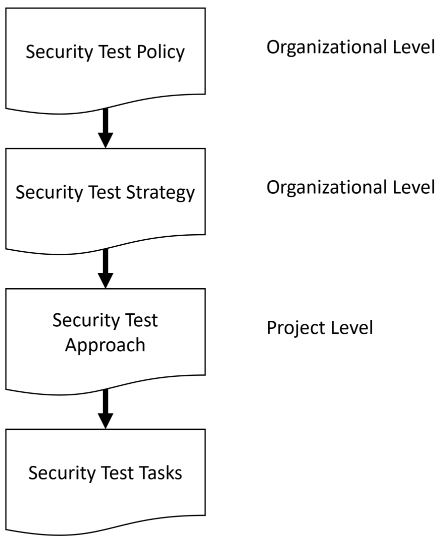


Figure 4.3.: Classification of the information from the ISTQB Security Tester Syllabus [Ist]

The result of this step is the ordered Information List. Further, the Information List should not contain information that is unnecessary to extend the security dimensions.

In the example of figure 4.1 this means that the “user of the system” information is removed from the Information List since it is at the organizational level and is not part of the system. Accordingly, “technologies of the system” and “access rights” remain in the Information List.

Step 3: Assigment

In the third step, the items of the Information List are iterated and each element is assigned to the best fitting dimensions. An information is assigned to a security dimension if it fits one of the following criteria:

- The information is covered by the concerns of the dimension
- The information is covered by the description of the dimension
- The information is covered by the goal of the dimension

If elements are not covered by security dimension concerns, a new concern is created for this element. If the criteria are met for more than one security dimension, the concern is assigned to all thematically proximate security dimensions. The result of this step is an overview of the extended security dimensions which contains the previously existing concerns, the newly added concerns, and the Policy List, a list with the information to which security dimensions a policy is assigned. Policies are not exclusive to the dimension they are assigned to. The idea is to guarantee that every policy from the information list can be modeled in at least one dimension.

In the example of figure 4.1, this means that the information "technologies of the system" and "access rights" are mapped to the security dimensions. The technologies are mapped to all dimensions since this information is relevant throughout and not limited to one dimension. The access rights refer to the authorization and are therefore assigned to the authorization dimension.

4.4. Results

In section 4.3 the Need Analysis, a method how to extract needs and information needed by a security tester from the ISTQB Security Tester Syllabus, is described. In this chapter, I describe the results of this method. The results are split in three parts: the list which contains the extracted needs and information in the Information List, the concerns which were newly created and assigned to the dimension and, in the Policy List, the policies which where mentioned in the ISTQB Syllabus and were not filtered out in step 2 of the Need Analysis.

4.4.1. Information List

The Information list is a result of the Need Analysis. It contains needs and information which are mentioned in the ISTQB Security Tester Syllabus and were not filtered out in the Need Analysis. The Information List is shown in section 4.4.1.

It is noticeable that from nine chapters, just the first five provided elements for the Information List. The first five chapters are larger than the other chapters. Further, they concentrate on security testing at the project level. The other chapter focus on the organizational level. For example, the last chapter is about industry trends and the use of standards.

Further, it is noticeable that some needs are mentioned in the syllabus more often than other needs. This could be caused by the abstractness of the often-mentioned needs.

4.4.2. Policy List

The Policy List is, as the Information List, a result of the Need Analysis. It contains the policies which are mentioned in the ISTQB Security Tester Syllabus and were not filtered out in the Need Analysis. The Policy List is shown in table 4.2.

It can be seen that policies are visible in all security dimensions. Further, not all security dimensions include the same number of policies as other security dimensions. The most policies are assigned to the secure communication dimension.

4.4.3. Extensions to the security dimensions

The concerns which extend the security dimension are a direct result of the information list which was created by the Need Analysis. It can be seen that the policy concern and a concern about the used technology are added to all security dimensions. This is a result of the importance of the policies which was described in the ISTQB Security Tester

Information	**Found in Sections**	**Assign to Dimension:**
Used technologies	1.1.1, 1.3.3, 2.4	All
Software components	1.1.1, 3.4	All
Networks	1.1.1, 3.4, 4.6	Secure Communication Monitoring
Environment of operation	1.1.1, 2.6, 3.4, 4.6	Secure Communication Secure Storage Secure Deployment Monitoring
Value of asset	1.1.2, 1.3.2	Authentication Secure Storage
Data	1.1.2, 3.4	Authentication Secure storage
Place of assets	1.1.2	Secure Storage
Security policies	1.2.1, 1.3.0, 2.4, 2.6 3.2.2	All
Malware detection mechanisms	1.3.0, 5.1, 5.6	Secure Communication Monitoring
Recovery of system	1.3.0	Monitoring
Reaction time for breaches	1.3.0	Monitoring
Open ports	1.3.1, 5.4	Secure Communication
Protection of data	1.3.1	Secure Storage
Application of security updates	1.3.1	Secure deployment view
Roles	1.3.3, 5.2	Authorization
Access rights Access limitations	1.3.3, 3.4, 4.5, 5.2	Authorization
Encryptions	2.6, 4.5, 5.1, 5.3	Secure Communication
Backups	2.6	Secure Storage Secure Deployment
Authentication	4.5, 5.2	Authentication
Update mechanisms	5.1	Secure Deployment
Sand boxes	5.1	Authorization
Certification mechanisms	5.2	Authentication
Password rules	5.2	Authentication

Table 4.1.: Information list of the extraction process after step 2 (section 4.3)

Policy Name	Description	Assigned to Dimension
Minimum Access	Defines the access rights to a software component or asset.	Authorization
Network Access	Defines criteria to access networks and defines permissions on network	Secure Communication
Remote Access	Defines required criteria to use a network remote	Secure Communication
Internet Access	Defines how internet access is filtered (e.g. whitelisting)	Secure Communication
User Account Management	Defines creation maintenance and deletion of user account	Authorization
Data Classification	Classification of sensitive data into classes: public, confidential, highly confidential, private, secret	All
Server Security	Defines how a server is configurated, with all settings, monitoring and auditing	Deployment
Mobile Devices	Policy which relates to mobile devices as these can get lost easier	Secure Communication
Guest Devices	Defines how and if guests can use the internet	Secure Communication
Password Policy	Defines password requirements	Authentication
Malware Protection	Defines how malware is prevented, detected and removed	Monitoring
Incident Response	Defines how to respond to an incident	Monitoring
Software Licencing	Defines where licences are defined and monitored	Deployment
Elektronic Monitoring and Privacy	Defines which communication in the company is monitored	Monitoring

Table 4.2.: Policies which are mentioned in the ISTQB Security Tester Syllabus. The Policies were assigned to dimensions in the Need Analysis, described in section 4.3.

Syllabus. As the used technology was described as a possible security vulnerability, it was added to all dimensions.

Further, concrete concerns were added to the dimensions. One example would be the concern SC-SS-BACKUP which relates to the used backup strategy of software components.

Dimension	Concerns
Authentication	SC-AN-TECH: Which technologies are used for the authentication? SC-AN-VALUE. How valuable are the assets? SC-AN-POLICY: Which security policies need to be followed during the authentication? SC-AN-PASSWORD: Which password rules are defined for authentication?
Authorization	SC-AN-TECH: Which technologies are used for the authorization process? SC-AN-POLICY: Which security policies need to be followed during the authorization process?
Secure Communication	SC-SC-TECH: Which technologies are used to secure the communication? SC-SC-POLICY: Which security policies need to be followed to secure the communication?
Secure Storage	SC-SS-TECH: Which technologies are used to secure the storage? SC-SS-POLICY: Which security policies need to be followed to secure the storage? SC-SS-VALUE: How valuable are the storage assets? SC-SS-BACKUP: How and when are backups created?
Secure Deployment	SC-SB-TECH: Which technologies are used on the deployed system? SC-SB-POLICY: Which security policies need to be followed on the deployed system? SC-SB-ENV: On which operating systems do the applications run? SC-SB-PORT: Which ports are open? SC-SB-UPDATES: How are security updates installed? SC-SB-SANDBOX: Which services run in a sandbox?
Secure Monitoring	SC-MT-TECH: Which technologies are used for the monitoring? SC-MT-POLICY: Which security policies need to be followed during the monitoring? SC-MT-MALWARE: Which antimaleware tools are used? SC-MT-RECOVER: How can the system recover?

Table 4.3.: Added concerns to the dimensions from the extraction process in section 4.3. The concerns extend the existing concerns (table 3.1).

5. Creation of the Fortress Modeling Language

In the previous chapter, the Need Analysis, which extracted the needed information for the security tester from the ISTQB Security Tester Syllabus, was described. This information was used to complete the security dimensions. Based on the results from the previous chapter, FML is created.

In this chapter, I describe the metamodel of FML and how the metamodel was developed. To do this, I describe my research method in section 5.1. In section 5.2 I describe the metamodel that was created by the research method.

5.1. Research method

In this section, I describe the research method for creating the metamodel of FML. For this purpose, in section 5.1.1 I present the information that is used to create the metamodel. In section 5.1.2 I describe the process how the metamodel is created from the information.

5.1.1. Information on which the metamodel is built

In this subsection I describe the input for the process used to create the metamodel of FML. The process for creating the metamodel is based on following information:

- The security dimensions by Sinkovec (table 3.1)
- The additional concerns that were added to the security dimensions during the need analysis (section 4.4)
- The results of the need analysis in the form of the information list and the policy list (section 4.4)

Additional input is provided by sinkovec's evaluation of the dimensions from his thesis [Sin22]. He conducted an expert interview evaluating the security dimensions. For this he created a prototype modeling language to let the experts evaluate the authentication dimension on the basis of an example. Additionally, the experts gave feedback on the prototype language for the authentication dimension. This feedback is directly incorporated into FML.

The following list of remarks was included in the evaluation:

Dimension	**Modeled security design concept**
Authentication	- Zero trust networking - No authentication / Authentication not required - Plaintext-based authentication - Protocol-based authentication - API gateway
Authorization	- Principle of least privilege - Deny access by default - API gateway
Secure communication	- Unencrypted communication - Use asynchronous messaging - TLS / mTLS
Secure storage	- Secret management
Secure deployment	- Secure-by-default
Monitoring	- Distributed tracing - Input validation - Security gateway

Table 5.1.: Overview of the randomly chosen design principles from the catalog of Sinkovec (see chapter 3), used to extend the case studies during the process to create FML.

- The description, where successful and failed connection attempts are logged, is missing
- It is unclear, how the secret management made
- Specific details about the implementation are needed
- Better use implementation agnostic notation
- There should be a notation for rulebreaking
- Network segments should be included
- More general notations than specific notation for each service are preferred

Sinkovec's modeling language for the security dimensions was not further evaluated in his work as the modeling language was not in his scope. Nervertheless the feedback is used to improve FML.

5.1.2. Creation process of the metamodel

The information listed in the subsection before are now used as an input for the incremental and iterative process to create FML. The process is is visualized by figure 5.1 and further explained in this chapter. The output is the metamodel of FML and the Modeling

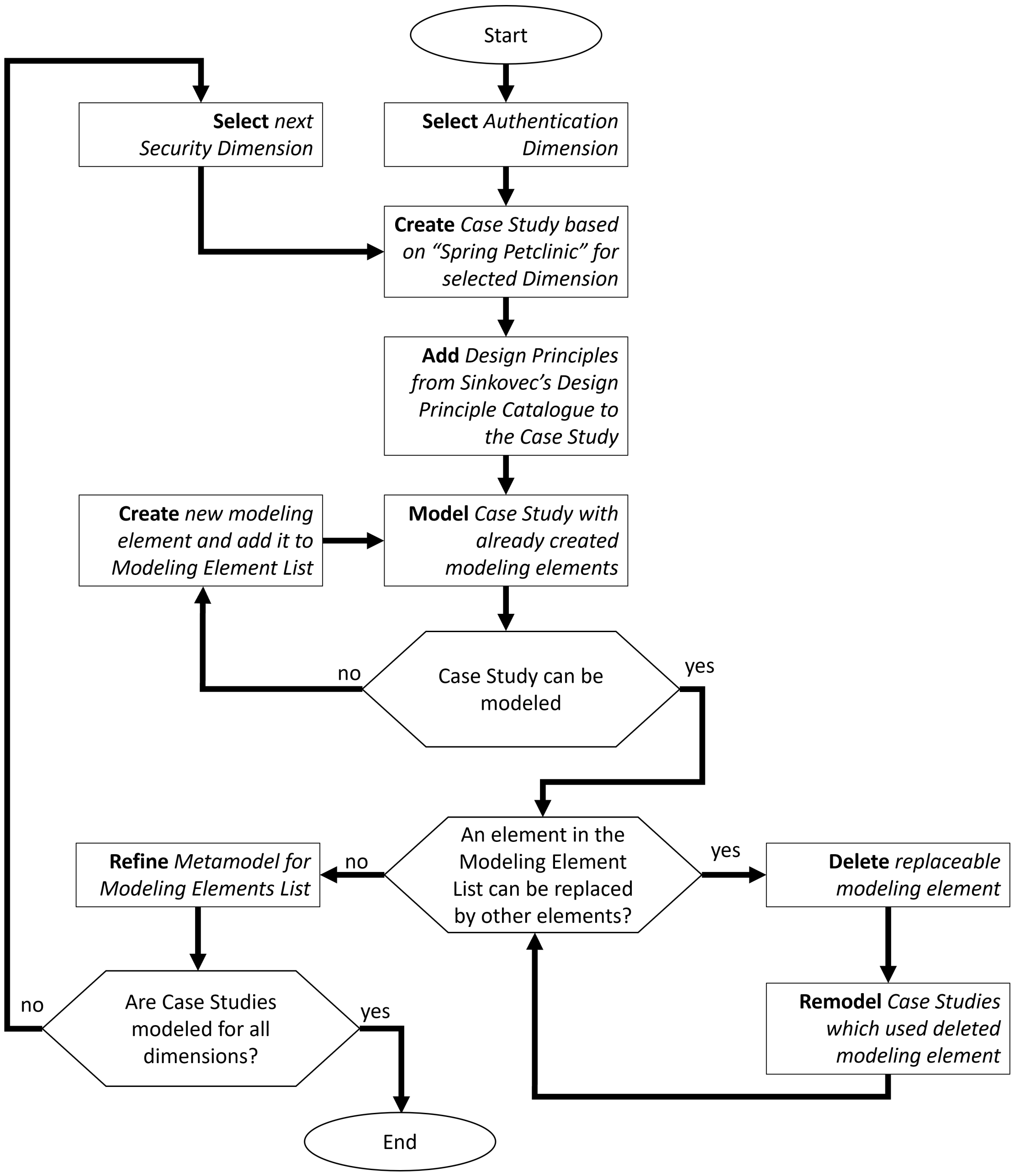

Start
Select *next Security Dimension*
Select *Authentication Dimension*
Create *Case Study based on "Spring Petclinic" for selected Dimension*
Add *Design Principles from Sinkovec's Design Principle Catalogue to the Case Study*
Create *new modeling element and add it to Modeling Element List*
Model *Case Study with already created modeling elements*
Case Study can be modeled
no
yes
An element in the Modeling Element List can be replaced by other elements?
no
yes
Refine *Metamodel for Modeling Elements List*
Delete *replaceable modeling element*
Remodel *Case Studies which used deleted modeling element*
Are Case Studies modeled for all dimensions?
no
yes
End

Figure 5.1.: Flowchart of the FML creation process

Element List, which also contains all elements. The process iterates over all security dimensions . For each security dimension a case study is built based on the “Spring Pet Clinic”, an example for microservice architectures.

Spring Pet Clinic

The Spring Pet Clinic[1] is an open source project that provides a sample application for developing web applications using the Spring Framework. It is designed to provide developers with a real-world example of best practices and design patterns in Spring application development.

The main goal of Spring Pet Clinic is to demonstrate the management of a veterinary practice. The application allows you to record and manage veterinarians, pets, owners and appointments. It provides features such as adding, editing and deleting animals and owners, creating and managing appointments, and searching for vets and animals.

The Spring Pet Clinic is a widely used sample project that is often used by developers as a starting point or learning resource to learn how to develop web applications using the Spring Framework. It provides extensive code and documentation to help developers understand and apply best practices and proven development practices.

The Pet Clinic consists of the following components: VetService: VetService is responsible for managing the veterinarians in the veterinary practice. It provides methods for adding, updating, and deleting veterinarians, as well as retrieving a list of all veterinarians. The VetService can access the VetRepository to persistently store vet data. CustomerService: The CustomerService is responsible for managing the owners of the pets. It provides methods for adding, updating, and deleting customers, as well as retrieving customers by name, last name, or phone number. VisitService: VisitService is responsible for the management of veterinary visits (appointments) in the PetClinic. It provides methods for adding, updating, and deleting visits, as well as retrieving visits for a specific pet or veterinarian. In addition, there is an AdminService to manage the system, as well as a logging service for monitoring.

The FML creation process

The process iterates over all security dimensions but starts with the authentication dimension. The reason behind this decision was the already existing feedback in the work by Sinkovec which contained feedback to his prototype modeling language which was created by him. For the selected authentication dimension a case study is created. The case study is based on “Spring Pet Clinic”.

To archive this, design principles of the Design Principle Catalogue by Sinkovec are used to extend the case study. The design principles used are shown in table 5.1. Further the case study should include all kinds of information of the Information List. Also all polices of the Policy List, which were assiged to the authentication dimension should be includable in the case study.

[1]https://github.com/spring-petclinic/spring-petclinic-microservices

As the authentication dimension is the first dimension, the Modeling Element List is empty. Therefore new elements are created until the case study can be modeled. To reduce the number of elements, all elements of the Modeling Element List are checked if they can be replaced by an other element in the Modeling Element List. If there are no replaceble elements, a metamodel is created for all elements in the Modeling Element List and the next security dimension is selected.

For the selected security dimension also a case study is created, like for the authentication dimension in the first step. The created case studies for all dimensions can be found in appendix A. Also for this dimension the case study is enriched with further information. To model the case study, the already existing modeling elements in the Modeling Elements List, are used. If the Case Study can not be modeled with the existing modeling elements, new modeling elements are created and added to the Modeling Elements List. If an existing element is a specialization of the needed element, the generalization is added as a new element and will probably replace this element in a later step. The goal is, not to have a number of specialized modeling elements which are hard to learn, but generalized elements which make it easy to understand all dimensions.

Therefore, the next step is to filter out unnecessary modeling elements. To archive this, all existing modeling elements of the Modeling Elements List are checked if they can be replaced by an existing modeling element. If no more modeling elements can be replaced, the meta model is updated and the next dimension is selected, until all dimensions got a case study. This leads to the final meta model.

5.2. The Fortress Modeling Language metamodel

In this section I describe FML the result of the research method, presented in the previous section. In FML there are three different categories of elements:

Component elements The *component elements* describe the artifacts and configurations of a system. These can be machines or databases, for example. Elements that define the configuration of a machine also fall under the *component elements.* It is important that in FML the assumption is made that the *component elements* represent the real existing system. This means that during the evaluation it is not necessary to consider that something might not be implemented.

