UNIVERSITÄT DUISBURG ESSEN

Open-Minded

Roman Wirtz
Master Thesis
Summer Semester 2016

Master Thesis

Risk-based Elicitation of Security Requirements

Roman Wirtz

September 2016

University of Duisburg-Essen · Faculty of Engineering
Department of Applied Computer Science and Applied Cognitive Science
Working Group Software Engineering – Prof. Dr. M. Heisel
Supervised By: Prof. Dr. Maritta Heisel
2nd Reviewer: Prof. Dr.-Ing. Torben Weis

Contents

I Foundations **1**

1 Introduction **3**
1.1 Motivation . . . 3
1.2 Content Overview . . . 5
1.3 Terminology . . . 5
1.3.1 Security Properties . . . 5
1.3.2 Roles . . . 6

2 Background **7**
2.1 Problem Frames . . . 7
2.1.1 Basics . . . 7
2.1.1.1 Domain Types . . . 7
2.1.1.2 Phenomena . . . 8
2.1.1.3 Interfaces . . . 8
2.1.1.4 Requirement . . . 8
2.1.2 Diagram Types . . . 8
2.1.3 UML4PF . . . 9
2.2 ADIT Software Development Process . . . 9
2.2.1 Analysis . . . 9
2.2.2 Design . . . 10
2.2.3 Implementation . . . 10
2.2.4 Testing . . . 10
2.3 CORAS . . . 10
2.3.1 CORAS Language . . . 10
2.3.1.1 CORAS Symbols . . . 10
2.3.1.2 CORAS Diagrams . . . 12
2.3.2 CORAS Method . . . 12
2.3.2.1 Step 1: Preparation for the analysis . . . 13
2.3.2.2 Step 2: Customer presentation of the target . . . 13
2.3.2.3 Step 3: Refining target description using asset diagrams . . . 13
2.3.2.4 Step 4: Approval of target description . . . 14
2.3.2.5 Step 5: Risk identification using threat diagrams . . . 14
2.3.2.6 Step 6: Risk estimation using threat diagrams . . . 15
2.3.2.7 Step 7: Risk evaluation using risk diagrams . . . 15
2.3.2.8 Step 8: Risk treatment using treatment diagrams . . . 15

3 Case Study - Smart Grid **17**
3.1 Description . . . 17
3.2 Requirements . . . 18
3.3 Workflow . . . 18

II The ProCOR Approach - Security Requirements Elicitation 21

4 Basics 23
4.1 Overview . . . 23
4.2 Step Description . . . 24
4.3 UML Profile for CORAS . . . 25
4.3.1 Diagram . . . 25
4.3.2 Asset . . . 25
4.3.3 ThreatDescription . . . 25
4.3.4 DataTypes . . . 25
4.3.5 Element . . . 25
4.3.6 Relations . . . 27

5 Step 1: Preparation 29
5.1 Customer Presentation . . . 29
5.2 Initial UML Model . . . 30
5.3 Adjust Problem Diagrams . . . 30
5.4 Phenomena Relations . . . 32
5.5 Validation Conditions . . . 32
5.6 Case Study . . . 33
5.6.1 Context Diagram . . . 33
5.6.2 Problem Diagrams . . . 33
5.6.2.1 Requirement 1: Setup . . . 34
5.6.2.2 Requirement 2: Establish Connection . . . 34
5.6.2.3 Requirement 3: Measuring . . . 34
5.6.2.4 Requirement 4: Retrieving . . . 36
5.6.2.5 Requirement 5: Invoicing . . . 36
5.6.2.6 Requirement 6: Control Power . . . 37
5.6.2.7 Requirement 7: Display Invoice . . . 38
5.6.2.8 Requirement 8: Change Personal Data . . . 38
5.6.3 Adjust Problem Diagrams . . . 40
5.6.4 Phenomena Relations . . . 42

6 Step 2: Defining Focus & Scope 43
6.1 Selection of Assets & Asset Diagram . . . 44
6.1.1 Profile Extension . . . 44
6.1.2 Asset Identification . . . 45
6.2 Information Flow Graph . . . 46
6.2.1 Graph Specification . . . 46
6.2.2 Profile Extension . . . 47
6.3 Scope Definition . . . 49
6.4 Validation Conditions . . . 49
6.5 Case Study . . . 49
6.5.1 Selection of Assets . . . 49
6.5.2 Information Flow Graph . . . 51
6.5.3 Scope Definition . . . 51

7 Step 3: Risk Identification 53
7.1 Risk Identification . . . 54
7.1.1 Resources . . . 54
7.1.1.1 Threat Classification . . . 54
7.1.1.2 Vulnerability Databases . . . 54
7.1.1.3 Threat Patterns . . . 57
7.1.2 Threat Elicitation . . . 58
7.1.3 Domain Knowledge Diagrams . . . 59

7.1.4 Unwanted Incidents . . . 59
7.2 Threat Overview Diagram . . . 60
7.3 Validation Conditions . . . 60
7.4 Case Study . . . 61
7.4.1 Threat Elicitation . . . 61
7.4.2 Domain Knowledge Diagrams . . . 61
7.4.2.1 Domain Knowledge for WirelessNetwork . . . 61
7.4.2.2 Domain Knowledge for EnergySupplier . . . 62
7.4.2.3 Domain Knowledge for SmartMeter . . . 62
7.4.2.4 Domain Knowledge for CommunicationHub . . . 62
7.4.2.5 Domain Knowledge for WAN . . . 62
7.4.3 Unwanted Incidents . . . 64
7.4.4 Threat Overview Diagram . . . 64

8 Step 4: Risk Estimation & Evaluation 67
8.1 Estimation of Likelihoods & Consequences . . . 68
8.2 Risk Evaluation . . . 69
8.3 Validation Conditions . . . 70
8.4 Case Study . . . 71
8.4.1 Scales and Risk Matrix . . . 71
8.4.2 Annotated Threat Overview Diagram . . . 71
8.4.3 Risk Evaluation . . . 72

9 Step 5: High-Level Security Requirements Instantiation 75
9.1 Instantiation of High-Level Security Requirements . . . 76
9.2 Threat Diagram per Unwanted Incident . . . 76
9.3 Validation Conditions . . . 76
9.4 Case Study . . . 77
9.4.1 High-Level Security Requirements . . . 77
9.4.2 Threat Diagram per Unwanted Incident . . . 78
9.4.2.1 Disclose transmitted data . . . 78
9.4.2.2 Manipulate transmitted data . . . 78
9.4.2.3 Wrong calculation for invoices . . . 78
9.4.2.4 Incorrect measuring of power consumption . . . 78
9.4.2.5 Connection unavailable . . . 79

10 Step 6: Treatment Selection 81
10.1 Profile Extension . . . 82
10.2 Treatment Selection & Likelihood Reduction . . . 82
10.2.1 Initial Treatment Selection . . . 82
10.2.2 Treatment Consolidation . . . 83
10.3 Validation Conditions . . . 84
10.4 Case Study . . . 84
10.4.1 Treatment Selection . . . 84
10.4.1.1 Initial Treatment Selection . . . 84
10.4.1.2 Treatment Consolidation . . . 86

11 Step 7: Treatment Problem Diagrams Creation 89
11.1 Profile Extension . . . 90
11.2 Creation of Treatment Problem Diagrams . . . 90
11.3 Validation Conditions . . . 91
11.4 Next Steps . . . 92
11.5 Case Study . . . 92
11.5.1 HLSR-1 . . . 92
11.5.2 HLSR-2 . . . 92

11.5.3 HLSR-3 . . . 94
11.5.4 HLSR-4 . . . 94
11.5.5 HLSR-5 . . . 94

12 Tool Support **97**
12.1 Environment . . . 97
12.2 UML4PF Language . . . 98
12.3 Combining UML4PF and CORAS . . . 98
12.4 ProCOR Tool . . . 99
12.4.1 Step 1: Preparation . . . 99
12.4.2 Step 2: Defining Focus & Scope . . . 100
12.4.3 Step 3: Risk Identification . . . 101
12.4.4 Step 4: Risk Estimation & Evaluation . . . 101
12.4.5 Step 5: High-Level Security Requirements Instantiation . . . 103
12.4.6 Step 6: Treatment Selection . . . 103
12.4.7 Step 7: Treatment Problem Diagrams Creation . . . 104
12.5 Plugins . . . 104

III Conclusion **105**

13 Related Work **107**

14 Evaluation **111**
14.1 Contributions in the Thesis . . . 111
14.2 Contributions to Research Questions . . . 112
14.2.1 Research Question 1 . . . 112
14.2.2 Research Question 2 . . . 112
14.2.3 Research Question 3 . . . 112
14.2.4 Research Question 4 . . . 112
14.2.5 Research Question 5 . . . 113
14.3 Restrictions & Future Work . . . 113
14.3.1 Indirect Assets . . . 113
14.3.2 Vulnerabilities . . . 113
14.3.3 Treatments . . . 113
14.3.4 Combination of elements . . . 114
14.3.5 Scalability . . . 114
14.3.6 Risk Identification . . . 114

15 Bibliography **115**

Eidesstattliche Erklärung **119**

List of Figures

1.1 Distribution of exploits used in cyberattacks, by the type of application attacked, 2015 [Kas15] . . . 4

2.1 CORAS symbols [LSS10] . . . 10
2.2 Eight Steps of the CORAS Method [LSS10] . . . 12
2.3 Risk Matrix Example [LSS10] . . . 14

3.1 Informal Description of the Case Study . . . 17

4.1 Overview ProCOR Method . . . 23
4.2 UML Profile for CORAS . . . 26

5.1 ProCOR - Step 1 . . . 29
5.2 Minimal Example for Connection Domain Discovery . . . 31
5.3 Minimal Example for Connection Domain Discovery Adjusted . . . 31
5.4 Phenomena Relations - Minimal Example . . . 32
5.5 Case Study: Context Diagram . . . 34
5.6 Case Study: Requirement 1 - Setup . . . 35
5.7 Case Study: Requirement 2 - Establish Connection . . . 35
5.8 Case Study: Requirement 3 - Measuring . . . 36
5.9 Case Study: Requirement 4 - Retrieving . . . 37
5.10 Case Study: Requirement 5 - Invoicing . . . 37
5.11 Case Study: Requirement 6 - Control Power . . . 38
5.12 Case Study: Requirement 7 - Display Invoice . . . 39
5.13 Case Study: Requirement 8 - Change Personal Data . . . 39
5.14 Case Study: Requirement 7 - Display Invoice (adjusted) . . . 41
5.15 Case Study: Requirement 8 - Change Personal Data (adjusted) . . . 41
5.16 Case Study: Phenomena Relations . . . 42

6.1 ProCOR - Step 2 . . . 43
6.2 ProCOR Profile: Asset . . . 44
6.3 ProCOR Profile: Information Flow Graph . . . 48
6.4 Case Study: Symbolic Phenomena . . . 50
6.5 Case Study: Asset Diagram . . . 50
6.6 Case Study: Information Flow Graph . . . 52

7.1 ProCOR - Step 3 . . . 53
7.2 CSA Threat - Data Breaches [Clo16] . . . 55
7.3 CVE-2015-8289 - Vulnerability Example [Nat15] . . . 56
7.4 Case Study: Domain Knowledge for WirelessNetwork . . . 62
7.5 Case Study: Domain Knowledge for EnergySupplier . . . 63
7.6 Case Study: Domain Knowledge for SmartMeter . . . 63
7.7 Case Study: Domain Knowledge for CommunicationHub . . . 64
7.8 Case Study: Domain Knowledge for WAN . . . 64
7.9 Case Study: Threat Overview Diagram . . . 66

8.1 ProCOR - Step 4 . . . 67

8.2 Case Study: Annotated Threat Overview Diagram with Consequences and Likelihoods . . . 72
8.3 Case Study: Threat Overview Diagram with Acceptable Likelihoods . . . 74

9.1 ProCOR - Step 5 . . . 75
9.2 Case Study: Threat Diagram for Disclose Transmitted Data . . . 78
9.3 Case Study: Threat Diagram for Manipulate Transmitted Data . . . 78
9.4 Case Study: Threat Diagram for Wrong Calculation for Invoice . . . 78
9.5 Case Study: Threat Diagram for Incorrect Measuring Of Power Consumption 79
9.6 Case Study: Threat Diagram for Connection Unavailable . . . 79

10.1 ProCOR - Step 6 . . . 81
10.2 ProCOR Profile: Addresses Dependency . . . 82
10.3 Case Study: Threat Diagram with Treatment Scenarios for HL-SR1 . . . 85
10.4 Case Study: Threat Diagram with Treatment Scenarios for HL-SR2 . . . 85
10.5 Case Study: Threat Diagram with Treatment Scenarios for HL-SR3 . . . 86
10.6 Case Study: Threat Diagram with Treatment Scenarios for HL-SR4 . . . 86
10.7 Case Study: Threat Diagram with Treatment Scenarios for HL-SR5 . . . 87

11.1 ProCOR - Step 7 . . . 89
11.2 ProCOR Profile: Treatment Problem Diagram . . . 90
11.3 Case Study: Treatment Problem Diagram for HL-SR1 . . . 93
11.4 Case Study: Treatment Problem Diagram for HL-SR2 . . . 93
11.5 Case Study: Treatment Problem Diagram for HL-SR3 . . . 94
11.6 Case Study: Treatment Problem Diagram for HL-SR4 . . . 95
11.7 Case Study: Treatment Problem Diagram for HL-SR5 . . . 95

12.1 UML4PF Language: Graphical Notation . . . 98
12.2 UML4PF Language: Problem Diagram Example . . . 99
12.3 Tool Support: Create Problem Diagram . . . 100
12.4 Tool Support: Phenomena Relations . . . 100
12.5 Tool Support: Asset Selection . . . 101
12.6 Tool Support: Scope Definition . . . 101
12.7 Tool Support: Estimation of Likelihoods . . . 102
12.8 Tool Support: Estimation of Consequences . . . 102
12.9 UML4PF Language: Symbol for HL-SR . . . 103
12.10 Tool Support: New Treatment . . . 103
12.11 UML4PF Language: Symbol for Treatment Domain . . . 104

13.1 Risk Management According to ISO 31000 [ISO09] . . . 107

List of Tables

4.1 ProCOR: Step Description 24

5.1 ProCOR - Step 1 29
5.2 Connection Domain Discovery Table - Minimal Example 31
5.3 Case Study: Connection Domain Discovery Table 40

6.1 ProCOR - Step 2 43

7.1 ProCOR: Step 3 53
7.2 Example Threat Pattern of First Class [UF14] 57
7.3 Example Threat Pattern of Second Class [UF14] 58
7.4 Threat Elicitation Results 59
7.5 Unwanted Incident Elicitation 59
7.6 Case Study: Threat Identification 61
7.7 Case Study: Unwanted Incidents 65

8.1 ProCOR: Step 4 67
8.2 Consequence Scale depending on CIA property 69
8.3 Risk Calculation Results 69
8.4 Case Study: Risk Matrix 71
8.5 Case Study: Calculated Risks 73

9.1 ProCOR: Step 5 75

10.1 ProCOR: Step 6 81

11.1 ProCOR: Step 7 89

Acronyms

The following variables are used in the list:
$x \in \mathbb{N}_{>0}$, where x stands for the number of the element

Ax Assumption x

CD Context diagram

CIA Confidentiality, Integrity, Availability

CSA Cloud Security Alliance

CVE Common Vulnerabilities and Exposures

DK Domain Knowledge

Fx Fact x

HL-SR High-Level Security Requirement

IFG Information Flow Graph

NIST National Institute for Standards and Technology

NVD National Vulnerability Database

PD Problem diagram

ProCOR Problem-based CORAS

ProPAn Problem-based Privacy Analysis

Rx Requirement x

RQ Research Question

UML Unified Modeling Language

TD Threat Diagram

UI Unwanted Incident

UML Unified Modeling Language

UML4PF Unified Modeling Language for Problem Frames

WAN Wide Area Network

Acknowledgements

I would like to thank some persons who supported me during my studies and this master thesis.

First, I would like to thank Maritta Heisel who was my supervisor for this thesis. I will always remember the long discussions in the evening. Thank you for your support, the valuable feedback and the possibility to continue as a PhD student in your working group.

Second, I would like to thank Rene Meis. He was the supervisor of my bachelor thesis and always took the time for discussions and questions during the master thesis as well. I also thank all my colleagues in the working group for their support and how they welcomed me.

I thank Christina Menges for her proof-readings and all the valuable comments which helped me to improve this thesis and my English.

Last, I would like to thank Ketil Stølen. He was the inventor of CORAS and during a meeting in Duisburg he was open for all discussions and provided a lot of feedback. Together with Maritta Heisel, he had the basic idea of combining CORAS with the problem frames.

Part I.

Foundations

1. Introduction

In this chapter, we give a motivation for our work together with an overview of the following chapters.

1.1. Motivation

Nowadays, there exist numerous security issues for software systems. The Kaspersky Lab provides a statistical overview for 2015 [Kas15]. There is a rating for the following application and application types: Adobe Reader, Microsoft Office, Adobe Flash Player, Java, Android and Browsers. Based on these groups, the number of recorded incidents has been evaluated. The incidents have been recorded by the Kaspersky Company[1], which is a well-known developer of security software. Figure 1.1 shows the distribution of exploits in cyberattacks by the type of application which has been attacked. The above stated categories are compared in this diagram with regard to the number of recorded incidents. Due to the growing sector of cloud applications, the group of applications used in browsers grows fast. 62% of all recorded incidents can be related to this type of application. The mobile operating system Android[2] is related to 14% of the recorded incidents.

This statistic shows that much of the software which has been developed and deployed still contains vulnerabilities, so that the software can be attacked. To fix a vulnerability is very cost intensive [Cor12]. And to provide a solution for the issue, it is necessary to identify the problem first. The first way to do this is to consider the lists of attacks provided by security software companies. The reasons for the recorded attacks can be used to identify vulnerabilities. Some companies also hire attackers to test the software against attacks. This test is called penetration test and is based on the expertise of the attacker. The goal is to identify all vulnerabilities of the software and the related threats.

The step to identify possible ways to attack a software is the starting point for a risk management process. This process aims at making a risk acceptable, where risk is defined as the combination of likelihood and consequence [ISO09]. For an identified risk which is not acceptable, the analysis team tries to find effective treatments to reduce the risk with regard to the costs. In some cases, it is not worth to apply a treatment, because the costs of the treatment are higher than the costs of the consequence without any treatment.

In most cases, a risk management is carried out when the software already has been deployed. Thus, all changes have to be done on an existing software. The later the phase of the software development in which the risk is identified, the higher the costs are to fix the problem. A deployed software has already passed all phases of the process. In this case, all changes in the software have to be spread backwards through all phases, e.g. the functional testing has to be redone and the requirements have to be rearranged. Most of the existing risk analysis technologies assume an existing software, e.g. the CORAS approach developed by SINTEF in Oslo [LSS10], and hence are confronted with this problem.

This thesis aims to provide a risk management for the early phases of the software development. Based on functional requirements and a detailed description of the environment, we propose a structured method that starts with identifying assets and possible threats. In the next step we evaluate the risk to be able to decide if it is acceptable. For non-acceptable risks we specify high-level security requirements together with diagrams which summarize

[1] `http://www.kaspersky.com` - Kaspersky Security
[2] `http://android.google.com` - Android by Google

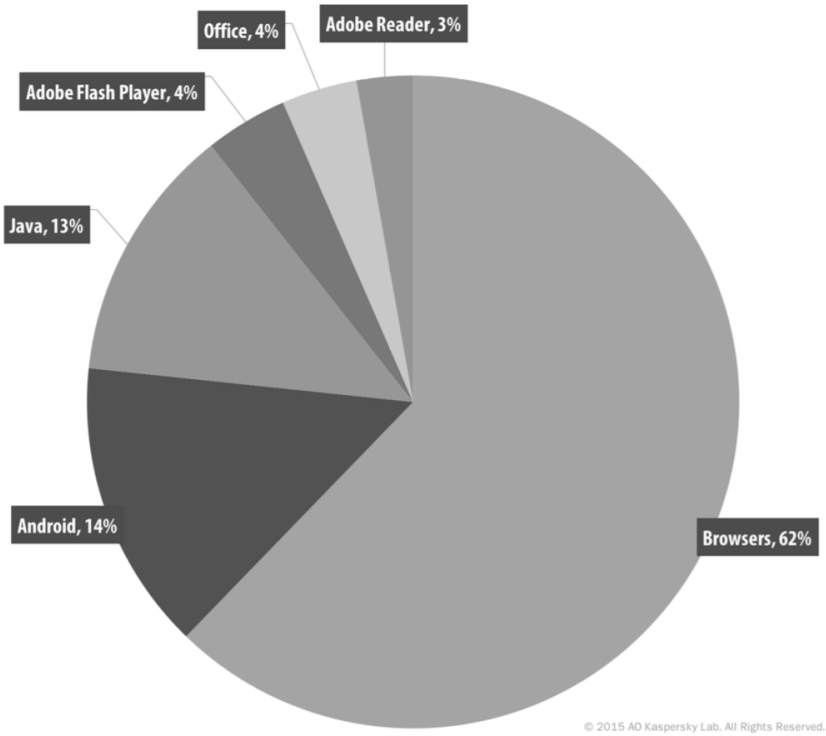


Figure 1.1.: Distribution of exploits used in cyberattacks, by the type of application attacked, 2015 [Kas15]

the reasons for the risk. This information is used to suggest treatments. In a last step, we add the treatments to the initial problem description. The security requirements complement the functional requirements, to describe the functionality on the one hand and the security aspect on the other hand. This reduces the effort to address the risk, because the necessary changes can be traced through all following steps of the development process. The software is then developed not only with regard to functional requirements, but also with regard to security requirements.

For this thesis we initially define the following research questions (RQ) which will be answered in the present document.

RQ1 How can we perform a risk analysis in the early phases of a software development process?

RQ2 How can the initial information of the system be used to perform a sufficient analysis? What additional information has to be elicited?

RQ3 How can existing methods be reused and combined?

RQ4 How can the results of the analysis be used in order to suggest treatments to reduce risk?

RQ5 How can the model be extended with the treatments so that they are available for the other phases of the software development lifecycle?

1.2. Content Overview

This thesis is structured into three parts which are structured by chapters. Here, we provide a short summary for each part and the contained chapters.

I Foundations: This part provides a motivation, background information and the description of a case study which is used in this thesis.

1 Introduction In the current chapter, we give a motivation for our work and describe the structure of the thesis. In the last section we introduce a terminology that is used in the following chapters.

2 Background We provide necessary background information to understand the context of the thesis. This information also includes the explanation of existing technologies on which our work is based.

3 Case Study - Smart Grid To illustrate the theoretical part of the document, we introduce a case study of a smart grid scenario. Together with an informal description, we introduce the functional requirements of the scenario.

II The ProCOR Approach - Security Requirements Elicitation: The main contribution of the thesis is a stepwise method for a model-based risk management which is called ProCOR (**Pro**blem-based **COR**AS). The method consists of seven steps and there is a chapter for each step. It starts with the initial requirements and finishes up with a selection of treatments to reduce the risk for the system to be developed. The case study is used in each chapter to show the result for each step. Last, we provide some initial ideas for a tool support.

III Conclusion: In the last part we give a short summary of our work. There are two chapters contained in this part.

Related Work We state some related work which can be used for example as an input for future work.

Evaluation We conclude the thesis with an overview about the contributions as well as a list of future work.

1.3. Terminology

In this section we provide a definition for the main security properties and the roles of stakeholders that take part in the method.

1.3.1. Security Properties

Security is the protection of the software from the environment. We consider the main security properties Confidentiality, Integrity and Availability (CIA properties) for our analysis as defined in [SJM08].

Confidentiality aims to protect information from being accessed by unauthorized parties. Authorized parties are the owners of the data or third parties that are allowed to access the data, e.g. an employee at the bank needs information about the account to be able to do his/her job.

Integrity describes the correctness of information. Information shall not be altered by unauthorized persons.

Availability means that the information shall be accessible by authorized persons.

1.3.2. Roles

The following stakeholders take part in the analysis. It is possible that several roles are assigned to one person depending on his/her expertise.

Analyst A person who takes part in the risk management process. He/she is familiar with the method and in case of interaction with other stakeholders, he/she structures such a meeting with regard of the results to be obtained.

Customer A customer can be a company or a person. He/she/it employs the analysts to perform a risk management on a given target.

Party Describes the point of view for the analysis, e.g. a person or a company. The party is not necessarily equal to the customer.

Expert An expert is someone who has a very good expertise on a specific topic. His/her knowledge is used to enhance the analysis with more details.

2. Background

In this chapter, we provide some background knowledge that is used in the following chapters. First, we describe the Problem Frames approach [Jac01] and corresponding extensions. Then, we describe a software development process called ADIT [Hat12]. Last, we introduce CORAS [LSS10], a risk management approach that is used in our proposed method.

2.1. Problem Frames

During the development of a software system, the main description of the system to be developed is divided into subproblems. System means a software which shall be developed together with connected devices, e.g. an input interface or sensors. Problem frames are patterns that can be used to describe these subproblems. Each problem frame consists of a set of domains and requirements. Domains describe elements that are contained in the environment of the software to be developed. These patterns are used in early phases of the development process to describe subproblems. Depending on the involved domains and how they are related to the requirement, an appropriate problem frame is chosen and instantiated. In this section we first describe the basics of this approach invented by Michael Jackson [Jac01]. Based on this, we introduce UML4PF [HH10], a UML profile to describe problem frames in UML.

2.1.1. Basics

In the following, we first describe the elements that are contained in the basic problem frame notation. This notation has been extended with additional elements which are presented as well.

2.1.1.1. Domain Types

In the basic problem frame notation, we distinguish five basic types of domains:

Biddable Domain A biddable domain describes an entity with an unpredictable behavior. For example, this can be a person which uses the software.

Lexical Domain A lexical domain represents data in the system.

Causal Domain A causal domain is an element with a predictable behavior, e.g. a technical component.

Connection domain A connection domain is a causal domain which describes a technical component used to connect to other domains, e.g. a webpage.

Machine The machine represents the software to be developed.

In this thesis we make use of one more domain type, introduced by Côté et al. [CHH+08]:

Display domain A display is a special causal domain, representing a technical component to show some kind of information.

2.1.1.2. Phenomena

We distinguish two types of phenomena:

Symbolic phenomenon A symbolic phenomenon represents values, truths and states. The value can be changed by external causation. For example, a symbolic phenomenon represents the data contained on a hard disk.

Causal phenomenon A causal phenomenon is an event, state or role which is directly caused or controlled by some domain. For example, a causal phenomenon can be a command caused by a person to change the data stored on a hard disk.

All phenomena have in common that they are controlled by one domain and observed by at least one other domain.

2.1.1.3. Interfaces

Interfaces are used to express the connection between two domains on which an information flow or an interaction exists. An interface contains several phenomena controlled by one of the two domains and observed by the other.

2.1.1.4. Requirement

A requirement is an optative statement that describes how the environment should behave when the machine is in action. It can refer to an arbitrary number of domains and must constrain at least one domain of the environment. A machine domain is not constrained or referred to because it is developed to fulfill the requirement.