Policy elements The *policy elements* describe policies that apply in a system, such as a policy about password length. Policies differ from *component elements* because they do not describe the system, but the rules that apply in the system. Also with the policies the assumption is made that the modeled policies are valid in the real system, if they were modeled in the same way in FML.

Security dimension elements The *security dimension elements* symbolize the security dimensions. The *security dimension elements* indicate which *component elements* are used in which security dimension.

In addition to the elements, FML defines relationships between the elements:

Relationships The *relationships* show how elements behave and interact with each other. The *relationships* also define how elements change when they interact with another element.

In the following the elements and *relationships* of FML are introduced and explained. For this purpose all elements and *relationships* are listed and explained in section 5.2.1, where also the hierarchy of the elements and *relationships* is explained. In section 5.2.2 it is explained how elements can be owned by other elements. Which *relationships* are used by which elements is described in section 5.2.3. In section 5.2.4 I describe which *relationships* and elements are used by which security dimension.

5.2.1. Hierachy of the elements in FML

In this section I introduce the elements of FML and explain, which element has which function and which elements inherit from other elements.

Hierachy of the security dimension elements

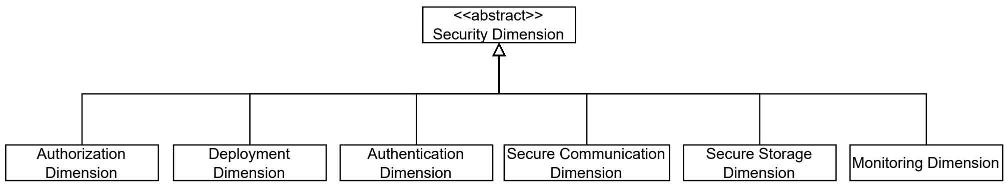


Figure 5.2.: Hierachy of the security dimension elements in the meta model of FML.

The *security dimension elements* in figure 5.2 represent the security dimensions which are a result of the need analysis from section 4.4. Each individual *security dimension element* is specified as an own object in the metamodel which inherits from the *abstract security dimension.* This allows to specify the used elements for every security dimension, as done in section 5.2.4. Further this leads to an extendability of the security dimension concept, as further security dimensions can be added in the hierachy in future work.

Hierachy of the component elements

The hierarchy of the *component elements* in FML is shown in figure 5.3. FML defines the following *component elements*:

Component The *component* stands for software components in the system such as services or applications. In FML databases are not explicitly considered as *components* since they are modeled as a separate element

Database The *database* is a special *component* that models only databases. it also contains information about the database software, such as mySQL and its version number. In addition, the *database* contains information about the classification of

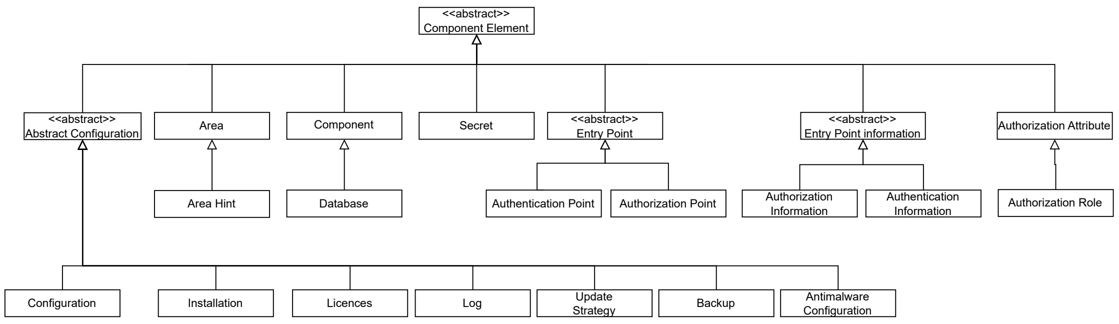


Figure 5.3.: Hierarchy of component elements in FML

the data it contains. Databases are displayed independently from other software components. Because FML is focused on data security also, *databases* are modeled separately due to their significance in data security

Secret A *secret* is private information that serves as a key to unlock protected resources or sensitive information in tools, applications, containers, DevOps, and cloud-native environments. Example of this is a private key to log in to a database as admin.

Abstract Configuration An *abstract configuration* defines the configuration of a *component element*. The *abstract configuration* can appear in different forms: 1. *Configuration*, which defines the configuration of a network, a machine or a software. 2. *Installation*, which defines the software installed on a machine. 3. *Licenses*, which specifies the licenses of a software. 4. *Log*, which defines what information is recorded when a system is operated. 5. *Update strategy*, which defines when and how updates are applied to a machine. 6. *Backup*, which defines when to create a backup, which trigger to create it and where to store the backup. 7. *Antimalware configuration*, which defines which antimalware programs and scanners run on a machine or in a network.

Area An *area* represents environments such as machines, networks or even physical environments such as houses, rooms or locations. Accordingly, an *area* shows where an assigned component is located.

Area Hint *Area hint* has the same meaning as an *area*. The difference is that the *area hint* occurs in dimensions in which environments would actually not be modeled, but special environments are nevertheless relevant for the evaluation of software system security. In this case the information of the *area* is borrowed from another dimension and modeled as an *area hint*. A dimension therefore always contains either *area*, *area hint* or none of these elements.

An example of the use of *area hint* is the authentication dimension, which models a behavior of the system. Since the environments of the software components are not always relevant for the evaluation of the security of the authentication process, the information is only relevant for communications that leave a network and may have to be secured separately. Let's assume that there is a poorly secured connection, but it takes place within a well-protected corporate network. Then the *area hint*, which models the corporate network, is necessary to assess the security of the connection, since the security software tester may assume that the connection could go over the internet and therefore assesses the connection as a security risk.

Authentication Point The *authentication point* indicates a communication interface of a *componen*t or *database*. It indicates that a *component* that connects to another component via an *authentication point* must authenticate itself.

Authentication Information The *authentication information* contains information about an authentication method. This includes the method used, e.g. an authentication via user and password, the technology like for example oauth2.0 and optionally the concrete implementation. The implementation is specified by a link to a sequence diagram.

Authorization Point The *authorization point* indicates a communication interface of a *component* or *database*. It is similar to an *authentication point*, but relates to the authorization.

Authorization Information The *authorization information* is similar to the *authentication information* but relates to the authorization. It also specifies the method, technology and optional the concrete implementation.

Authorization Attribute An *authorization attribute* defines an attribute that is required to access a component or database. Attributes are properties or characteristics of users, resources or the environment. They are used as a basis for access control decisions. The *authorization attribute* is created to be able to model attribute-based access control (see chapter 2) in FML.

Authorization Role An *authorization role* defines an role that is required to access a component or database. An authorization role defines the permissions and privileges assigned to a user based on their role in the organization or system. The role serves as an abstract representation of a user's duties and responsibilities and enables efficient management of access rights because permissions are assigned at the role level rather than at the individual user level. The *authorization role* is created to be able to model role-based access control (see chapter 2) in FML.

Hierachy of the relationships in FML

The relationships between the elements of FML are shown in figure 5.4. In FML the following relationships are defined:

Upgrade The Upgrade-Relationship connects two elements, a parent element and the child element. An *Upgrade* is interpretet, that information and properties from the

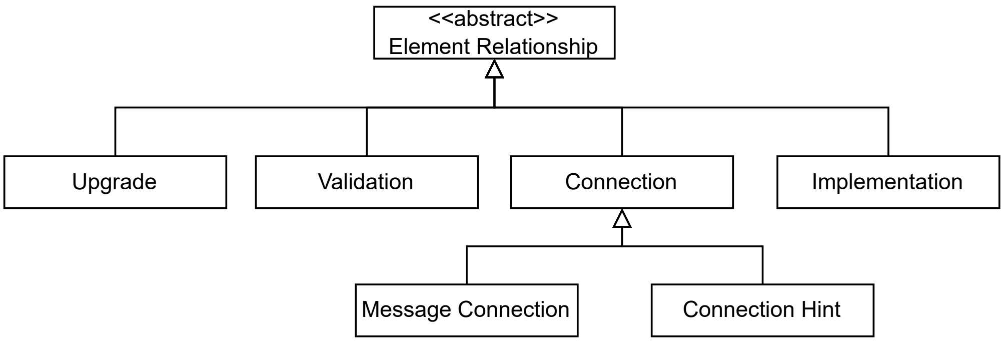


Figure 5.4.: Hierachy of the relationship elements in the meta model of FML.

parent element are copied to the child element, but the information and properties of the child element overwrite the information and properties from the parent element.

Validation The *Validation* is a relationship between two component elements, a validator and an element to be validated, like for example an authentication. A *Validation* specifies who validates authentication and authorization access.

Connection The *Connection* stands for a possible connection between two components. This indicates that the components are communicating with each other. If two components need to authenticate themselves before establishing a connection, this is indicated by the *Connection* pointing to the authentication point.

Message Connection The *Message Connection* is a special *Connection*. Unlike the standard *Connection*, the *Message Connection* contains one or more messages that are exchanged between the components. This is used, for example, in the secure communication dimension to track how messages are sent through a system.

Connection Hint *Connection Hint* is a special *Connection* that occurs only in security dimensions that do not contain the *Connection*. The security dimension borrows the information about the *Connection* from another security dimension, if this is useful for the evaluation. This also indicates that not all *Connections* were modeled in this security dimension, just important *Connections*, in form of a *Connection Hint*.

Implementation The *Implementation* indicates that an *Abstract Element* implements an *abstract configuration* or an *entry point configuration*. This means that the *Abstract Element* conforms to the specifications of the *Abstract Configuration* or *Entry Point Configuration*. An example would be the implementation of *Authentication Information* by an *Authentication Entry Point*. The Implementation indicates how the *Authentication Entry Point* is specified.

Policy elements in FML

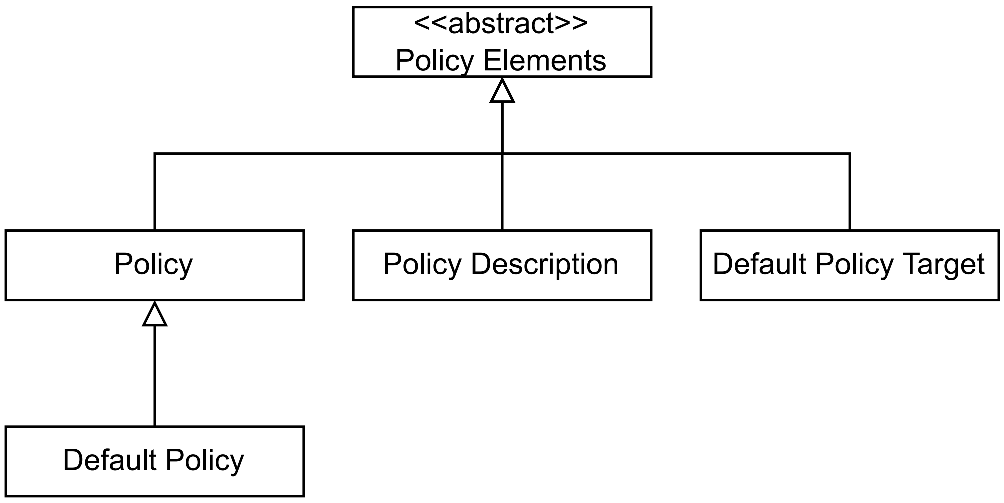


Figure 5.5.: Hierachy of the policy elements of FML.

The hierarchy of the *policy elements* in FML is shown in figure 5.5. In FML the following *policy elements* are defined:

Policy The *policy* models a set of rules or regulations that define how security aspects are handled in a system. It defines the allowed or disallowed actions, behaviors, or configurations and which are used to ensure the security of data, resources, and systems.

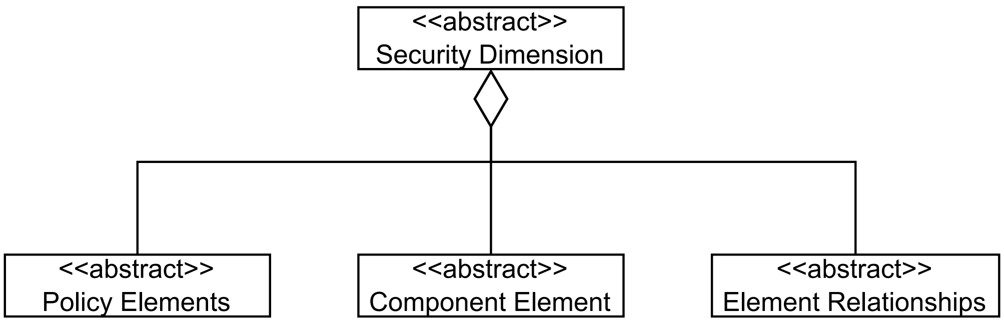
Default Policy The *default policy* is a special *policy*. It differs from the *policy* in that in an FML model *default policies* are always valid and only violations are modeled.

Policy description A *policy description* describes the content, configuration and the specification of a *policy* or a *default policy*. In a *policy description* of a password policy, for example, the minimum length of passwords could be defined.

Default Policy Target A *default policy target* describes the target elements of a *default policy*. With a *default policy target* you can for example define that a policy that was modeled as a *default policy* only refers to *databases*.

5.2.2. Ownership of the elements in FML

In this section I introduce the ownership of the elements of FML. It is explained which elements can own other elements and which elements consist of other elements.



<<abstract>> Security Dimension
<<abstract>> Policy Elements
<<abstract>> Component Element
<<abstract>> Element Relationships

Figure 5.6.: Ownership of the general security dimension.

Security dimension ownership

The ownership of the *security dimensions* in general is shown in figure 5.6. Each *security dimension* consists of *policy elements*, *component elements* and *element relationships*. The concrete *component elements* for each *security dimension* are defined in section 5.2.4.

Component element ownership

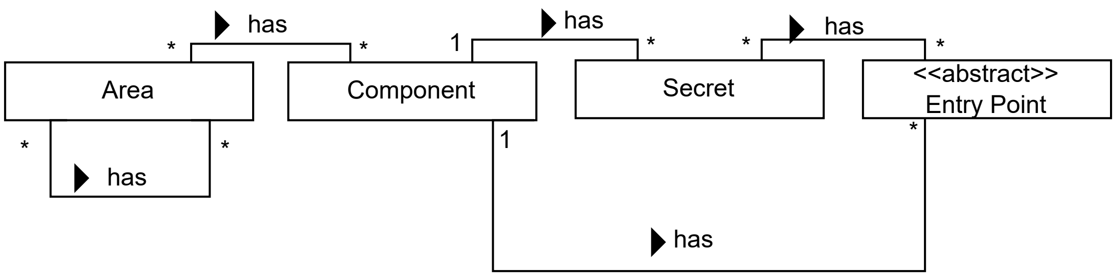


Figure 5.7.: Ownership of the component elements

The ownership of the *component elements* is shown in figure 5.7.

Areas have any number of *components*. This specifies where *components* are located in the system. For example, it is possible to define on which machine, modeled as an *area*, which application, modeled as a *component*, is running. A *component* can belong to several *areas*. Thus it is possible to model, for example, if an application can access several networks. Additionally *areas* can have further *areas*. This allows to define, for example, which networks exist in the system and which machines are connected to which network.

To be able to model different secured connections to a *component*, *components* can have any number of *entry points*. These are either *authentication points* or *authorization points*.

Secrets are assigned to *components* in which they are stored. *Secrets* can additionally be assigned to *authentication points.* In this case they contain authentication information, like credentials, which can be used for authentication at the assigned *authentication point.*

Policy element ownership

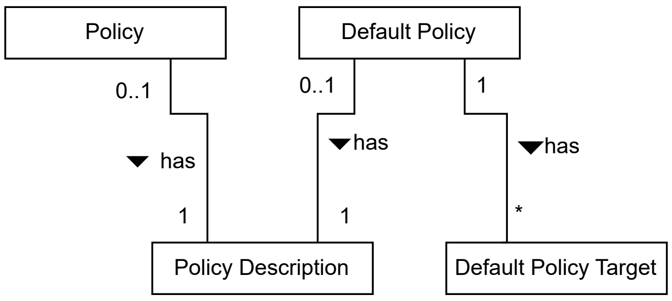


Figure 5.8.: Ownership of the policy elements.

The ownership of the *policy elements* is shown in figure 5.8. *Policies* and *default policies* have a *policy description* that describes the *policy.* This must clearly explain how the *policy* is defined. For example, the *policy description* of a password policy must clearly define what the minimum strength of passwords should be. *Default policies* can have any number of *default policy targets.* This defines which elements the *default policy* applies to. For example, the password policy applies only to *authentication points.* If the password policy is modeled as a *default policy*, it is possible to define the *authentication point* as a *default policy target.* This would make it clear that the password policy does not apply to, for example, *areas.*

5.2.3. Behaviour of the elements in FML

In this chapter I present the behavior of the elements of FML. Thereby it is explained how the elements behave to other elements.

Policy element behaviour

The behaviour of the *policy elements* is shown in figure 5.9. All *component elements* and *element relationships* can implement or violate any number of *policies* and *default policies.* For *policies*, the implementation is explicitly modeled and the violation is implicitly assumed by not modeling. For *default policies* it is the other way round, here the violation is modeled explicitly and the implementation is assumed implicitly.

For example, if an *authentication information* satisfies a password policy modeled as a *policy*, the implementation is modeled. If a zero trust policy is modeled as a *default policy*, which states in the *policy description* that all *connections* in the system must be authenticated, only violations are modeled. These violations would point to the *connection* and not to the *components* in this case.

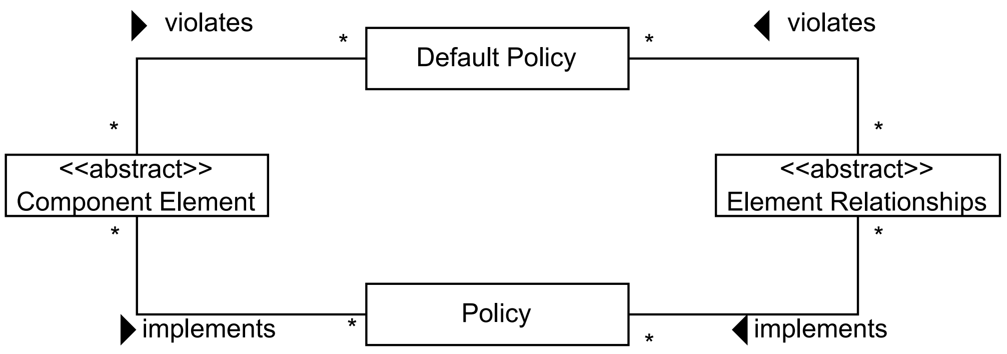


Figure 5.9.: Behaviour of the policy elements.