2.1.2. Diagram Types

In the context of problem frames, we distinguish three types of diagrams:

Context diagram A context diagram describes where the problem is located. It contains the domains of the environment which are relevant for the overall problem description together with the relations between the domains expressed as interfaces.

Problem diagram A problem diagram can be an instance of a problem frame and is derived from the context diagram. It contains at least one requirement, a machine that provides the functionality described by the requirement and all domains that are relevant for the requirement. The domains contained in a problem diagram are not restricted to the domains that are contained in the context diagram. For example, a domain to describe a webpage is not necessary to describe where the problem is located, but is necessary to describe what the problem is. This domain will then be added to the problem diagram. Hence, it is also possible to introduce new phenomena and to extend or to restrict the interfaces.

Domain knowledge diagram A domain knowledge diagram describes additional information about the environment that cannot be controlled or observed by the machine and hence is no functional requirement. For example, when a person is talking directly to another person, this cannot be observed by the machine. This interaction between the two domains is not part of the problem but takes place in the environment. A domain knowledge diagram is then used to model the communication between both domains. Domain knowledge diagrams are not part of the Jackson terminology. They have been introduced for UML4PF [HH10]. We distinguish between facts and assumptions as described in [vL09]. Assumptions are conditions to ensure the requirement. Usually, they describe required user behavior. For example, a person does not provide his/her password to third parties. Facts describe fixed properties of the problem environment

which cannot be influenced by the machine, e.g. a door cannot be opened and closed at the same time.

In a domain knowledge diagram, a fact/an assumption refers to and constrains domains that are relevant to describe the domain knowledge.

2.1.3. UML4PF

UML for Problem Frames (UML4PF) [HH10] provides a UML profile to express the diagrams mentioned in the previous section in form of UML class diagrams. This profile consists of stereotypes which can be applied to the classes and associations. For each element of a diagram, a stereotype exists. To describe the type of the diagram, a UML package is used.
To assist the software engineer, OCL conditions have been defined to check the correctness of the diagrams.

In the original UML4PF, the interfaces have been expressed with UML interface classes. Phenomena are expressed as operations in these classes. An interface class is then controlled by one domain and observed by another. For our work, we make use of an alternative representation that has been proposed by Meis and Heisel [MH16a]. Each phenomenon is realized with a separate class with the stereotype ≪`symbolicPhenomenon`≫ or ≪`causalPhenomenon`≫ depending on its type. This stereotype has an attribute *controlledBy* which is linked to the domain that controls the phenomenon. To express the interface between two domains, an association is used with the stereotype ≪`connection`≫ which has an attribute **phenomena**. The value of this attribute is a set of all phenomena that are observed by one of the domains and controlled by the other.

Remark: We do not show examples for diagrams in this section. In Chapter 5 on page 29, we explain how a problem diagram is created using the UML4PF profile. There, we also provide example diagrams for our case study.

2.2. ADIT Software Development Process

Our proposed method is placed in the early phases of a software development process. The ADIT process described by Hatebur [Hat12] is model-based and divided into four main phases: **A**nalysis, **D**esign, **I**mplementation, **T**esting. We briefly describe the main aspects of the different phases and explain which phases are considered in our work.

2.2.1. Analysis

The analysis phase is used to understand the problem and to specify important aspects of the software to be developed. This phase is again divided into six other phases from which we make only use of the first two ones in our work.

A1 Phase A1 contains the problem elicitation and description. The requirements and the domain knowledge are elicited and the context diagram is created.

A2 Phase A2 contains the problem decomposition. The output of this phase is a set of problem diagrams to describe the previously stated requirements. The problem diagrams are classified according to its problem frame.

The method we propose in this thesis is based on the first two phases *A1* and *A2*. We elicit security requirements together with a problem diagram for each of these requirements. In contrast to the previously described problem diagrams, there are no machines in these diagrams but treatments. This output of the method complements the results of phases *A1* and *A2*. Hence, the phases starting with *A3* have to be performed after applying our method.

2.2.2. Design

Based on the analysis, the design decisions for the system are made. The security requirements have to be taken in account. For example, to preserve confidentiality of data, an encryption component is added to the software design. The encryption component corresponds to a treatment. Hence, each treatment has to be considered in this phase.

2.2.3. Implementation

In the implementation phase, the software is expressed as running code based on the analysis and the design decision. The used programming language and techniques depend on the environment and on the needs of the customer.

2.2.4. Testing

The testing phase is used to ensure the correctness of the software and that all requirements are fulfilled. Different test methods are used depending on the requirements and the techniques and languages that are used for the implementation. The security requirements also have to be checked.

As our method is carried out after *A2*, the results of the method have to be considered in all following phases. We ensure that security is considered from the beginning. We do not investigate how the phases starting with *A3* have to be adjusted to apply the results of our method.

2.3. CORAS

CORAS [LSS10] is a risk management process. In this section, we describe the CORAS language and the CORAS method. Extensions of the language and the method will not be part of this work.

2.3.1. CORAS Language

The CORAS language is a graphical semi-formal notation. It consists of different symbols, each representing one element in the analysis. There are different types of diagrams used in the method which is described in Section 2.3.2. The creation of these diagrams is supported by a tool. Based on a meta model, the tool is able to check the syntactical correctness of the diagrams. For non-experts, there are translation rules of the diagrams to natural language.

2.3.1.1. CORAS Symbols

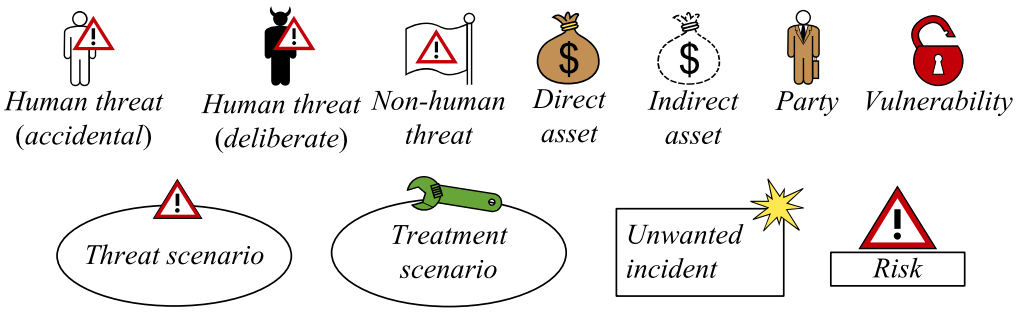


Figure 2.1.: CORAS symbols [LSS10]

Figure 2.1 shows the symbols which are contained in the language. We describe the symbols, starting from the left top symbol.

Threat types

Human threat (accidental) A human-threat accidental is a person which harms the system in an unwanted way, e.g. an employee who deletes some data by pressing the wrong button.

Human threat (deliberate) A human-threat deliberate is a person who performs an attack on the system, e.g. an attacker who breaks into the firewall.

Non-human threat A non-human threat is an issue which harms the system, but is not caused by humans, e.g. a storm destroys a cable.

Asset types

Direct asset A direct asset is some kind of value which needs protection, e.g. some kind of data. Depending on the target of the analysis one has to decide about the items which shall be considered as an asset. The term direct means that the consequences on the asset are easy to measure.

Indirect asset For an indirect asset the consequences are more difficult to measure, e.g. the reputation of a company. These assets are harmed when a direct asset is harmed.

Other elements

Party The role of a party has already been defined. Using this symbol, it is possible to represent the party in a diagram.

Vulnerability A vulnerability is something which makes it possible for a threat to harm a system, e.g. a weak password to protect a database or the absence of power redundancy for a non-human threat.

Threat scenario A threat scenario is the state in which the asset can be harmed, e.g. an attacker breaks the encryption of a wireless connection. A threat scenario can have a likelihood.

Treatment scenario A treatment scenario is a way to reduce the likelihoods or consequences of something which can lead to harming an asset, e.g. defining stronger rules for a password to eliminate the vulnerability of a weak password.

Unwanted incident An unwanted incident is the event which harms the asset, e.g. disclosing the data after breaking the encryption harms the confidentiality of the data. An unwanted incident has a likelihood.

Risk A risk is the combination of a likelihood and a consequence for an asset as defined in ISO 31000:2009 [ISO09]. The unwanted incident has a likelihood and its influence on the asset has a consequence. Hence, a risk is derived from this relation. Different risk levels can be defined, e.g. low, medium and high.

2.3.1.2. CORAS Diagrams

The CORAS language contains different diagram types which are described in the following.

Asset diagram An asset diagram contains direct and indirect assets and their relations. The party in this diagram describes the point of view.

Threat diagram A threat diagram contains threats, threat scenarios, unwanted incidents and direct assets together with their relations.

Risk diagram A risk diagram shows identified risks which are derived from the threat diagrams. In these diagrams, indirect assets are included again.

Treatment diagram A treatment diagram is based on a threat diagram, but possible treatment scenarios are connected to the elements of the original diagram. The connected elements are treated by the treatment scenario.

Treatment overview diagram A treatment overview diagram summarizes all treatment scenarios in one diagram. The treatments are now related to the risk which they address.

Remark: We do not show examples of the diagrams here. For our method we introduce a UML Profile on page 25 to express the CORAS language.

2.3.2. CORAS Method

The CORAS method consists of three main parts which are divided in a total of eight steps. This is shown in Figure 2.2.

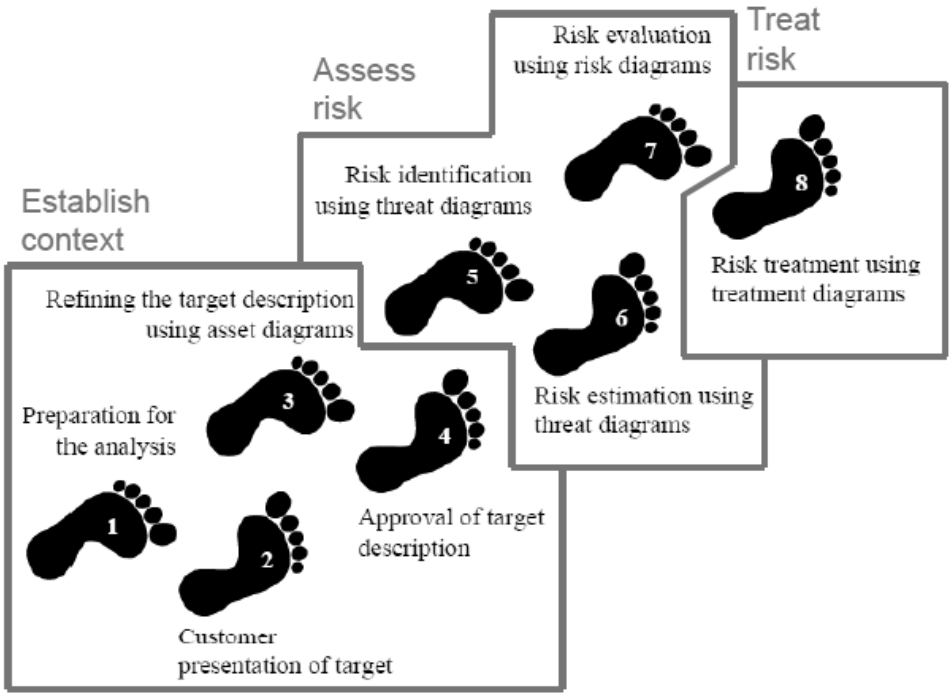


Figure 2.2.: Eight Steps of the CORAS Method [LSS10]

In the following we give a short summary of each step.

2.3.2.1. Step 1: Preparation for the analysis

The purpose of this first step is to get an initial understanding of what shall be analyzed. In a first meeting with the customer, all participants of the analysis agree on the person-hours for the analysis, the initial scope, different responsibilities and contact persons. The different steps of the analysis are presented by the analysts and it is shown which expertise is needed. Based on that, the amount of person-hours for the analysis is decided. At the end of this step, all parties have an initial understanding of how the analysis will work in the next steps.

2.3.2.2. Step 2: Customer presentation of the target

The objective of this step is that the analysts gather information about the item which shall be analyzed and the goals that shall be achieved during the analysis. The following questions have to be answered:

- What shall be analyzed?
- What is the scope? Which parts of the software and its environment shall be considered?
- What assumptions can be made for the system, e.g. existing protection technologies?
- What is the focus of the analysis? What shall be protected?

This meeting mainly consists of the presentation by the customer where the analysts are the audience. The results are then discussed to avoid misunderstandings. To be able to communicate, all participants agree on a common terminology. There are no formal semantics for the representation of the results. They can be documented in form of sketches or in natural language.

At last, all participants agree on the dates for the following meetings, the participants and the delivering of reports.

2.3.2.3. Step 3: Refining target description using asset diagrams

This step is divided in three substeps:

1. Presentation of the target as understood by the analysts: Misunderstandings which have occurred in the previous step are corrected. There are no semantic rules how to document the understanding. It is suggested to use UML diagrams for this as a well known standard.
2. Asset identification: Pinpointing on the most important valuables of the parties. Parties can be the customer but also other stakeholders, like a patient of a hospital where the customer is the board of the hospital. The results are documented in an asset diagram as described in Section 2.3.1.
3. High-Level risk analysis: Get an overview of the main threats and risks based on the asset identification. It helps to identify the most important items within the scope that need to be addressed in the analysis. Normally this is done in a table which answers the following questions:
   - Who/what causes it? - Threats
   - How? What is the incident? What does it harm? - Threat scenario, unwanted incident, asset
   - What makes it possible? - Vulnerabilities

   Remark: The results are only high-level. The detailed analysis is performed in step five of the CORAS method as described in Section 2.3.2.5.

2.3.2.4. Step 4: Approval of target description

In this step, there is again a meeting to agree on the description of the target. It is necessary to define scales to measure the likelihoods and consequences and to rate the assets based on their importance. For each asset, a seperate consequence scale is defined but there is only one likelihood scale.

For each asset a risk evaluation matrix is used to define the risk level. It is necessary to agree on a risk scale. The rows of the matrix are the different values for the likelihood and the columns are the values for the consequence. Figure 2.3 shows an example for a risk matrix. Here, the risk scale only contains the values high and low. The combinations of likelihood and consequence that are rated as a high risk are colored red in the matrix and the ones with a low risk are colored green. It is necessary to define which risk levels are unacceptable. Unacceptable risk means that it is necessary to consider this for the treatment selection to make the risk acceptable. In the provided risk matrix the high risk is considered as unacceptable whereas the low risk is considered as acceptable. There is no further investigation for the acceptable risk.

		Consequence				
		Insignificant	*Minor*	*Moderate*	*Major*	*Catastrophic*
Frequency	*Rare*					
	Unlikely					
	Possible					
	Likely					
	Certain					

Figure 2.3.: Risk Matrix Example [LSS10]

The results of this step serve as a contract for the analysis. This information is the starting point of the analysis and describes the conditions under which the analysis takes place.

2.3.2.5. Step 5: Risk identification using threat diagrams

Step 5 is organized as a meeting with a structured brainstorming. The objective is to identify threats, threat scenarios and unwanted incidents which harm the assets that have been identified in the previous step.

In the meeting, experts with different backgrounds come together. It is structured in the sense that the analysis team asks questions to the experts and guides them through the identification process. The results are captured on the fly using threat diagrams. These diagrams are displayed to the participants so that they can make remarks. The analysis leader must already have experience with the brainstorming technique to be able to lead the discussion to a result. It is useful to have checklists for a specific domain and to have knowledge about vulnerabilities and possible threats.

As a starting point for the brainstorming session, the analysts create threat diagrams based on the high-level analysis and the identified assets. The way how a threat diagram is structured and grouped depends on the domain. For example, it is possible to have one diagram per threat type. An initial set of unwanted incidents is created to have a minimal threat diagram. These threat diagrams are extended during the brainstorming session.

The results of this step have to be documented in a concise and understandable manner, because in further steps there are participants with different backgrounds. The CORAS language provides support for natural language and corresponding rules, so that these diagrams can easily be translated into natural language.

The annotations for likelihood and consequences will be added in the next step.

2.3.2.6. Step 6: Risk estimation using threat diagrams

In this step, the threat diagrams are augmented with corresponding values of likelihood and consequences. As risk is defined as the combination of likelihood and consequence, where the main focus lies on the relation between unwanted incident and asset. The unwanted incident has a consequence on the asset and is annotated on the relation between them. The likelihood of the unwanted incident is annotated on the element itself. In the case that the likelihood of the incident cannot be determined, it is possible to estimate the likelihoods of the threat scenarios which lead to this unwanted incident. Then the likelihood of the incident is derived from this estimation. The scales defined in step 4 as described in Section 2.3.2.4 provide the values to be attached to the elements within the threat diagrams.

The values are estimated with help of the expert knowledge. The experts have different backgrounds and have an expertise from past projects. It is also possible that external resources are used, for example statistics about the most recent threats.

In this step, no new elements will be added to the diagrams. The existing ones are only annotated.

2.3.2.7. Step 7: Risk evaluation using risk diagrams

The objective of this step is to identify risks that must be considered for possible treatments. In this step, the indirect assets are considered again. A consequence scale has to be defined together with the risk evaluation criteria in form of a risk matrix.

For a direct asset, the unwanted incident together with the asset is considered. The likelihood of the unwanted incident and corresponding consequence is inserted in the correct field of the risk evaluation matrix. The risk level can then be retrieved from the matrix. Based on the definition of the risk level one has to decide whether it is an acceptable or unacceptable risk.

The identified risks and corresponding assets are represented in risk diagrams. A risk diagram contains the asset, the risk and the threat which leads to the unwanted incident. The risk itself is named as the unwanted incident starting with a unique prefix as an abbreviation. The indirect assets are considered again and the consequences of the risk on these are estimated.

Using the information of the risk matrix, the risk level is annotated on the risk objects in the diagrams. All risks with a risk level that has been defined as unacceptable are considered for the treatment selection in the next step.

2.3.2.8. Step 8: Risk treatment using treatment diagrams

Based on the risk evaluation, the objective of this step is to reduce the risk by choosing treatments with respect to cost-benefit. The results of the treatment selection are represented in a treatment diagram. Such a diagram is based on a threat diagram. It contains all elements of the threat diagram which lead to an unwanted incident that is related to an unacceptable risk. The unwanted incident is represented by the corresponding risk element that has been created in step 7. Following the paths of the diagram, it is possible to identify the causes of the risk. These causes need treatment to reduce the risk. The treatment scenarios are then added to the diagram and linked to the element that is addressed by it.

As a final result, there is a treatment overview diagram. It contains all risks that have been identified and the threats that cause the risks. All treatment scenarios are contained in this diagram and are linked to the risk which is reduced by them.

3. Case Study - Smart Grid

In this chapter we introduce a case study which is used as an example in the next chapters to explain and validate our work.

3.1. Description

Our case study is a smart grid system inspired by the OPEN meter [OPE09]. The description is provided by the OPEN meter Consortium which has been funded by the European Commission and provides a report on the specification of a multi-metering infrastructure. The original specification has been modified in some ways to fit for our needs. Figure 3.1 shows an informal description of the smart grid scenario. A smart grid is an intelligent and connected power supply network, in which different participants are able to interact and control the grid. For example, it is possible to retrieve the measurements of the power consumption remotely. In the following, we describe the main components of the scenario.

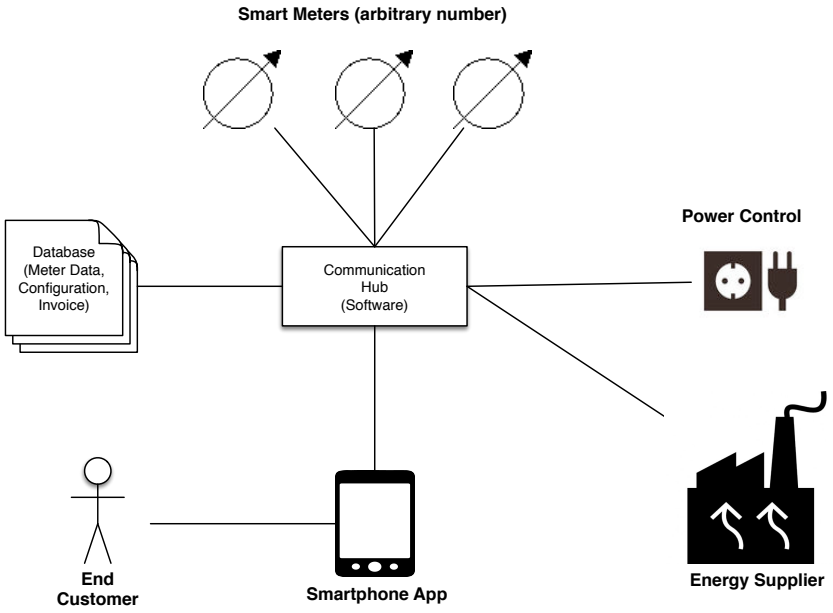


Figure 3.1.: Informal Description of the Case Study

The software to be developed is the *Communication Hub*. It serves as the connection between all other components and actors and is used to do some calculation, e.g. to provide an invoice.

The *Database* contains the data that is used by the software. It contains the data that has been measured (meter data), the configuration of the software (login data, personal data of the end customer and tariff parameters) and the invoices that have been calculated based on the meter data.

There are some *Smart Meters* measuring the consumption of energy with sensors. They are connected to the Communication Hub using the wireless network.

The *Power Control* is a technical device which enables the energy supplier to cut off the power of the end customer. The reasons for cutting off the power can be an invoice that has not been paid.

The *Energy Supplier* is the provider of the smart grid and is considered as a company. It is able to do an initial setup locally on the communication hub. After the initial setup, there is a remote connection to the communication hub via the wide area network (WAN). This connection is used by the energy supplier for further interaction with the communication hub.

The *End Customer* is the one who has ordered the communication hub. He/she is paying the energy supplier's invoices.

There is a *Smartphone App* that is connected to the communication hub using the wireless network. With the app the end customer is able to interact with the software and the software is able to display some information such as an invoice.

3.2. Requirements

The original specification of the OPEN meter scenario suggests different sets of requirements. To apply our method, we derived eight functional requirements. These eight requirements are stated in the following. For our method, we assume that the requirement analysis for functional requirements has already taken place.

R1 The energy supplier can perform an initial setup. This is done locally on the communication hub by providing initial information.

R2 After the initial setup or after turning on the communication hub it establishes the connection with the energy supplier. This request is confirmed.

R3 In given intervals, a smart meter transmits the measured data to the communication hub. The data is then stored.

R4 The energy supplier can request all meter data. This meter data consists of the different measurements of the smart meters. It is possible to set a timestamp from which on the meter data is provided.

R5 In given intervals, an invoice based on the current meter data is created.

R6 The energy supplier can enable or disable the power of the customer, e.g. when an invoice is not paid.

R7 When a new invoice has been created, it is shown to the end customer using the smartphone app.

R8 An end customer can change the personal data using the smartphone app.

3.3. Workflow

To express the relations between the requirements and the workflow of the system, we make use of life-cycle expressions as proposed in step A6 of the ADIT software development process [Hat12].

In the following, we first introduce the alphabet and the operators of these expressions. In the original process life-cycles are used to express the relations between sequence diagrams. Here, the expressions are replaced by the names of the requirements.

The following operators are used to connect the requirements. If `x` and `y` are life-cycle expressions then so are

`x;y`	`x` is followed by `y`
`x | y`	either `x` or `y`
x^*	`x` is executed 0 or more times
x^+	`x` is executed at least once
`[x]`	`x` is optional
`x || y`	`x` and `y` are executed concurrently

Operator precedences: `[]`, *, $^+$, `;`, `|`, `||`

Using this notation, we can derive the following life-cycle where $LC_{CommunicationHub}$ stands for the life-cycle of the communication hub:

$LC_{CommunicationHub} = R1;\ (R2;\ (R3 \mid R4 \mid R5 \mid R6 \mid R7 \mid R8)^*)^*$

First of all, it is necessary to do the initial setup. Afterwards, the communication hub establishes the connection to the energy supplier. Using this connection, the requirements R3 to R8 can be repeated an arbitrary number of times and in any order. We do not consider a function to shutdown the communication hub. It is turned off using some kind of hardware switch. When switching on the communication hub again, the workflow starts at requirement R2 to establish the connection.

Part II.

The ProCOR Approach - Security Requirements Elicitation

4. Basics

4.1. Overview

Our method is based on the CORAS method which has been introduced in Section 2.3.2. In contrast to the original CORAS method, we do not assume an existing system. Our method is based on the problem frame approach. We aim to address security issues in the early phases of the software development process. Thus, we propose a risk management approach in the analysis phase. We call this method ProCOR, which stands for ***Pro****blem-based* ***COR****AS*. Figure 4.1 shows an overview of the seven steps of the method together with the corresponding inputs and outputs. The stakeholders that participate in the different steps are annotated as actors. As the analysts perform the method, they take part in each step and are not stated explicitly.

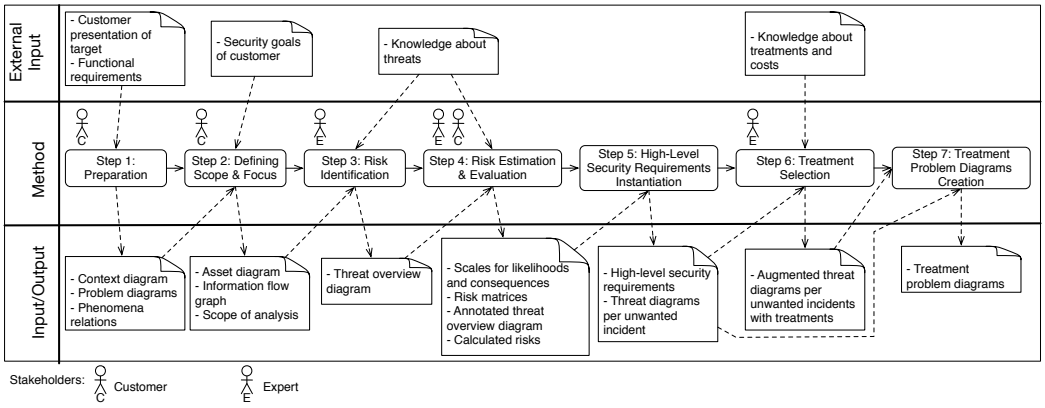


Figure 4.1.: Overview ProCOR Method

Step 1: Preparation In the first step, we create the initial UML model based on the customer description and the functional requirements. It contains a context diagram and a set of problem diagrams. The relations between the different phenomena contained in the problem diagrams are expressed in one domain knowledge diagram.

Step 2: Defining Focus & Scope The assets are defined based on the security goals of the customer and the symbolic phenomena. As symbolic phenomena represent some kind of data, they are considered as values with regard to confidentiality, integrity and availability. These phenomena related to an asset together with the problem diagrams are used to identify domains that process the data. A graph is created that shows the domains where the symbolic phenomena are available together with the flow of information between these domains. Based on the set of domains that are contained in the graph, the scope of the analysis is defined.

Step 3: Risk Identification For those domains that are in the scope of the analysis, possible threats, threat scenarios and unwanted incidents are identified. The information about the threats is considered as domain knowledge and is represented with domain knowledge diagrams. Using a CORAS UML Profile that has been developed in this thesis, a threat overview diagram that contains all assets and the identified elements is created.