Component element behaviour

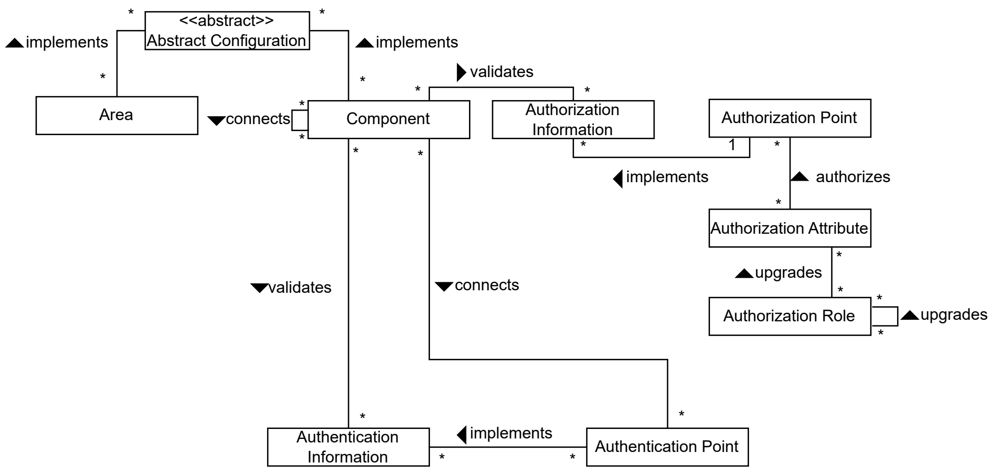


Figure 5.10.: Behaviour of the component elements

The behaviour of the *component elements* is shown in figure 5.10.

Components can connect to other *components*. This can be modeled by 2 possibilities: 1. the *component* is directly connected to another *component*. In this case no authentication takes place during this communication. 2. the *component* is connected to an *authentication point*, which belongs to a *component* (see figure 5.7).

Another behavior of *components* is the validation of any number of *entry point information* in the form of *authentication information* or *authorization information* (see figure 5.3). An *authentication information* is implemented by an *authentication point*, indicating how authentication is performed at the implementing *authentication point*. An *authentication information* can be implemented by any number of *authentication points*.

The implementation of *authorization information* by an *authorization point* is similar.

This behavior is best explained with an example: A client communicates with an api gateway and must authenticate itself. The validation of the authentication is done by an auth service. In FML the *components* client, api gateway and auth service are modeled. The communication between client and api gateway is modeled by a communication via an *authentication point*, which belongs to the api gateway. This *authentication point* implements an *authentication information.* The *authentication information* is validated by the auth service. A graphical visualization of the example can be found in the case study in section 7.1.2.

The *authorization point* indicates that access to a *component* must be authorized. *Authorization attributes*, which authorize the *authorization point*, indicate that for the access the corresponding *authorization attribute* must be held by the accessor. See also the description of ABAC in chapter 2.

Authorization roles are an upgrade of any number of *authorization attributes.* Thereby the *authorization role* gets the *authorization attribute* assigned. *Authorization roles* can also be upgrades of other *authorization roles.* Thus the upgraded *authorization roles* get the authorization rights of the upgrading *authorization roles.*

Abstract configurations can be implemented by any number of *components* and *areas.* Thus, the owned elements guarantee that they fulfill the configurations of the *abstract configuration.* For example, if an *installation* as a special *abstract configuration* (see section 5.2.1) contains the firefox browser in version 113, all machines which implement this installation are guaranteed to have this browser installed in the defined version 113.

Configuration behaviour

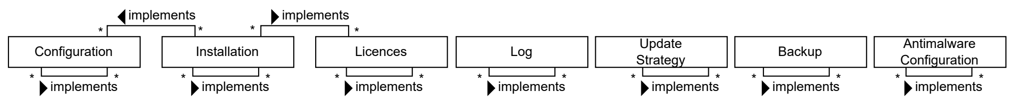


Figure 5.11.: Behaviour of the subclasses of the abstract configuration

The behaviour of the subclasses of the *abstract configuration* is shown in figure 5.11. An element of a subclass can implement other elements of the same subclass. Thereby additional information is added to the implementing element. If properties conflict with each other, the properties of the implementing element apply.

For clarification the *installation* serves as an example. An *installation* can be implemented by other *installations.* In this example, let's assume that there is a base installation in a system, which should be available on all machines in an example system. This base installation contains python 3.10 and firefox 113. Additionally there should be machines in the example system, which have just python 2.6 installed, for compatibility reasons.

In FML the example can be modeled by defining the *installation* "base installation" and the *installation* "compatibility installation". The "compatibility installation" implements the “base installation”. This will keep firefox 113 on all machines, the python version would be 3.10 for the “base installation” and 2.6 for the “compatibility installation”.

Installations can additionally implement *configurations* and *licenses* which then refer to the *installation*. For example, if there is an *installation*, which contains the web server software nginx and the containerization software docker, the implemented *configuration* could define the ports used by nginx, while the implemented *license* specifies the docker license.

5.2.4. Elements of the security dimensions

In section 5.2.2 it is defined that security dimensions in FML consist of *policy elements*, *component elements* and *element relationships*. Additionally, FML defines which concrete elements are used in a security dimension. The limited number of elements is intended to ensure that the modeled security dimensions do not become too large and model too much information that actually belongs to other security dimensions. Thereby all concerns of the respective security dimension should be able to be modeled with the elements. These are represented in section 4.4.

Authentication dimension

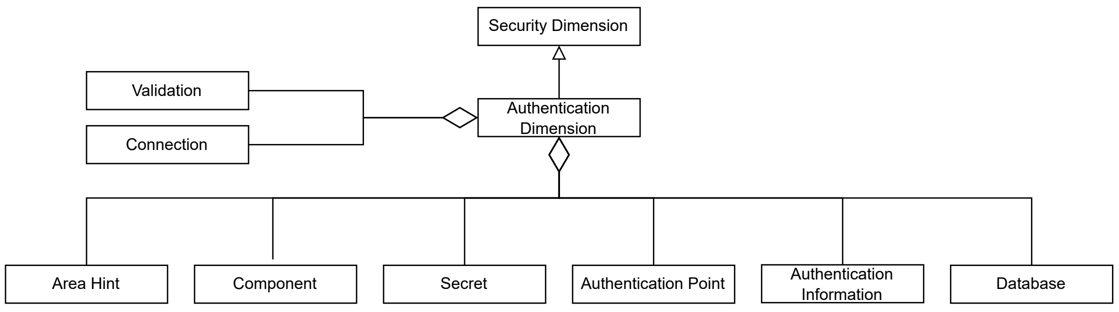


Figure 5.12.: Elements of the authentication dimension.

Figure 5.12 shows the elements that are allowed to be used for modeling an authentication dimension. In an authentication dimension the *component*, *database*, *secret*, *authentication point* and *authentication information* can be used. The secret in combination with the *authentication point*, which implements *authentication information*, is the core of an authentication dimension, because it models the actual authentication process. The *validation* and *connection* are used to show where connections exist and which component validates authentications.

The *area hint* is used to evaluate the authentication. With the *area hint*, it is modeled whether the communication between components takes place in the same network or even on the same machine. In this way, it can be evaluated whether an authentication method is sufficient, for example, if it takes place in a partitioned network.

Authorization dimension

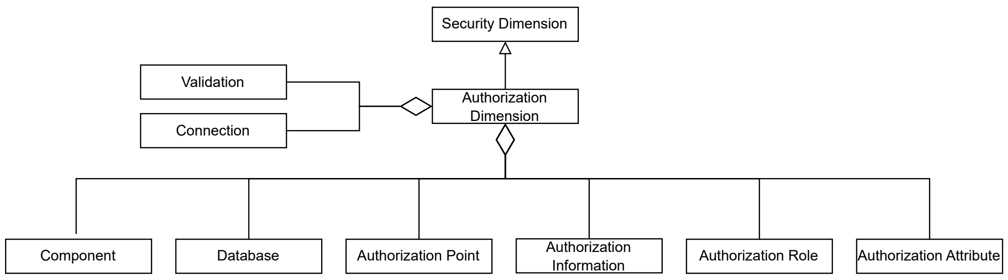


Figure 5.13.: Elements of the authorization dimension.

Figure 5.13 shows the elements that are allowed to be used for modeling an authorization dimension. In an authorization dimension the *component*, *database*, *authorization attribute*, *authorization role*, *authorization point* and *authorization information* can be used. The *authorization role* and *authorization attribute* in combination with the*authorization point*, which implements an *authorization information*, model the authorization process. The *validation* and *connection* are used to show where connections exist which need to be authorized and which component validates authorization.

The authorization dimension does not contain *area hints*. The goal of the dimension is to evaluate the authorization and to see where individual users may have too many rights. Since this is very complex, the number of elements here should be particularly small.

Secure communication dimension

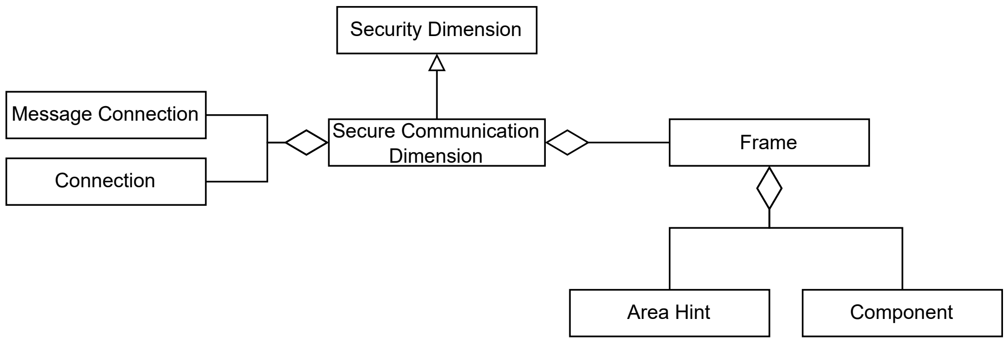


Figure 5.14.: Elements of the secure communication dimension.

Figure 5.14 shows the elements that are allowed to be used for modeling a secure communication dimension. The secure communication consists of several frames. These frames contain *components* and *area hints*. The frames always represent a specific communication sequence. This is for example an API call. They show all messages between the components, which are triggered by this api call. A finished secure communication dimension strongly resembles a communication diagram of *Unified Modeling Language (UML)*.

In a secure communication dimension, existing connections between components in the system are represented by *connections*. Connections on which communication takes place within the modeled communication are replaced by a *message connection*. Thus the dimension should always represent a complete picture of the communication in a system.

The area hint is used to evaluate the communication, since it may, for example, make a difference in the evaluation whether an unsecured connection exists in the same network or not.

Secure storage dimension

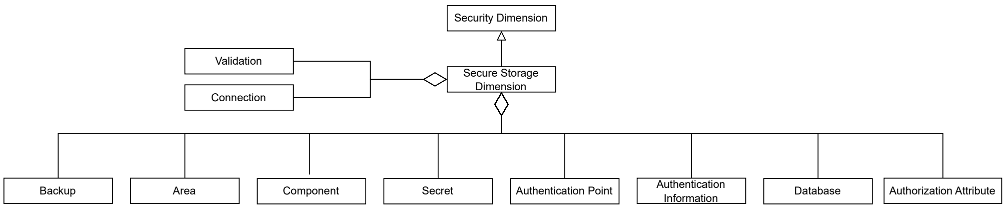


Figure 5.15.: Elements of the secure storage dimension.

Figure 5.15 shows the elements that are allowed to be used for modeling a secure storage dimension. In an secure storage dimension the *component*, *database*, *secret*, *authentication point*, *authentication information*, *authorization attribute*, *area* and *backup* can be used. The *secret* in combination with the *authentication point*, which implements *authentication information*, is only included in the secure storage dimension to model the authentication of *databases*. In the secure storage dimension only authentication processes in the system are modeled if they belong to the *connection* to a *database*. The authentication processes are included to evaluate the data security of a system.

In the authorization dimension, the *secret* therefore refers to passwords or access codes for databases. In order to evaluate how secure the *secrets* are, *authorization attributes* belonging to the *secret* are also modeled in the secure storage dimension to show who has access to the *secret*.

The *area* in the secure storage dimension shows, where *databases* are located, both the virtual and physical. This is intended to enable an evaluation of the resilience of a system, as it can be used, for example, to model the location of backups.

Secure deployment dimension

Figure 5.16 shows the elements that are allowed to be used for modeling a secure deployment dimension. In a secure storage dimension the relationship *connection* and the elements *component*, *database*, *area*, *configuration*, *update strategy*, *installations* and *licences* can be used. The *connection* and the *area* can be used to model the infrastructure

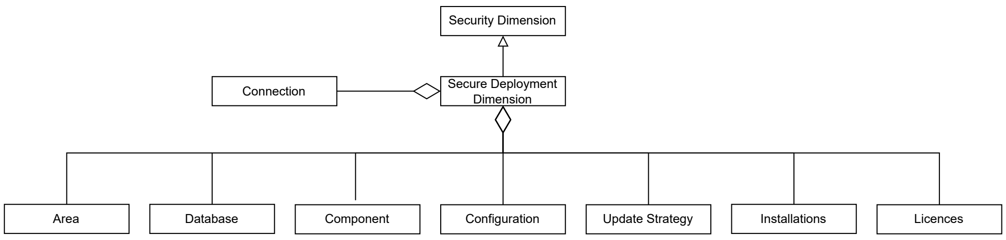


Figure 5.16.: Elements of the deployment dimension.

of a system. Additionally, the *configuration*, the *update strategy*, the *installations* and the *licenses* define how the *components*, *areas* and *databases* of the system are configured.

The *update strategy* makes it possible to read how a *component* or an *installation* is updated and what the trigger is. This makes it possible to easily identify where an update strategy may be insufficient.

The *installations* allows a good overview of the different software including the version numbers. Thus outdated versions can be found quickly. In addition, dependencies are also modeled with the *installation*. Thus outdated dependencies are also included in the secure deployment dimensions.

The *licenses* are also part of the secure deployment dimension. Missing and expired licenses are also a security risk, as the need analysis in section 4.4 has shown.

Secure monitoring dimension

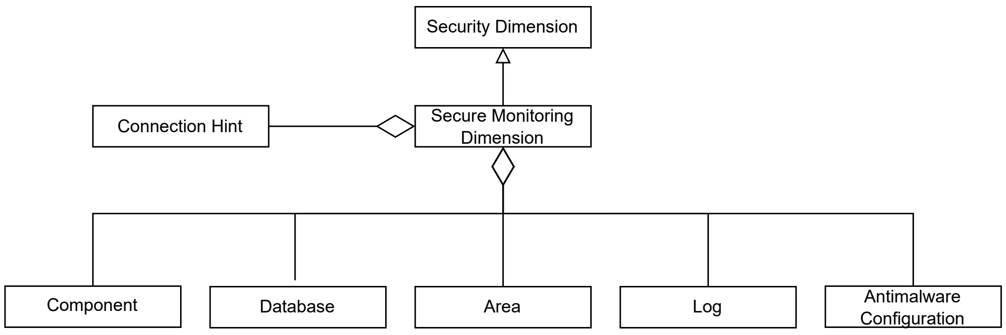


Figure 5.17.: Elements of the monitoring dimension.

Figure 5.17 shows the elements that are allowed to be used for modeling a secure monitoring dimension. In a secure storage dimension the relationship *connection hint* and the elements *component*, *database*, *area*, *log* and *antimaleware configuration* can be used.

The *area* is used to model networks and machines. Networks and machines, as well as the *components*, implement *logs* and *antimaleware configurations.* This aims to make it easy to read which activities are recorded and to discover missing logs. In addition, the *antimalware configuration* can be used to read whether and how the network or the machine is monitored.

6. Realization of FML

The previous chapter describes the process of creating the FML metamodel. During this process, case studies were created for all security dimensions. For this purpose and for the evaluation, a sample realization of FML was developed, which is presented in this chapter.

6.1. Component elements

In this section I describe the realization of the *component elements* presented in figure 5.3. These have been divided into three groups for a better overview: 1. the basic components: database, secret, area and area hint. 2. the authentication and authorization specific *component elements*. 3. the abstract configuration subclasses.

6.1.1. Realization of the basic components

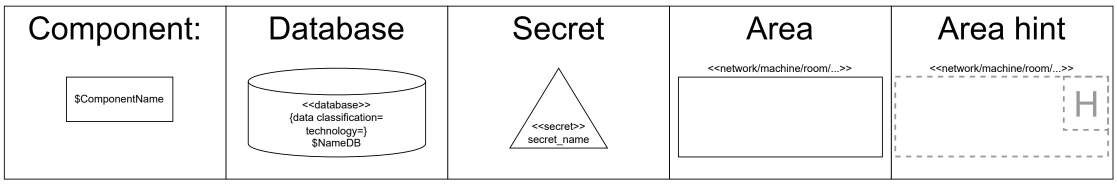


Figure 6.1.: Example realization of the basic components in FML.

Figure 6.1 shows the example realization of the basic components. The *component* is symbolized by a rectangle, which contains the name of the *component*. The rectangular representation of the *component* is based on the representation from other realizations, e.g. often used UML component diagrams.

Databases are represented as cylinders as commonly used in other representations. Additionally the stereotype «database» refers to the type. The database attributes are written in curly brackets inside the cylinder. Furthermore the *database* is named.

The *secret* is represented by a triangular shape. Additionally the name of the *secret* is written in the triangle, as well as the stereotype «secret». Since the *secret* is a FML specific element, the annotation by the stereotype is an important information for understanding the language.

Area and *area hint* are each represented by a box. The stereotype above the box indicates what the *area* or *area hint* represents. This can be e.g. a network or a machine.

Unlike the *area*, the *area hint* is colored in gray instead of black and depicted an H for hint.

6.1.2. Realization of the authentication and authorization components

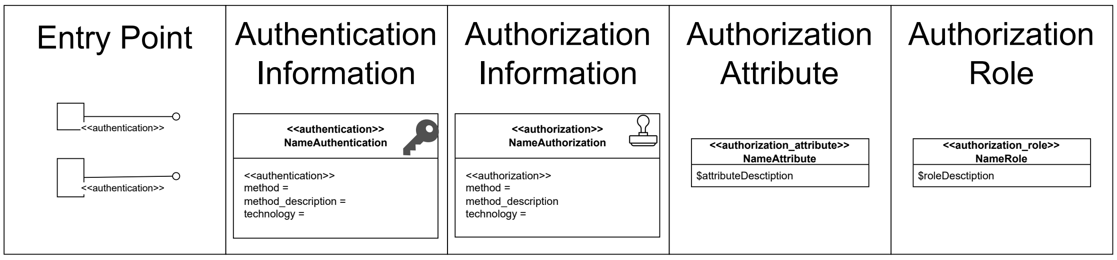


Figure 6.2.: Example realization of the authentication and authorization components in FML.

Figure 6.2 shows the example realization of the authentication and authorization elements. The two *entry points*, *authentication point* and *authorization point*, are represented by a box, which has a connection point. Additionally, the stereotype indicates whether it is an *authentication point* or *authorization point.* Since *authentication point* and *authorization point* are not used in the same dimension, a stronger separation of both elements is not necessary. The design is based on interfaces in representations of UML component diagrams.

The *authentication information* consists of a rectangle divided by a bar. The upper part of the rectangle contains the stereotype «authentication» and the name of the *authentication information.* Additionally the upper part contains a key symbol. This is to distinguish the *authentication information* from the *authorization information.* The *authorization information* looks exactly like the *authentication information*, with the difference that the stereotype “authorization” is used and a stamp is used as recognition symbol. The stamp is supposed to remind of authorizations in the real world, where for example authorized contracts are stamped. In the lower part of the rectangle, *authentication information* and *authorization information* define the technology and authentication, respectively authorization method. The design is reminiscent of that of a class in a UML class diagram. Since the *authentication information* specifies how an *authentication point* is defined, this analogy is intentional.