Step 4: Risk Estimation & Evaluation In this step likelihoods and consequences are added to the threat overview diagram and a risk matrix for each asset is defined. Based on these risk matrices, it is evaluated whether the risk is acceptable or not. An unacceptable risk means that the likelihood of the unwanted incident that harms an asset is too high and needs to be reduced.

Step 5: High-Level Security Requirements Instantiation For the unwanted incidents with an unacceptable likelihood, we set up high-level security requirements together with a threat diagram. This threat diagram is an extract of the threat overview diagram and only contains the threats and threat scenarios that lead to the unwanted incident. Additionally, the assets that are harmed by the unwanted incident are added to the diagram. Hence, we have one high-level security requirement and one threat diagram per unwanted incident with an unacceptable likelihood. These requirements have to be fulfilled to reduce all risks to be acceptable.

Step 6: Treatment Selection To reduce the likelihood of an unwanted incident to be acceptable, it is necessary to select appropriate treatments. These treatments are expressed as treatment scenarios and are added to the threat diagrams of the unwanted incidents. It is annotated which domain is treated and which likelihood is addressed. For each treatment scenario, there is a likelihood reduction. After the treatment selection, the likelihoods are estimated again with regard to the likelihood reduction. The likelihood of the unwanted incidents should then be acceptable.

Step 7: Treatment Problem Diagrams Creation For each high-level security requirement, a treatment problem diagram is created. All domains that are related the corresponding unwanted incident are contained in the diagram. For each treatment scenario, a treatment domain is defined. This treatment is connected to the domain that is treated. The requirement refers to the domains without a treatment and constrains the domains that are treated. The treatment domains need to be applied to fulfill the requirement. Hence, there is a treatment problem diagram for each high-level security requirement.

4.2. Step Description

Each step is described in a table like Table 4.1. This table consists of the in- and outputs of the method, the related CORAS steps and validation conditions. To ensure that errors by applying the method are detected as early as possible, the validation conditions have to be checked after each step. The table is inspired by the Agenda concept introduced by Heisel [Hei98].

Step x: Name of Step	
CORAS Steps	Related steps of the original CORAS method.
Input	Inputs that are needed to perform the step.
Output	Outputs that are produced by this step.
Validation Conditions	Conditions to detect errors as early as possible.

Table 4.1.: ProCOR: Step Description

4.3. UML Profile for CORAS

To be able to combine the CORAS language with UML4PF, we developed a UML Profile expressing the CORAS language. This profile is based on the CORAS meta model [LSS10]. To express the elements of the CORAS language, we make use of stereotypes. In the original CORAS language, there are no connections to the problem frame approach. To be able to combine these approaches, we add attributes to the stereotypes.

Figure 4.2 shows the CORAS profile. The profile *Core* indicates only those elements that are part of the original CORAS language. To be able to make use of UML4PF, we extended the stereotypes with attributes to establish links between UML4PF and CORAS. In the following sections, we introduce extensions of the profile which are then not part of the *Core* profile.

The following profiles are all contained in the *Core* profile and are used to categorize the elements of the CORAS language.

4.3.1. Diagram

The profile *Diagram* describes the different diagram types of the CORAS language. The abstract stereotype *CORASDiagram* extends the meta class *Package*. Hence, all elements of a diagram are contained in a package.

There are four different diagram types that specialize the stereotype *CORASDiagram*. We distinguish the stereotypes *TreatmentDiagram*, *RiskDiagram*, *ThreatDiagram* and *AssetDiagram*. The last one has the attribute *party* of type String. This attribute is used to describe the point of view on the assets.

4.3.2. Asset

The profile *Asset* defines the stereotypes to express assets. There is a stereotype *DirectAsset* and a stereotype *IndirectAsset*. Both stereotypes extend the meta class *Class* and have an attribute *description* of type String to describe the asset.

4.3.3. ThreatDescription

The profile *ThreatDescription* defines stereotypes to describe threats. The abstract stereotype *Threat* extends the meta class *Class*. Hence, threats are represented as classes. The stereotype *NonHumanThreat* is a specialization of the stereotype *Threat* and contains an attribute *domain* of type Domain. This attribute is used to link the threat to the corresponding domain of the problem or domain knowledge diagrams. The abstract stereotype *HumanThreat* is a specialization of the stereotype *Threat* and has again two specializations called *HumanThreatAccidental* with the attribute *person*, and *HumanThreatDeliberate* with the attribute *attacker*. Both attributes are of type BiddableDomain and are used to link the corresponding domain in the model.

4.3.4. DataTypes

The profile *DataTypes* defines datatypes that are used in the profile. The datatype *Consequence* is used to express a consequence and the datatype *Likelihood* is used to express likelihoods. To be able to make use of qualitative and quantitative values, there are two attributes for each stereotype. The attribute *quantitative* is of type Real and the attribute *qualitative* is of type String.

4.3.5. Element

The profile *Element* describes stereotypes for all elements of a CORAS diagram. The abstract stereotype *Element* has the attribute *description* of type String and extends the meta

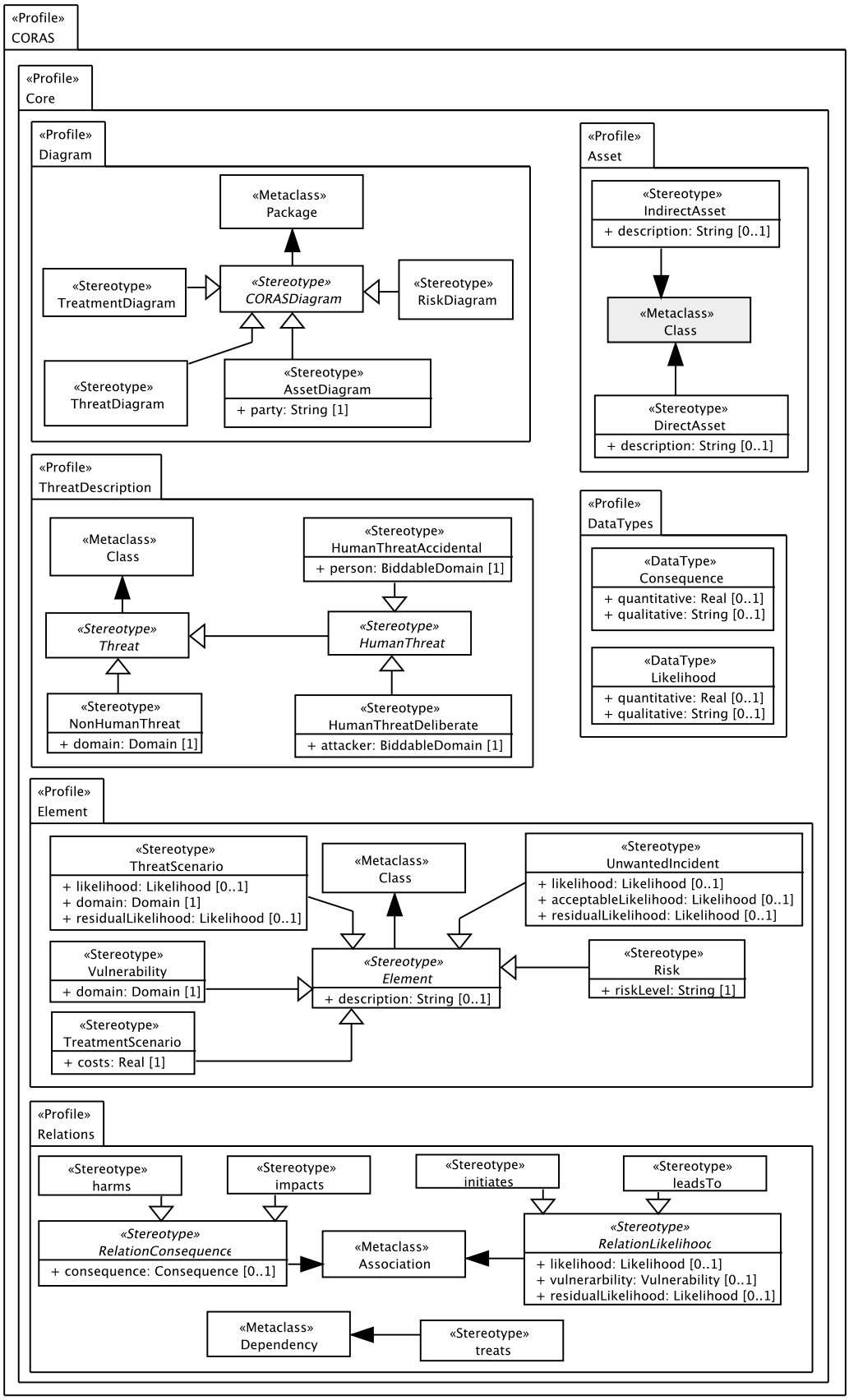


Figure 4.2.: UML Profile for CORAS

class *Class*. Hence, all elements are represented as classes. There are several specializations of the stereotype *Element*. The stereotype *ThreatScenario* has three attributes. The attribute *likelihood* is of type Likelihood and expresses the likelihood without any treatments. The attribute *domain* is of type Domain and is used to annotate the domain where the threat scenario takes place. After applying a treatment, the residual likelihood is calculated and stored in the attribute *residualLikelihood*.

The stereotype *UnwantedIncident* has an attribute *acceptableLikelihood* of type Likelihood to express the likelihood that is acceptable for this unwanted incident. There is also an attribute *likelihood* to express the likelihood without treatments and an attribute *residualLikelihood* to express the likelihood after the treatment selection.

The stereotype *Vulnerability* has the attribute *domain* of type Domain to state the domain where the vulnerability exists.

To express a treatment scenario, the stereotype *Treatment Scenario* is used. It has the attribute *costs* of type Real which makes it possible to annotate the costs for this treatment.

The stereotype *Risk* is used to express a risk element with an UML class. There is one attribute *riskLevel* of type String. The type String enables the user of the profile to use an arbitrary scale for the risk.

4.3.6. Relations

The profile *Relations* defines all stereotypes to express relations between the elements of a diagram. There are two abstract stereotypes called *RelationConsequence* and *RelationLikelihood* which extends the meta class *Association*. Hence, these relations are expressed with an association in a diagram. The stereotype *RelationConsequence* has the attribute *consequence* of type Consequence. There are two specializations of this stereotype, called *harms* and *impacts*. The stereotype *RelationLikelihood* has three attributes. The attributes *likelihood* and *residualLikelihood* are of type Likelihood and are used to annotate the likelihood without any treatment and the residual likelihood after applying a treatment. The attribute *vulnerability* is of type Vulnerability. It is used to annotate the vulnerability that enables this relation. There are two specializations of the abstract stereotype *RelationLikelihood*, called *leadsTo* and *initiates*.

The stereotype *treats* is an extension of the meta class *Dependency*. A treats relation is used to express which element is treated by a treatment scenario.

In the following, we make use of this profile to express CORAS diagrams in our approach.

5. Step 1: Preparation

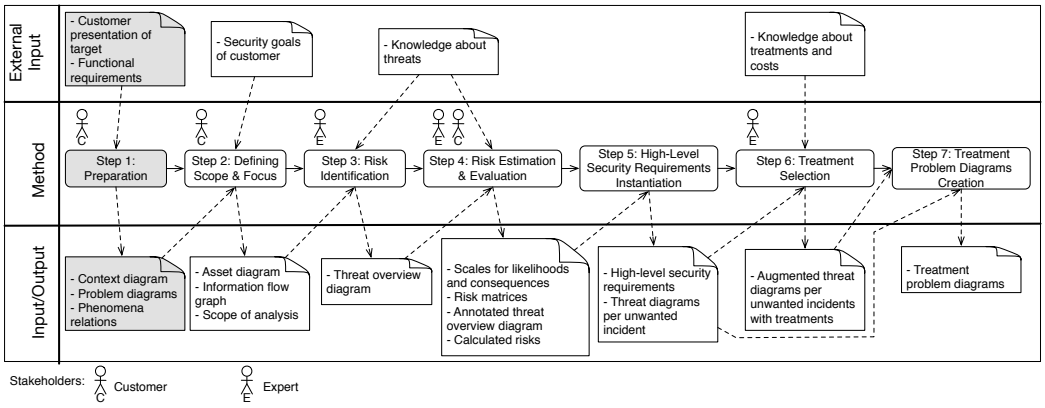


Figure 5.1.: ProCOR - Step 1

In a first step the customer presents the software to be built to the analysts. Based on the description, a first model is created. At this point, we assume that the functional requirements already exist. Table 5.1 presents an overview of this step. We describe the procedure in a more detailed way in the following subsections.

Step 1: Preparation	
CORAS Steps	Steps 1 and 2
Input	- Customer presentation of the target - Functional requirements
Output	- Context diagram - Set of problem diagrams - Phenomena relations in a domain knowledge diagram
Validation Conditions	- There exists exactly one context diagram. - Each functional requirement is covered in at least one problem diagram. - The customer has agreed on the initial model. - All phenomena that are contained in the problem diagrams are contained in the domain knowledge diagram about phenomena relations.

Table 5.1.: ProCOR - Step 1

5.1. Customer Presentation

In a first meeting, the customer and the analysts come together. The customer presents the context of the software together with a list of functional requirements. It is also important to know what kind of information is processed by the software. If this cannot be derived from the requirements, it is necessary that the analysts ask the customer for more details.

During this meeting, the following questions have to be answered:

1. What stakeholders interact with the software? ⇒ List of biddable domains.
2. What technical equipment is used, e.g. input devices or sensors? ⇒ List of causal domains.
3. What kind of information is processed by the software? ⇒ List of lexical domains and symbolic phenomena.
4. What information flow exists between the domains? ⇒ List of causal phenomena and interfaces between the domains.
5. How is the information flow realized from a technical view? For example, what types of networks are used? ⇒ List of connection domains.

Companies can be considered as a biddable domain or a causal domain. A company can have an unpredictable behavior, so it classified as a biddable domain. When only using provided and automatic services of the company, e.g. an automatic checkout process, one can consider the company as a causal domain. In the case of a causal domain, only the technical view on the company is considered.

5.2. Initial UML Model

Based on the description of the customer, the analysts create an initial UML model for the software to be built. This model consists of a context diagram to describe all related domains and a set of problem diagrams, each of which describes a subproblem. We make use of the steps *A1* and *A2* of the ADIT process [Hat12]. The diagrams are created using UML4PF as described in Section 2.1.3 to perform this step.

To ensure that the target of the analysis has been understood correctly, the initial model is presented to the customer. We assume that the customer is able to understand UML, so it might be necessary to consider a software specialist of the company. Improvements and comments are applied immediately on the model. From now on, changes on the initial model shall be avoided because this results in inconsistencies in all other steps. If improvements are unavoidable, they have to be propagated to all parts of the model.

5.3. Adjust Problem Diagrams

The set of initial problem diagrams describes the functional requirements that have been provided by the customer. Problem diagrams allow some freedom concerning their construction. To perform a sufficient risk analysis with regard to security, it is necessary to identify all domains that are processing data. Connection domains are often left out, because they are no essential part of the functional requirement. However, these domains are often used for an attack or are often a point of failure, e.g. a denial of service attack on the Internet or an attacker who sniffs on this connection.

To address this problem, it is necessary to add missing domains to the initial problem diagrams. Faßbender et al. [FHM14] propose a way to discover all connection domains. The connection domain discovery is achieved with a questionnaire. The results are captured in a table.

As a minimal example, we consider the problem diagram shown in Figure 5.2. A person wants to store some data. To discover all connection domains, we now investigate each connection between two domains and answer the following questions:

- Information transmitting domain involved? Name? ($Q1$)
- Domain displaying information involved? Name? ($Q2$)

- Domain storing information involved? Name? ($Q3$)

From the scenario description of the minimal example we conclude that the person connects to the software using the Wide Area Network (WAN). The results of the questionnaire are shown in Table 5.2. We discover one new domain, called WAN, that needs to be added between *Software* and *Person*. It is needed to add a new interface and new phenomena that are forwarded by the newly introduced connection domain. The new phenomena are controlled by the connection domain and are called *fX*, where *f* stands for forward and *X* stands for the original name of the phenomenon. In Figure 5.3, we show the adjusted problem diagram for the minimal example.

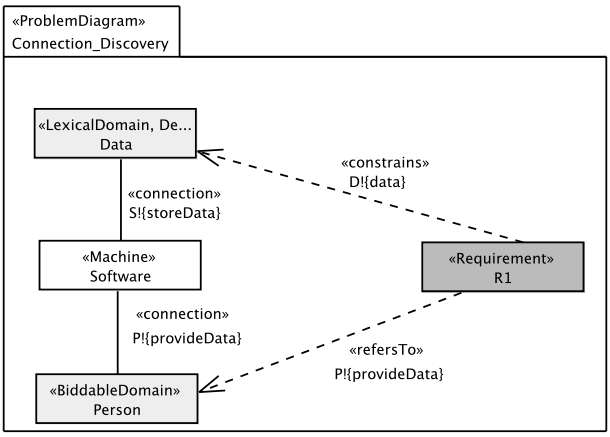


Figure 5.2.: Minimal Example for Connection Domain Discovery

Domain 1	Domain 2	$Q1$	Name?	$Q2$	Name?	$Q3$	Name	PD
Person	Software	yes	WAN	no	-	no	-	Connection_Discovery
Software	Data	no	-	no	-	no	-	Connection_Discovery

Table 5.2.: Connection Domain Discovery Table - Minimal Example

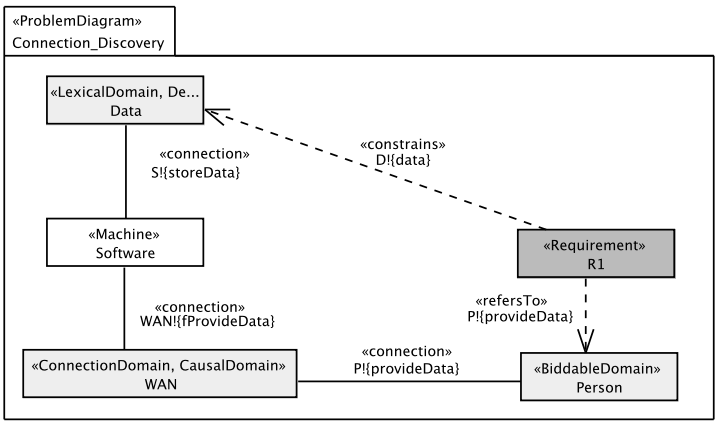


Figure 5.3.: Minimal Example for Connection Domain Discovery Adjusted

5.4. Phenomena Relations

Depending on the complexity of the system and the amount of information that is processed by the software, the number of phenomena contained in the initial model might be high. There exist phenomena which are composed of other phenomena. For example, the command to send an e-mail consists of the sender, the receiver and the text. To be able to reuse the information of the sender and receiver, they are modeled with separate UML classes. To express that these phenomena are contained in the mail command, we model the relations in a domain knowledge diagram.

During the following steps of the method, only the problem diagrams of the model will be considered. Hence, only the relations of the phenomena contained in the problem diagrams are modeled. All phenomena are contained in a package with the stereotype ≪`DomainKnowledge`≫ to make clear that this is additional knowledge. This package is called *PhenomenaRelations* to be easily identifiable in the model.

For each phenomenon, one has to decide what symbolic phenomena are contained in it. If a symbolic phenomenon is not already contained in the model, it is added. A causal phenomenon cannot be contained in another causal phenomenon, but a symbolic phenomenon can be contained in another symbolic phenomenon.

To express the relation that a phenomenon is contained in another, an aggregation is used together with the stereotype ≪`contains`≫ between the phenomena. This stereotype has been proposed by Meis and Heisel [MH16b].

Finally, all phenomena contained in at least one problem diagram and the newly introduced ones are part of the domain knowledge diagram together with their relations. As the selection of assets in the next step will be based on the information represented by symbolic phenomena, this step needs to be carried out carefully.

In Figure 5.4 we show a minimal example with three causal phenomena and two symbolic phenomena. The causal phenomenon *storeData* contains the data and the customer ID. To read the data, it is necessary to provide the corresponding customer ID. The answer of this request *provideData* contains the data.

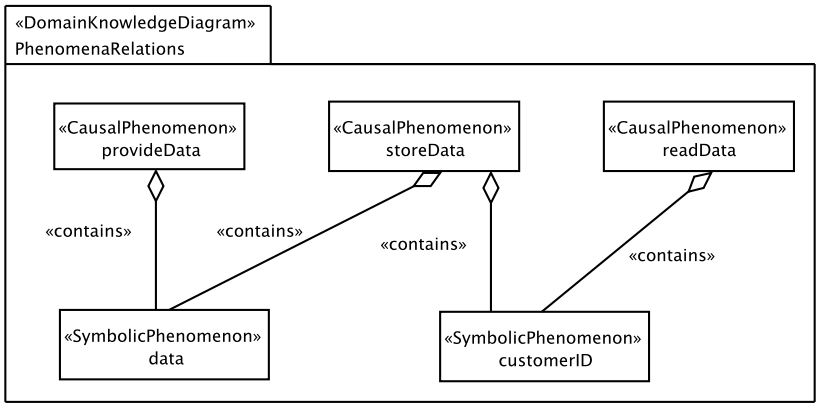


Figure 5.4.: Phenomena Relations - Minimal Example

5.5. Validation Conditions

There are four validation conditions for this step.

1. There exists exactly one context diagram.

To describe the context, there is only one context diagram in the initial model. This is necessary to preserve consistency with the informal description of the scenario. Based on this context model, the problem diagrams are created.

2. Each functional requirement is covered in at least one problem diagram.

   The set of problem diagrams is the starting point for risk analysis. To perform a sufficient analysis, it is necessary to have a complete set of problem diagrams. Complete means that each functional requirement is covered by at least one problem diagram. Normally, each requirement is covered by exactly one problem diagram.

3. The customer has agreed on the initial model.

   The model is presented to the customer and he/she has to agree on it. The model is part of the documentation and with the agreement of the customer, the analysts make sure not to be made responsible if the model later does not meet the customer's demands.

4. All phenomena that are contained in the problem diagrams are contained in the domain knowledge diagram about phenomena relations.

   Based on the relations between the phenomena, the selection of the assets is performed. Additionally, the flow of information is analyzed with the relations. Hence, it is necessary to have one domain knowledge diagram containing all phenomena and the corresponding relations.

5.6. Case Study

The description of the case study presented in Chapter 3.2 on page 18 is considered as the customer presentation that is normally provided in the first meeting. The customer is an energy supplier.

5.6.1. Context Diagram

Based on the description, the analysts create the context diagram which is shown in Figure 5.5. The software to be developed is called *CommunicationHub* and is represented as a machine domain. There are three lexical domains. The first one is called *Invoice*. It represents all invoices that are calculated. The domain *MeterData* represents the consumption of energy that has been measured. The last lexical domain is called *Configuration*. It represents the information about the customer, the tariff parameters and the login data to connect with the energy supplier. The causal domain *PowerControl* is technical equipment to cutoff the power of the customer. The domain *SmartMeter* is again a causal domain representing the meters to measure the energy consumption. The company of the energy supplier is modeled with a biddable domain called *EnergySupplier*. As the interaction with the company might be initiated by humans, the company has an unpredictable behavior. The customer of the company is called *EndCustomer*. He/she is modeled as biddable domain and is connected to the software via the connection domain *Smartphone*. As the context diagram is only used as a description of the environment, the interfaces are very abstract. Depending on the needs for the requirements, they are refined in the problem diagrams.

5.6.2. Problem Diagrams

The problem diagrams are created based on the functional requirements that have been stated on page 3. In the following, we describe each problem diagram. The corresponding textual description of the requirement is provided in the diagram as a comment which is linked to the requirement.

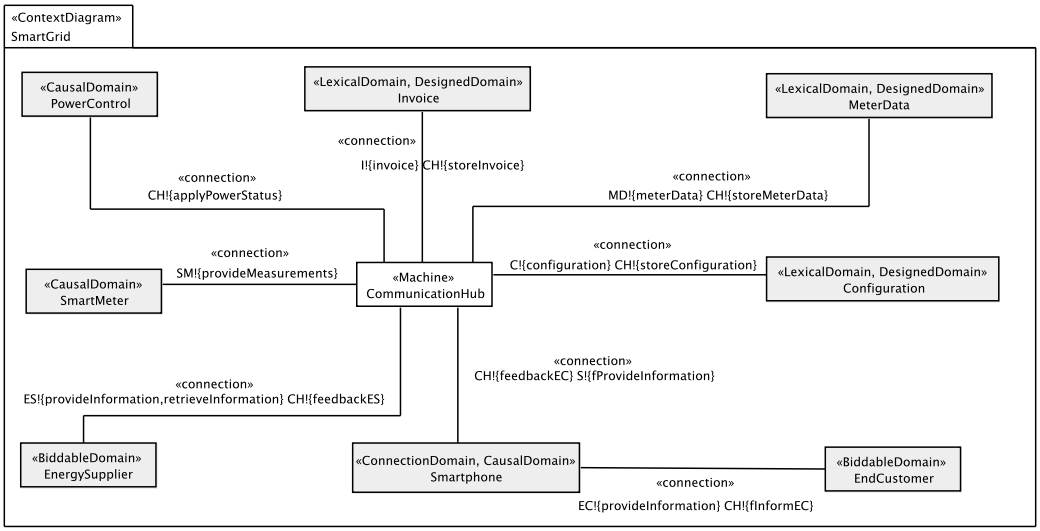


Figure 5.5.: Case Study: Context Diagram

5.6.2.1. Requirement 1: Setup

Figure 5.6 shows the problem for the requirement *R1: Setup*. It contains three domains. The biddable domain *EnergySupplier* is referred by the requirement. This domain is connected to the machine *CommunicationHub*. Between these domains, there is an interface which describes that the energy supplier provides some initial information to the machine. The lexical domain *Configuration* is constrained by the machine, because the initial information is stored in the configuration.

As a biddable domain is referred to and a lexical domain is constrained, this problem diagram fits to the problem frame **Simple Workpiece** [Jac01].

5.6.2.2. Requirement 2: Establish Connection

Figure 5.7 shows the problem diagram for the requirement *R2: Establish Connection*. The diagram contains four domains. To establish the connection, the machine *CommunicationHub* needs the login data that is available at the lexical domain *Configuration*. Hence, the requirement refers to the domain *Configuration*. There is a connection domain called *WAN (Wide Area Network)* that is used to represent the remote connection between machine and energy supplier. The energy supplier is described as a biddable domain called *EnergySupplier*. Between the domain *CommunicationHub*, *WAN* and *EnergySupplier*, the interfaces describe the following flow of information. The communication hub transmits the login data to the energy supplier using the wide area network. The energy supplier replies with a confirmation on the same way. The domain *WAN* is constrained by the requirement and the domain *EnergySupplier* is referred.

As a biddable domain and a lexical domain are referred to, a connection domain is constrained and the operation is initiated by the machine, this problem diagram fits to the problem frame **commanded data-based control** [CHH+08].

5.6.2.3. Requirement 3: Measuring

Figure 5.8 shows the problem diagram for the requirement *R3: Measuring*. It contains four domains. The causal domain *SmartMeter* initiates the operation. It is referred by the requirement and there is an interface which is connected to the connection domain *WirelessNetwork*. This connection domain is connected to the machine *CommunicationHub*.

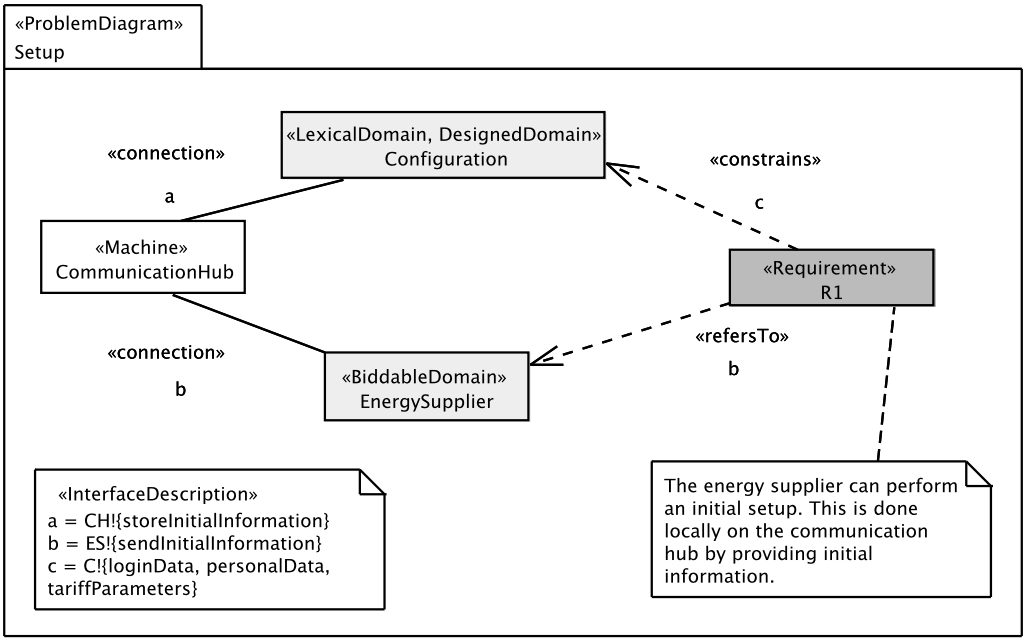


Figure 5.6.: Case Study: Requirement 1 - Setup

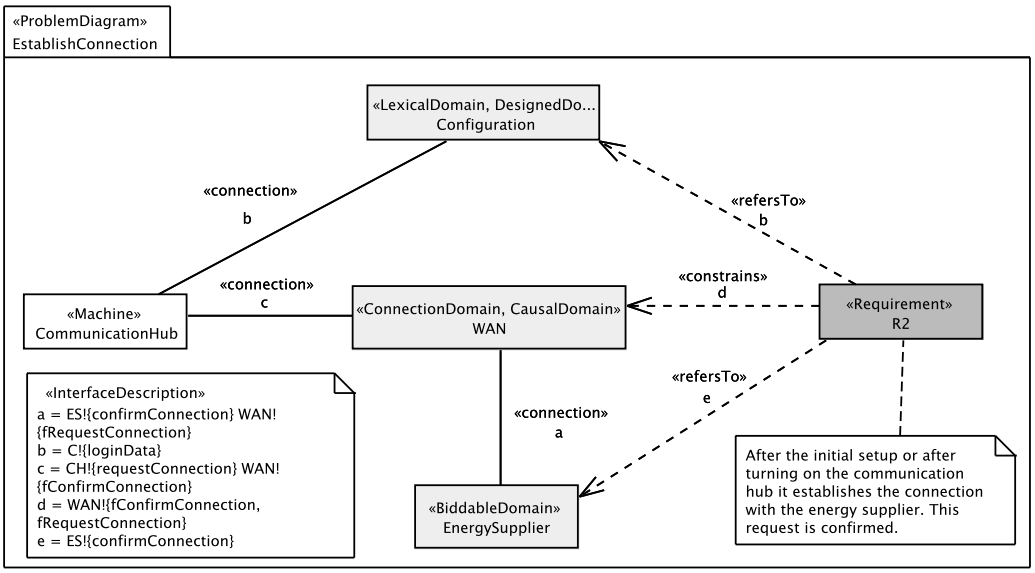


Figure 5.7.: Case Study: Requirement 2 - Establish Connection

This means that the data measured by the domain *SmartMeter* is forwarded to the machine via the connection domain *WirelessNetwork*. The connection domain is constrained by the requirement. The machine then forwards the data to the lexical domain *MeterData* where all measured data is stored. The lexical domain is also constrained.

As the causal domain is referred to and the lexical and connection domain are constrained, this problem diagram fits to the problem frame **model-building + required behavior** [CHH+08].

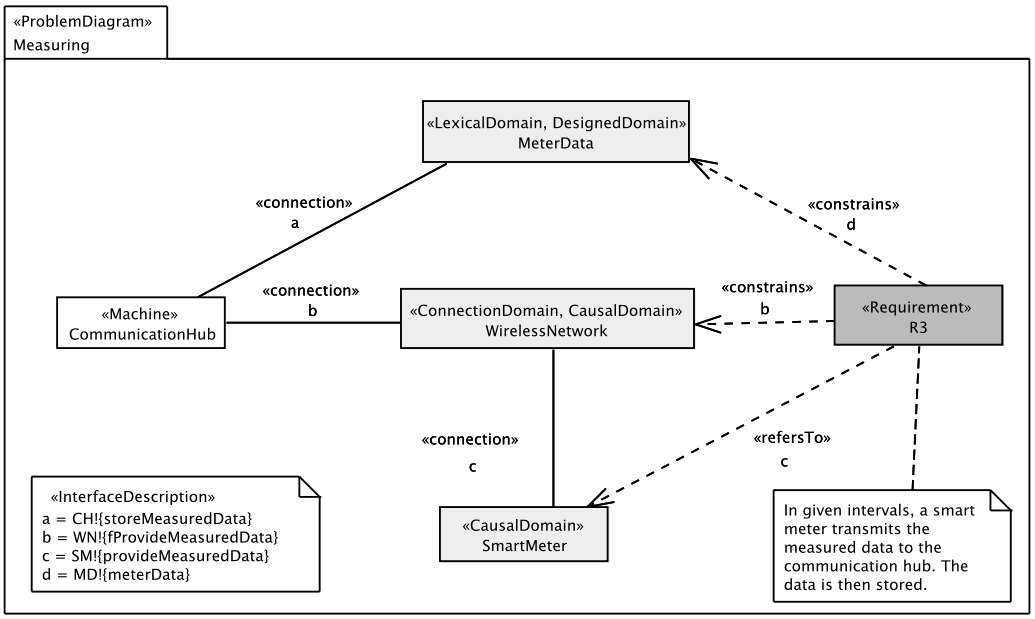


Figure 5.8.: Case Study: Requirement 3 - Measuring

5.6.2.4. Requirement 4: Retrieving

Figure 5.9 shows the problem diagram for the requirement *R4: Retrieving*. The biddable domain *EnergySupplier* is referred to by the requirement and connected to the machine *CommunicationHub* via the connection domain *WAN*. The connection domain is constrained by the requirement. The energy supplier sends a request to the machine to retrieve all measured data. The data is available at the lexical domain *MeterData*. Hence, there is an interface between machine and lexical domain and the lexical domain is referred to by the requirement. The requested data is then provided to the energy supplier via the connection domain *WAN*.

As the biddable and lexical domains are referred to and the connection domain is constrained, this problem diagram fits to the problem frame **query** [CH04].

5.6.2.5. Requirement 5: Invoicing

Figure 5.10 shows the problem diagram for the requirement *R5: Invoicing*. The operation described in this requirement is initiated by the machine. Based on the measured data and the personal data contained in the configuration, an invoice is created and stored. The measured data is available at the lexical domain *MeterData* and the personal data is available at the domain *Configuration*. Both domains are referred to by the requirement and for both, there is an interface to the machine to provide the data. The created invoice is then stored at the lexical domain *Invoice*. Hence, the domain *Invoice* is constrained.

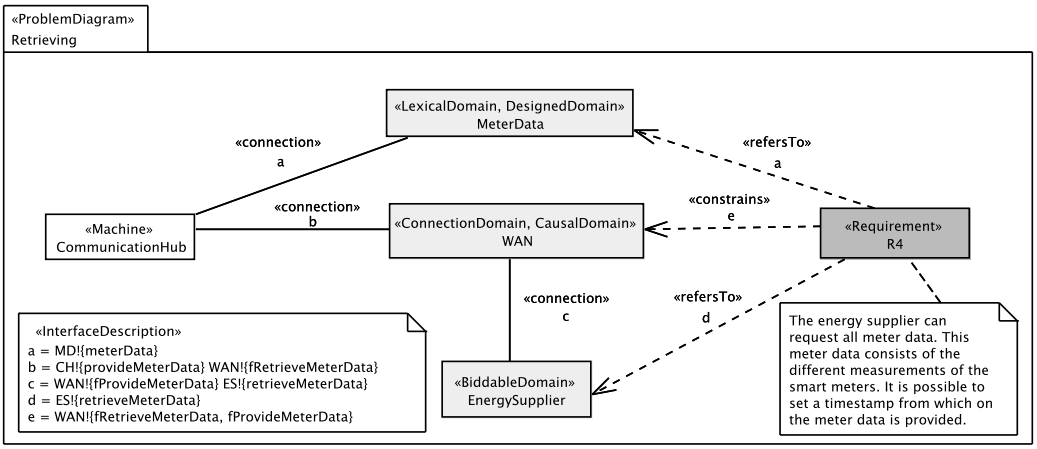


Figure 5.9.: Case Study: Requirement 4 - Retrieving

As two lexical domains are constrained and one lexical domain is referred, this problem diagram fits to the problem frame **transformation** [Jac01].

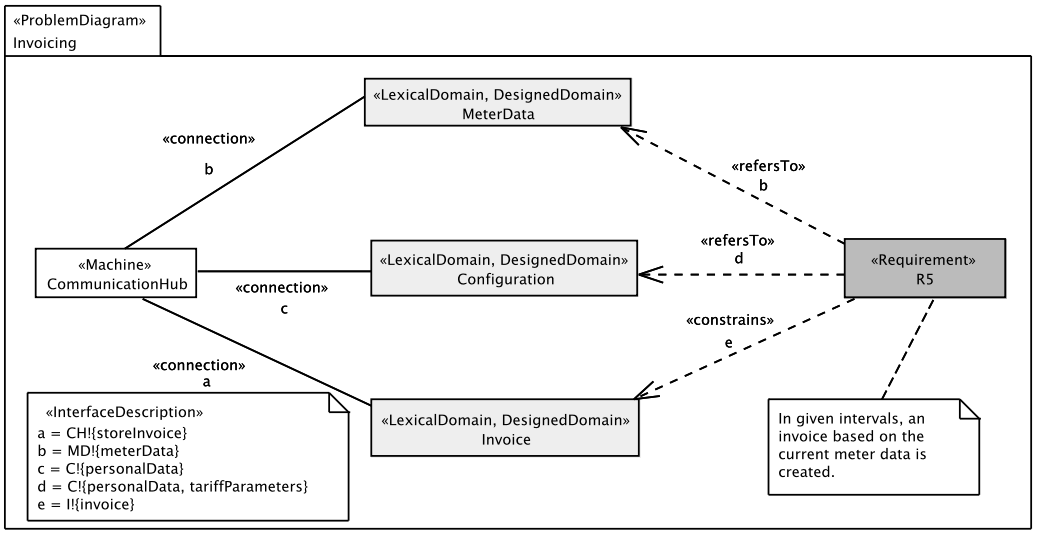


Figure 5.10.: Case Study: Requirement 5 - Invoicing

5.6.2.6. Requirement 6: Control Power

Figure 5.11 shows the problem diagram for the requirement *R6: Control Power*. The operation to change the status of the causal domain *PowerControl* is initiated by the biddable domain *EnergySupplier*. The new power status is sent to the machine using the connection domain *WAN*. The biddable domain is referred to by the requirement and the connection domain is constrained. The machine then provides the status to the domain *PowerControl*. Hence, the causal domain is constrained and there is an interface to the machine.

As the biddable domain is referred and the connection and causal domain are constrained, this problem frame fits to the problem frame **commanded behavior** [Jac01].

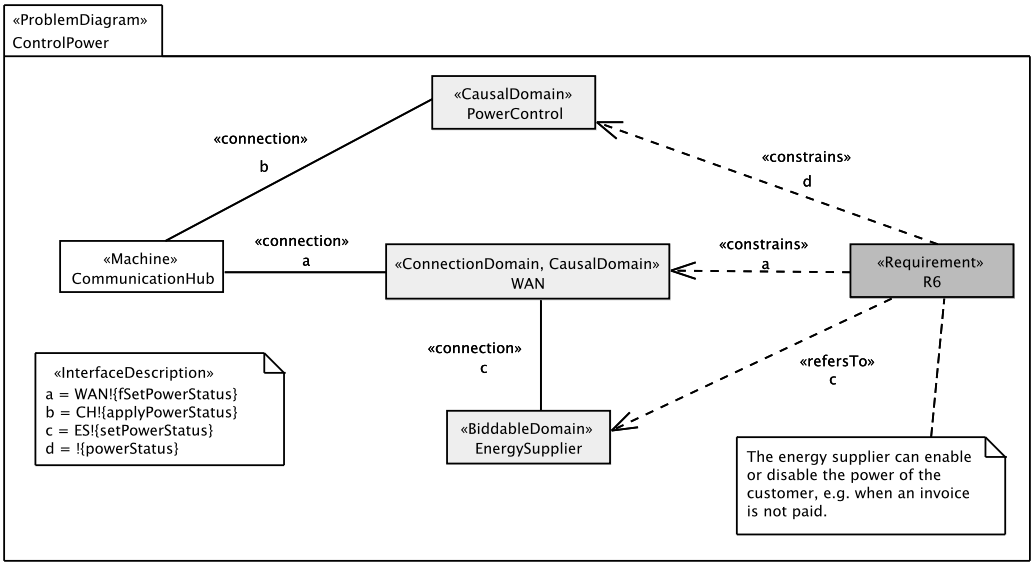


Figure 5.11.: Case Study: Requirement 6 - Control Power

5.6.2.7. Requirement 7: Display Invoice

Figure 5.12 shows the problem diagram for requirement *R7: Display Invoice*. After creating a new invoice, the machine displays the invoice to the customer using the smartphone app. The invoice to be shown is available at the lexical domain *Invoice*. The domain is referred to and there is an interface to the machine to provide the invoice. The connection domain *Smartphone* represents the device on which the invoice is shown. As this domain is also used as an input device for *R8: Change Personal Data*, it is not realized as a display domain. The connection domain is constrained and there is an interface from the machine to it to provide the invoice. The biddable domain *EndCustomer* is contained in the diagram but is neither referred nor constrained because it is not part of the requirement that the end customer has a look at the smartphone.

As the lexical domain is referred to and the connection domain constrained, this problem diagram fits to the problem frame **model display** [Jac01].

5.6.2.8. Requirement 8: Change Personal Data

Figure 5.13 shows the problem diagram for the requirement *R8: Change Personal Data*. The end customer is able to change his/her personal data using the smartphone app. The biddable domain *EndCustomer* is referred to by the requirement and the connection domain *Smartphone* is constrained. The new personal data are provided to the machine *CommunicationHub* via the domain *Smartphone*. There are interfaces between these domains. The personal data is available at the lexical domain *Configuration* and the new data is stored in this domain. The domain is constrained and there is an interface to the machine.

As the biddable domain is referred to and the connection and causal domain are constrained, this problem diagram fits to the problem frame **update** [CH04].

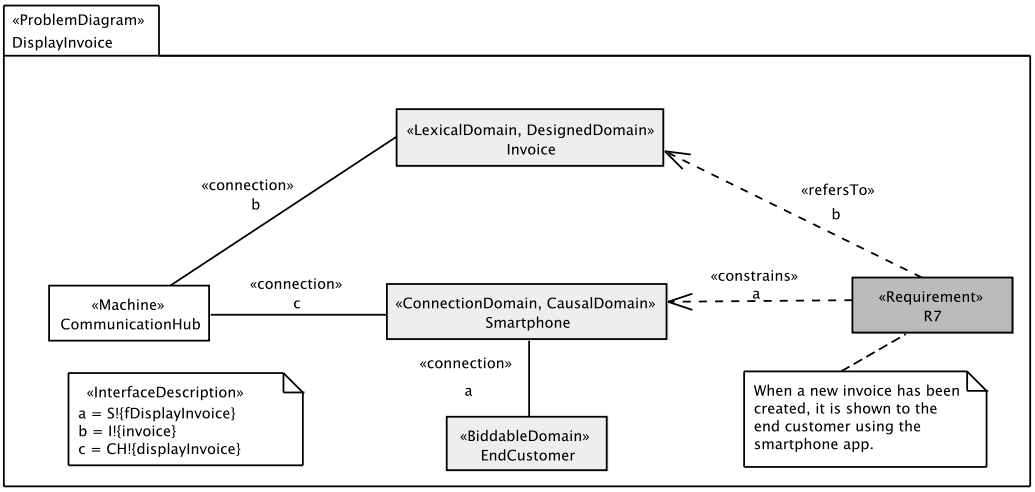


Figure 5.12.: Case Study: Requirement 7 - Display Invoice

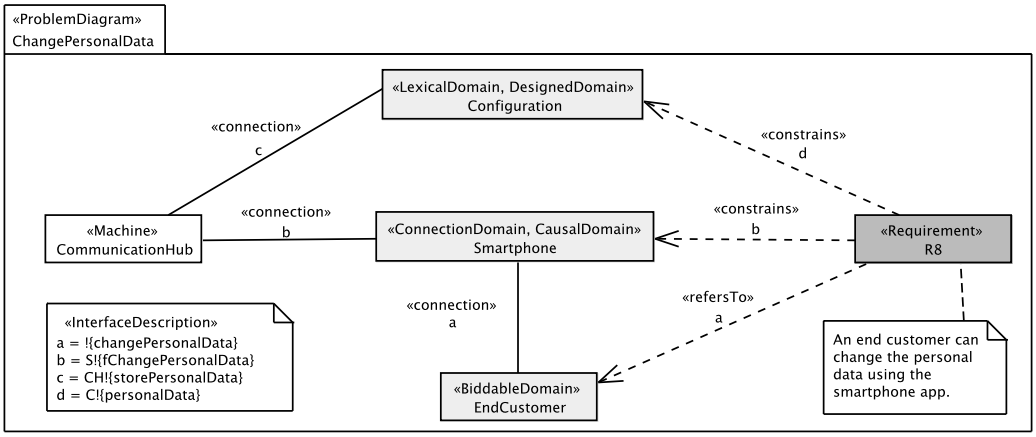


Figure 5.13.: Case Study: Requirement 8 - Change Personal Data

5.6.3. Adjust Problem Diagrams

The left out connection domains are discovered as proposed in Section 5.3. The results are shown in Table 5.3. The results show that adjustments are needed for the diagrams *ChangePersonalData* and *DisplayInvoice*, because the connection domain *WirelessNetwork* has been left out.

The connection domain is added between the machine *CommunicationHub* and the connection domain *Smartphone*. The phenomena controlled by the adjacent domains of the newly introduced domain are forwarded to the domains that are observing them in the original diagram.

The adjusted problem diagram for the requirement *R7: Display Invoice* is shown in Figure 5.14. The adjusted problem diagram for the requirement *R8: Change Personal Data* is shown in Figure 5.15.

Domain 1	Domain 2	$Q1$	Name?	$Q2$	Name?	$Q3$	Name?	PD
Energy-Supplier	CommunicationHub	no	-	no	-	no	-	Setup
Configuration	CommunicationHub	no	-	no	-	no	-	Setup
CommunicationHub	Configuration	no	-	no	-	no	-	Establish-Connection
CommunicationHub	Energy-Supplier	yes	WAN	no	-	no	-	Establish-Connection
CommunicationHub	MeterData	no	-	no	-	no	-	Measuring
CommunicationHub	Smart-Meter	yes	Wireless-Network	no	-	no	-	Measuring
CommunicationHub	MeterData	no	-	no	-	no	-	Retrieving
CommunicationHub	Energy-Supplier	yes	WAN	no	-	no	-	Retrieving
CommunicationHub	MeterData	no	-	no	-	no	-	Invoicing
CommunicationHub	Configuration	no	-	no	-	no	-	Invoicing
CommunicationHub	Invoice	no	-	no	-	no	-	Invoicing
CommunicationHub	Power-Control	no	-	no	-	no	-	Control-Power
CommunicationHub	Energy-Supplier	yes	WAN	no	-	no	-	Control-Power
CommunicationHub	Invoice	no	-	no	-	no	-	Display-Invoice
CommunicationHub	End-Costumer	yes	Wireless-Network	yes	Smart-phone	no	-	Display-Invoice
CommunicationHub	Configuration	no	-	no	-	no	-	Change-Personal-Data
CommunicationHub	End-Costumer	yes	Wireless-Network, Smart-Phone	no	-	no	-	Change-Personal-Data

Table 5.3.: Case Study: Connection Domain Discovery Table

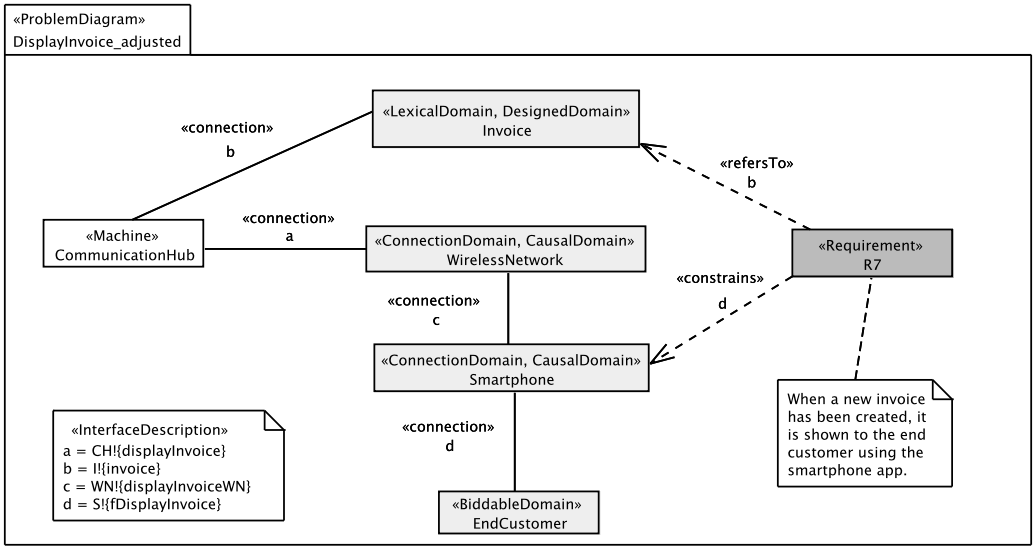


Figure 5.14.: Case Study: Requirement 7 - Display Invoice (adjusted)

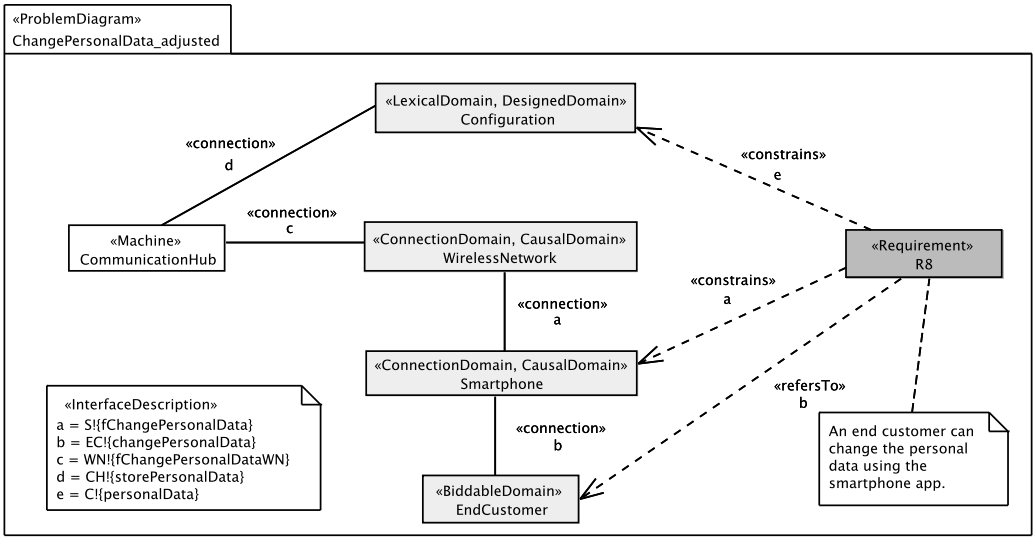


Figure 5.15.: Case Study: Requirement 8 - Change Personal Data (adjusted)

5.6.4. Phenomena Relations

Based on the problem diagrams, the analysts and the customer elicit the relations between the different phenomena. They make use of the process described in this section on page 32. The results of this step are shown in Figure 5.16. Because of the complexity of this diagram, we do not describe each element in detail. The diagram has to be read in the following way: The causal phenomenon *fRequestConnection* (left upper corner) contains the symbolic phenomenon *loginData*. The symbolic phenomenon *loginData* is contained in the causal phenomena *sendInitialInformation*, *storeInitialInformation* and *requestConnection*.