The *authorization attribute* is also symbolized by a two-part rectangle. Analogous to the *authentication information*, the upper part of the rectangle for the *authorization attribute* consists of the stereotype "authorization attribute" and its name. The lower part contains the description of the *authorization attribute.* The representation of the *authorization role* is analogous to the representation of the *authorization attributes.*

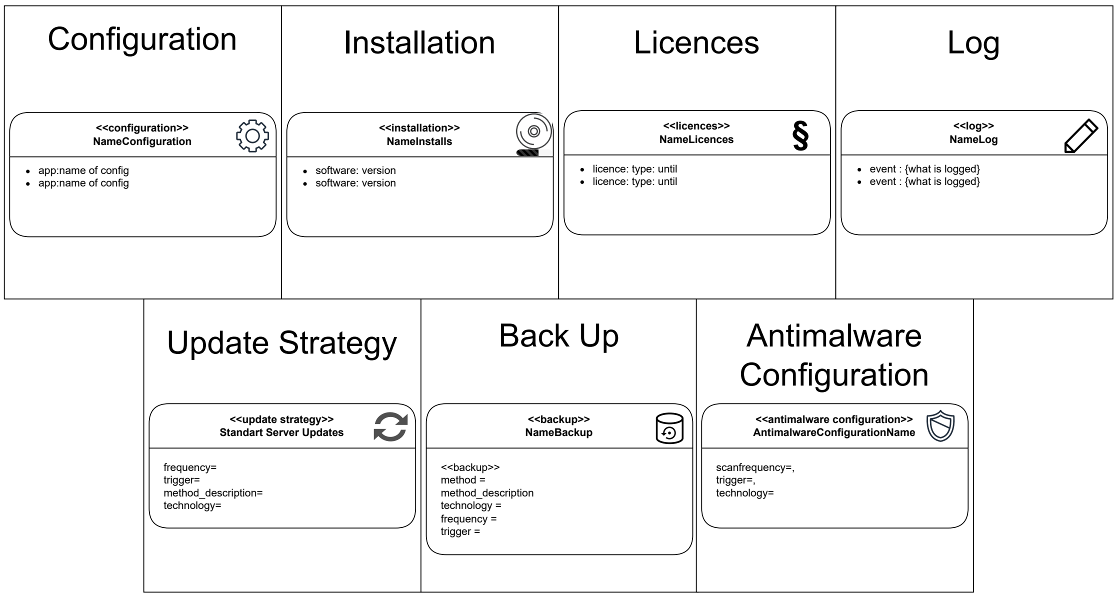


Figure 6.3.: Example realization of the configuration components in FML.

6.1.3. Realization of the configuration components

Figure 6.3 shows the example realization of the configuration elements. Similar to the *authentication information*, the configuration components consist of a rectangle divided into two parts. However, unlike the *authentication information* and the *authorization information*, the rectangle has rounded corners. This difference is intended to clearly distinguish the configuration components from *authentication information* and *authorization information*, since their behavior differ significantly, as defined in section 5.2.3. Analogous to the *authentication information*, the upper part of the rectangle contains the configuration element's type in the form of a stereotype and the name of the element. Additionally there is a symbol for each configuration element. This symbol is introduced to better distinguish the configuration components from each other. The lower part contains the content of the configuration component.

6.2. Policy Elements

Figure 6.4 shows the example realization of the policy elements in FML as defined in section 5.2.1. A *policy* is symbolized by an octagon. The octagon contains the name of the *policy*. In addition, the added stereotype «policy» in the octagon indicates that it is a policy. The *default policy* is defined analogously, but with a double bordered octagon. The *policy description* is symbolized by a document with a bent corner and contains a textual description of the *policy* assigned to it.

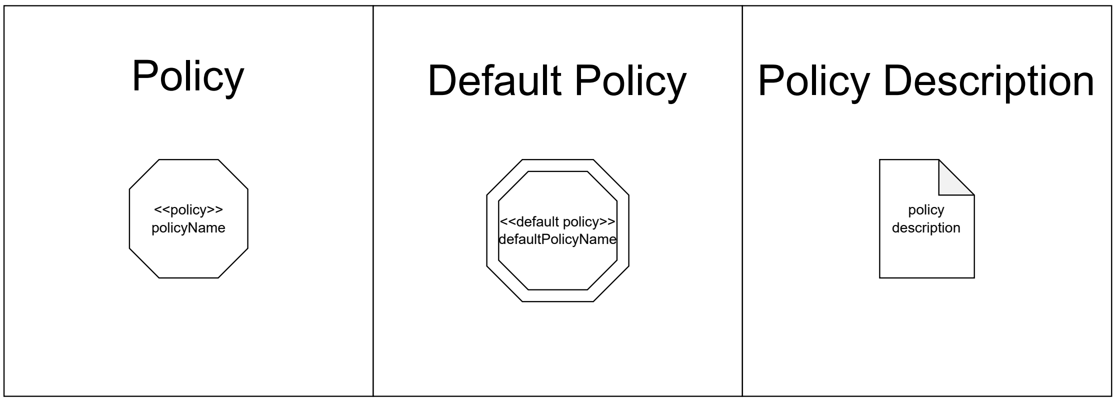


Figure 6.4.: Example realization of the policy elements in FML.

6.3. Ownership

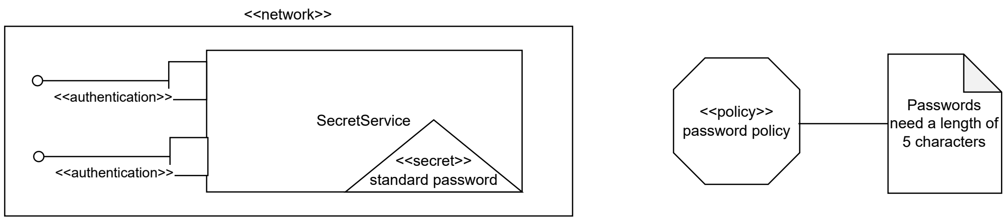


Figure 6.5.: Example realization of the relationships in FML.

Figure 6.1 shows an example how ownership is visualized in the example representation of FML as defined in section 5.2.2. In figure 5.7 it is shown that an *area* can have any number of *components*. This is represented by enclosing a *component* in an *area*. The relationship between a *component* and a *secret* is analogous. If a *component* has a *secret*, the *secret* is enclosed by the *component*.

If *entry points* belong to another element, they are docked to the element. The point of the *entry point* always points away from the docked element.

The ownership relationship between a *policy* and a *policy description* is represented by a black line.

6.4. Behavior

Figure 6.1 shows the example realization of the relations between the elements in FML as defined in section 5.2.1. The relation *upgrade* is represented by an arrow and marked with the stereotype «upgrade>. The arrow points to the element that is being upgraded. The relation *validation* works analogously. Since in FML both relations do not occur at the same place, no further graphical distinction is defined.

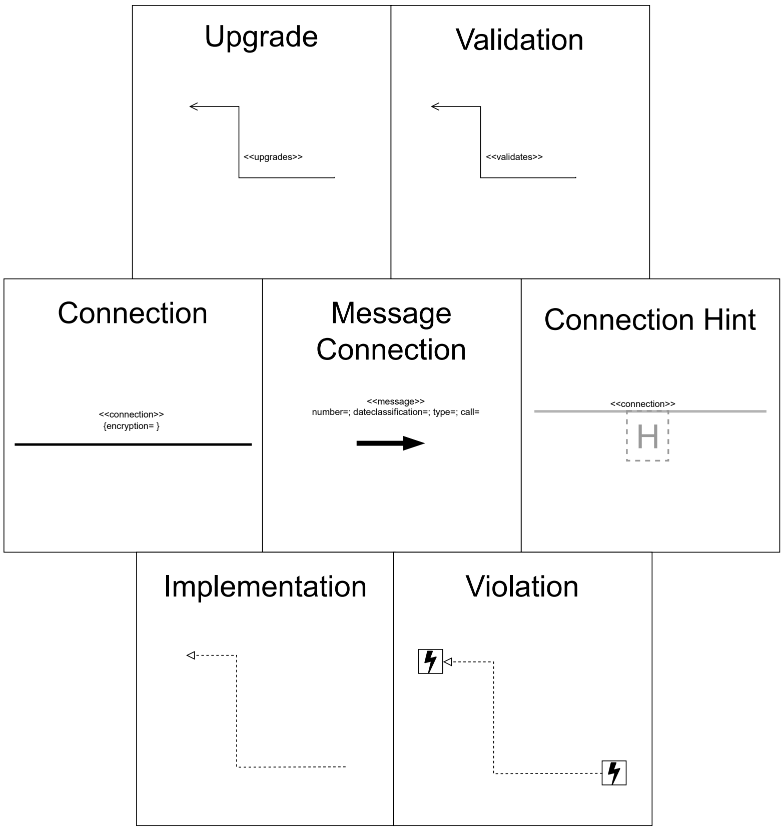


Figure 6.6.: Example realization of the *relationships* in FML.

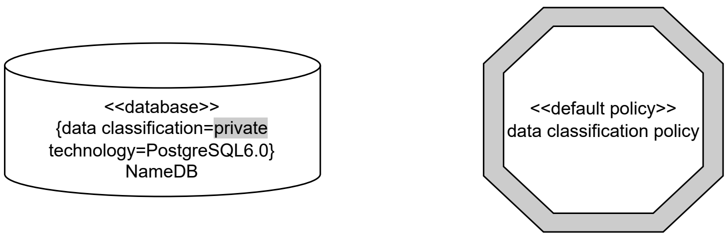

<<database>>
{data classification=private
technology=PostgreSQL6.0}
NameDB
<<default policy>>
data classification policy

Figure 6.7.: Example realization of the *default policy implementation hint* in FML.

The *connection* is undirected and is only defined by a line. In addition the stereotype «connection» is used. Furthermore, an encryption technology can be defined in the form of a tagged value.

A *connection hint* is represented like a *connection*. The difference to the *connection* is the missing encryption information and a gray color instead of a black one. Additionally, the *connection hint* is extended by an H, which stands for “hint”.

The *message connection* consists of an arrow, which indicates the direction of the message. Additionally the stereotype «message» indicates that it is a *message connection*. A *message connection* contains any number of messages, which contain tagged values which assign a number, a data classification, a type and possibly a call to the message.

The *implementation* is represented by an arrow with a dotted line. The direction indicates that the target is implemented. For example, if an *implementation* points from a *component* to a *policy*, this defines that the *policy* applies at that *component*. The *violation* is displayed like an *implementation*, with the difference that rectangles with lightning bolts are added at the beginning and end of the arrow. These represent a negativity, which is supposed to symbolize the *violation*.

Figure 6.7 shows an example of a *default policy implementation hint*. This gives a hint that a *default policy* is implemented by an element, even if *default policies* are implemented by all elements by default. For this purpose the *default policy* is given a color and information concerning the *default policy* is underlined with the corresponding color. In the example figure 6.7 this concerns the classification of the data in the *database*, which originates from the *default policy*.

7. Evaluation

In this chapter, I evaluate the *Fortress Modeling Language (FML)* to see how suitable the language is for modeling security in software systems. For this purpose, I describe the evaluation method in section 7.1. Then I describe the results of the evaluation in section 7.2.

7.1. Method

The semi-structured interview serves as the evaluation method. In this interview style, the interview is guided by predefined questions. However, the interviewer can deepen the answers through specific follow-up questions. This allows both a common thread of the interview, whereby predefined topics are discussed, and a more in-depth conversation about the content. The focus of the interview was also maintained. All interviews were conducted online via videoconferencing and visually supported by a screen presentation. Each interview lasted between 30 and 45 minutes.

7.1.1. Interview structure

The interview consists of four parts: The introduction of the interview, the introduction of the research project, the presentation of a case study, and the outro. The questions asked can be seen in table 7.1. EQ1 and EQ2 are general questions about the participant. The questions EQ3 to EQ7 are asked during the introduction of the case study, which builds up during the interview. The outro is optional and should round off the interview. During the interview, the questions were asked in an open manner, so that the participant has enough space to imagine the modeling on his own projects.

Figure 7.1 shows the case study, presented in the third part of the interview. The case study is an authentication dimension of a fictitious system and is further described in section 7.1.2. The goal of the case study is to represent all elements of the authentication dimension. As the elements and the language are new to the participants, the size and complexity of the case study could confuse the participants. To avoid confusion, the case study is built iteratively, with two or three new elements being added at each step. After each iteration, a question was asked about the newly added elements, which directly relate to a specific part of the FML metamodel.

In the following, I present the course of the interview.

General Questions	
EQ1	What is your current role?
EQ2	Do you have experience in modeling security in software architectures or software architectures in general?
Case study questions	
EQ3	How do you rate the authentication element modeling concept with respect to granularity and scope?
EQ4	Which of the area concepts, area hint, area, no area, do you prefer?
EQ5	How do you rate the policy modeling concept with respect to scope and granularity?
EQ6	Would you add or delete any element of the authentication dimension?
EQ7	Could you imagine using an authentication dimension to evaluate the authentication security of a software system for security testing?

Table 7.1.: Questionaire for the expert interview to evaluate FML

Introduction of the interview (EQ1+2)

The interview starts with the general questions to get an overview of the participant. First questions were asked about the current position of the participant (EQ1). Then the participant was asked about previous experience in the areas of security, software architectures, and security modeling (EQ2).

Introduction to the research project

Afterward, the research project was presented and the background of FML was presented. The presentation of the background is similar in content to chapter 3. The aim of the presentation is to familiarize the participant with the security dimensions. This is important for the classification of the case study, as missing information may be in another security dimension. After that, the participant was able to ask questions about the research project. This should prevent participants from being at different levels of knowledge to ensure comparability.

Evaluation of the authentication elements (EQ3)

At this point, the case study is introduced. The introduction of the case study starts by showing the software component elements of the case study and explaining them. Then the *authentication point* elements were inserted and explained. Afterward, the *authentication information* was added to the case study and also explained. Thus all authentication elements were in the case study. At this point, the participant was allowed to ask questions to make sure the concept was understood. Subsequently, the question “How do you rate the authentication element modeling concept with respect to granularity and scope?” (EQ3) was asked. Question EQ3 mainly evaluates the behavior and ownership of the *authentication point* and *authentication information*, shown in figure 5.7 and figure 5.10. The aim is to evaluate whether the *authentication point* in combination with the *authentication information* provides enough information to evaluate the existence of the authentication process. Additionally, the answer shall be used to derive whether the concept that component elements implement an information element to indicate how they are configured makes sense. This should provide first conclusions about the element *abstract configuration* which works the same way.

Evaluation of the area concept (EQ4)

In the subsequent course of the interview, the *database* element is introduced and shown in the case study. Then the elements *area* and *area hint* are introduced and explained. Now the participant is shown three versions of the case study: 1. Figure 7.1 without the *policies*, where the *area hint* element was removed 2. Figure 7.1 without the *policies*, where the *area hint* element was replaced by an *area* element 3. Figure 7.1 without the *policies*. Afterward, the participant could ask questions about the concepts if something was not understood. Then the question “Which of the area concepts, area hint, area, no area, do you prefer?” (EQ4) was asked.

The question is intended to evaluate the area concept shown in figure 5.7. Thus the question aims at whether the element *area* is meaningful at all in the context of the authentication dimension. For this, the participant was shown the case study without *area* or *area hint*. Furthermore, the whole concept of hints, which was also used for the *connection* in the form of a *connection hint*, shall be evaluated. So it should be shown, how possible users of FML evaluate the mixture of the dimensions. Therefore the question is also intended to show whether the hint concept adds some value to the language.

Evaluation of the policy concept (EQ5)

After the explanation and questions about the area concept, the *policy* and the *policy description* are explained. Further, the *default policy* is explained and the differences between the *policy* and the *default policy* are pointed out. Then the complete case study is shown, in which the policy elements were added (see figure 7.1). After that, the participant can ask questions if the policy modeling concept was not understood. Then the question “How do you rate the policy modeling concept with respect to scope and granularity?” (EQ5) is asked.

The question aims at evaluating whether the policy elements contain enough information to assess to what extent policies are fulfilled and violated in the system. In addition, it should be possible to assess whether the information is sufficient to classify policy violations, i.e. to determine whether, for example, it is a critical violation. Thus the behavior shown in figure 5.9 is evaluated. Furthermore, the distinction between *policy* and *default policy* (see figure 5.5) is to be evaluated.

Evaluation of the full authentication dimension (EQ6+7)

After asking about the policy concept, the participant is informed that the case study is now fully modeled. Then "Would you add or delete any element of the authentication dimension? " (EQ6) is asked. EQ6 refers to the elements of the authentication dimension, modeled in the metamodel in figure 5.12. It is to be evaluated whether the information is missing, which would be necessary for the evaluation of an authentication concept of a system. Also, EQ6 is used to evaluate whether elements are unnecessary for the evaluation of the authentication concept.

Afterward "Could you imagine using an authentication dimension to evaluate the authentication security of a software system for security testing?" (EQ7) is asked. EQ7 aims at evaluating whether the elements in interaction generate an added value for the evaluation of an authentication concept and whether FML is useful for modeling it.

Outro

The interview ends with a loose outro in which another case study is shown, representing a secure storage dimension of a fictitious system. The outro rounds off the interview and is intended to show the participant a further dimension. The goal is to get ahead of the question for an example of a further dimension by the participant. This is also intended to maintain interest in the research project.

7.1.2. Case study

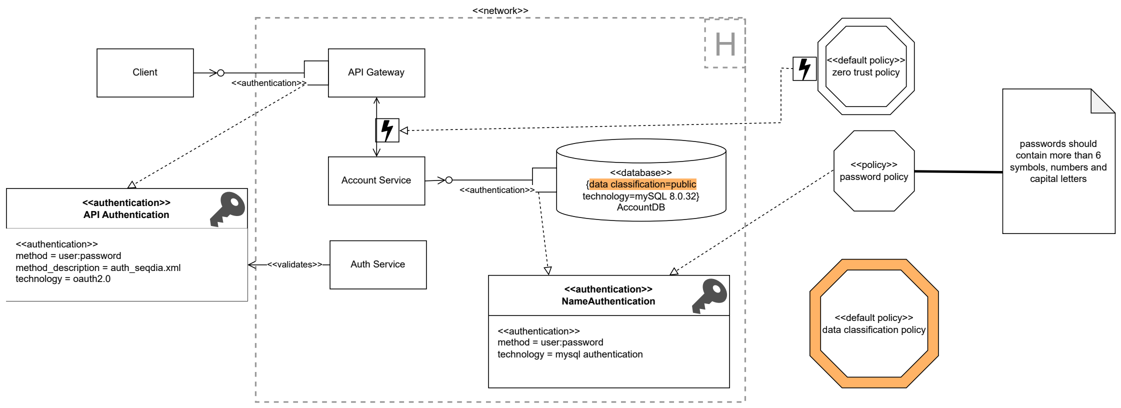


Figure 7.1.: Case study for the evaluation of FML

In this section, I describe the structure of the case study used for the interview. The case study, shown in figure 7.1, displays a fictitious system mapped to an authentication dimension modeled in FML. The requirements for the system are that all elements of the authentication dimension are represented. The authentication dimension was chosen because it contains many concepts of FML. Thus, by evaluating the authentication dimension, conclusions can be drawn about other dimensions and FML.