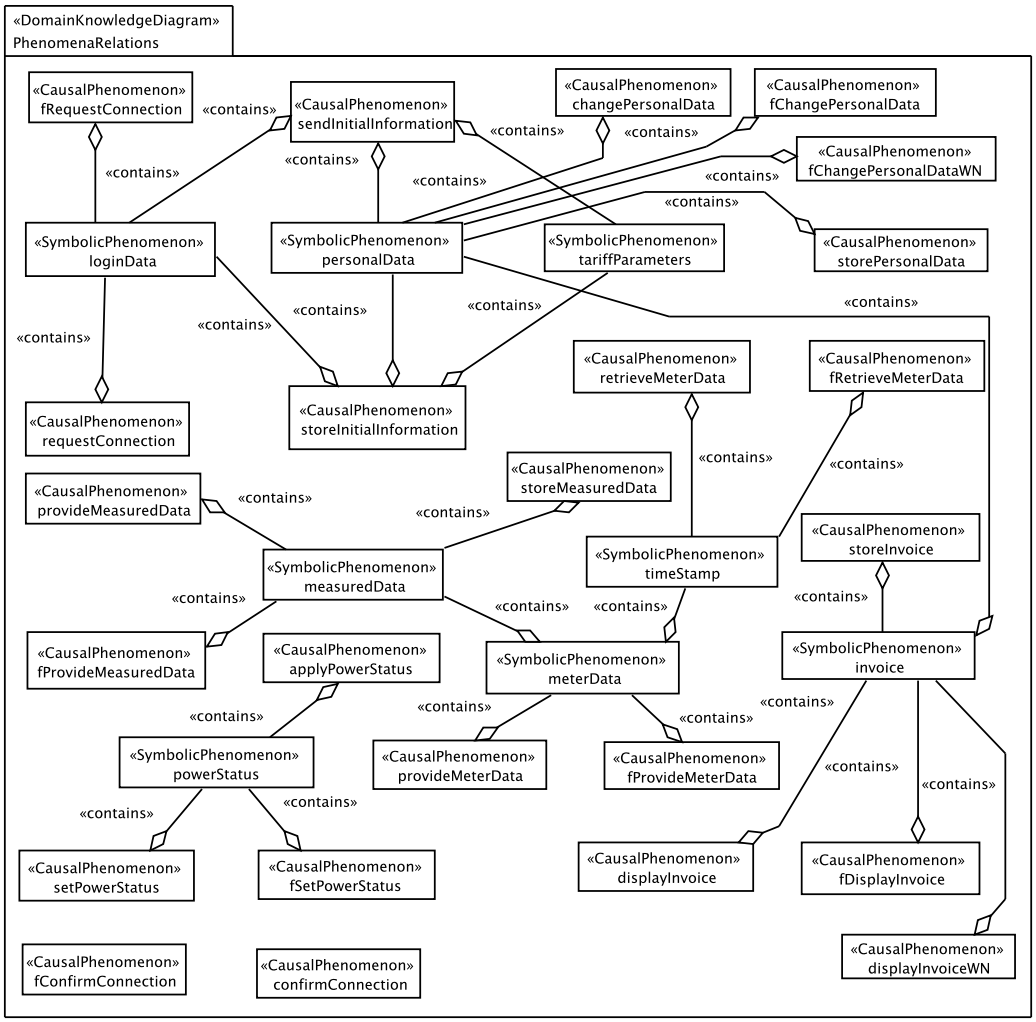


Figure 5.16.: Case Study: Phenomena Relations

6. Step 2: Defining Focus & Scope

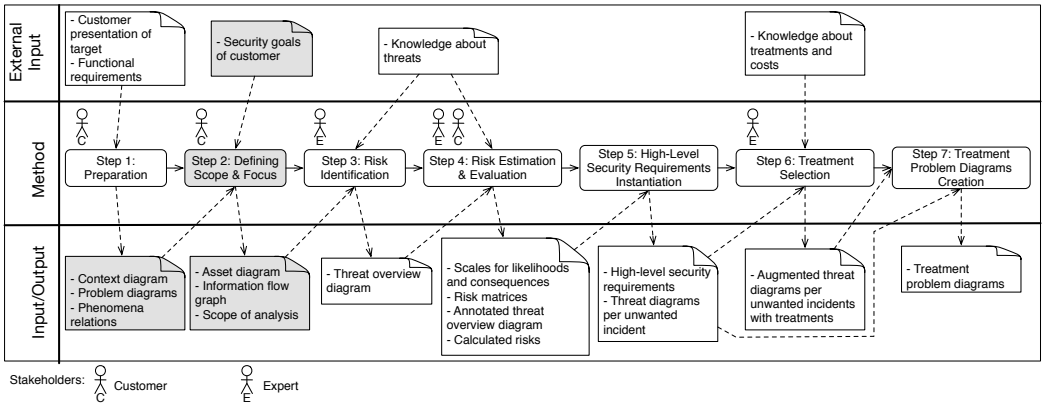


Figure 6.1.: ProCOR - Step 2

Based on the preparation, decisions are made for the further steps of the risk analysis. Table 6.1 gives an overview of the step and the contained sub-tasks.

This step is similar to *Step 3: Refining target description using asset diagrams* of the original CORAS method. As the model serves as a documentation for the target, we do not consider *Step 4: Approval of target description* of the CORAS method separately. The model has already been presented to the customer.

Step 2: Defining Focus & Scope	
CORAS Steps	Step 3
Input	- Problem Diagrams - Domain knowledge about phenomena relations - Security goals of customer (informal)
Output	- Asset diagram describing the security goals of the customer - Information flow graph - Definition of the scope for the analysis
Validation Conditions	- Only those domains are defined as to be in the scope of the analysis on which at least one phenomenon related to an asset is available. - There exists exactly one asset diagram. - Each customer goal is considered in the asset diagram.

Table 6.1.: ProCOR - Step 2

6.1. Selection of Assets & Asset Diagram

To be able select the assets, we first extend the CORAS UML profile. Afterwards, we describe what is considered as an asset in the method and how one can identify an asset. In the present method, we only consider direct assets. The consideration of indirect assets will be subject to future work.

6.1.1. Profile Extension

In Section 4.3 on page 25 we introduced the CORAS UML profile. For our method, we describe an asset in the following way: An asset is some kind of information represented as a symbolic phenomenon in the model with regard to the security properties Confidentiality, Integrity and Availability (CIA). Hence, a security goal is the protection of the information with regard to the related security property.

Therefore, we extend our profile as shown in Figure 6.2.

The stereotype *DirectAsset* contained in the CORAS Core profile is specialized with a new abstract stereotype called *SecurityGoal*. This new stereotype has the attribute *phenomenon* of type SymbolicPhenomenon to be able to express the relation of a security goal to some kind of information. To be able to represent the different security properties, the stereotype *SecurityGoal* is again specialized by three other stereotypes called *Confidentiality*, *Integrity* and *Availability*. Each asset can then be represented as a UML class with one of these stereotypes and the link to the corresponding phenomenon.

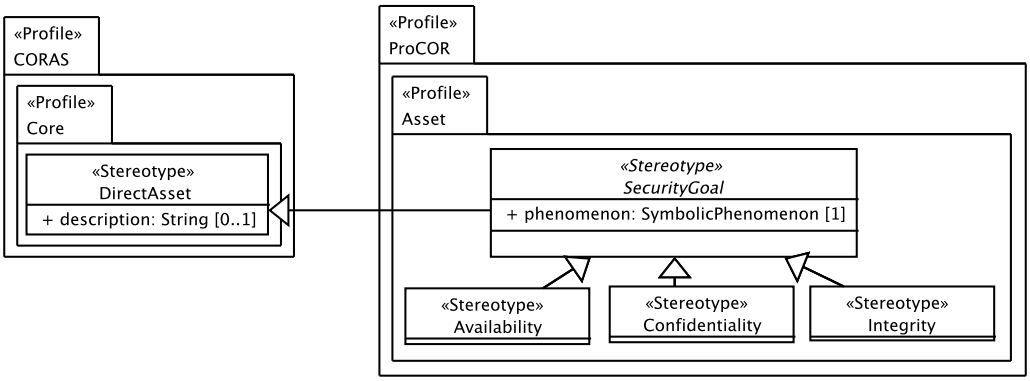


Figure 6.2.: ProCOR Profile: Asset

6.1.2. Asset Identification

The domain knowledge diagram called *PhenomenaRelations* contains all phenomena of the problem diagrams. For the selection of the assets, only the symbolic phenomena are relevant because they represent some kind of information.

First, a new domain knowledge diagram is created which contains the symbolic phenomena and the relations between these phenomena. The causal phenomena and the relations to them are left out.

To be able to select the assets, the point of view for the analysis has to be defined. The point of view is called *party* and is annotated in the stereotype ≪`AssetDiagram`≫ with the attribute *party*.

The analysts now present the domain knowledge diagram about the symbolic phenomena to the customer. For each contained phenomenon, the customer has to answer the following questions:

1. From the view of the party, is this information valuable with regard to the security property confidentiality?
2. From the view of the party, is this information valuable with regard to the security property integrity?
3. From the view of the party, is this information valuable with regard to the security property availability?

For the combination of the symbolic phenomenon and security property for which the above mentioned questions have been answered positively, a UML class is created to which the stereotype of the security property is applied and the attribute *phenomenon* is set to the symbolic phenomenon. This class is then added to the package to which the stereotype ≪`AssetDiagram`≫ is applied. This package represents the asset diagram.

6.2. Information Flow Graph

The Information Flow Graph is used to identify domains on which the information that is related to an asset is available. For this identification, we consider all problem diagrams, the phenomena relations and a set of all symbolic phenomena which are related to an asset regardless of their CIA properties. The Information Flow Graph is inspired by the Problem-based Privacy Analysis (ProPAn) [BFHM14].

6.2.1. Graph Specification

The information flow graph is specified using the Z-Notation [Spi89]. The first schema is called *InitialModel* and specifies the elements of the initial model which are needed to create the Information Flow Graph. There are problem diagrams and the *contains* relations to express the relation between the different phenomena. The second schema is called *InformationFlowGraph* and imports the schema for the initial model to specify the nodes and edges of the graph. There is a function *informationAt* that returns the symbolic phenomena related to an asset which are available on a given domain. The nodes of the information flow graph are a set of domains for which the result of the function is not empty. The edges of the graph describe an information flow between two domains where a symbolic phenomenon related to an asset is transmitted. The problem diagrams in which the information flow takes place and the transmitted symbolic phenomena are annotated at the edge.

$[Domain, Phenomenon, Requirement]$
$Connection == (Domain \times \mathbb{P}\, Phenomenon \times Domain)$
$ProblemDiagram == \mathbb{P}\, Domain \times \mathbb{P}\, Connection \times \mathbb{P}\, Requirement$

InitialModel

$domains : \mathbb{P}\, Domain$
$phenomena : \mathbb{P}\, Phenomenon$
$requirements : \mathbb{P}\, Requirement$
$assets : \mathbb{P}\, Phenomenon$
$controlledBy : Phenomenon \nrightarrow Domain$
$connections : \mathbb{P}\, Connection$
$contains : \mathbb{P}(Phenomenon \times Phenomenon)$

$assets \subseteq phenomena$

$\mathsf{dom}\ controlledBy = phenomena$
$\mathsf{ran}\ controlledBy \subseteq domains$
$\forall\, p : \mathbb{P}\, Phenomenon;\ d_1, d_2 : Domain \mid (d_1, p, d_2) \in connections \bullet \forall\, x : p \bullet$
$\quad d_1 \neq d_2 \land p \subseteq phenomena \land d_1 \in domains \land d_2 \in domains$
$\quad \land\ controlledBy(x) = d_1 \lor controlledBy(x) = d_2$

$\forall\, d : \mathbb{P}\, Domain;\ c : \mathbb{P}\, Connection;\ r : \mathbb{P}\, Requirement \mid (d, c, r) \in problemDiagrams \bullet$
$\quad d \subseteq domains \land c \subseteq connections \land \forall\, d_1, d_2 : Domain;\ p : \mathbb{P}\, Phenomenon \mid$
$\quad (d_1, p, d_2) \in c \bullet d_1 \in domains \land d_2 \in domains \land r \subseteq requirements$

$\mathsf{dom}\ contains \subseteq phenomena$
$\mathsf{ran}\ contains \subseteq phenomena$
$\forall\, x_1, x_2, x_3 : phenomena \bullet (x_1, x_2) \in contains \land (x_2, x_3) \in contains \Rightarrow (x_1, x_3) \in contains$

InformationFlowGraph

$$
\begin{array}{l}
\mathit{InitialModel} \\
\mathit{nodes} : \mathbb{P}\, \mathit{Domain} \\
\mathit{informationAt} : \mathit{Domain} \nrightarrow \mathbb{P}\, \mathit{Phenomenon} \\
\mathit{edges} : \mathbb{P}(\mathit{Domain} \times \mathbb{P}\, \mathit{Phenomenon} \times \mathbb{P}\, \mathit{ProblemDiagram} \times \mathit{Domain}) \\
\hline
\forall\, d : \mathit{domains} \bullet \mathit{informationAt}(d) = \{a : \mathit{assets} \mid \exists\, d' : \mathit{domains};\ p' : \mathbb{P}\, \mathit{phenomena} \bullet \\
\quad ((d, p', d') \in \mathit{connections} \lor (d', p', d) \in \mathit{connections}) \land \\
\quad (a \in p' \lor (\exists\, a' : p' \bullet (a', a) \in \mathit{contains}))\} \\
\\
\mathit{nodes} = \{d : \mathit{domains} \mid \mathit{informationAt}(d) \neq \varnothing\} \\
\mathit{edges} = \{d_1 : \mathit{domains};\ p : \mathbb{P}\, \mathit{assets};\ \mathit{pds} : \mathbb{P}\, \mathit{problemDiagrams};\ d_2 : \mathit{domains} \mid \\
\quad p = \mathit{informationAt}(d_1) \cap \mathit{informationAt}(d_2) \cap (\mathit{controlledBy}^{\sim}(\!|\{d_1\}|\!)) \land p \neq \varnothing \land \\
\quad \mathit{pds} = \{D : \mathbb{P}\, \mathit{domains};\ C : \mathbb{P}\, \mathit{connections};\ R : \mathbb{P}\, \mathit{requirements} \mid \\
\quad (D, C, R) \in \mathit{problemDiagrams} \land \exists\, a : p;\ d_1, d_2 : \mathit{domains};\ p' : \mathbb{P}\, \mathit{phenomena} \bullet \\
\quad (d_1, p', d_2) \in C \land (a \in p' \lor (\exists\, a' : p' \bullet (a', a) \in \mathit{contains}))\}\}
\end{array}
$$

6.2.2. Profile Extension

To embed the information flow graph into the existing UML model, we define new stereotypes to describe the elements of the graph. This part of the ProCOR UML profile is shown in Figure 6.3. The names of the elements are chosen according to the graph specification.

All elements of an information flow graph are contained in a package with the stereotype ≪`InformationFlowGraph`≫. The stereotype ≪`InformationAtDomain`≫ is applied to the nodes of the graph and contains the property *domain*, to link it to the original domain, and the property *availableInformation*, which is a set of all symbolic phenomena that are related to an asset and that are available on this domain. In the graph specification, this attribute is defined by the function *informationAt* for the corresponding domain. The property *inScope* of type Boolean is used to document the decision of the customer whether a domain shall be considered during the risk analysis. This property is not contained in the specification as it is not part of the graph generation. The nodes of the graph are represented as classes. Hence, the stereotype is an extension of the meta class *Class*.

The stereotype ≪`informationFlow`≫ is used to express an information flow between two domains. So it represents the edges of the graph. It consists of two properties: the property *phenomena* is a set of symbolic phenomena that shows which information flows between the two domains, and the property *origin*, which describes the problem diagrams in which the information flows occur. These attributes are used for the traceability between the information flow graph and the initial model.

The information flow graph now provides an overview of all information flows that occur due to the functional requirements and provides a list for each domain of available information related to an asset. Each domain on which some kind of information related to an asset is available is a potential point to harm an asset.

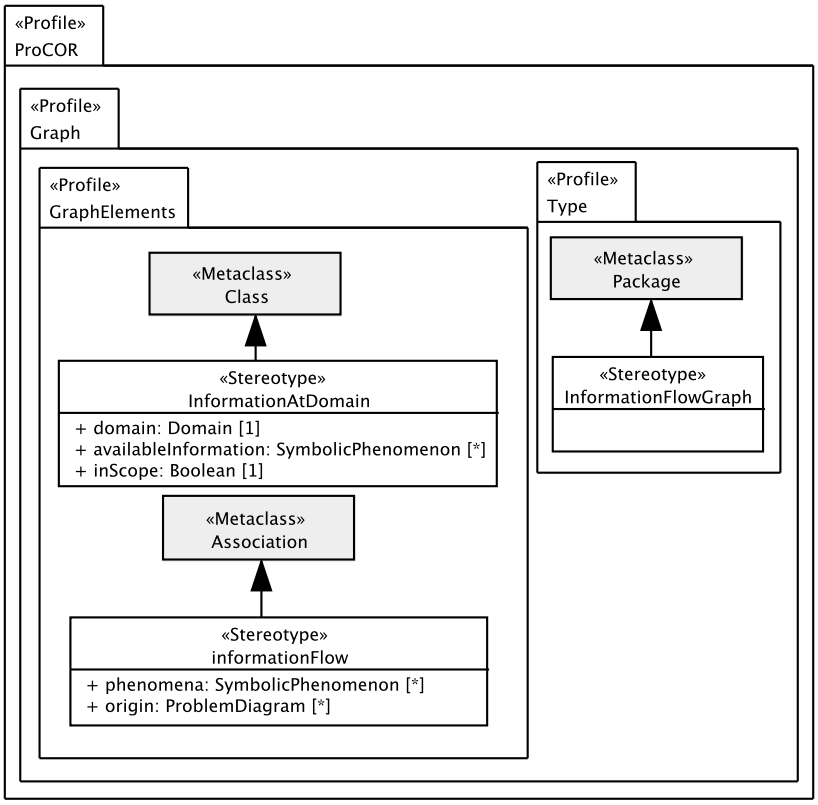


Figure 6.3.: ProCOR Profile: Information Flow Graph

6.3. Scope Definition

Based on the graph, the scope for the analysis is defined. An implicit definition of the scope has already taken place during asset identification. Only symbolic phenomena which are related to an asset have been considered for the graph generation.

The analysts present the information flow graph to the customer and have to explain what information is available on which domain. With regard to the asset diagram, the customer has to decide which domains shall be considered for the risk identification. As the analysts have a deeper knowledge of possible attacks, they can support the customer, e.g. that a wireless connection has maybe more possible attack points than a wired connection. To document the decisions of the customer, the value for the attribute *inScope* of each node is set accordingly.

Remark: As we propose a model-based approach that is not based on brainstorming with the customer, *Step 4: Approval of target description* of the original CORAS method is left out. The model elements on which the following steps will be based have already been presented to the customer.

6.4. Validation Conditions

There are three validation conditions.

1. Only those domains are defined as to be in the scope of the analysis on which at least one phenomenon related to an asset is available.

   Domains on which information related to an asset is available have to be considered for the risk analysis. Other domains are not considered because they do not have access to the information. Considering them would increase the effort without new results.

2. There exists exactly one asset diagram.

   All assets are contained in one diagram. This is necessary to make the definition of assets unambiguous.

3. Each customer goal is considered in the asset diagram.

   If an asset is not contained in the asset diagram, it will not be considered in the analysis. Therefore, it has to be checked that the diagram is complete.

6.5. Case Study

In this section, we apply Step 2 to the case study.

6.5.1. Selection of Assets

Figure 6.4 shows the domain knowledge diagram with all symbolic phenomena which are contained in the model. This diagram is created based on the phenomena relations shown on page 42. It is now presented to the customer and the customer decides about the assets with regard to the CIA properties. The party as the point of view is the energy supplier company.

Based on the list of symbolic phenomena, the customer decided to consider the following assets:

loginData shall be integer and confidential. Using these credentials, it is possible to establish a false connection to the energy supplies. Hence, the energy supplier is no longer able to retrieve the correct meter data. When manipulating the data, it is no longer possible to establish a connection with the energy supplier.

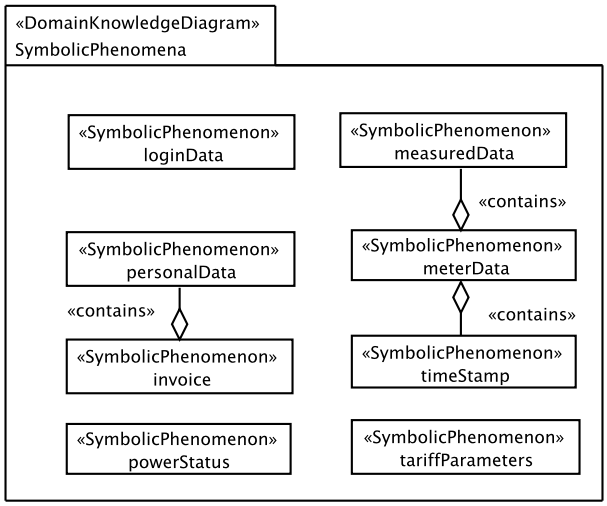


Figure 6.4.: Case Study: Symbolic Phenomena

measuredData shall be available and integer. Without correctly measured data, it is not possible to calculate the correct invoice.

meterData shall be available and integer as well, because the measured data are contained in the meter data.

personalData shall be confidential and integer. When this kind of data is disclosed, the customer might sue the energy supplier. As the personal data is part of the invoice, the correctness of the data is necessary for a correct invoice.

invoice shall be confidential, integer and available. As an invoice contains personal data, the confidentiality is important. Incorrect invoices might lead to a wrong payment. When invoices are not available, it is not possible to pay in time.

powerStatus shall be integer and available. When manipulating the power status it is possible to enable or disable the customer's energy. When the status is not available, it is not ensured that the power control works correctly.

tariffParameters shall be integer and available. Incorrect tariff parameters lead to a wrong invoice. When the tariff parameters are not available, it is not possible to calculate an invoice.

The asset diagram is shown in Figure 6.5. For each of the above mentioned security goals, a class is created with the corresponding stereotype and symbolic phenomenon as attribute.

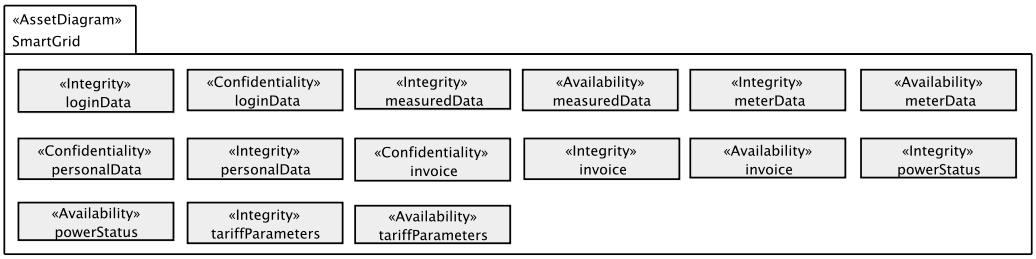


Figure 6.5.: Case Study: Asset Diagram

6.5.2. Information Flow Graph

The information flow graph is generated for the selected assets based on the introduced problem diagrams. The resulting graph is shown in Figure 6.6. The available information for each domain is annotated in a comment linked to the nodes in the graph. The names of the nodes are chosen according to the names of the annotated domains. As the attributes of the edges are only used for the traceability, they are not shown here. We do not explain the complete graph because of its complexity.

For example, the graph shows that there is an information flow from the node *SmartMeter* to the node *WirelessNetwork*. At the *SmartMeter*, the symbolic phenomenon *measuredData* is available and it is also available at the node *WirelessNetwork*.

6.5.3. Scope Definition

The analysts present the information flow graph to the customer. The customer decides to consider all identified domains in the analysis.

Hence, all values of the attributes *inScope* are set to *true*. This documents the customer decision in the model.

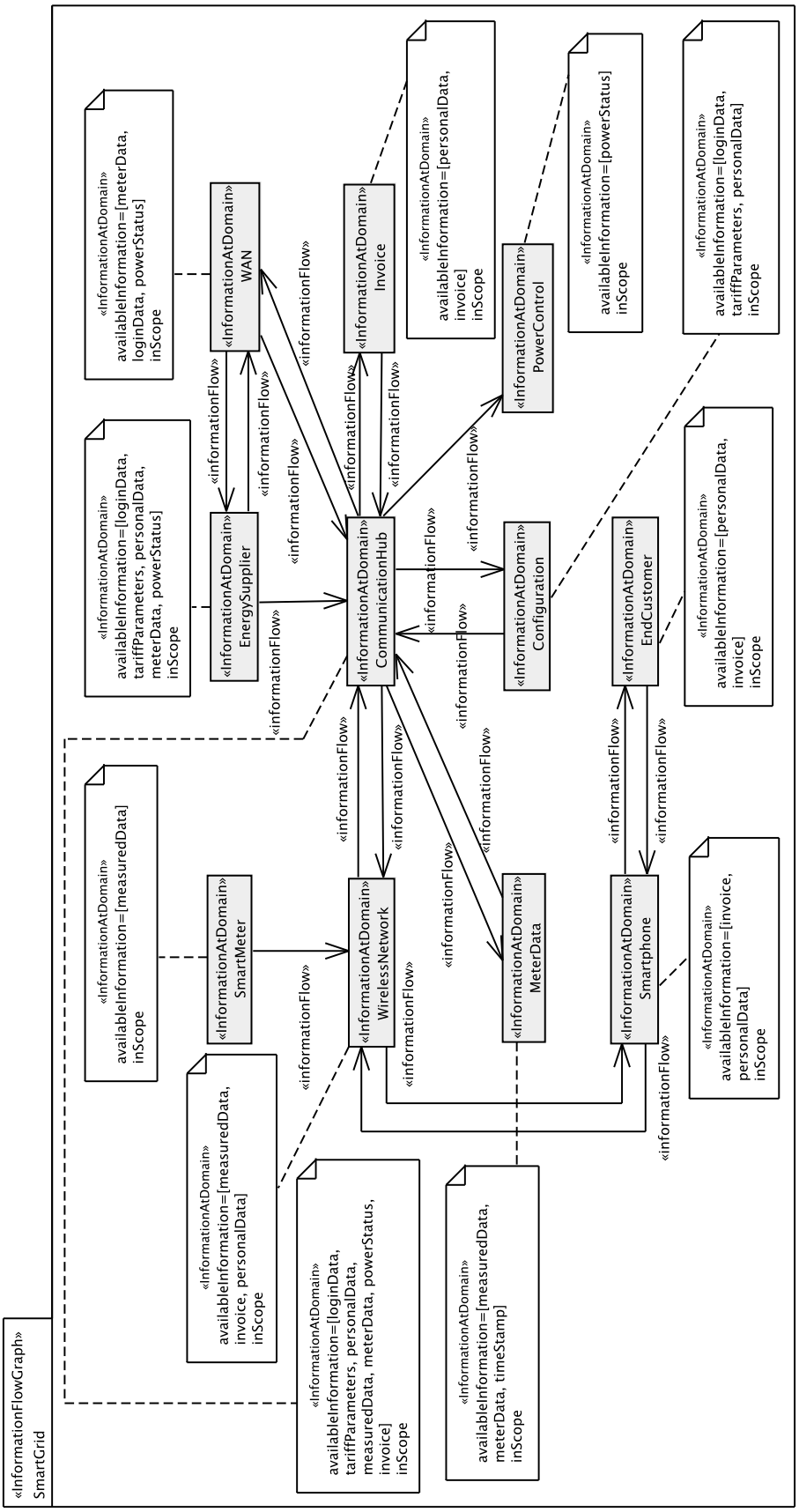


Figure 6.6.: Case Study: Information Flow Graph

7. Step 3: Risk Identification

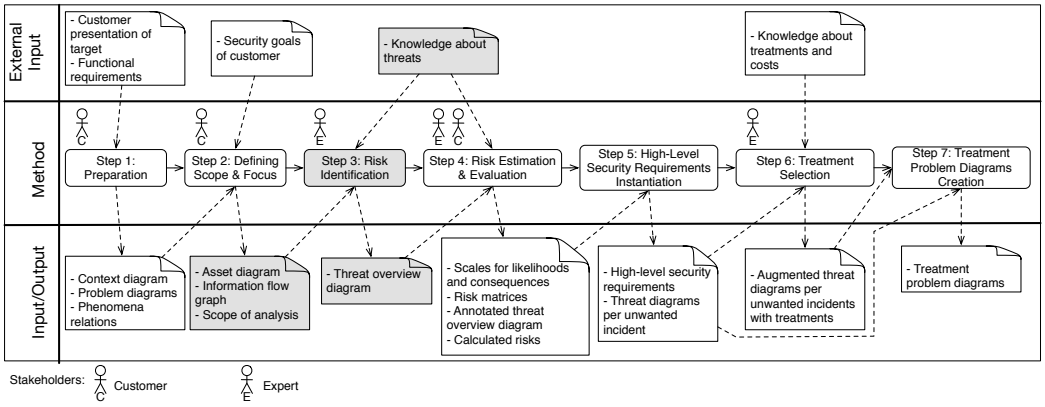


Figure 7.1.: ProCOR - Step 3

Based on the results of the previous steps, the analysts now analyze the system with regard to risk. Hence, it is necessary to identify threats, threat scenarios and unwanted incidents. This step is quite similar to *Step 5: Risk identification using threat diagrams* of the original CORAS method. We make use of the CORAS UML profile as introduced on page 26 to create threat diagrams. Table 7.1 provides an overview of the step. We do not consider vulnerabilities in our method explicitly but we make use of them to identify possible threats.

Step 3: Risk Identification	
CORAS Steps	Step 5
Input	- Domains in scope - Information Flow Graph - External knowledge about threats
Output	- Threat Overview Diagram
Validation Conditions	- All assets are considered in the threat overview diagram. - Only domains where the symbolic phenomenon related to an asset is available can have a threat scenario that leads to an unwanted incident that harms the asset.

Table 7.1.: ProCOR: Step 3

7.1. Risk Identification

The identification of risk consists of three substeps: threat identification, threat scenarios and unwanted incidents. For the different elements, we use the definition as described on page 10. First of all, we introduce some resources which can be used during the identification process.

7.1.1. Resources

To be able to identify risk, we need expert knowledge. Some knowledge can be accessed for free, as it is provided by non-profit organizations. We introduce three resources in this section.

7.1.1.1. Threat Classification

There are several organizations which try to classify threats. One example for such an organization is the Cloud Security Alliance[1] (CSA). It focusses on security for cloud scenarios and aims to support international policy makers, for example the NIST[2] as well as the European Commission. Based on a search of known incidents which harm some kind of cloud computing system, these incidents are categorized according to their causes. Using this approach, in 2016 twelve different threat classes have been identified [Clo16].

Each threat, called Security Concern in the document, comes with additional information. There is a textual description of the incident, which contains a definition, the business impact and examples for this type. There is also a threat analysis, which describes the influences of the incident, e.g. information disclosure. To address the threat, there are references to the CSA Security Guidance. This information can be used to avoid the threat. For example, there is a guide for encryption and key management.

Figure 7.2 shows a screenshot of the first security concern, Data Breaches. As this is a description for a cloud scenario, the service models that are potentially influenced by this threat are stated. A data breach is an incident in which non-public data is harmed. The security guidance addresses different types of security, e.g. *Domain 12: Identity, Entitlement and Access Management* in which the CSA aims to support the avoidance of the threat by guiding the company to provide different levels of access for the users of the system. Last, there is a threat analysis which shows that information might be disclosed due to the threat. The business impacts are left out in this thesis.

As this threat classification is very high-level, it can be used in early phases of the software development, in which we do not have information about the different technical components of the software. Also for non-cloud computing, the given information provides a general idea for possible threats.

7.1.1.2. Vulnerability Databases

A vulnerability database is a collection of known vulnerabilities in software or hardware. It can be used as a search engine, e.g. by entering the name of a router. Then, detailed information is provided how this object can be harmed. In contrast to threat classifications, there is no information how often this vulnerability is exploited. The *National Vulnerability Database*[3] is provided by the NIST. As an example, we use the keyword *Netgear* as an input. Netgear is a well known company for network components. The first result leads to the vulnerability *CVE-2015-8289*. CVE stands for Common Vulnerability and Exposure, a standard for a common namespace for vulnerabilities. The incident has been discovered in 2015 and 8289 is its number in this year. Figure 7.3 shows a screenshot of the result. The

[1] `https://cloudsecurityalliance.org/` - Cloud Security Alliance
[2] `http://nist.gov` - National Institute of Standards and Technology
[3] `https://nvd.nist.gov/` - NVD

1. Security Concern: Data Breaches

1.1 Description

A data breach is an incident in which sensitive, protected or confidential information is released, viewed, stolen or used by an individual who is not authorized to do so. A data breach may be the primary objective of a targeted attack or may simply be the result of human error, application vulnerabilities or poor security practices. A data breach may involve any kind of information that was not intended for public release including, but not limited to, personal health information, financial information, personally identifiable information (PII), trade secrets and intellectual property.

An organization's cloud-based data may have value to different parties for different reasons. For example, organized crime often seeks financial, health and personal information to carry out a range of fraudulent activities. Competitors and foreign nationals may be keenly interested in proprietary information, intellectual property and trade secrets. Activists may want to expose information that can cause damage or embarrassment. Unauthorized insiders obtaining data within the cloud are a major concern for organizations.

The risk of data breach is not unique to cloud computing, but it consistently ranks as a top concern for cloud customers. A cloud environment is subject to the same threats as a traditional corporate network as well as new avenues of attack by way of shared resources, cloud provider personnel and their devices and third party partners of the cloud provider. Cloud providers are highly accessible and the vast amount of data they host makes them an attractive target.

1.2 Business Impacts

SERVICE MODELS

IaaS | PaaS | SaaS

CSA SECURITY GUIDANCE REFERENCE

Domain 5: Information Management and Data Security
Domain 10: Application Security
Domain 11: Encryption and Key Management
Domain 12: Identity, Entitlement and Access Management
Domain 13: Virtualization

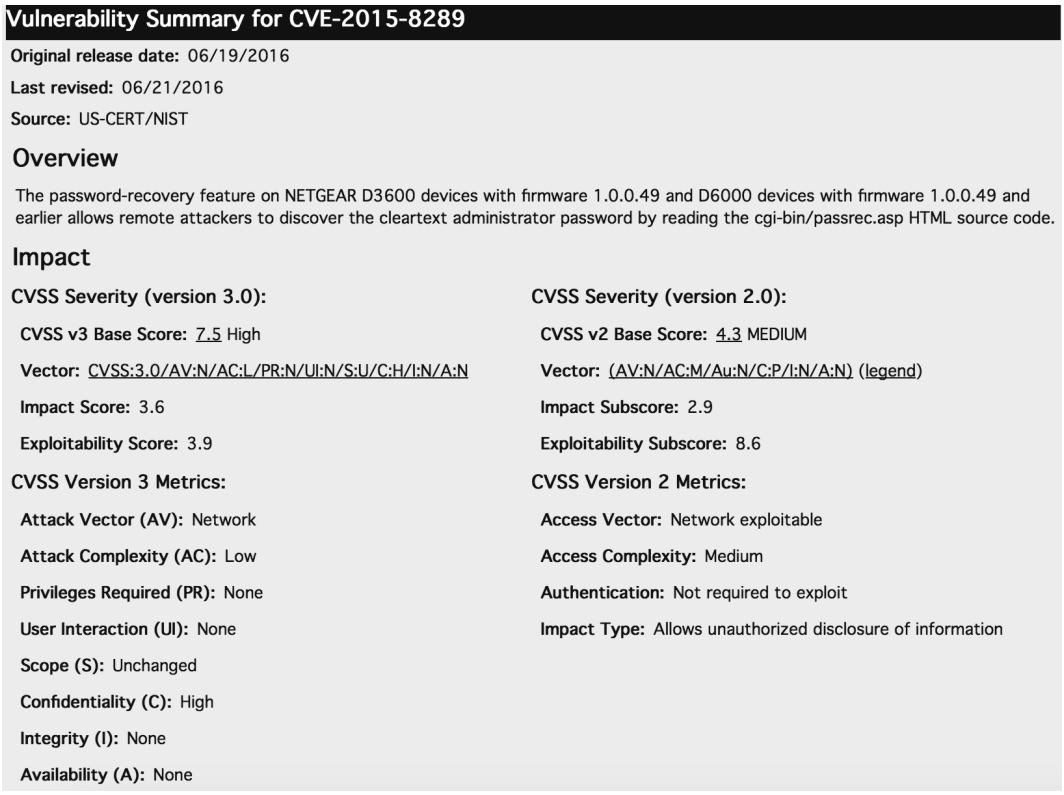
THREAT ANALYSIS

STRIDE:

- ☒ Spoofing Identity
- ☒ Tampering with data
- ☒ Repudiation
- ☑ Information Disclosure
- ☒ Denial of Service
- ☒ Elevation of Privilege

Figure 7.2.: CSA Threat - Data Breaches [Clo16]

information about the vulnerability is structured using the CVSS (Common Vulnerability Scoring System) metric [FIR].



Vulnerability Summary for CVE-2015-8289

Original release date: 06/19/2016

Last revised: 06/21/2016

Source: US-CERT/NIST

Overview

The password-recovery feature on NETGEAR D3600 devices with firmware 1.0.0.49 and D6000 devices with firmware 1.0.0.49 and earlier allows remote attackers to discover the cleartext administrator password by reading the cgi-bin/passrec.asp HTML source code.

Impact

CVSS Severity (version 3.0):

CVSS v3 Base Score: 7.5 High

Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Impact Score: 3.6

Exploitability Score: 3.9

CVSS Version 3 Metrics:

Attack Vector (AV): Network

Attack Complexity (AC): Low

Privileges Required (PR): None

User Interaction (UI): None

Scope (S): Unchanged

Confidentiality (C): High

Integrity (I): None

Availability (A): None

CVSS Severity (version 2.0):

CVSS v2 Base Score: 4.3 MEDIUM

Vector: (AV:N/AC:M/Au:N/C:P/I:N/A:N) (legend)

Impact Subscore: 2.9

Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

Access Vector: Network exploitable

Access Complexity: Medium

Authentication: Not required to exploit

Impact Type: Allows unauthorized disclosure of information

Figure 7.3.: CVE-2015-8289 - Vulnerability Example [Nat15]

The different values of the metric are used to calculate a score for the vulnerabilities. The result is a value between 0.0 and 10.0. The higher the scoring, the bigger the need to consider the vulnerability in the analysis. The value is calculated using the following basic values:

Attack Vector Local (local access to component), Adjacent Network (local network access) and network (also remote access).

Attack Complexity Low, Medium, High. Describes the skills which are needed to perform the attack.

Privileges Required None, Low, High. Is an authorization required to perform the attack?

User Interaction None, Required. Can the attack be performed by the attacker himself/herself, or is a third-party in form of a user required?

Scope Unchanged, Changed. Can other resources be affected which are not in scope of the attack?

Confidentiality Impact Low, Medium, High. Is the confidentiality of parts of the component harmed?

Integrity Impact Low, Medium, High. Is the integrity of parts of the component harmed?

Availability Impact Low, Medium, High. Is the availability of parts of the component harmed?

We do not describe the calculation here. For the given example, the attacker only needs network access and the complexity of the attack is low. There are no privileges or user interaction required. The scope is unchanged but the impact on the confidentiality is high. This leads to a CVSS v3 Base Score of 7.5 (High).

A vulnerability database provides a detailed view on possible attacks for a system. Additionally, the impacts are described and a rating based on the CVSS can be derived. The disadvantage of this resource is to need information about the technical equipment to perform a sufficient search in the database. In our approach, we only consider the functional requirements and a general description. We know that there is a database to store data, but we do not yet know of which kind it is. One solution is to perform a search with well known representatives of the term *Database*, e.g. MySQL or Microsoft SQL Server. This approach has a high workload, because the result set will be large. Another approach is to agree on the technical concepts in the early phases of the development process. This can be done using a technical context diagram [Hat12] which in contrast to the original context diagram describes the technical context of the system to be developed.

7.1.1.3. Threat Patterns

Uzunov and Fernandez propose an extensible pattern-based threat taxonomy for a distributed system [UF14]. In this sense it is restricted to this kind of system, but most relevant systems are built on a distributed architecture. If this is not the case, the approach can be adapted. The non-relevant threats, which only occur due to the communication, can be left out.

A threat pattern is described as a triple of the form: (A, T, SP). *A* is the architectural context, *T* is a generic threat description and *SP* contains the mitigating security policies. The patterns are categorized into classes. The classes themselves are distributed on two levels.

The architectural context describes the abstract relation to a part of the system architecture. For this purpose, different functionality decomposition layers are defined. For each layer, there is a brief description and the related classes of the first level are stated. For example, one layer is *User interaction*. It describes all interaction and interfacing with users and is related to five threat classes of the first level.

Threats of the first levels' classes describe threats that can be harmful by a single attack. For example, the first threat class is called *Identity attacks*. The threats of the first level are described by a name, a threat description, the architectural context expressed as the decomposition layer and the mitigating security policy. An example of a threat pattern is shown in Table 7.2.

Security threat pattern	Threat description	Architectural context	Mitigating security policy
Identity Spoofing	The attacker fabricates a new identity or claims to possess an existing identity of some principal in the system	User Interaction (input ports); Distribution control (software component interfaces, execution abstraction, processes); Addressing (all)	Identity management (authentication)

Table 7.2.: Example Threat Pattern of First Class [UF14]

Threats of the second levels' classes are more high-level than the ones from the first class. The classes contain threats that may consist of more than only one attack and are described independently from the architectural context. They are called meta-security threats. For example, one class is called *Cryptographic attacks*. An example is shown in Table 7.3. The abuse of a weak algorithm can lead to different cryptographic attacks. Thus, the meta-

security describes a cause for a group of different attacks. Using this information, general threats can be identified which are independent of the architecture.

Meta-security threat pattern	Threat description	Mitigating security policy
Abuse of weak algorithm	A cryptographic attack (brute force, known plaintext, etc.) is launched on a weak algorithm used for authentication, message encryption, hashing or some other security measure.	Identity management (authentication), Secure communication, Storage security

Table 7.3.: Example Threat Pattern of Second Class [UF14]

Uzunov and Fernandez provide a summary of known threats based on their own expertise [UF14]. The idea is to build up a threat library for the system to be built. The previously stated ways to identify threats and vulnerabilities can be used as an input for the library.

All approaches have in common that we need external knowledge to identify the risk in our method. In the following, we provide questionnaires to guide the analysts through the discussion with the customer and optional (external) experts.

7.1.2. Threat Elicitation

In CORAS, we distinguish three types of threats: human-threat (accidental), human-threat (deliberate) and non-human threat. For each domain contained in the information flow graph and considered to be in scope, the following questions of the top level have to be answered to identify threats and threat scenarios. The questions of the next level only have to be answered if the question of the current level has been answered positive.

1. Are there domains in the model that have a negative influence on the domain under investigation?
   - 1.1 What is the negative influence (threat scenario)?
   - 1.2 Is the domain that has the negative influence human?
     - 1.2.1 Is this influence accidental (human-threat accidental)?
     - 1.2.2 Is this influence deliberate (human-threat deliberate)?
   - 1.3 Is the domain that has the influence not human (non-human threat)?
2. Are there domains not yet mentioned in the model that have a negative influence on the domain under investigation?
   - 2.1 What is the negative influence (threat scenario)?
   - 2.2 Is the domain that has the negative influence human (new biddable domain)?
     - 2.2.1 Is this influence accidental (human-threat accidental)?
     - 2.2.2 Is this influence deliberate (human-threat deliberate)?
   - 2.3 Is the domain that has the influence not human (non-human threat)?
     - 2.3.1 Is this influence caused by a technical item (new causal domain)?
     - 2.3.2 Is this influence caused by a vis major (new biddable domain)?

A force majeure is described as a biddable domain, because the threat has an unpredictable behavior, e.g. a thunderstorm. Domains that are not yet mentioned in the model but are related to a threat have to be added to the model. This is done in the next step in which domain knowledge diagrams are created to document the results of the questionnaire in the model.

The results of the questionnaire are documented on-the-fly in Table 7.4.

Domain	Threat scenarios	Human-threat accidental	Human-threat deliberate	Non-human threat
...	...	...	...	...

Table 7.4.: Threat Elicitation Results

7.1.3. Domain Knowledge Diagrams

As the information of the threats and related elements is elicited using external knowledge, this information is not yet contained in the UML model. We make use of domain knowledge diagrams to describe threats and the initiated threat scenarios. There is one domain knowledge diagram per domain on which a threat scenario occurs. It contains the domain on which the threat scenario occurs and all threats which initiate a threat scenario on this domain. For each threat scenario, there is an assumption that describes the threat scenario. The domain that has the negative influence (threat) is referred to, whereas the domain on which the negative influence takes place is constrained by this assumption. We use an assumption because it is not sure that the threat and the threat scenario really exist. Between the threat and the threat scenario, there is an association with the stereotype ≪`connection`≫. This interface contains the phenomena which describe the way of initiating the scenario. The *refers to* dependency is annotated with the phenomena which are needed to initiate the threat scenario by the threat. The phenomena annotated at the constrains dependency describe the changes on the behavior or state of the domain on which the threat scenario occurs.

This information is used later to specify high-level security requirements to reduce risks for the system-to-be.

7.1.4. Unwanted Incidents

An unwanted incident is the event which is caused by a threat scenario and harming an asset. Hence, we analyze the combination of threat scenarios and assets.

An asset is described as a tuple (s, p) where s denotes some kind of data, represented as a symbolic phenomenon and p denotes a security property. Let $\mathcal{A}$ be the set of assets. A threat scenario is described as a tuple (d, t), where d denotes the domain on which the threat scenario occurs and t denotes a textual description. Let $\mathcal{T}$ be the set of threat scenarios.

To identify the unwanted incidents we consider the set $\mathcal{U} = \mathcal{A} \times \mathcal{T}$. For each element in $\mathcal{U}$, the analysts have to investigate if the threat scenario may lead to an unwanted incident which then harms the asset. To decide this, we make use of the information flow graph, created in Step 2 of the ProCOR method on page 46. If the phenomenon s of the asset is not available at the domain d of the threat scenario, the combination is discarded for the analysis. Otherwise, the unwanted incidents are elicited. For this, we again need expert knowledge.

As a result, we obtain a list of unwanted incidents as shown in Table 7.5. For each unwanted incident, it is documented which assets are harmed and which threat scenarios lead to this unwanted incident.

Threat Scenario	Unwanted Incident	Asset
...	...	...

Table 7.5.: Unwanted Incident Elicitation

7.2. Threat Overview Diagram

Based on the information which has been elicited in this step, we create a threat diagram. The diagram is represented as a UML class diagram. Using the CORAS profile, the user is able to state links to elements which are contained in other diagrams of the model.

A threat diagram is indicated by a package with the stereotype ≪`ThreatDiagram`≫. All elements of this diagram are contained in this package together with the relations between them.

All assets that are contained in the asset diagram are added to the threat diagram on the right-hand side.

Next, the unwanted incidents are added on the left of the assets. It is important to consider the unwanted incidents as a set. This means that we add each incident only once in the diagram.

An unwanted incident and the harmed asset are connected using a directed association from the unwanted incident to the asset with the stereotype ≪`harms`≫.

The threat scenarios are added on the left of the unwanted incidents. Here again, each scenario is only added once. The attribute *domain* is set to with the domain on which the threat scenario exists. Between threat scenario and unwanted incident, a directed association is added with the stereotype ≪`leadsTo`≫.

In a last step, the threats are added to the diagram on the left-hand side. Each threat is added once to the diagram, so that there are no duplicates. The attributes *person* and *attacker* for human-threats are set to the biddable domain that represents the person of the threat. The attribute *domain* of a non-human threat is set to the domain on which the threat occurs.

If necessary, it is possible to add a more detailed description to an element using the attribute *description*.

The attributes for likelihood and consequence will be set in Step 4 of the ProCOR method as described on page 67.

Depending on the number of elements, the threat diagram grows very fast. Based on the UML model, it is possible to create different views on the diagram depending on the needs of the analysts or customer. In Step 5 of the ProCOR method, we generate a view for each unwanted incident. Other views are not part of this thesis.

7.3. Validation Conditions

There are two validation conditions.

1. All assets are considered in the threat overview diagram.

   It is necessary to consider all assets for the threat analysis. Therefore, it is necessary to add all assets to the threat overview diagram.

2. Only domains where the symbolic phenomenon related to an asset is available can have a threat scenario that leads to an unwanted incident that harms the asset.

   When a symbolic phenomenon is not available at the domain, it cannot have a negative influence on the asset.

7.4. Case Study

In this section, the risk identification is performed for the case study.

7.4.1. Threat Elicitation

The customer has decided to consider all domains that are related to a node of the information flow graph for the analysis. Hence, the analysts and the experts have to answer the questionnaire for all domains.

The results are documented in Table 7.6. There are threat scenarios on five different domains. In total, we have six threat scenarios initiated by five different threats. For example, there is a threat scenario called *Broken connection due to maintenance* at the domain *WAN*. The threat scenario is initiated by the provider of the WAN who is responsible for the maintenance of the WAN connection.

Domain	Threat scenarios	Human-threat accidental	Human-threat deliberate	Non-human threat
WAN	Broken connection due to maintenance	Provider		
WirelessNetwork	Break into wireless connection		Network-Attacker	
	Disrupt frequency of own wireless network			Other-Wireless-Network
EnergySupplier	Energy supplier is bribed		EndCustomer	
SmartMeter	Manipulate sensor		EndCustomer	
Communication-Hub	Incorrect implementation of algorithms		Developer	

Table 7.6.: Case Study: Threat Identification

7.4.2. Domain Knowledge Diagrams

For each domain on which a threat scenario exists, the analysts have to create a domain knowledge diagram. In the following, the resulting five diagrams are described. The corresponding assumptions are stated as well.

7.4.2.1. Domain Knowledge for WirelessNetwork

Figure 7.4 shows the domain knowledge diagram for the domain *WirelessNetwork*. This diagram contains two assumptions:

A1 A network attacker breaks into the wireless connection.

A2 The frequency used by the wireless network is disrupted by another wireless network.

Assumption *A1* refers to the biddable domain *NetworkAttacker* and constrains the connection domain *WirelessNetwork*. Assumption *A2* refers to the causal domain *OtherWireless-Network* and constrains the connection domain *WirelessNetwork*.

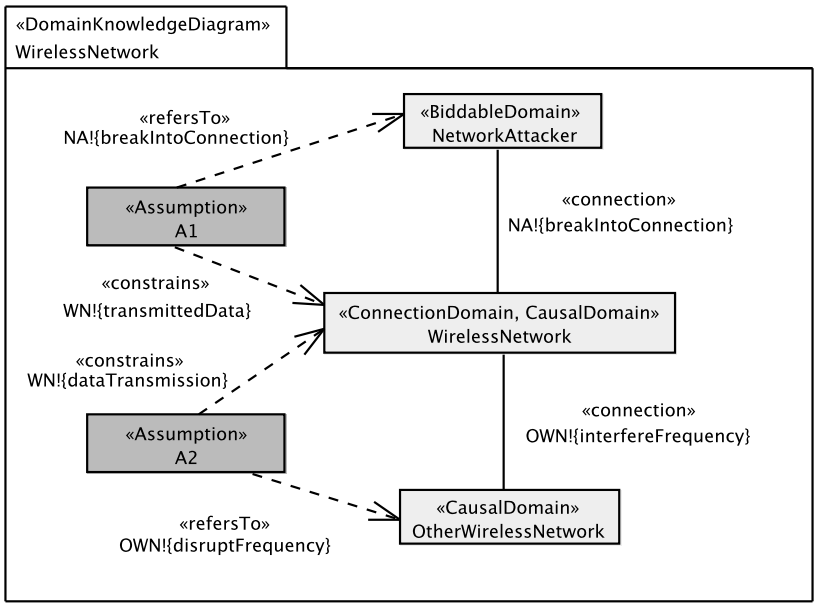


Figure 7.4.: Case Study: Domain Knowledge for WirelessNetwork

7.4.2.2. Domain Knowledge for EnergySupplier

Figure 7.5 shows the domain knowledge diagram for the domain *EnergySupplier*. This diagram contains one assumption:

A3 The energy supplier is bribed by the end customer to disclose the information of other customers.

Assumption *A3* refers to the biddable domain *EndCutomer* and constrains the biddable domain *EnergySupplier*.

7.4.2.3. Domain Knowledge for SmartMeter

Figure 7.6 shows the domain knowledge diagram for the domain *SmartMeter*. This diagram contains one assumption:

A4 The end customer manipulates some sensor.

Assumption *A4* refers to the biddable domain *EndCutomer* and constrains the causal domain *SmartMeter*.

7.4.2.4. Domain Knowledge for CommunicationHub

Figure 7.7 shows the domain knowledge diagram for the domain *CommunicationHub*. This diagram contains one assumption:

A5 A developer implements an algorithm incorrectly.

Assumption *A5* refers to the biddable domain *Developer* and constrains the machine *CommunicationHub*.

7.4.2.5. Domain Knowledge for WAN

Figure 7.8 shows the domain knowledge diagram for the domain *WAN*. This diagram contains one assumption:

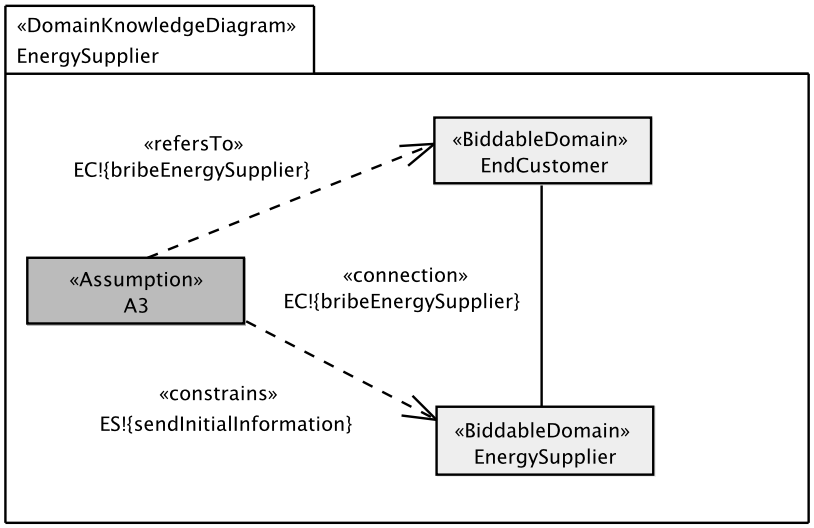


Figure 7.5.: Case Study: Domain Knowledge for EnergySupplier

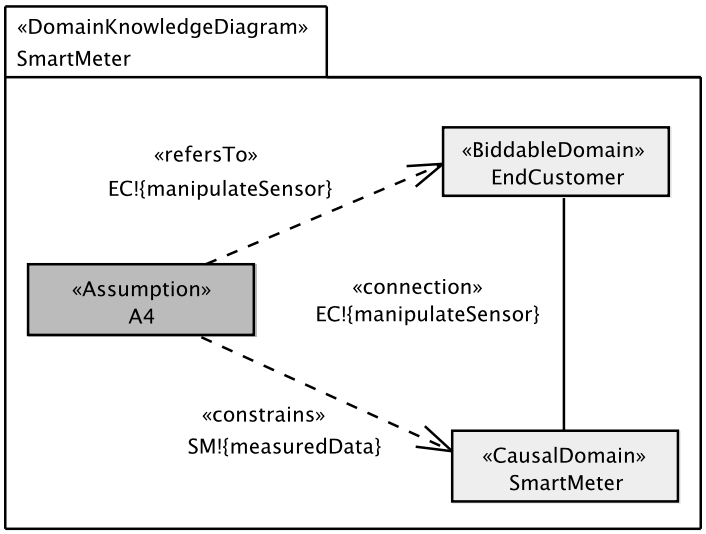


Figure 7.6.: Case Study: Domain Knowledge for SmartMeter

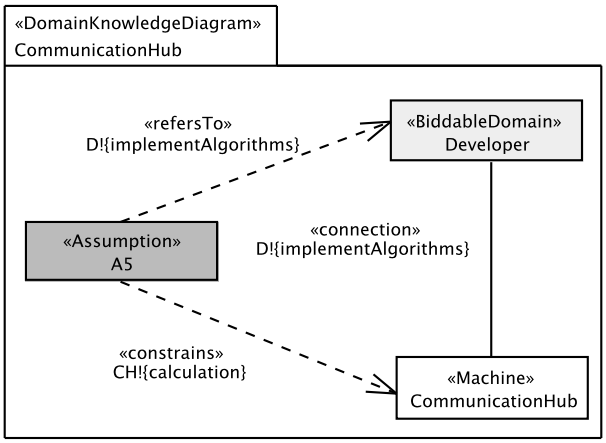


Figure 7.7.: Case Study: Domain Knowledge for CommunicationHub

A6 The WAN connection is broken due to a maintenance.