In terms of content, the case study sees a client, an API gateway, an account service, and an auth service. An API gateway is a tool that sits between a client and the backend services. It receives all API calls and aggregates the different required services and returns the corresponding result. The account service is connected to a database. The account service must authenticate itself to the database with a user and password. The data contained in the database is classified as public and the database software is MySQL 8.0.032 and therefore on the current version at the time of this thesis. Besides the connection between the account service and the database, two other connections around the system have been modeled: There is a connection between the client and the API gateway. In this connection, the client has to authenticate itself to the API gateway. The Auth Service validates this authentication. In the communication between the API gateway and account service, there is no authentication.

In addition, three policies were modeled: 1. a zero trust policy, which specifies that every communication, inside and outside every network, must be authenticated. This is modeled as a default policy. Therefore, only the violation in the form of the communication between the API gateway and account service occurs, since no authentication takes place there. 2. a password policy. This has a policy description, which describes the required password strength. The policy is modeled as a normal policy and is only fulfilled by the database authentication. Therefore, the authentication between the client and API gateway violates the required password strength. 3. the data classification policy, which classifies data. This was modeled as default policy and no violations were modeled. However, a hint was modeled in the account service database that attributes the origin of the data classification to the data classification policy.

In the case study, the zero trust policy and the data classification policy were modeled without a policy description. This is since the interviews were conducted via a screen presentation and the policy descriptions would have greatly increased the size of the model due to their volume, which would have resulted in significantly lower readability.

The selected example does not claim to be a high-quality architecture but was chosen for the purpose of evaluation to be able to model and present the developed concepts as meaningfully and comprehensibly as possible, taking care to consider realistic use cases.

7.2. Results

In this section, I describe the results of the expert interview described in the previous section. In the following, I will go through the individual questions that were asked in the expert interview and summarize the participants' answers. An overview of all questions is shown in table 7.1. The shortened and translated transcriptions of the expert interview are displayed in appendix B. In the following, I will first describe the general questions, which were asked to get an overview of the participants. Then the case study questions are asked, which refer to the case study described in section 7.1.2.

7.2.1. General questions

In this section, I present the participants' answers to the general questions from the expert interview. This includes an explanation of their current position and their experience in the field of security modeling or software architecture as a whole.

EQ1 *What is your current role?*

A total of six people from industry and science took part in the expert interview. The participants from the industry are P1, P2, P3, and P6. The industry participants P2, P3, and P6 are software architects, while P1 is a cyber security consultant. From the academic environment, two scientists from the field of cyber security, P4, and P5, participated. P4 is a private lecturer at a German university and P5 is a Ph.D. student.

EQ2 *Do you have experience in modeling security in software architectures or software architectures in general?*

Participant P1 claims to have two years of experience in designing and implementing security systems for customers. Participants P2 and P3 state a high level of experience as software architects with 20 and 13 years respectively. Additionally, P2 states that he has been reengineering the software of a German concern with a focus on security for four years of the 20 years. Participant P6 states that he has had experience in the area of software architecture for three years, with a focus on the area of infrastructure. Participants P4 and P5, who come from an academic environment, report 15 and three and a half years of experience in security research, respectively.

7.2.2. Case study questions

In this section, the case study questions are asked, which refer to the case study described in section 7.1.2.

EQ3 *How do you rate the authentication element modeling concept with respect to granularity and scope?*

For this question, the case study from section 7.1.2 was presented, in which only the *components*, *authentication points* and *authentication information* was shown. Of the six participants, five said that the elements *component*, *authentication point* and *authentication information* are sufficient to evaluate an authentication process and rated the scope and depth of information of the authentication modeling concept as sufficient. Only P3 did not make a clear statement in this direction.

However, the participants made additional comments about missing information. For example, three of the five participants from the industry said that more information about the authentication process was needed. In addition, two of the four participants from the industry wanted more information about the communication methods used for authentication. P4 and P6 noted that the configuration of the authentication method or software is missing. According to P4, this is a reason for many security breaches. P6 noted that the version number of the software responsible for authentication would be missing.

EQ4 *Which of the area concepts, area hint, area, no area, do you prefer?*

For this question, the case study from section 7.1.2 was shown three times, in which only the components, authentication points, authentication information, and databases were faded in. The first time no area or area hint was shown, the second time the *area hint* was replaced by an *area* and shown, and the third time the *area hint* element was shown. The *area* or *area hint* provided the information that a network exists in the case study. Afterward, the participants were asked about their preferred version.

None of the participants stated that they preferred the variant without an *area.* All participants said that information about the existence of a network in the authentication dimension is useful.

The participants did not agree on whether the *area hint* was useful. Two participants, P1 and P2, stated that they would rather use only *area.* Both participants found the concept of a hint that borrows information from other dimensions confusing and difficult to understand. P2 would say that in practice it could lead to misusing only *area* or *area hint* in all dimensions.

Four of the experts, two from industry and two from research stated that they preferred the *area hint.* However, the importance of separating the dimensions of P4 was emphasized. The separation is possibly lost by the overlapping of the dimensions by hints. Otherwise, P3 noted that the concept of hints might be too complex if it only occurred within *areas.* Also, P3 noted that dimensions could become too large. Finally, P1 noted that with the term network, it was not quite clear whether it was a logical or physical network.

EQ5 *How do you rate the policy modeling concept with respect to scope and granularity?*

For this question, the case study from section 7.1.2 was presented completely with all elements. Therefore the participants were able to see three policies: one *policy* and two *default policies.* Further, one of the *default policies* is violated, one *default policy* has an implementation hint and the *policy* has one implementation as well as a *policy description.*

Of the six participants, P1, P3, and P6 said directly, that the depth and scope of the information was sufficient. P1 and P2 said that the description of the policy in the form of the *policy description* would be useful. Further, P4 suggested replacing the policy description with a link to a broader description. Four of the six participants, P1, P2, P5, and P6, expressed positive views about modeling policy violations.

P1 stated that the concept would be suitable for modeling common policies in a system. According to P2, the concept needs the possibility to model positive deviations. For example, it is only possible to model at which point a password policy is followed and not at which point it is exceeded. P2 also suggests checking whether too many policies need to be modeled on one dimension, which could lead to a lack of clarity, in future work. P6 notes that it is unclear where *policies* should be used and where *default policies* should be used. In addition, P6 notes that *default policies* complicate changes in the system because the documentation has to be revised each time.

EQ6 *Would you add or delete any element of the authentication dimension?*

For this question, the case study from section 7.1.2 was still presented completely with all elements. With this question, the participants were asked which elements they would miss, or which elements might be unnecessary for an *authentication dimension*.

P1 suggests that classifications could be modeled separately from *policies*, as a separate element. Additionally, P1 suggests combining the *policy description* with the *policy* or the *default policy* to one element. However, P2 wants information about the transport protocols used in the system. P3 suggests that grouping elements could be added. These could group different elements together and thus reduce the number of modeled relationships. P5 suggests an asset value dimension, which models the value of assets. Subsequently, asset hints could be added to the *authentication dimension*.

EQ7 *Could you imagine using an authentication dimension to evaluate the authentication security of a software system for security testing?*

The final question was to find out whether the participants could imagine using FML. Five of the six participants said that they could imagine using it. Only P6 considered the language useful at first sight, but the abstraction level would be too detailed for the current use in the company.

8. Discussion

In this chapter, I discuss and evaluate FML based on the expert interview. For this purpose, I first discuss in section 8.1 the results of the expert interview which was described in chapter 7. Subsequently, I elaborate on the implications of these discussion results for the research questions of this thesis described in chapter 1 in section 8.2.

8.1. Discussion of the evaluation results

The goal of the evaluation was to evaluate the metamodel of FML. For the discussion of the evaluation results in this section, I revisit the interview questions about the case study (EQ3 to EQ7) and analyze the participants' responses.

EQ3 *How do you rate the authentication element modeling concept with respect to granularity and scope?*

Question EQ3 should mainly evaluate the behavior and ownership of the *authentication point* and *authentication information*, shown in figure 5.7 and figure 5.7. The aim is to evaluate whether the *authentication point* in combination with the *authentication information* provides enough information to evaluate the existence of the authentication process. Additionally, the answer shall be used to derive whether the concept, that component elements implement an information element to indicate how they are configured, makes sense. This should provide first conclusions about the element *abstract configuration* which works the same way.

Overall, the authentication concept seems to have been well received, as five out of six experts were overall satisfied with the scope and depth of information. Thus, the *authentication information* and the *authentication point* in combination seem to be suitable for a rough evaluation of an authentication process.

However, the participants made additional comments about information that could be included in the *authentication dimension.* For example, three of the five participants from the industry said that more information about the authentication process was needed. This affects the *authentication information.* In FML, the authentication process is only named and referred to other documents. In the case study presented, this was done with a link to a sequence diagram, which was not included in the case study. It was therefore unclear to the participants whether the requested information could have been found in this sequence diagram. The reason for this is that a new diagram might have taken too much time to explain, so it was not included in the case study. It may be that further evaluation of FML requires the inclusion of the linked material.

In addition, two of the four industry participants stated that they would like to have information about the communication protocols used for authenticated communications in the software system. This information would be assigned to the *secure communication dimension* in FML. It may be considered whether the information would be useful as a *communication hint* in the *authentication dimension*. For this purpose, a further study would have to be conducted to find out whether the participants, having seen a *secure communication dimension* and a corresponding *authentication dimension*, consider the use of a *communication hint* to be useful.

Additionally, two of the participants noted that the configuration of the *components* involved in the authentication is relevant for the evaluation of the authentication. The element *configuration* would be suitable for this purpose. It might make sense to add the *configuration* to the *authentication dimension* for this purpose.

Another criticism was that the version number of the *components* that are responsible for performing the authentication was missing. I fully agree with this point of criticism. Since version numbers are also information from the need analysis (see chapter 4), the information should be added in the next version of FML in any case.

Overall, there was no criticism of the concept that an element containing information, such as the *authentication information*, defines or configures a *component element*. Accordingly, I would pursue this concept in future versions of FML.

EQ4 *Which of the area concepts, area hint, area, no area, do you prefer?*

The question EQ4 is intended to evaluate the area concept shown in figure 5.7. Thus the question aims at whether the element *area* is meaningful at all in the context of the *authentication dimension*. For this, the participant was shown the case study in three versions: 1. Without *area* or *area hint*. 2. With *area* 3. With *area hint*. In the case study, the *area* and *area hint* display a network. By evaluating the *area hint*, the whole concept of hints, which was also used for the *connection* in the form of a *connection hint*, shall be evaluated. It shall be evaluated, how potential users of FML evaluate the mixture of the dimensions. Therefore, the question is also intended to show whether the hint concept is too confusing.

Since none of the participants in the expert interview preferred the variant shown without the network information and each of the experts stated that the network information is useful, an *area* or *area hint* would in principle be seen as useful in the authentication dimension.

However, two of the participants from industry found the concept of *area hints* confusing and would only use *areas*. This criticism did not come from the research participants at all. It could be that the principle of a sharp separation between logical and infrastructural information is kept rather strict in security research, while it is not so important in the industry. Thus, research participants may perceive the concept of intentional violation of the separation between the logical and infrastructural viewpoints to be useful.

It is also noticeable that the first two participants found the *area hint* concept confusing. Therefore it could also be possible that the explanations of the concept got better in later interviews, even though the same slides and script were used. Additionally, P3 commented that the hint concept is complicated, especially when it is only present in the form of an *area*. It could be considered to revise the hint concept so that it can be used consistently for all elements. However, P3 notes that the dimensions may become too large if hints are used too often. This indicates, that clear rules have to be found, when a hint is allowed and when not.

In summary, I would not make a statement about whether an area or an area hint should be used in the *authentication dimension*. Overall, the hint concept should be redefined, making the borrowing of information by using hints clearer. Possibly this would lead to greater acceptance of the *area hint*.

EQ5 *How do you rate the policy modeling concept with respect to scope and granularity?*

The question aims at evaluating whether the policy elements contain enough information to assess to what extent policies are fulfilled and violated in the system. In addition, it should be possible to assess whether the information is sufficient to classify policy violations, i.e. to determine whether, for example, it is a critical violation. Thus the behavior shown in figure 5.9 is evaluated. Furthermore, I want to evaluate the distinction between *policy* and *default policy* (see figure 5.5).

Overall, participants were rather positive about the modeling of policies in FML. Participants P1, P3, and P6 clearly stated that the scope and depth of information were sufficient. The fact that the violations were well received by four of the six participants and that the usefulness of the *policy description* was also emphasized by three of the six participants shows that the policy modeling concept was overall well received.

However, there were also suggestions for improvement. For example, according to p2 policy modeling lacked a way to model when policies are not only adhered to but also exceeded. This could be achieved by changing the relationship *implementation*, but the use of more complex relationships could affect the clarity of the security dimensions. In general, the idea could be considered for the next version.

In addition, P6 noted that *default policies* mean that if changes are made to the system, the documentation may have to be completely revised. However, this could also happen with *policies*, as these would also have to be revisited. Additionally, P6 was unclear when *policies* and when *default policies* are used. The idea of FML is that this should be the decision of the user. However, it might make sense to define rules, as to when to use which type of policy. These rules could then also depend on the severity of not following the policy. This should possibly be considered in the next version of FML.

In summary, the policy modeling concept was received rather positively. I would suggest using it as a starting point for further versions of FML and extending it.

EQ6 *Would you add or delete any element of the authentication dimension?*

EQ6 refers to the elements of the *authentication dimension*, modeled in the metamodel in figure 5.12. It is to be evaluated whether the information is missing, which would be necessary for the evaluation and evaluation of an authentication concept of a system. Also, EQ6 is used to evaluate whether elements are not necessary for the evaluation of the authentication concept.

Overall, no participant in the study suggested that any element should be removed. However, there were ideas for elements that could be added.

For example, participant P1 suggested adding an element for classifications to the *authentication dimension*. This could then replace *policies* that classify. The idea sounds quite reasonable and should be investigated in future versions.

Apart from that, according to p2 the authentication dimension lacked information about the transport protocols used in the communication within the system. This was already discussed in EQ3.

The suggestion of P3, that elements could be grouped to form classes by using abstract elements, seems reasonable to me. This could be well realized with the relationship *upgrade* and should be considered in the next version of FML.

Also, the suggestion of P5 that asset value hints could be added to the *authentication dimension* should be considered. These hints could refer to the *secure storage dimension*.

EQ7 *Could you imagine using an authentication dimension to evaluate the authentication security of a software system for security testing?*

EQ7 aims at evaluating whether the elements in interaction generate an added value for the evaluation of an authentication concept and whether FML is useful for modeling it.

It was found that five of the six participants could imagine using FML to model an *authentication dimension* to evaluate the authentication security of a software system. Only one participant did not make a direct statement but indicated that the language was currently too detailed for the current application in his company.

Overall, it can be concluded that the idea of a modeling language such as FML is meeting with interest in industry and research. Due to the broad population of interview participants from both research and industry, with different backgrounds and different levels of experience, the value of FML and the research direction of security modeling is thus supported.

8.2. Discussion in the context of the research questions

In this section, I evaluate *Fortress Modeling Language (FML)* with regard to the research questions, defined in chapter 1. Since the two research questions are closely intertwined, I analyze them together and evaluate whether FML can be seen as being an answer to both.

RQ1 *How can a metamodel for modeling security dimensions of software architecture views look like?*

RQ2 *Which modeling elements can be used to model an Authentication, Authorization, Secure Communication, Secure Storage, Secure Build and Deployment, and Monitoring Dimension as defined by Sinkovec?*

The goal of this thesis is to create a modeling language that can be used to model the security dimensions. For this purpose, the FML was created in this work and evaluated in an expert interview. The experts did not directly evaluate the metamodel and its suitability for modeling security in system architectures. For reasons of comprehensibility, the modeling elements were evaluated by using a case study modeled with the representation, described in chapter 6. It was evaluated whether the information scope and the information depth of the representation of selected core concepts of FML are sufficient. For this, a case study was created which explained the modeling elements to the experts. This process was based on the assumption that if the metamodel is rather insufficient to model the security of a software system, the representation is also considered insufficient. This assumption is called transfer assumption in the following. The transfer assumption allows to conclude the evaluation to the metamodel. To evaluate RQ1 and RQ2, I analyze the responses of the expert interview participants regarding the metamodel and with focussing on the modeling elements.

Overall, the evaluation of the authentication dimension was rather positive, the authentication concept was well received, and the experts found the scope and depth of information to be sufficient. However, some participants suggested additional information, such as details about the authentication process. These were isolated comments for missing information, but there was no conceptual criticism of the language. FML thus provides a first draft for a metamodel to model the authentication dimension, based on the transfer assumption. The metamodel of FML therefore offers sufficient possibilities to model the authentication. In addition, conclusions can be drawn about other dimensions. Other dimensions take over concepts of the authorization dimension. For example, the secure communication dimension, which uses the authentication concept for authentication related to databases. In addition, other concepts, such as the authorization concept, are based on the authentication concept in order to create uniformity within FML. Accordingly, initial conclusions can also be drawn about the authorization dimension. For example, the combination of *entry point* and *abstract information* for describing access to a software component was well received by the experts. Cross-dimensional concepts such as the policy concept were also well received by the experts. This also indicates that the metamodel was a good first draft. However, the metamodel of FML cannot be seen as complete because of the limitations of the evaluation. For this, more extensive studies are needed, both for the authentication dimension and for the other security dimensions.

Further, the elements of FML can serve as a basis to model the security dimensions, and further enhancements and additions can be made based on the feedback received during the further evaluation process. The modeling elements have proven to be suitable in the evaluation. It was possible to model the case study, with the existing elements. In addition, there was no major criticism of the modeling elements or comments that certain scenarios or edge cases could not be modeled. However these edge cases and scenarios which cannot be modeled in FML cannot be ruled out completely due to the limited scope of the evaluation Rough information gaps by the elements of FML would have probably been noticed during the evaluation. This suggests that it makes sense to let further versions of FML build on these elements.

However, due to the lack of a large scale study, it cannot be guaranteed that the elements of FML are suitable for modeling the entirety of security dimensions. For the other security dimensions, such as authorization, secure communication, secure storage, secure build and deployment, and monitoring, it would be necessary to conduct further research and evaluations to identify possibly missing modeling elements required for these dimensions. The expert interview focused on the authentication dimension, and although some participants mentioned elements related to other dimensions, a detailed evaluation of those dimensions was not conducted. Therefore, additional studies and expert opinions would be needed for each security dimension.

Additionally, there are some limitations of the evaluation to be considered. Due to the shortage of experts in the field of security modeling and the time constraints of this thesis, the evaluation has to be interpreted taking into account of the following limitations:

- Only six participants took part in the expert interview. Thus, the study cannot be considered large scale. Accordingly, the results should be seen as a rough overview rather than a complete evaluation of FML.
- The expert interview covers only the authentication dimension. Although the authentication dimension was chosen because it contains most of the basic concepts of FML, the other dimensions were not evaluated. Thus, some elements of FML have not been evaluated by experts.
- The experts did not evaluate the metamodel directly, but with the help of a case study. Although this was intended to illustrate the concepts of FML, there are no direct statements about the metamodel.
- Since the case study had to be comprehensible within a short period of time, it was limited in scope. This meant that it was not possible to model a large real system. Therefore, FML was only evaluated within a predefined context.

In conclusion, the evaluation of the metamodel and modeling elements in FML for modeling security dimensions provided valuable insights. The assessment of the authentication dimension was positive, indicating that FML offers a promising first draft for modeling authentication and potentially other dimensions. However, the evaluation had limitations, including a small number of participants, focus on authentication, and indirect evaluation through a case study. Further studies are necessary to validate and enhance FML for all security dimensions, addressing potential missing elements. Nevertheless, FML serves as a foundation for modeling security dimensions, can be a base for future improvements.

9. Related Work

In this chapter, I will describe related work for already existing security modeling approaches in software architectures. I first describe the *Unified Modeling Language (UML)*, which is already suitable as a modeling language for software system security without extensions. Then I describe the two extensions of UML, UMLSec and SecureUML. After that I describe CySeMol, a language specialized for the analysis of enterprise systems. This is followed by short descriptions of other relevant languages, which are suitable for security modeling. Finally I distinguish FML from the previously described languages.

9.1. UML

The *Unified Modeling Language (UML)* is a modeling language for software engineering to design and visualize software systems from different perspectives. UML is defacto the industry standard and many developers are familiar with the language [MiD10; RREAT17]. The language does not provide a specific diagram type for security. But it is possible to extend UML with built-in extension mechanisms [Obj17]. These so-called lightweight extensions are realized with stereotypes, profiles, constraints, and model libraries. Further, extensions can also be realized by adding new concepts and relationships to the UML meta-model [Obj17; Fak].

An example for the use of UML without extensions for the modeling of security of software systems is the approach of Pavlich-Mariscal, Michel, and Demurjian [PMMD07]. In this approach, constraints related security information is added to UML. This allows to use UML already without further extensions for modeling the security of a system.

9.2. UMLsec

The first language I describe is UMLSec, which is a lightweight extension of UML. UMLsec focuses on secure-critical distributed systems and provides the possibility to evaluate the security aspects of a system design by referring to a formal semantics of a simplified fragment of UML [Jü02]. UMLsec extends the UML meta-model and therefore does not provide its one. It uses the extension mechanisms stereotypes, tagged values, and constraints to extend the semantics of existing types in UML [Jü02]. Constraints are used to specify the security requirements. These constraints are written in a not further specified language [MiD10]. UMLsec supports the whole set of UML diagrams. With the use of UMLsec, the system security can be assessed by performing a formal analysis. A UMLsec model can contain confidentiality, integrity, and authenticity criteria which are realized with solution stereotypes. This could be for example fair exchange, Role-based

Access Control, authenticity, or secure information flow definitions. UMLsec further supports standard risk management. [MiD10]

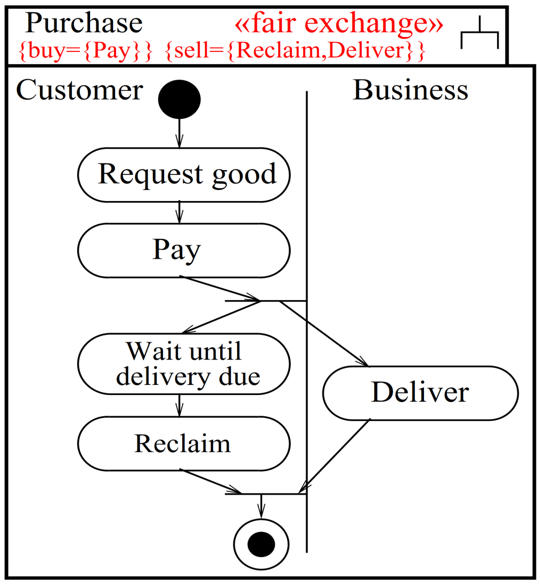


Figure 9.1.: Example of an UMLSec model [Jü02]

The example figure 9.1 shows the modeling of a purchase transaction in UMLSec. The fair exchange stereotype shows that the customer can either complete the purchase or get his money back. The keywords buy and sell define that each payment must lead to either a reclaim or a delivery.

9.3. SecureUML

Another UML extension for security modeling is SecureUML. Like UMLSec it uses the UML extension possibilities stereotypes, tagged values, and constraints to extend the UML [LBD02]. In comparison to UMLsec, SecureUML has its meta-model which is defined on the Role-based Access Control Model (RBAC) [LBD02].

Further, in comparison to UMLsec, SecureUML extends mainly the UML class diagrams and not the complete sets of UML diagrams. They further differ in the modeling target. [MiD10] SecureUML is designed to model solutions with the use of RBAC, as defined in chapter 3, and not to model different security criteria. [MiD10]

The example in figure 9.2 shows a modeling of a calendar application in SecureUML. The example consists of two classes, Calendar and Entry. A calendar contains several entities, which simulate appointments. These have different attributes like location, start and end date, as well as an owner.

The secuml.constraint element to the right of the Entry element indicates that access to an Entry is restricted to working days. Additionally, the sec.uml.permission to the left

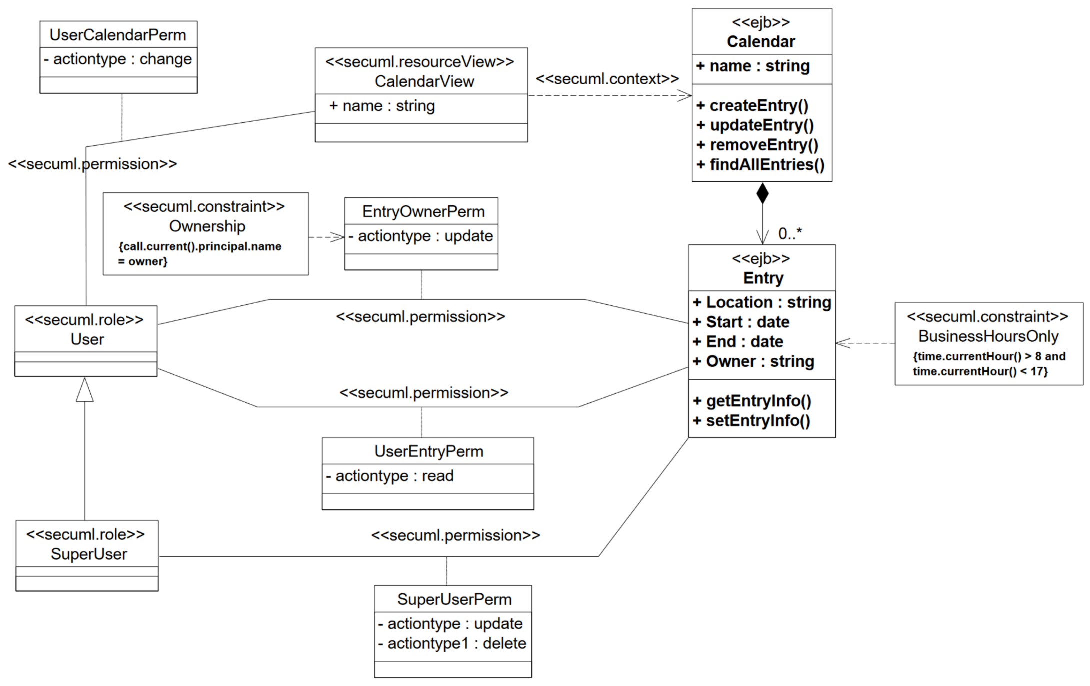


Figure 9.2.: Example of an UMLSec model [LBD02]

of the Entry element indicates that only the owner of an entry can update the element, but all users can see it.

9.4. CySeMoL

CySeMoL is a modeling language, which is made to analyze the security of enterprise system architectures [SEH13]. A CySeMoL model contains a probabilistic relational model (PRM) which is used to estimate the securitz of a system. CySeMoL is defined with its own meta-model. The PRM specifies how a Bayesian Network should be constructed from an object model like a UML object model [SEH13]. A possible result can be seen in figure 9.3. A CySeMoL model can be generated in two steps: First, the qualitative structure is created. This structure contains the assets, attacks, and defenses and how they are associated. In the second step, quantitative data is added with the information on how likely different attacks succeed. This is done with the information about the presence or absence of different defenses [SEH13]. CySeMoL can analyze 22 assets, 102 attacks and defenses, and 32 asset relationships.

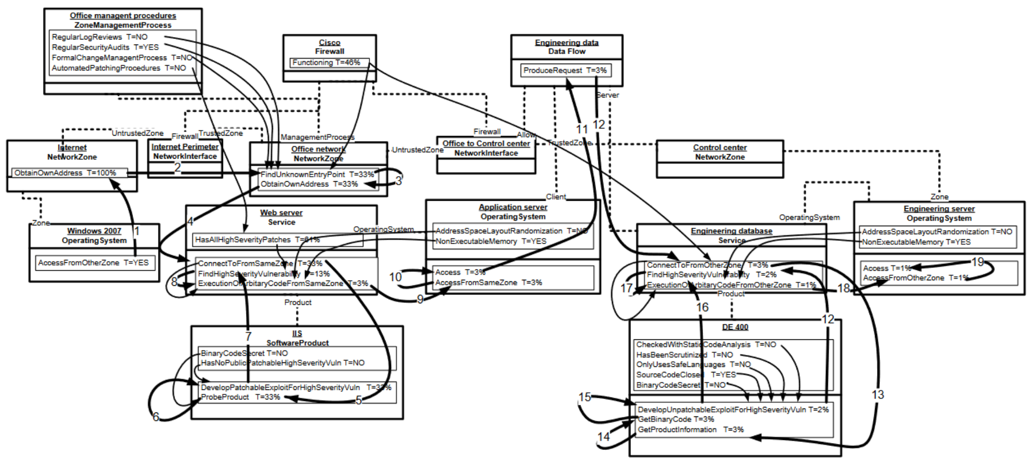


Figure 9.3.: Example of a CySeMoL model with a 19-step attack path. The numbers display the steps on the attack graph. The probability that each step on the path is reached is given with probability T [SEH13]

9.5. Further modeling languages

Next to the already presented approaches exist some more mentionable approaches. IoTSec is a UML extension for security modeling which specializes in the Internet of things [RREAT17]. The extension is made with stereotypes and notation extensions. The extension aims to be used in the design stage and claims that the user does not need to be familiar with cybersecurity concepts completely.

Another approach is SecREAM, which is described to crater different system types like cloud-based or real-time [GGS15]. SecREAM was created with a focus to improve the communication between the different stakeholders of the system. Further, SecREAM is described as precise enough to be used to advise management decisions.

Rodríguez, Fernández-Medina, and Piattini approach focus on the security of business processes [RFMP06]. For this purpose, they extended UML activity diagrams with stereotypes and tagged values. The approach allows us to evaluate a system in a very early design stage.

9.6. Differentiation to the Fortress Modeling Language

The difference which exists between the *Fortress Modeling Language (FML)* and the other presented approaches is the connection of the security dimensions. FML is strongly connected to the security dimensions to model the different aspects of security. UMLSec provides a security extension to already existing UML diagrams. This allows the usage of UMLSec in behavior diagrams. The actual focus of FML is the use in structural diagrams. Thereby FML gives a specification, which contents are modeled in which

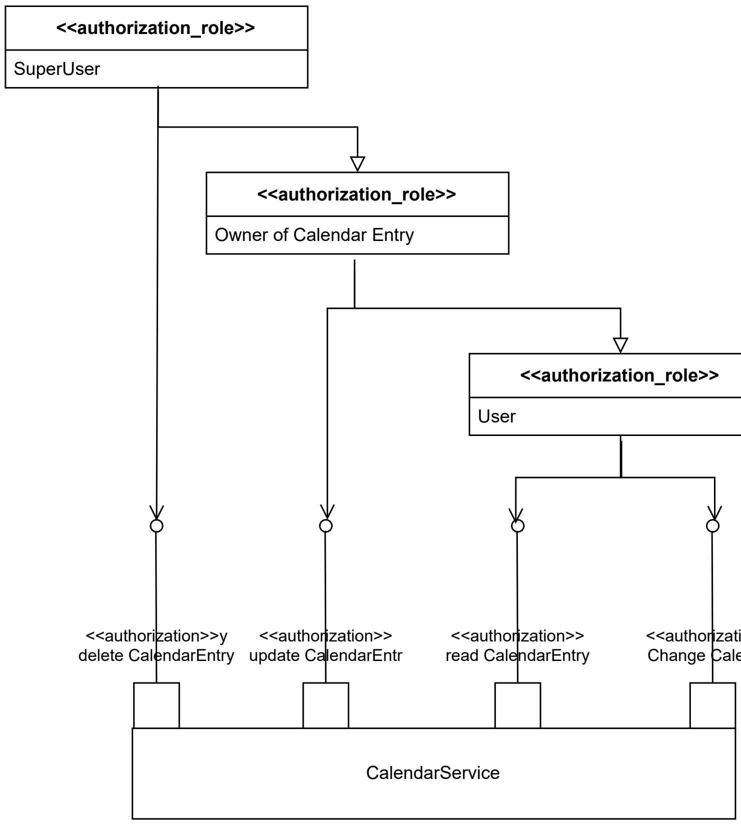


Figure 9.4.: Example of figure 9.2 remodeled in FML as an authentication dimension.

dimension. UMLSec and SecureUML do not provide such a strict specification [MiD10].

To better compare the differences between FML and SecureUML, figure 9.2 is remodeled in figure 9.4. Figure 9.4 shows an authorization dimension that displays a comparable system with the same functionality as the example from figure 9.2. The direct comparison shows how the focus of FML is on the entire system. Therefore, individual classes of a software are not modeled as in SecureUML, but system components, such as a service. SecureUML is more formal and clearly more precise with respect to the functions of the individual software components. FML, on the other hand, shows an overview of which authorization is required for which system components. Additionally, it would be possible to model, where the validation of the authorization is done in FML.

The scope of CySeMol is to model enterprise architectures [SEH13]. FML has the focus on system architectures. Further CySeMol is analyzed automatically based on common attacks. FML has the focus to model common defenses. Attacks are not in the scope of FML but are modeled in CySeMol. Also, the analysis of the security based on an FML model is made by hand.

FML differs from IoTSec in the sense that it does not specialize in one application. In addition, FML is aimed at security software testers, who need to have knowledge in the field of security. SecREAM differs from FML in the sense that SecREAM models are created in the early stages of development, while FML models are based on fully developed systems in the late stage of testing. In an earlier stage of development are also models of the language of Rodríguez, Fernández-Medina, and Piattini. In addition, this language is concerned with securing the processes, while FML is concerned with the system.

10. Conclusion

This exploratory work aims to present a first draft of a language to model the security of software systems. In future work, it is proposed to further refine this draft and to complete it to a comprehensive modeling language for software system security. In this chapter, a brief summary of the research method and contribution of this thesis is given in section 10.1. Through the knowledge gained during this work, many opportunities for further development and research have been revealed. Therefore, a brief outlook on the directions in which future work can be approached to extend the contribution of this thesis is provided in section 10.2.

10.1. Summary

This thesis has addressed an approach to meet the increasing threat to the security of software systems in a digitized world. Given our heavy dependence on software in various domains, it is crucial that software developers take measures to avoid security vulnerabilities and strengthen resilience against potential attacks. Existing approaches such as threat modeling mainly focus on identifying vulnerabilities from an attacker's perspective. Therefore, there is a research gap in considering security from a defender's perspective during the software design process. By introducing a defender's point of view and considering design principles, software developers can actively improve the security of software systems and proactively defend against cyber threats.

In his thesis, Sinkovec introduced the concept of his "security views". This concept divides the security of system architecture into different "security views", which together should form a complete picture of the system security from a defender's point of view[Sin22].

In this thesis, the concept of "security views" is extended under the name security dimensions by creating the *Fortress Modeling Language (FML)* as a modeling language with the goal to model the security of system architectures by dividing them into the security dimensions authentication dimension, authorization dimension, secure communication dimension, secure storage dimension, secure build dimension and monitoring dimension, described in chapter 3.

Since the security dimension has not yet been specifically defined for a stakeholder, the creation of FML starts by adding a stakeholder, which is the security software tester in this work. The decision was made because there exists good documentation about the tasks and needs of the security software tester. Therefore, the security software tester's requirements for a security software tester view are extracted from a document from the security software tester training. The security dimensions are then supplemented by the security software tester's requirements to finally be underlying information of FML. (chapter 4).

Based on this information, the FML metamodel is created in an iterative process, described in chapter 5. In addition, an example representation of FML is created and described in chapter 6 to be used in the evaluation.

In the evaluation, a subsequent semi-structured expert interview, in which experts from research and industry are presented with an authentication dimension in the form of a case study, the suitability of FML for modeling the security of a software system is evaluated (chapter 7).

The subsequent discussion of the results of the expert interview, chapter 8, shows that FML is suitable as a basis for a language for modeling security, but cannot yet be considered complete.

The thesis shows with FML an effective method to model security aspects in software systems. The experts interviewed in this study responded positively to the concepts behind FML and showed great interest in its application. Of the six experts interviewed, five indicated that they could imagine using FML in their projects. It is also noteworthy that when using a variable level of abstraction, even all six experts considered FML to be useful.

These positive responses underscore the relevance and potential value of FML modeling for software system security. Future research can build on this and continue to explore the application of FML in practice to further improve the security of software systems.

10.2. Future work

The work completed within the scope of this thesis has laid the groundwork for further investigation of the present state of FML. Futhermore, opportunities to expand FML to meet the growing requirements of industry and research for the security of software systems have arisen The research methods and results used in this thesis provide a starting point for future work, which can be divided into further extensions of FML and further evaluations of the present state of FML.

10.2.1. Extension of FML

Within the scope of this thesis, FML was designed with a focus of an a software security tester as a stakeholder to provide a tool to re-model and evaluate the security of an existing software system In a future version, it should be developed to the point where it can be used to model software systems and their security. Thus it could be possible to use FML not only after the development process but already in the design phase of the software development process. The use of FML could then constitute a new step when designing software architectures and thus support the security of software systems.

Additionally, it would be a useful extension of FML, if the security dimensions could be evaluated automatically and this would not have to be done manually as is the case now. This could be used to detect security holes, which are not easily detected by hand. This would provide a user-independent and accurate way to evaluate the security of software systems.

For this purpose, a tool support for FML should be established. Complementary tools should allow an easy change between the dimensions of a software architecture. This would significantly improve the ability of FML to evaluate the security of a software system.

10.2.2. Further evaluation of FML

Within the time constraints of this thesis, FML was evaluated with the help of an expert interview with a limited number of participants. Although this lead to a good foundation the evaluation of FML needs to be expanded to provide a clearer picture of the current state of FML and its advantages and limitations and whether it meets the requirements of industry and research in a satisfactory way.