Assumption *A4* refers to the biddable domain *Provide* and constrains the connection domain *WAN*.

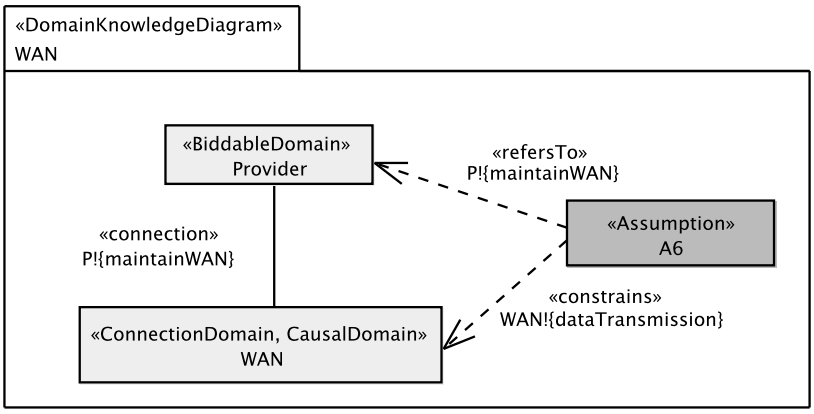


Figure 7.8.: Case Study: Domain Knowledge for WAN

7.4.3. Unwanted Incidents

Based on the threat elicitation, the analysts and the experts collect possible unwanted incidents. The results are shown in table 7.7. For example, the threat scenario *Break into wireless connection* leads to the unwanted incident *Disclose transmitted data*. This unwanted incident then harms the confidentiality of personal data and the confidentiality of invoices.

7.4.4. Threat Overview Diagram

Using the results stated in Table 7.6 and Table 7.7, the analysts create the threat overview diagram. This diagram is shown in Figure 7.9. One has to read the diagram by following the paths starting with a threat on the left side. For example, the human-threat deliberate *NetworkAttacker* initiates the threat scenario *Break into wireless connection*. The threat scenario leads to the unwanted incidents *Disclose transmitted data* and *Manipulate*

Threat Scenario	Unwanted Incident	Asset
Break into wireless connection	Disclose transmitted data	≪Confidentiality≫ personalData ≪Confidentiality≫ invoice
Break into wireless connection	Manipulate transmitted data	≪Integrity≫ measuredData ≪Integrity≫ personalData ≪Integrity≫ invoice
Broken connection due to maintenance	No connection between end customer and energy supplier	≪Availability≫ meterData ≪Availability≫ powerStatus
Disrupt frequency of own wireless network	Connection unavailable	≪Availability≫ measuredData ≪Availability≫ invoice
Energy supplier is bribed	Disclose customer data	≪Confidentiality≫ loginData ≪Confidentiality≫ personalData
Energy supplier is bribed	Manipulate tariff parameters	≪Integrity≫ tariffParameters
Manipulate sensor	Incorrect measuring of power consumption	≪Integrity≫ measuredData
Incorrect implementation of algorithms	Wrong calculation for invoice	≪Integrity≫ invoice

Table 7.7.: Case Study: Unwanted Incidents

transmitted data. The unwanted incident *Manipulate transmitted data* harms the assets ≪Confidentiality≫ *personalData* and ≪Confidentiality≫ *invoice*. The unwanted incident *Manipulate transmitted data* harms the assets ≪Integrity≫ *measuredData*, ≪Integrity≫ *personalData* and ≪Integrity≫ *invoice*. There are no unwanted incidents for the assets ≪Integrity≫ *meterData*, ≪Integrity≫ *loginData*, ≪Integrity≫ powerStatus and ≪Availability≫ *tariffParameters*. These assets are placed to the bottom of the diagram.

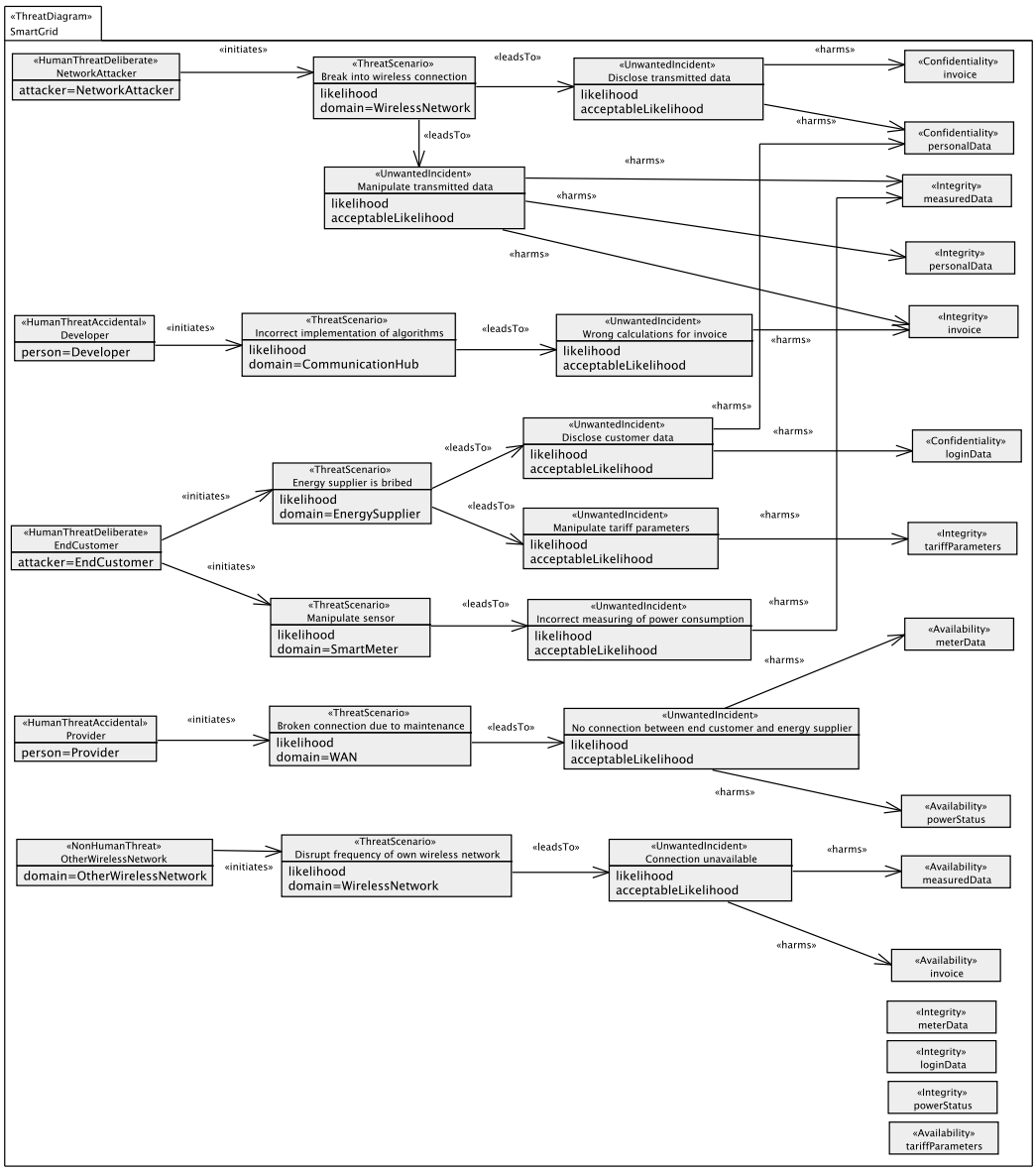


Figure 7.9.: Case Study: Threat Overview Diagram

8. Step 4: Risk Estimation & Evaluation

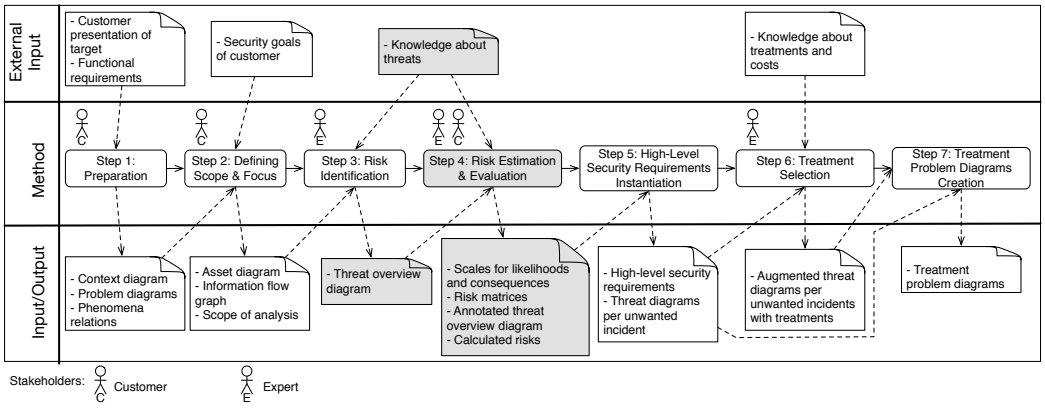


Figure 8.1.: ProCOR - Step 4

In this step, the threat overview diagram is annotated with the values for likelihoods and consequences. Then, these values are evaluated to identify whether a risk is acceptable.

Step 4: Risk Estimation & Evaluation	
CORAS Steps	Steps 6 and 7
Input	- Threat overview diagram - External knowledge about threats, especially likelihoods and consequences
Output	- Scales for likelihoods and consequences - Risk matrices - Calculated risks - Annotated threat overview diagram
Validation Conditions	- There is a likelihood annotated on each *initiates* relation. - There is a likelihood annotated on each *leadsTo* relation. - There is a consequence annotated on each *harms* relation. - For each threat scenario and unwanted incident, the likelihood has been calculated and documented in the model. - For each asset, there exists a risk matrix. - For each unwanted incident that is related to an unacceptable risk, the value for the acceptable likelihood is given.

Table 8.1.: ProCOR: Step 4

8.1. Estimation of Likelihoods & Consequences

For the likelihood estimation, we distinguish between likelihoods that are estimated based on external knowledge and likelihoods which are derived from the estimated likelihoods. For both types we make use of quantitative values.

In a threat diagram, it is possible to add a likelihood on the relations *initiates* and *leadsTo* as well as on a threat scenario and on an unwanted incident.

The likelihoods for the following relations are estimated based on external knowledge:

initiates It is estimated how likely it is that the threat initiates the threat scenario. For example, one has to decide how likely it is that an attacker tries to break into a connection.

leads to It is estimated how likely it is that the threat scenario leads to the unwanted incident. For example, one has to decide how likely it is that an attacker is able to disclose some kind of information when breaking into a connection.

Based on these estimated values, the other likelihoods are derived in the following way:

Threat scenario The likelihood of a threat scenario is derived from the likelihoods of the incoming *initiates* associations. If there is only one incoming association, the likelihood is equal to the likelihood of the association. Otherwise, it has to be decided how the likelihoods of the incoming associations are related. This likelihood has to be determined according to the rules of probability theory, and we do not cover this case here.

Unwanted incident The likelihood of an unwanted incident is derived from the likelihood of the threat scenario that leads to the incident and the likelihood of the *leads to* association. For each incoming association, the following calculation is applied: $ui = (1 - lt) * ts$, where lt is the likelihood of the association and ts is the likelihood of the threat scenario. If there is only one incoming association, ui is the likelihood of the unwanted incident. Otherwise, ui is calculated for all incoming associations. Then, it has to be decided how the calculated values are related. This is related to probability theory again.

This step is very similar to the original CORAS as described in Section 2.3.2.6 on page 15. In CORAS, the estimation of the likelihoods is started at the unwanted incident. The other likelihoods are considered if it is not possible to estimate the likelihood of the unwanted incident directly. The unwanted incident's likelihood is then derived as described above.

In our approach, the estimation starts on the associations and the likelihoods of the threat scenarios and the unwanted incidents are derived from that. All likelihoods are added to the threat overview diagram using the attribute *likelihood* of the corresponding stereotype.

The consequence for an unwanted incident on an asset is determined independently of its likelihood. For each pair of unwanted incident and asset between which there is a *harms* relation, we estimate the consequence. The value is also added in the diagram using the attribute *consequence* of the *harms* relation.

For our approach, we suggest qualitative consequence scales based on the CIA property that is considered as an asset. These scales are shown in Table 8.2. For the availability of information we suggest a scale based on the duration for which the information is unavailable. For the confidentiality of information we distinguish between sensitive and non-sensitive data and whether the data is encrypted. For the integrity of information we distinguish between the effort to identify and to correct changes.

As mentioned, we use a quantitative way to determine the likelihood whereas we use a qualitative way to determine the consequences. However, our method is not restricted to this way of determining likelihoods and consequences. In the case that qualitative scales are

Scale / Property	Availability	Confidentiality	Integrity
Insignificant	Unavailable for a short period; new data is cached	Disclosure of encrypted non-sensitive data	Minor changes of data can be identified and corrected
Minor	Unavailable for a long period; new data is cached	Disclosure plain non-sensitive data	Minor changes of data can be identified but not corrected
Moderate	Unavailable for a short period; Unable to store new data during this period	Disclosure of encrypted sensitive data; Encryption is hard to break	Minor changes can neither be corrected nor identified
Major	Unavailable for a long period; Unable to store new data during this period	Disclosure of encrypted sensitive data; Encryption is easy to break	Major changes with identification but without correction
Severe	Data loss of existing data and new data	Disclosure of plain sensitive data	Major changes without identification and correction

Table 8.2.: Consequence Scale depending on CIA property

used to estimate the likelihoods, it has to be defined how these values can be handled to derive the likelihoods of threat scenarios and unwanted incidents.

The result of this step is an augmented threat diagram. We do not add new elements to the diagram but we set the values of some attributes of already contained elements.

8.2. Risk Evaluation

As a risk is the combination of likelihood and consequence [ISO09], the calculation of the risk is performed using a risk matrix. In contrast to the original CORAS method, we only define acceptable and unacceptable risks. For each asset, a risk matrix is defined. On the x-axis of the matrix, we add the values of the consequence scale starting from left to right in increasing order. On the y-axis of the matrix, we add the values of the likelihood scale starting from the top in increasing order. We make use of intervals to define the likelihood scale. For each combination of likelihood and consequence, one has to decide if this is acceptable for the asset or not. If not, the corresponding cell of the matrix is marked red. Otherwise, the cell is marked green.

Now, for each combination of asset and unwanted incident that is contained in the threat overview diagram, the risk level is calculated according to the risk matrix of the corresponding asset. The results are documented as shown in Table 8.3 and grouped by the unwanted incidents.

Unwanted Incident	Asset	Risk level
...	...	☐acceptable ☐unacceptable

Table 8.3.: Risk Calculation Results

For the following steps of the ProCOR method, we only consider the unwanted incidents that are related to an unacceptable risk.

In our CORAS profile presented on page 26, we added an additional attribute to the stereotype ≪`UnwantedIncident`≫ called *acceptableLikelihood*.

Starting from an asset, we consider the risk for each unwanted incident that harms the asset. If it is unacceptable, the maximal acceptable likelihood for the consequence that is annotated on the relation between incident and asset is derived from the risk matrix. Using the annotated consequence, the highest acceptable likelihood is chosen. This value is lower than the calculated likelihood. Otherwise, there will be no risk which is unacceptable for this incident. The value is added to the threat diagram.

For all incidents that do not lead to an unacceptable risk, we do not set this attribute.

The result of this step is a set of unwanted incidents that need to be considered for a possible treatment. It consists of all incidents where the attribute *acceptableLikelihood* has been set.

8.3. Validation Conditions

There are six validation conditions.

1. There is a likelihood annotated on each *initiates* relation.

   The estimated likelihoods on this relation are needed to calculate the likelihood of the threat scenarios.

2. There is a likelihood annotated on each *leadsTo* relation.

   The estimated likelihoods on this relation are needed to calculate the likelihood of the unwanted incidents.

3. There is a consequence annotated on each *harms* relation.

   The estimated consequence is needed to derive the risk.

4. For each threat scenario and unwanted incident, the likelihood has been calculated and documented in the model.

   To evaluate the risk and to select treatments, it is necessary that all likelihoods have been calculated.

5. For each asset, there exists a risk matrix.

   The risk matrix for an asset is needed to evaluate a risk. The acceptable risk depends on the asset. Therefore, it is necessary to define a risk matrix for each asset.

6. For each unwanted incident that is related to an unacceptable risk, the value for the acceptable likelihood is given.

   To be able to select treatments for an unwanted incident with unacceptable risk, it is necessary to state the acceptable likelihood of this incident.

8.4. Case Study

In this section, the risk estimation and evaluation is applied to the case study.

8.4.1. Scales and Risk Matrix

To estimate the consequences, we make use of the scales that have been introduced in Table 8.2.

The scale for the likelihood definition is divided into five intervals based on the occurrence of an incident per year.

]0.001,0.003] more than every three years, at most once a year

]0.003,0.011] more than once a year, at most once a quarter of a year

]0.011,0.033] more than once a quarter of a year, at most once of a month

]0.033,0.14] more than once a month, at most once a week

]0.14,1] more than once a week

100% or 1 means an occurrence at least 365 times a year, so at least once a day. Only values between 0 and 1 are allowed. This means that all likelihoods bigger than once a day are represented by 100%. Everything that is less or equal 0.001 is acceptable. The lower boundary 0.001 is equal to once every three years. As it is not possible to reduce a likelihood to 0, it is necessary to define a lower boundary from which the likelihood is acceptable in general.

The risk matrix is shown in Table 8.4. In the case study, the analysts and the experts decided to have only one risk matrix for all assets. The risk that is acceptable is colored green and the risk that is unacceptable is colored red.

		Consequence				
		Insignificant	*Minor*	*Moderate*	*Major*	*Severe*
Likelihood	*]0.001,0.003]*					
	]0.003,0.011]					
	]0.011,0.033]					
	[0.033,0.14]					
	]0.14,1]					

Table 8.4.: Case Study: Risk Matrix

8.4.2. Annotated Threat Overview Diagram

Based on the scales, the analysts and experts estimate and calculate the values for the likelihood and consequences. The annotated threat overview diagram is shown in Figure 8.2. The estimation and calculation has been done in the following way: The likelihood that the human-threat deliberate *NetworkAttacker* initiates the threat scenario *Break into wireless connection* is estimated with $3\% = 0.03$. As this is the only incoming association for the threat scenario, its likelihood is also $3\% = 0.03$. The likelihood that the threat scenario leads to the unwanted incident *Disclose transmitted data* is estimated with $50\% = 0.5$. As this is the only incoming association, the likelihood for the unwanted incident is calculated with $(1 - 0.5) * 0.03 = 0.015 = 1.5\%$.

When the unwanted incident *Disclose transmitted data* happens, the consequence for the asset «`Confidentiality`» *invoice* is severe. An invoice contains sensitive data and is not

encrypted during the transmission. Therefore, we set the attributes for likelihoods and consequences to these values.

The described procedure is repeated for all elements of the diagram.

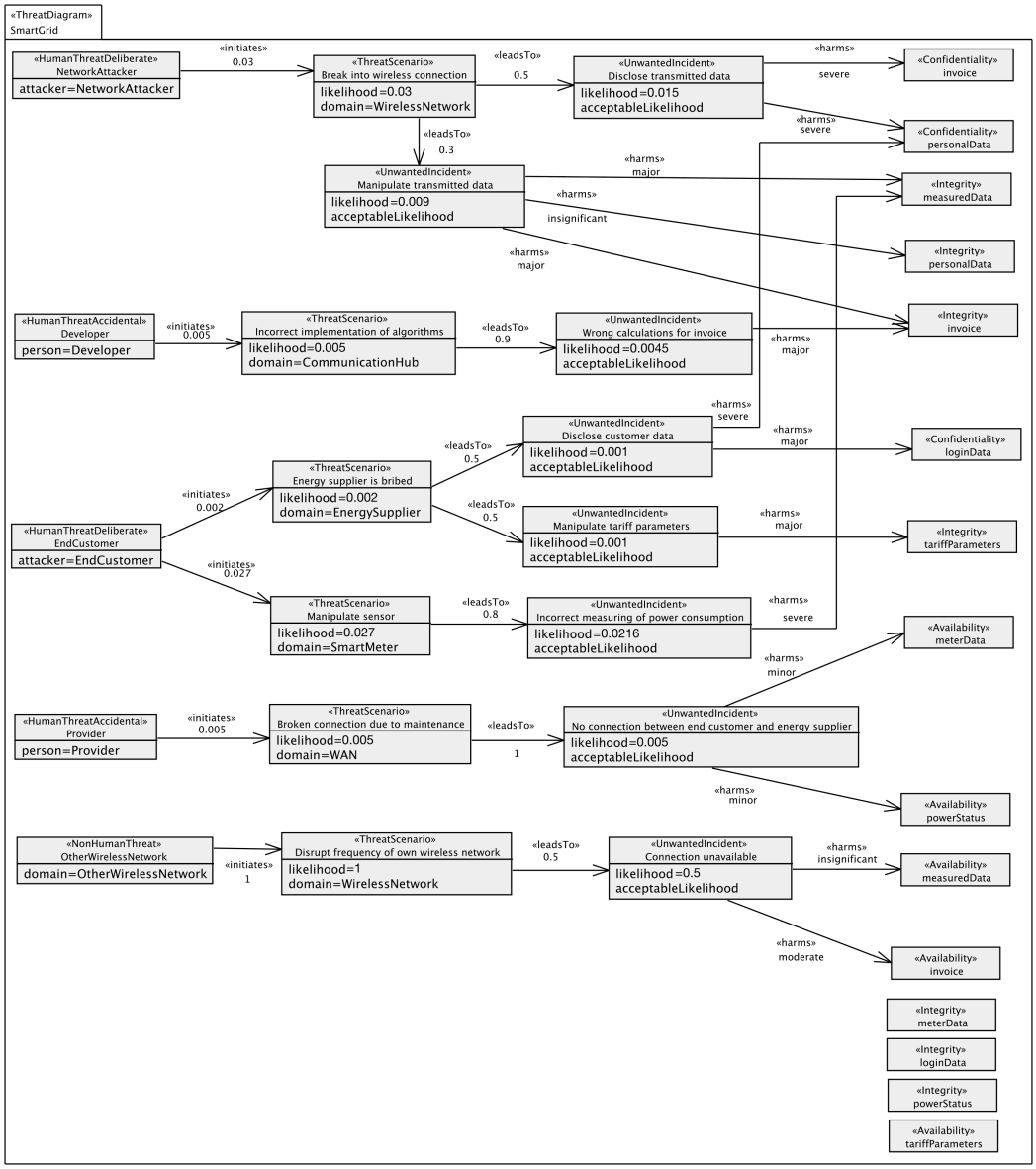


Figure 8.2.: Case Study: Annotated Threat Overview Diagram with Consequences and Likelihoods

8.4.3. Risk Evaluation

In the next step, the analysts calculate the risk for each related pair of an unwanted incident and an asset based on the risk matrix. In the risk matrix, the analysts look up the combination of the unwanted incident's likelihood and the consequence for the asset. If this is a green field, the risk is acceptable. If it is a red field, the risk is unacceptable. The results are shown in Table 8.5. For example, the unwanted incident *Disclose transmitted data* has a likelihood of 0.015 and the consequence for the asset ≪`Confidentiality`≫ *invoice* is severe.

Using the risk matrix, the risk is evaluated as unacceptable.

Unwanted Incident	Asset	Risk level
Disclose transmitted data	≪Confidentiality≫ invoice	☐acceptable ☒unacceptable
Disclose transmitted data	≪Confidentiality≫ personalData	☐acceptable ☒unacceptable
Manipulate transmitted data	≪Integrity≫ measuredData	☐acceptable ☒unacceptable
Manipulate transmitted data	≪Integrity≫ personalData	☒acceptable ☐unacceptable
Manipulate transmitted data	≪Integrity≫ invoice	☐acceptable ☒unacceptable
Wrong calculation for invoice	≪Integrity≫ invoice	☐acceptable ☒unacceptable
Disclose customer data	≪Confidentiality≫ personalData	☒acceptable ☐unacceptable
Disclose customer data	≪Confidentiality≫ loginData	☒acceptable ☐unacceptable
Manipulate tariff parameters	≪Integrity≫ tariffParameters	☒acceptable ☐unacceptable
Incorrect measuring of power consumption	≪Integrity≫ measuredData	☐acceptable ☒unacceptable
No connection between end customer and energy supplier	≪Availability≫ meterData	☒acceptable ☐unacceptable
No connection between end customer and energy supplier	≪Availability≫ powerStatus	☒acceptable ☐unacceptable
Connection unavailable	≪Availability≫ measuredData	☐acceptable ☒unacceptable
Connection unavailable	≪Availability≫ invoice	☐acceptable ☒unacceptable

Table 8.5.: Case Study: Calculated Risks

There are five out of eight unwanted incidents for which at least one unacceptable risk exists. The analysts now determine the acceptable likelihood for each of the unwanted incidents with an unacceptable risk. For example, the unwanted incident *Disclose transmitted data* may harm two different assets, and the corresponding risks are unacceptable. For the assets ≪Confidentiality≫ invoice and ≪Confidentiality≫ personalData, the consequence is severe. Using the risk matrix, the analysts derive that the acceptable likelihood for severe is 0.001. Hence, this value is added to the attribute *acceptableLikelihood* of the unwanted incident. Figure 8.3 shows the threat overview diagram with all acceptable likelihoods. An unwanted incident for which the value for the attribute *acceptableLikelihood* is not given needs no further inspection. Hence, the analysts only consider five out of eight unwanted incidents for the following steps.