The first evaluation of FML, described in chapter 7, consisted of an expert interview. Since there were only six participants due to the lack of time and experts, FML should be further evaluated. For this purpose, it would be a possibility to repeat the first evaluation on a larger scale. This would allow a broader view of the experts to be obtained in order to uncover any weaknesses in the metamodel and to identify information gaps.

Furthermore, only an authentication dimension was shown to the experts during the first evaluation of FML. Other dimensions were not evaluated directly. This should be done in a future expert interview, which includes both the individual evaluation of other security dimensions, as well as the evaluation of the interaction of several security dimensions, in which the experts are presented with several security dimensions that refer to the same system. In this way, it could be examined to what extent the concept of dimensions, modeled by FML, is suitable to provide a holistic evaluation of the security of a software system.

In addition to the evaluation by expert interviews, it is useful to perform an evaluation of FML with a case study based on a real software system from the industry. Since real software systems are often very complex, this might reveal some weaknesses or deficiencies of FML.

A. Security dimension case studies

This appendix chapter includes the case studies as vector graphics, which were used to create FML:

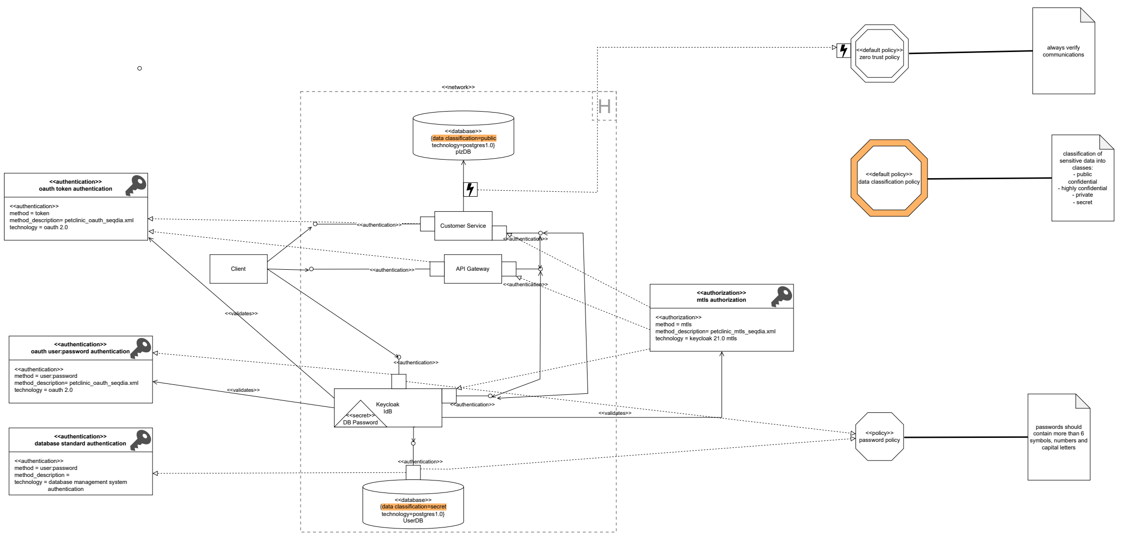


Figure A.1.: Case study for designing the authentication dimension. The existing case study was extended to include mtls authentication between the services. The client now has to authenticate itself at the api gateway with oauth2.0. Additionally, all databases were set to an outdated technology and the customer service communicates with an open database that does not require authentication. In addition, a password policy and a data classification policy were introduced, as well as a zero trust policy, which is violated by the unprotected database.

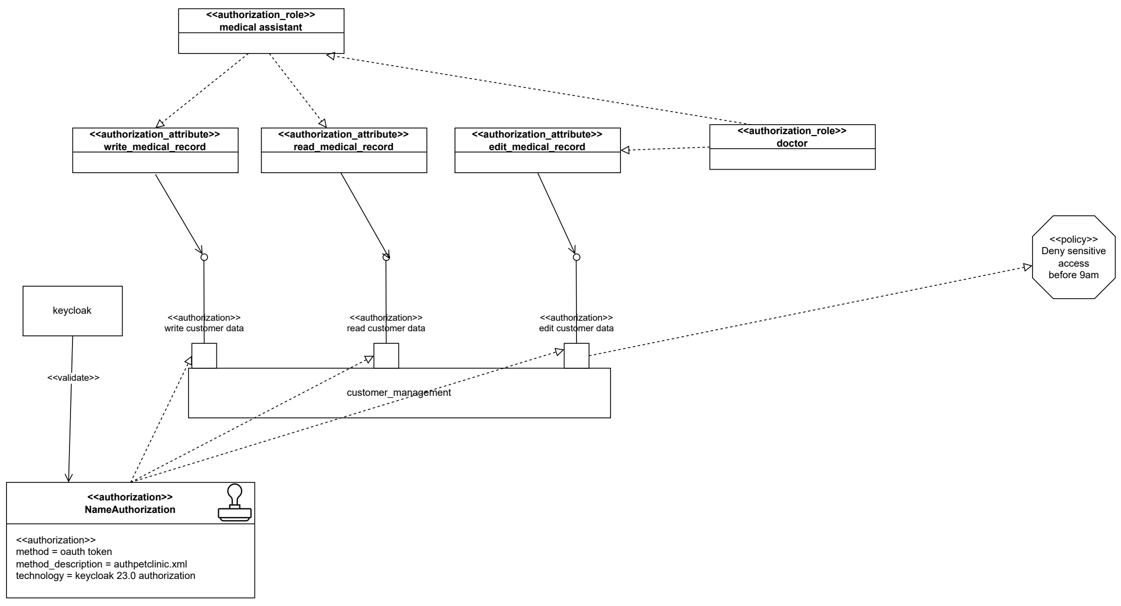
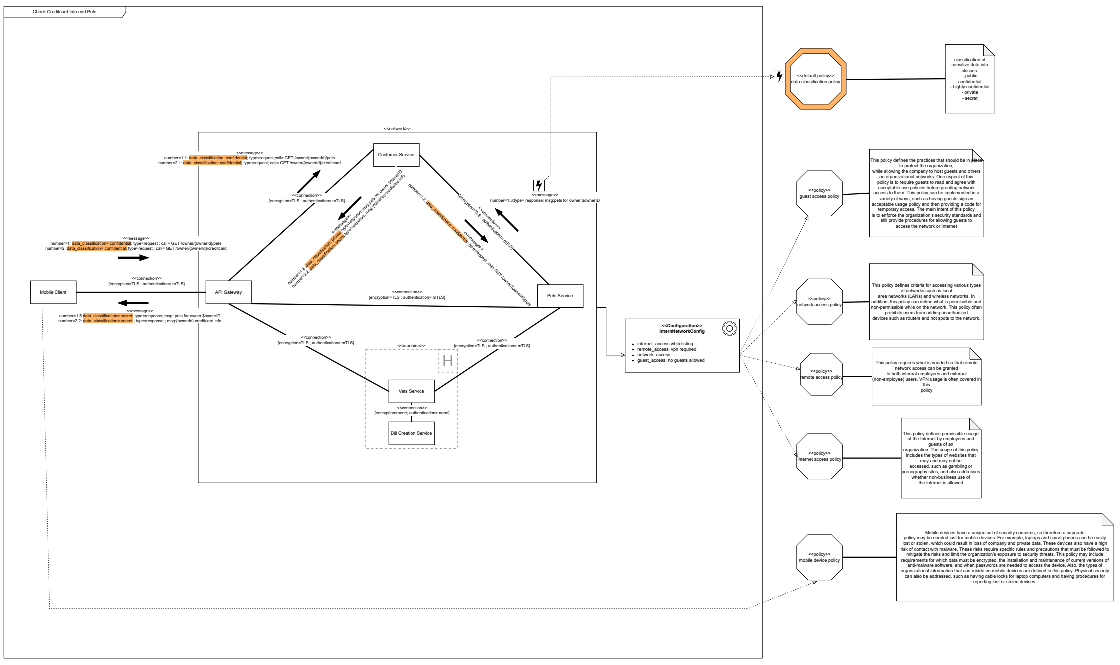


Figure A.2.: Case study for designing the authorization dimension. Various authorizations were added to the customer service. The writing and reading of customer data is allowed to be done by users with the role of medical assistant. Doctors have all the same rights as medical assistants, but are also allowed to make changes. The authorization validation is done by a keycloak service.


Check Creditcard Info and Pets
<<network>>
Customer Service
API Gateway
Pets Service
Mobile Client
Vets Service
Bill Creation Service
<<machine>>
H
<<message>>
number=1.1: data_classification: confidential; type=request;call= GET /owner/{ownerId}/pets
number=2.1: data_classification: confidential; type=request; call= GET /owner/{ownerId}/creditcard
<<message>>
number=1; data_classification= confidential; type=request ; call= GET /owner/{ownerId}/pets
number=2; data_classification= confidential; type=request ; call= GET /owner/{ownerId}/creditcard
<<message>>
number=1.5 data_classification= secret; type=response; msg: pets for owner $ownerID
number=2.2: data_classification= secret ; type=response ; msg:{ownerId} creditcard info
<<message>>
number=1.4 : data_classification: private;type=response; msg:pets for owner $ownerID
number=2.2:: data_classification: secret;type=response ; msg:{ownerId} creditcard info
<<message>>
number=1.2: data_classification: confidential; type=request ;call= GET /owner/{ownerId}/pets
<<message>>
number=1.3:type= response; msg:pets for owner $ownerID
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=TLS ; authentication= mTLS}
<<connection>>
{encryption=none; authentication= none}
<<Configuration>>
InternNetworkConfig
internet_access:whitelisting
remote_access: vpn required
network_access:
guest_access: no guests allowed
<<default policy>>
data classification policy
<<policy>>
guest access policy
<<policy>>
network access policy
<<policy>>
remote access policy
<<policy>>
internet access policy
<<policy>>
mobile device policy

classification of sensitive data into classes:
- public
confidential
- highly confidential
- private
- secret

This policy defines the practices that should be in place to protect the organization, while allowing the company to host guests and others on organizational networks. One aspect of this policy is to require guests to read and agree with acceptable use policies before granting network access to them. This policy can be implemented in a variety of ways, such as having guests sign an acceptable usage policy and then providing a code for temporary access. The main intent of this policy is to enforce the organization's security standards and still provide procedures for allowing guests to access the network or Internet

This policy defines criteria for accessing various types of networks such as local area networks (LANs) and wireless networks. In addition, this policy can define what is permissible and non-permissible while on the network. This policy often prohibits users from adding unauthorized devices such as routers and hot spots to the network.

This policy requires what is needed so that remote network access can be granted to both internal employees and external (non-employee) users. VPN usage is often covered in this policy

This policy defines permissible usage of the Internet by employees and guests of an organization. The scope of this policy includes the types of websites that may and may not be accessed, such as gambling or pornography sites, and also addresses whether non-business use of the Internet is allowed

Mobile devices have a unique set of security concerns, so therefore a separate policy may be needed just for mobile devices. For example, laptops and smart phones can be easily lost or stolen, which could result in loss of company and private data. These devices also have a high risk of contact with malware. These risks require specific rules and precautions that must be followed to mitigate the risks and limit the organization's exposure to security threats. This policy may include requirements for which data must be encrypted, the installation and maintenance of current versions of anti-malware software, and when passwords are needed to access the device. Also, the types of organizational information that can reside on mobile devices are defined in this policy. Physical security can also be addressed, such as having cable locks for laptop computers and having procedures for reporting lost or stolen devices.

Figure A.3.: Case study for designing the secure communication dimension. A network was specified, which has a configuration. This configuration fullfills various policies. In addition, a message chain was defined over various services to query credit card information for users, as well as to query a user's pets.

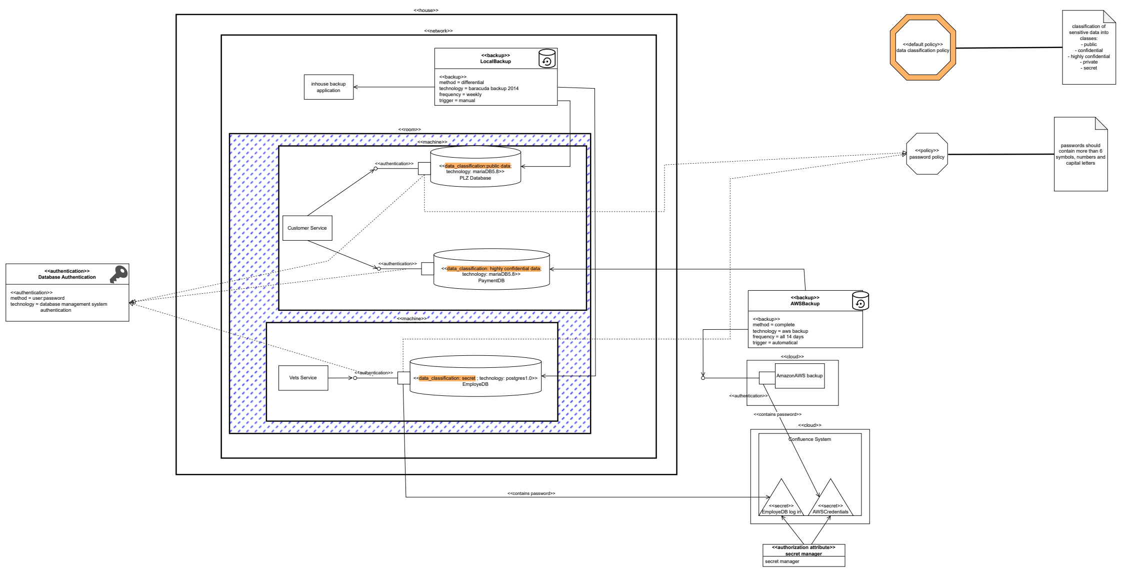

<<house>>
<<network>>
<<backup>> LocalBackup
<<backup>>
method = differential
technology = baracuda backup 2014
frequency = weekly
trigger = manual
inhouse backup application
<<room>>
<<machine>>
<<authentication>>
<<data_classification:public data; technology: mariaDB5.8>> PLZ Database
Customer Service
<<authentication>>
<<data_classification: highly confidential data; technology: mariaDB5.8>> PaymentDB
<<machine>>
Vets Service
<<authentication>>
<<data_classification: secret ; technology: postgres1.0>> EmployeDB
<<contains password>>
<<authentication>> Database Authentication
<<authentication>>
method = user:password
technology = database management system authentication
<<default policy>> data classification policy
classification of sensitive data into classes: - public - confidential - highly confidential - private - secret
<<policy>> password policy
passwords should contain more than 6 symbols, numbers and capital letters
<<backup>> AWSBackup
<<backup>>
method = complete
technology = aws backup
frequency = all 14 days
trigger = automatical
<<cloud>>
AmazonAWS backup
<<authentication>>
<<contains password>>
<<cloud>>
Confluence System
<<secret>> EmployeDB log in
<<secret>> AWSCredentials
<<authorization attribute>> secret manager
secret manager

Figure A.4.: Case study for designing the secure storage dimension. It was added that the system runs in a house where there is a corporate network. In one room of the house there are two machines running the customer and vets service. In addition, two databases with a local backup were added, where the backup is in the same building. Further, a database with cloud backup was added. Additionally, an application was added that contains two passwords modeled as secret. One for the cloud backup service and one for a database of the system.

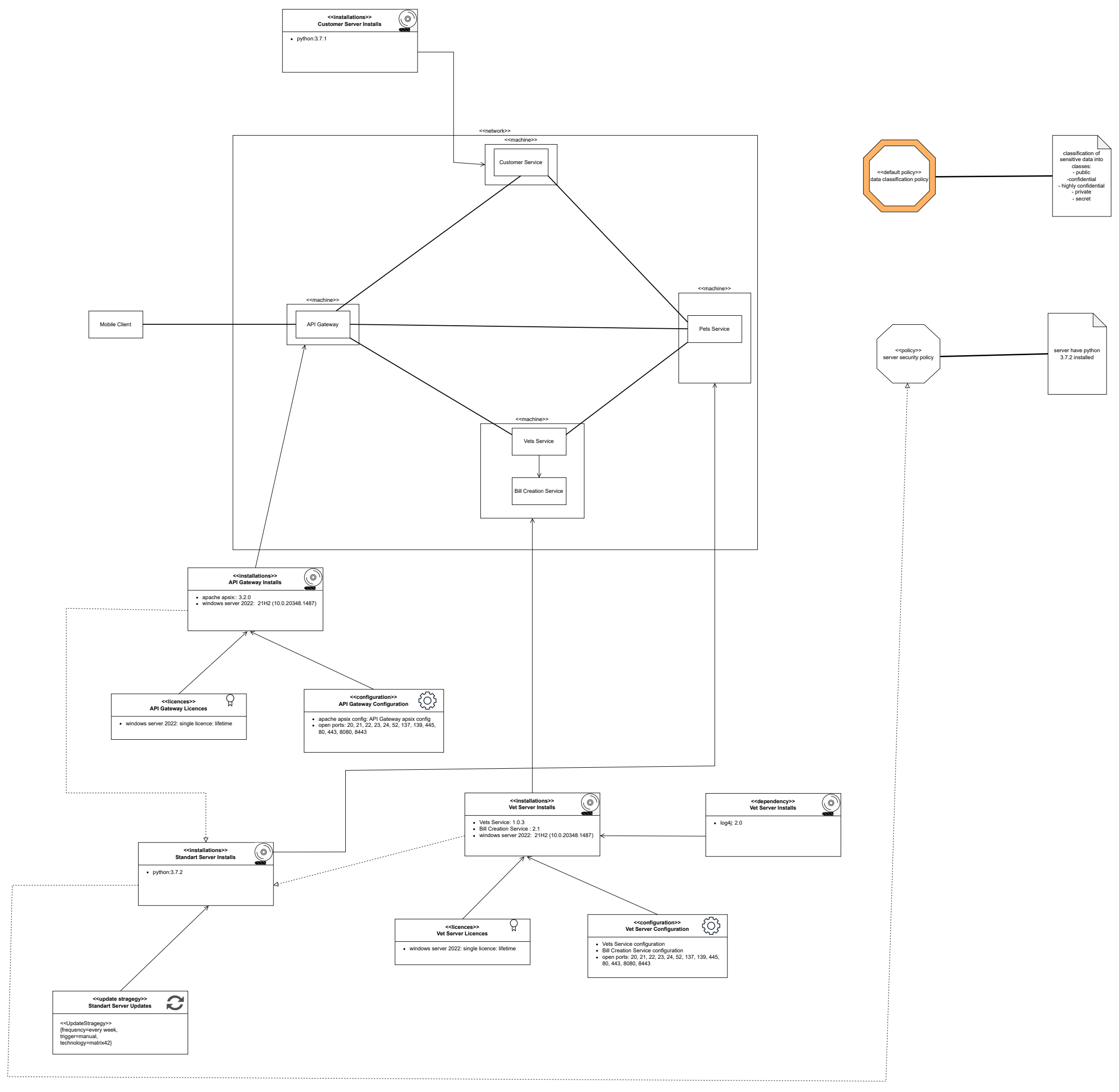


Figure A.5.: Case study for designing the secure deployment dimension. It was added that each service runs on one machine. In addition, a configuration for machines was defined. One of them is the default configuration, which is intended as a minimum configuration for all systems. In addition, there is a server security policy, which is violated by the customer server install.

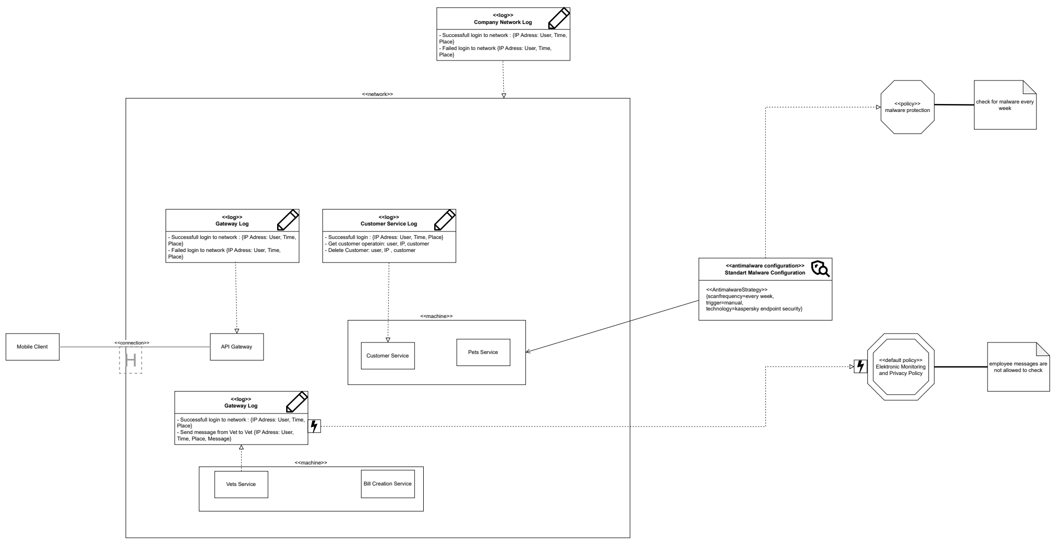
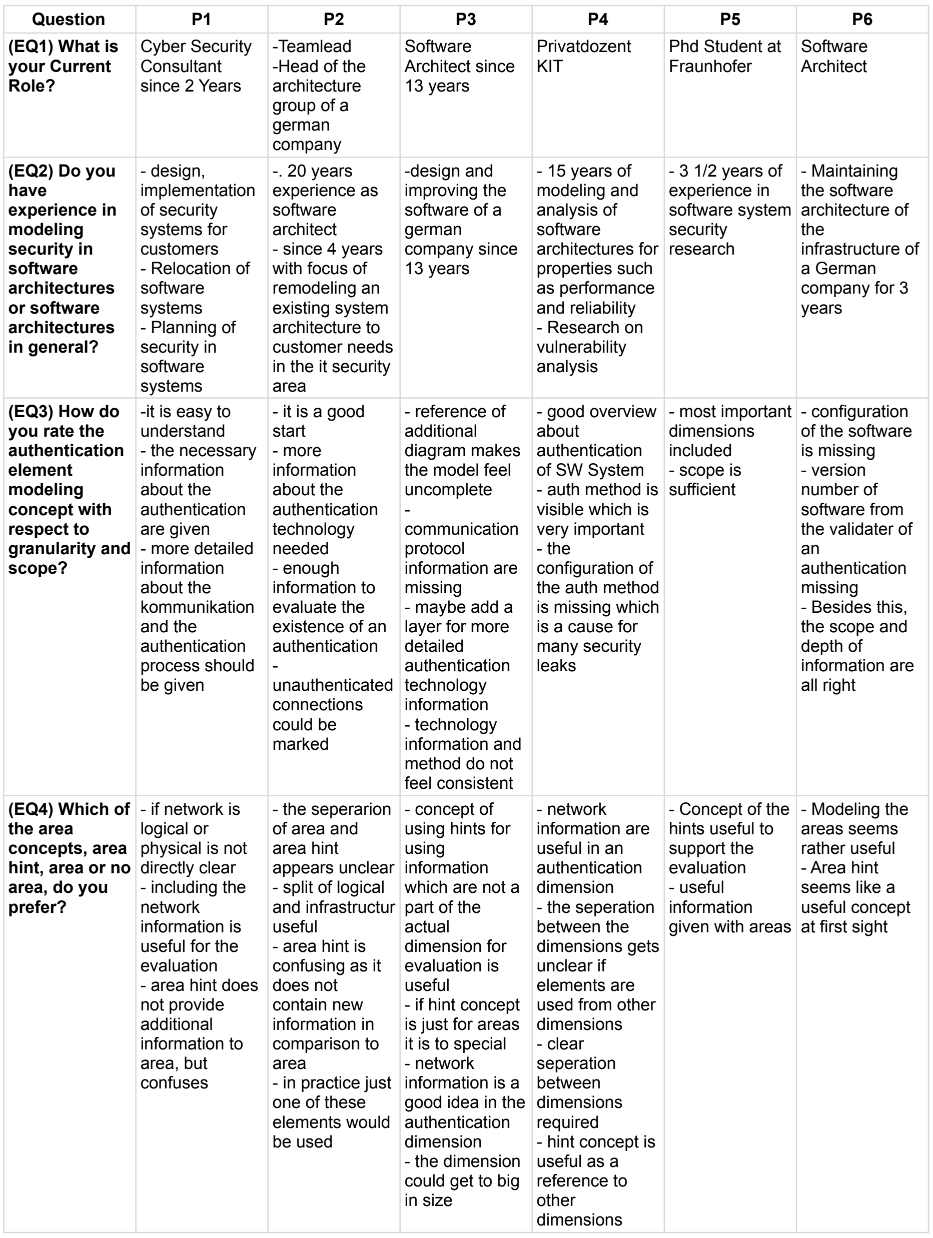

<<log>>
Company Network Log
- Successfull login to network : {IP Adress: User, Time, Place}
- Failed login to network {IP Adress: User, Time, Place}
<<network>>
<<policy>>
malware protection
check for malware every week
<<log>>
Gateway Log
- Successfull login to network : {IP Adress: User, Time, Place}
- Failed login to network {IP Adress: User, Time, Place}
<<log>>
Customer Service Log
- Successfull login : {IP Adress: User, Time, Place}
- Get customer operatoin: user, IP, customer
- Delete Customer: user, IP , customer
<<antimalware configuration>>
Standart Malware Configuration
<<AntimalwareStrategy>>
{scanfrequency=every week,
trigger=manual,
technology=kaspersky endpoint security}
<<machine>>
Mobile Client
<<connection>>
API Gateway
Customer Service
Pets Service
<<default policy>>
Elektronic Monitoring and Privacy Policy
employee messages are not allowed to check
<<log>>
Gateway Log
- Successfull login to network : {IP Adress: User, Time, Place}
- Send message from Vet to Vet {IP Adress: User, Time, Place, Message}
<<machine>>
Vets Service
Bill Creation Service

Figure A.6.: Case study for designing the monitoring dimension. It was added that each machine will be equipped with a log. The electronic and privacy policy was added, which defines what must not be logged. The API gateway violates this by logging the content of messages between vets. In addition, there is an antimalware configuration that is run manually once a week. However, this is only active on one machine, which means that only this machine complies with the antimalware policy.

B. Evaluation transcription

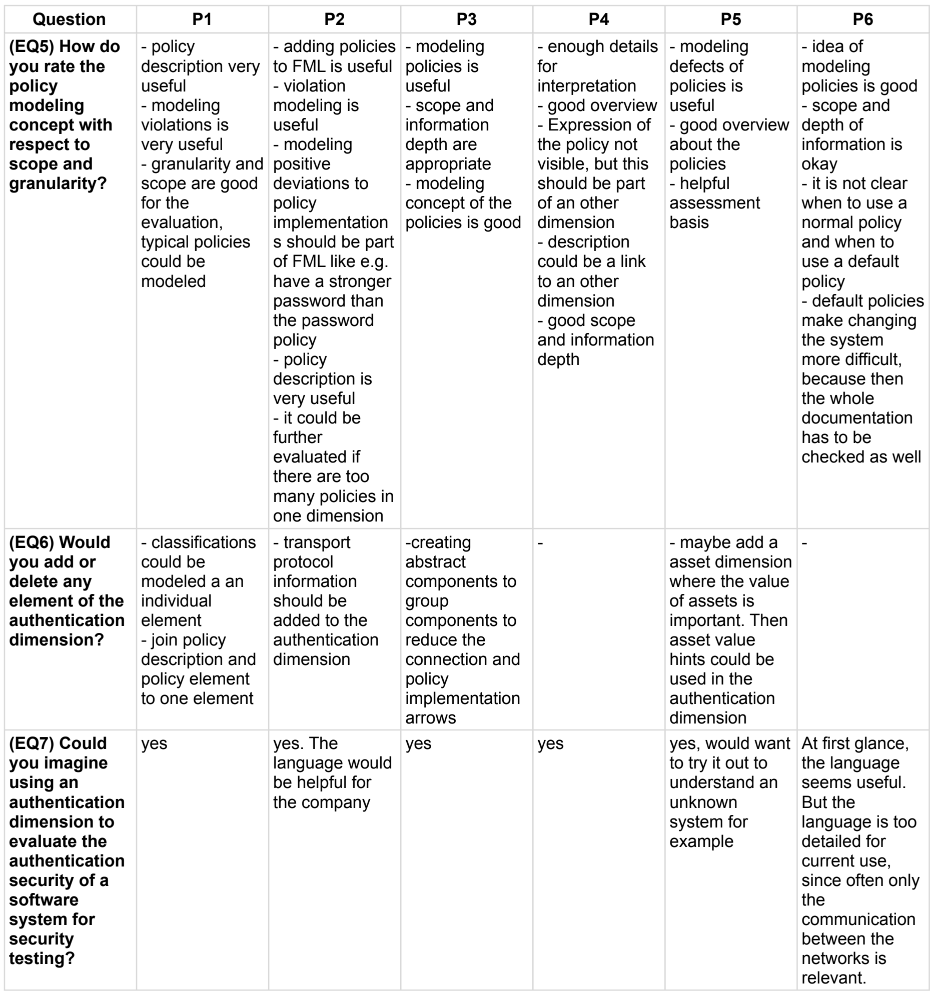
This appendix contains the shortened and translated transcriptions of the expert interview.

Figure B.1.: Interview transcription EQ1-4



Question	P1	P2	P3	P4	P5	P6
(EQ1) What is your Current Role?	Cyber Security Consultant since 2 Years	-Teamlead -Head of the architecture group of a german company	Software Architect since 13 years	Privatdozent KIT	Phd Student at Fraunhofer	Software Architect
(EQ2) Do you have experience in modeling security in software architectures or software architectures in general?	- design, implementation of security systems for customers - Relocation of software systems - Planning of security in software systems	-. 20 years experience as software architect - since 4 years with focus of remodeling an existing system architecture to customer needs in the it security area	-design and improving the software of a german company since 13 years	- 15 years of modeling and analysis of software architectures for properties such as performance and reliability - Research on vulnerability analysis	- 3 1/2 years of experience in software system security research	- Maintaining the software architecture of the infrastructure of a German company for 3 years
(EQ3) How do you rate the authentication element modeling concept with respect to granularity and scope?	-it is easy to understand - the necessary information about the authentication are given - more detailed information about the kommunikation and the authentication process should be given	- it is a good start - more information about the authentication technology needed - enough information to evaluate the existence of an authentication - unauthenticated connections could be marked	- reference of additional diagram makes the model feel uncomplete - communication protocol information are missing - maybe add a layer for more detailed authentication technology information - technology information and method do not feel consistent	- good overview about authentication of SW System - auth method is visible which is very important - the configuration of the auth method is missing which is a cause for many security leaks	- most important dimensions included - scope is sufficient	- configuration of the software is missing - version number of software from the validater of an authentication missing - Besides this, the scope and depth of information are all right
(EQ4) Which of the area concepts, area hint, area or no area, do you prefer?	- if network is logical or physical is not directly clear - including the network information is useful for the evaluation - area hint does not provide additional information to area, but confuses	- the seperarion of area and area hint appears unclear - split of logical and infrastructur useful - area hint is confusing as it does not contain new information in comparison to area - in practice just one of these elements would be used	- concept of using hints for using information which are not a part of the actual dimension for evaluation is useful - if hint concept is just for areas it is to special - network information is a good idea in the authentication dimension - the dimension could get to big in size	- network information are useful in an authentication dimension - the seperation between the dimensions gets unclear if elements are used from other dimensions - clear seperation between dimensions required - hint concept is useful as a reference to other dimensions	- Concept of the hints useful to support the evaluation - useful information given with areas	- Modeling the areas seems rather useful - Area hint seems like a useful concept at first sight

Figure B.2.: Interview transcription EQ4-7



Question	P1	P2	P3	P4	P5	P6
(EQ5) How do you rate the policy modeling concept with respect to scope and granularity?	- policy description very useful - modeling violations is very useful - granularity and scope are good for the evaluation, typical policies could be modeled	- adding policies to FML is useful - violation modeling is useful - modeling positive deviations to policy implementations should be part of FML like e.g. have a stronger password than the password policy - policy description is very useful - it could be further evaluated if there are too many policies in one dimension	- modeling policies is useful - scope and information depth are appropriate - modeling concept of the policies is good	- enough details for interpretation - good overview - Expression of the policy not visible, but this should be part of an other dimension - description could be a link to an other dimension - good scope and information depth	- modeling defects of policies is useful - good overview about the policies - helpful assessment basis	- idea of modeling policies is good - scope and depth of information is okay - it is not clear when to use a normal policy and when to use a default policy - default policies make changing the system more difficult, because then the whole documentation has to be checked as well
(EQ6) Would you add or delete any element of the authentication dimension?	- classifications could be modeled a an individual element - join policy description and policy element to one element	- transport protocol information should be added to the authentication dimension	-creating abstract components to group components to reduce the connection and policy implementation arrows	-	- maybe add a asset dimension where the value of assets is important. Then asset value hints could be used in the authentication dimension	-
(EQ7) Could you imagine using an authentication dimension to evaluate the authentication security of a software system for security testing?	yes	yes. The language would be helpful for the company	yes	yes	yes, would want to try it out to understand an unknown system for example	At first glance, the language seems useful. But the language is too detailed for current use, since often only the communication between the networks is relevant.

Bibliography

[250] “ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)”. In: *ISO/IEC/IEEE 25010:2011* (2011), pp. 1–46 (cit. on p. 7).

[612] “ISO/IEC/IEEE Systems and software engineering – Architecture description”. In: *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)* (2011), pp. 1–46. DOI: `10.1109/IEEESTD.2011.6129467` (cit. on p. 6).

[BCK03] L. Bass, P. Clements, and R. Kazman. “Software Architecture In Practice”. In: Jan. 2003. ISBN: 978-0321154958 (cit. on p. 5).

[BJE16] S. Bromander, A. Jøsang, and M. Eian. “Semantic Cyberthreat Modelling”. In: *Semantic Technologies for Intelligence, Defense, and Security.* 2016 (cit. on p. 1).

[Ele90] I. Electronics Engingeers. “IEEE Standard Glossary of Software Engineering Terminology”. In: *Office* 121990 (1990), p. 84 (cit. on p. 5).

[Ell+96] W. J. Ellis et al. “Toward a recommended practice for architectural description”. In: *Proceedings of ICECCS ’96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)* (1996), pp. 408–413 (cit. on p. 7).

[Fak] K. Fakhroutdinov. *UML, Meta Meta Models and profiles.* URL: `https://www.uml-diagrams.org/uml-meta-models.html` (cit. on p. 65).

[Fer+01] D. F. Ferraiolo et al. “Proposed NIST standard for role-based access control”. In: *ACM Transactions on Information and System Security (TISSEC)* 4 (2001), pp. 224 –274 (cit. on p. 8).

[GGS15] R. Goel, M. C. Govil, and G. Singh. “Security Requirements Elicitation and Assessment Mechanism (SecREAM)”. In: *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2015), pp. 1862–1866 (cit. on p. 68).

[Hu+14] V. C. Hu et al. “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”. In: 2014 (cit. on p. 9).

[Ist] *ISTQB Certified Tester Advanced Level Syllabus - Security Tester.* International Software Testing Qualifications Board. Version 1.0. 2016. URL: `https://www.german-testing-board.info/wp-content/uploads/2020/12/ISTQB-CTAL-SEC_Syllabus_V2016_EN.pdf` (cit. on pp. 16, 19).

[Jü02] J. Jürjens. “UMLsec: Extending UML for secure systems development”. In: vol. 2460. Jan. 2002, pp. 412–425. ISBN: 978-3-540-44254-7. DOI: 10.1007/3-540-45800-X_32 (cit. on pp. 65, 66).

[KK21] A. Kotov and J. Klein. *SEI Software Architecture Principles and Practices Overview Training*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA, 2021 (cit. on p. 1).

[LBD02] T. Lodderstedt, D. Basin, and J. Doser. “SecureUML: A UML-based modeling language for model-driven security”. In: vol. 2460. Jan. 2002, pp. 426–441. ISBN: 978-3-540-44254-7. DOI: 10.1007/3-540-45800-X_33 (cit. on pp. 66, 67).

[MiD10] R. Matulevi, ius, and M. Dumas. “A Comparison of SecureUML and UMLsec for Role-based Access Control”. In: 2010 (cit. on pp. 65, 66, 69).

[Obj17] Object Management Group. *OMG Unified Modeling Language (OMG UML)*. Version 2.5.1. Dec. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (cit. on p. 65).

[PMMD07] J. Pavlich-Mariscal, L. Michel, and S. Demurjian. “Enhancing UML to Model Custom Security Aspects”. In: Jan. 2007, p. 10 (cit. on p. 65).

[RFMP06] A. Rodríguez, E. Fernández-Medina, and M. Piattini. “Towards a UML 2.0 Extension for the Modeling of Security Requirements in Business Processes”. In: *Trust and Privacy in Digital Business*. Ed. by S. Fischer-Hübner, S. Furnell, and C. Lambrinoudakis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 51–61. ISBN: 978-3-540-37752-8 (cit. on pp. 68, 70).

[RREAT17] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, and T. Tryfonas. “IoTsec: UML Extension for Internet of Things Systems Security Modelling”. In: *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*. 2017, pp. 151–156. DOI: 10.1109/ICMEAE.2017.20 (cit. on pp. 65, 68).

[SEH13] T. Sommestad, M. Ekstedt, and H. Holm. “The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures”. In: *Systems Journal, IEEE* 7 (Sept. 2013), pp. 363–373. DOI: 10.1109/JSYST.2012.2221853 (cit. on pp. 67–69).

[Sin22] B. Sinkovec. “Towards a software engineering view of security for microservice-based applications”. MA thesis. RWTH Aachen University, 2022 (cit. on pp. c, 2, 3, 11–13, 25, 26, 28, 71).

[TMD09] R. N. Taylor, N. Medvidović, and E. M. Dashofy. “Software architecture: foundations, theory, and practice”. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering* 2 (2009), pp. 471–472 (cit. on p. 6).