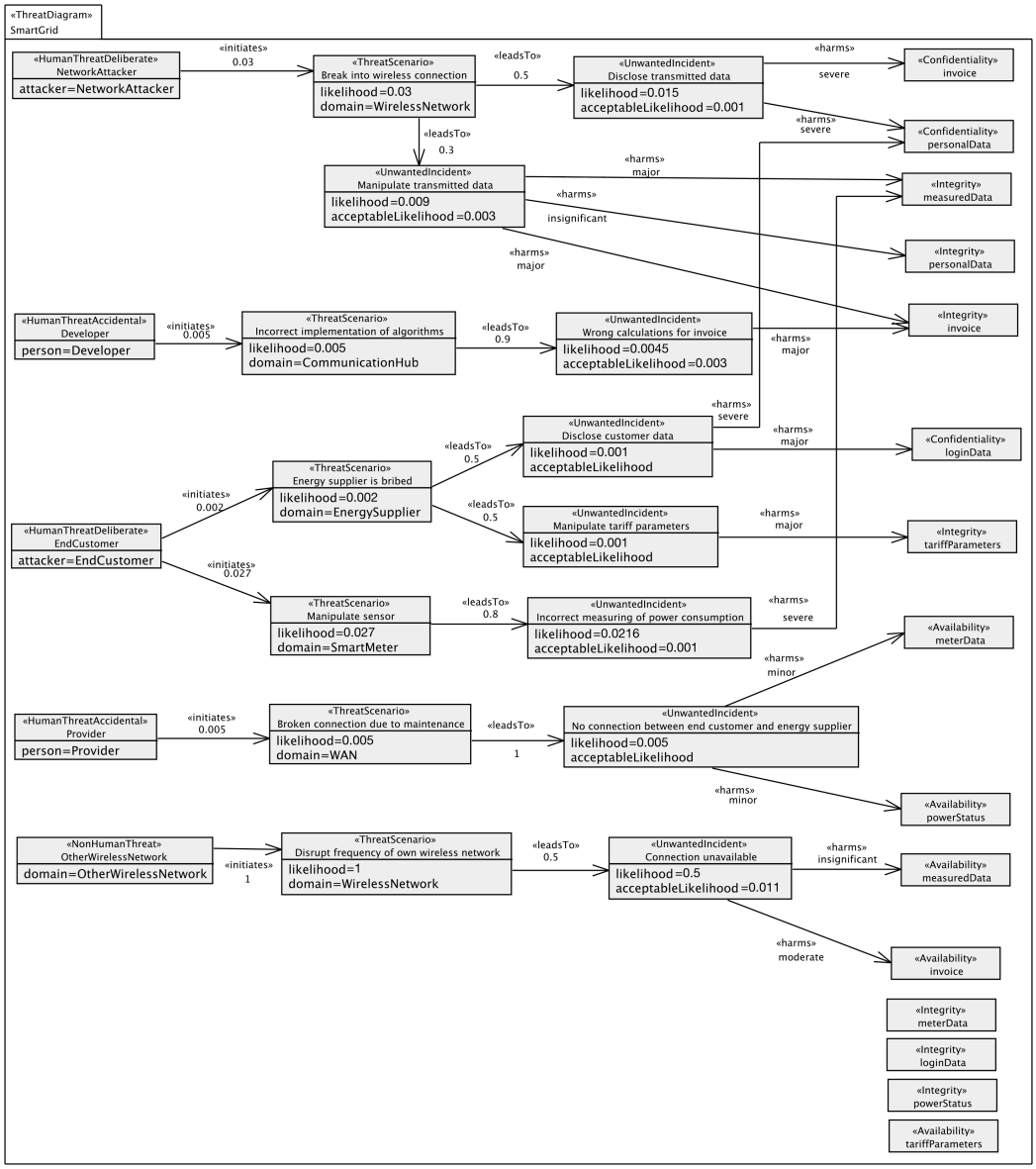


Figure 8.3.: Case Study: Threat Overview Diagram with Acceptable Likelihoods

9. Step 5: High-Level Security Requirements Instantiation

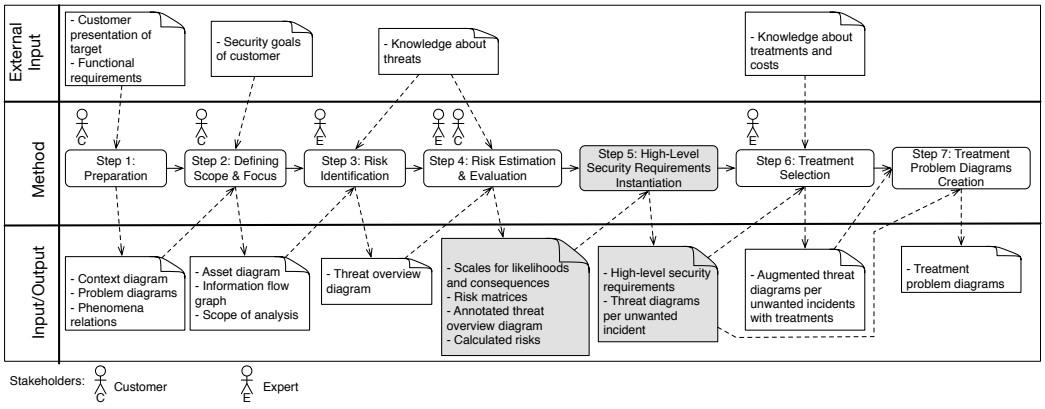


Figure 9.1.: ProCOR - Step 5

In this step, we define high-level security requirements (HL-SR) to address the risk. These requirements have to be fulfilled to ensure an acceptable risk. Based on these requirements, the causes for an unwanted incident are analyzed. An overview of the step is shown in Table 9.1. Ketil Stølen[1] has proposed the basic idea of these high-level security requirements.

Step 5: High-Level Security Requirements Instantiation	
CORAS Steps	Does not exist in CORAS.
Input	- Annotated threat overview diagram
Output	- High-level security requirements - Threat diagram per unwanted incident
Validation Conditions	- For each unwanted incident which needs treatment, a high-level security requirement is defined. - For each high-level security requirement, there is one threat diagram containing only the elements that are related to the corresponding unwanted incident.

Table 9.1.: ProCOR: Step 5

[1]SINTEF Institute Oslo

9.1. Instantiation of High-Level Security Requirements

A high-level security requirement is stated for each unwanted incident with an unacceptable likelihood. To reduce all risks to be acceptable, all of the high-level security requirements have to be fulfilled. In the following, we describe how such a requirement is derived.

A high-level security requirement has the following form:

> Ensure that the likelihood of **UI** caused by $\mathcal{TS}$ and initiated by $\mathcal{T}$ is not higher than **UI.acceptableLikelihood**.

The words printed in bold are variables which are replaced by concrete values. UI is an unwanted incident, $\mathcal{TS}$ is a set of threat scenarios and $\mathcal{T}$ is a set of threats.

To identify the elements for the requirement, we go backwards through the threat overview diagram. Starting from the unwanted incident we identify all threat scenarios which lead to the incident. This is done by considering the incoming edges with the stereotype «`leadsTo`». These threat scenarios are added to the set $\mathcal{TS}$. For each identified threat scenario, the incoming edges with the stereotype «`initiates`» are considered to identify the threats. The threats are then added to the set $\mathcal{T}$.

9.2. Threat Diagram per Unwanted Incident

Based on such a high-level security requirement, a new view on the overall threat diagram is generated. This view only contains the elements that are referred to the requirement. This view is generated for each high-level security requirement. As a threat diagram contains the harmed assets, we also add the assets that are harmed by this unwanted incident to the diagram. The threat diagram is then called *TD_UI* where UI is replaced with the name of the unwanted incident.

In the next step, the set of threat diagrams is used to identify possible treatments to fulfill the high-level security requirement. The threat overview diagram contains information that is not necessary for the treatment selection. The newly created threat diagram allows a more focused view during the treatment selection.

9.3. Validation Conditions

There are two validation conditions.

1. For each unwanted incident which needs treatment, a high-level security requirement is defined.

   A high-level security describes a risk that is unacceptable. This unacceptable risk is caused by an unwanted incident with an unacceptable likelihood. Hence, it is necessary to have such a high-level security requirement for each unwanted incident which needs treatment.

2. For each high-level security requirement, there is one threat diagram containing only the elements that are related to the corresponding unwanted incident.

   To be able to select appropriate treatments to fulfill the high-level security requirement, a focussed view on the threat overview diagram is created. It is important that all related elements are contained to ensure that all information has been considered for the treatment selection. There is exactly one diagram per high-level security requirement to avoid redundancy and consistency errors.

9.4. Case Study

The analysts have identified five unwanted incidents with an unacceptable likelihood. For those, they set up high-level security requirements as well as a threat diagram per unwanted incident.

9.4.1. High-Level Security Requirements

To be able to instantiate the high-level requirements, the analysts collect the elements of the threat overview diagram that are related to the corresponding unwanted incident.

Disclose transmitted data

$\mathcal{TS}$={Break into wireless connection}

$\mathcal{T}$ = {≪HumanThreatDeliberate≫ NetworkAttacker}

Manipulate transmitted data

$\mathcal{TS}$={Break into wireless connection}

$\mathcal{T}$ = {≪HumanThreatDeliberate≫ NetworkAttacker}

Wrong calculation for invoice

$\mathcal{TS}$={Incorrect implementation of algorithms}

$\mathcal{T}$ = {≪HumanThreatAccidental≫ Developer}

Incorrect measuring of power consumption

$\mathcal{TS}$={Manipulate sensor}

$\mathcal{T}$ = {≪HumanThreatDeliberate≫ EndCustomer}

Connection unavailable

$\mathcal{TS}$={Disrupt frequency of own wireless network}

$\mathcal{T}$ = {≪NonHumanThreat≫ OtherWirelessNetwork}

Based on the previously stated sets and the unwanted incidents, the analysts instantiate the following high-level security requirements:

HL-SR1 Ensure that the likelihood of **Disclose transmitted data** caused by **{Break into wireless connection}** and initiated by **{**≪HumanThreatDeliberate≫ **NetworkAttacker}** is not higher than **0.001**.

HL-SR2 Ensure that the likelihood of **Manipulate transmitted data** caused by **{Break into wireless connection}** and initiated by **{**≪HumanThreatDeliberate≫ **NetworkAttacker}** is not higher than **0.003**.

HL-SR3 Ensure that the likelihood of **Wrong calculation for invoice** caused by **{Incorrect implementation of algorithms}** and initiated by **{**≪HumanThreatAccidental≫ **Developer}** is not higher than **0.003**.

HL-SR4 Ensure that the likelihood of **Incorrect measuring of power consumption** caused by **{Manipulate sensor}** and initiated by **{**≪HumanThreatDeliberate≫ **EndCustomer}** is not higher than **0.001**.

HL-SR5 Ensure that the likelihood of **Connection unavailable** caused by **{Interfere frequency of own wireless network}** and initiated by **{**≪NonHumanThreat≫ **OtherNetwork}** is not higher than **0.011**.

9.4.2. Threat Diagram per Unwanted Incident

For each high-level security requirement, the analysts extract the elements from the threat overview diagram that are related to the unwanted incident.

9.4.2.1. Disclose transmitted data

The threat diagram for the high-level security requirement **HL-SR1** is shown in Figure 9.2. It is called *TD_DiscloseTransmittedData*.

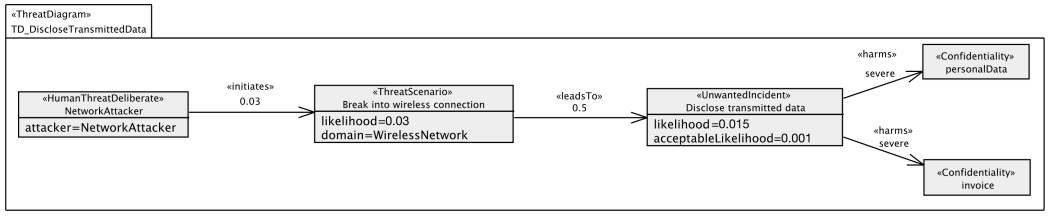


Figure 9.2.: Case Study: Threat Diagram for Disclose Transmitted Data

9.4.2.2. Manipulate transmitted data

The threat diagram for the high-level security requirement **HL-SR2** is shown in Figure 9.3. It is called *TD_ManipulateTransmittedData*.

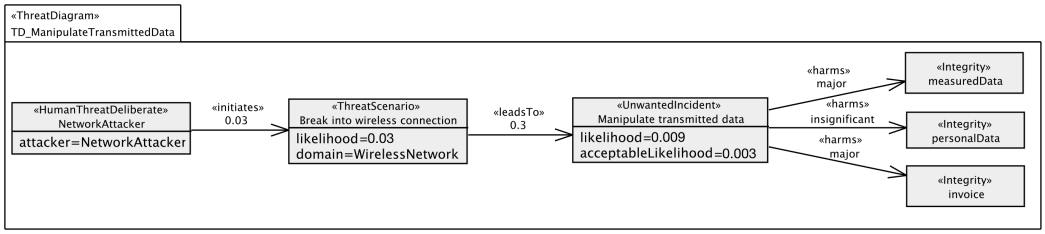


Figure 9.3.: Case Study: Threat Diagram for Manipulate Transmitted Data

9.4.2.3. Wrong calculation for invoices

The threat diagram for the high-level security requirement **HL-SR3** is shown in Figure 9.4. It is called *TD_WrongCalculationForInvoice*.

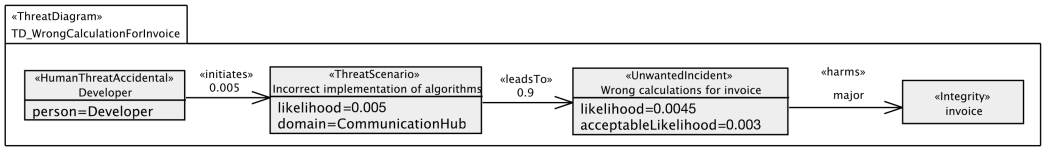


Figure 9.4.: Case Study: Threat Diagram for Wrong Calculation for Invoice

9.4.2.4. Incorrect measuring of power consumption

The threat diagram for the high-level security requirement **HL-SR4** is shown in Figure 9.5. It is called *TD_IncorrectMeasuringOfPowerConsumption*.

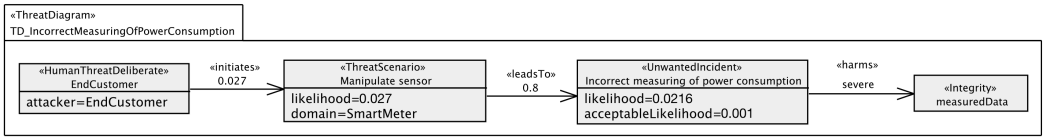


Figure 9.5.: Case Study: Threat Diagram for Incorrect Measuring Of Power Consumption

9.4.2.5. Connection unavailable

The threat diagram for the high-level security requirement **HL-SR5** is shown in Figure 9.6. It is called *TD_ConnectionUnavailable*.

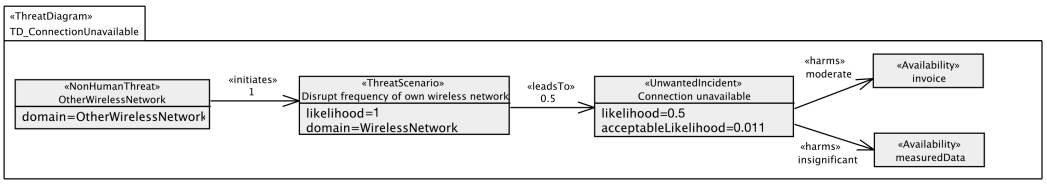


Figure 9.6.: Case Study: Threat Diagram for Connection Unavailable

10. Step 6: Treatment Selection

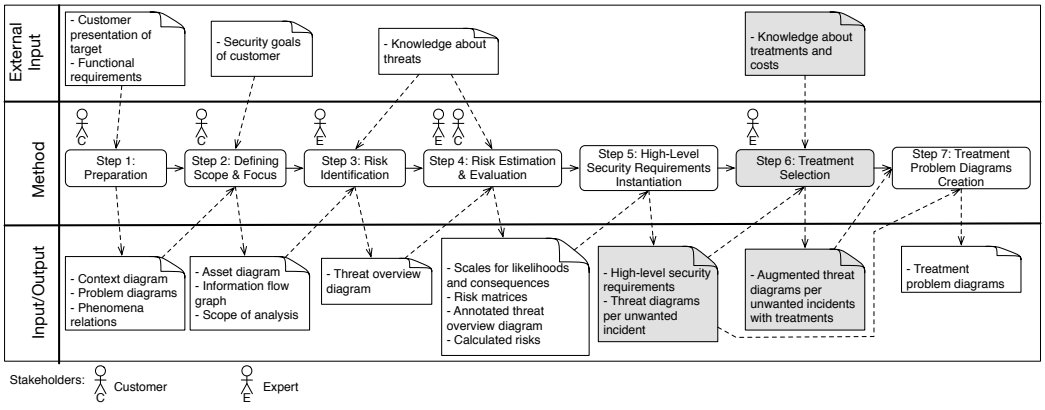


Figure 10.1.: ProCOR - Step 6

After the specification of high-level security requirements, we now consider treatments to reduce the current likelihood of an unwanted incident to be acceptable. This reduction is necessary to fulfill the high-level security requirement. To express the treatments in the model, we first extend our CORAS UML profile with a new stereotype. An overview of the step is given in Table 10.1.

Step 6: Treatment Selection	
CORAS Steps	Step 8
Input	- High-level security requirements - Threat diagrams per unwanted incident
Output	- Augmented threat diagrams per unwanted incident
Validation Conditions	- For each unwanted incident where an unacceptable likelihood exists, there is a residual likelihood that is less or equal to the acceptable likelihood. - A dependency to which the stereotype ≪`addresses`≫ is applied only points to an *initiates* or *leadsTo* relation. - A dependency on which the stereotype ≪`treats`≫ is applied only points to a threat scenario or threat. - The value for the residual likelihood is stated on all threat scenarios, unwanted incidents, initiates relations and leads to relations contained in the threat diagrams per unwanted incident.

Table 10.1.: ProCOR: Step 6

10.1. Profile Extension

In CORAS, there is a *treats* relation between a treatment scenario and a treatable element. In our approach, this relation is realized as a dependency that points to the treatable element. The element can either be a threat or a threat scenario. Other elements are not considered as being treatable. The likelihoods of these elements are derived from other likelihoods. This means that a treatment scenario needs to point to the likelihoods that are addressed by it. With the existing ≪`treats`≫ relation it is not possible to point to the likelihood to be reduced. The relation indicates only on which domain the treatment is applied. Therefore, we introduce a new stereotype called *addresses*. This stereotype can be applied to a dependency. The dependency then points from the ≪`TreatmentScenario`≫ to an association on which the stereotype ≪`initiates`≫ or ≪`leadsTo`≫ is applied. There is an attribute called *likelihoodReduction* of type *Real*. It expresses the factor to which the likelihood of the association on which it points to is reduced. For example, a likelihood reduction of *0.6* on an association with the likelihood *0.5* means that the original likelihood is reduced by 60%. This is calculated with the formula $residualLikelihood = likelihood - (likelihood * likelihoodReduction)$. A residual likelihood is the remaining likelihood on an element or association after applying a treatment. When no treatment is applied, the residual likelihood is equal to the likelihood without treatment.

The profile that provides the new stereotype is shown in Figure 10.2.

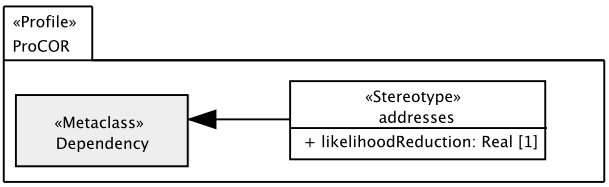


Figure 10.2.: ProCOR Profile: Addresses Dependency

10.2. Treatment Selection & Likelihood Reduction

To identify treatments, expert knowledge or experience of the analysts about best practices is needed. The selection of treatments is divided into two steps. In the first step, the threat diagrams are investigated separately to find appropriate treatments for the necessary likelihood reduction. This step is called *Initial Treatment Selection*. In the second step, the selected treatments for a domain are considered pairwise and it is decided whether a treatment subsumes another treatment. We call this step *Treatment Summary*. The threat diagrams are then updated accordingly.

10.2.1. Initial Treatment Selection

For each treatable element, the related domain is described with an attribute in the applied stereotype. It is possible to treat the annotated domains with a treatment scenario. This is indicated by adding a new treatment scenario to the threat diagram for an unwanted incident. The estimated costs of the scenario are described in the attribute *costs* of the stereotype ≪`TreatmentScenario`≫. Between the treatment scenario and the threat respectively the threat scenario that is treated, a dependency is added on which the stereotype ≪`treats`≫ is applied. It is possible that one treatment scenario treats more than one threat or threat scenario. In this case, several dependencies are added. To express which likelihood is reduced by the treatment scenario, a dependency between the treatment scenario and the association that is addressed is added. On this dependency the stereotype ≪`addresses`≫ is

applied. The likelihood reduction is expressed with the attribute *likelihoodReduction* of the stereotype. Again, it is possible that one treatment scenario addresses more than one likelihood.

After adding a new treatment scenario, the residual likelihoods are calculated. First, the residual likelihood on the addressed associations are calculated. In the case that more than one dependency with a likelihood reduction points to an association, additional knowledge is needed about how to combine the likelihood reductions. The likelihoods of the threat scenarios and the unwanted incidents are then derived as explained in Step 3 on page 53. After adding a treatment scenario, the analysts have to check if the residual likelihood is less than or equal to the acceptable likelihood. In this case, the treatment selection has been successful. Otherwise, additional treatments have to be selected.

10.2.2. Treatment Consolidation

Based on the initial treatment selection, it is necessary to decide whether a treatment is subsumed by another. Subsumed means that a treatment can be replaced by another treatment with at least the same calculated likelihood reduction for the unwanted incident. For the treatment consolidation all threat diagrams are considered.

It is only possible to replace a treatment by another if both treatments treat the same domain. Therefore, the summary is performed domain-wise. For each treatment scenario the *treats* dependency is considered, and the domain that is annotated at the corresponding treatable element is used to collect all treatment scenarios for a domain. The result of this first step is a set of all treatment scenarios for each domain that treat that domain.

Next, we only consider the sets with more than one treatment scenario. The treatment scenarios contained in a set are inspected pair-wise. If two treatments are the same or if both are contained in the same threat diagram per unwanted incident, no changes are needed. Otherwise, we distinguish two cases:

1. The *addresses* dependencies of both treatment scenarios point to the same *initiates* relation. This means that threat and threat scenario for both relations are the same. In this case, the treatment scenario is chosen that has the lower costs and whose likelihood reduction suffices to achieve the necessary risk reduction for the related unwanted incidents.

2. The *addresses* dependencies point to different relations. In this case, the experts have to decide whether one treatment can be used to replace the other to achieve a sufficient likelihood reduction. This decision is done with regard to the costs. It might be necessary to adjust the *addresses* dependencies as well, because the original dependencies of both treatments point to different relations.

The replacements are applied to the threat diagrams. The treatment scenarios are then updated in the model as well. The residual likelihoods have to be recalculated after replacing a treatment.

Remark: The selection of treatments is performed with regard to the costs. The problem how the highest reduction with the lowest costs can be achieved is an optimization problem. Solutions to such problems are not discussed in this thesis. The above-mentioned treatment consolidation is a preliminary idea how to avoid unnecessary treatments. This step has to be elaborated in more detail in the future. Tran et al. proposed a method to select treatments with regard to costs and effects [TSS13]. This method has been exemplified by CORAS but is also applicable to other risk analysis approaches.

10.3. Validation Conditions

There are four validation conditions.

1. For each unwanted incident where an unacceptable likelihood exists, there is a residual likelihood that is less or equal to the acceptable likelihood.

   To fulfill the high-level security requirement, it is necessary to have a residual likelihood that is less or equal to the acceptable likelihood for the corresponding unwanted incident. The treatments have to be selected with regard to a sufficient likelihood reduction.

2. A dependency to which the stereotype ≪`addresses`≫ is applied only points to an *initiates* or *leadsTo* relation.

   An *addresses* dependency points to an estimated likelihood. These likelihoods are annotated on an *initiates* relation or a *leadsTo* relation.

3. A dependency on which the stereotype ≪`treats`≫ is applied only points to a threat scenario or threat.

   A *treats* dependency indicates for which threat scenario or which threat the treatment has been chosen and on which domain the treatment is applied. Only threats and threat scenarios are considered as treatable elements.

4. The value for the residual likelihood is stated on all threat scenarios, unwanted incidents, initiates relations and leads to relations contained in the threat diagrams per unwanted incident.

   It is necessary to calculate all residual likelihoods in the model to ensure that all treatments have been considered and that the residual likelihoods for the unwanted incidents have been calculated correctly.

10.4. Case Study

There are five high-level security requirements with five corresponding threat diagrams. Hence, the analysts inspect each threat diagram one by one to select appropriate treatments. There are different experts with different backgrounds who take part in a meeting with the analysts to provide support for the selection.

10.4.1. Treatment Selection

The two steps of the treatment selection are performed in this section.

10.4.1.1. Initial Treatment Selection

In the following, the augmented threat diagrams are described. On the relations, there are two values separated by a slash. The left value states the likelihood without any treatment and the right value states the residual likelihood after applying the treatments.

HL-SR1 Figure 10.3 shows the augmented threat diagram for HL-SR1. There is one treatment scenario called *Use TLS to transmit data via the wireless connection*. The software of the communication hub has to be extended as well as the software of the smart meters. Therefore, the expected costs are 5.000 € for the developers and the deployment of the software update. The treatment scenario treats the threat scenario *Break intro wireless connection* and addresses the likelihood that the threat scenario leads to the unwanted incident *Disclose transmitted data*. The likelihood reduction is stated with 0.95, so there is a residual

likelihood of 0.025 for the *leadsTo* relation. There are no changes to the *initiates* likelihood and the threat scenario. The residual likelihood for the unwanted incident is 0.0008. This is lower than the acceptable likelihood. Hence, applying the treatment leads to an acceptable risk and the high-level security requirement can be fulfilled using the chosen treatment.

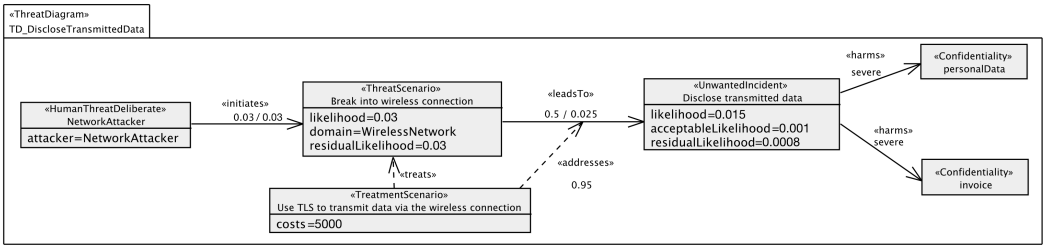


Figure 10.3.: Case Study: Threat Diagram with Treatment Scenarios for HL-SR1

HL-SR2 Figure 10.4 shows the augmented threat diagram for HL-SR2. There is one treatment scenario called *Use checksum*. The effort to implement the checksum is smaller than the effort for the implementation of the encryption, namely 1.000 €. The treatment scenario treats the threat scenario *Break intro wireless connection* and addresses the likelihood that the threat scenario leads to the unwanted incident *Manipulate transmitted data*. The likelihood reduction is stated with 0.7 so there is a residual likelihood of 0.09 for the *leadsTo* relation. There are no changes to the *initiates* likelihood and the threat scenario. The residual likelihood for the unwanted incident is 0.0027. This is lower than the acceptable likelihood. Hence, applying the treatment leads to an acceptable risk and the high-level security requirement can be fulfilled using the chosen treatment.

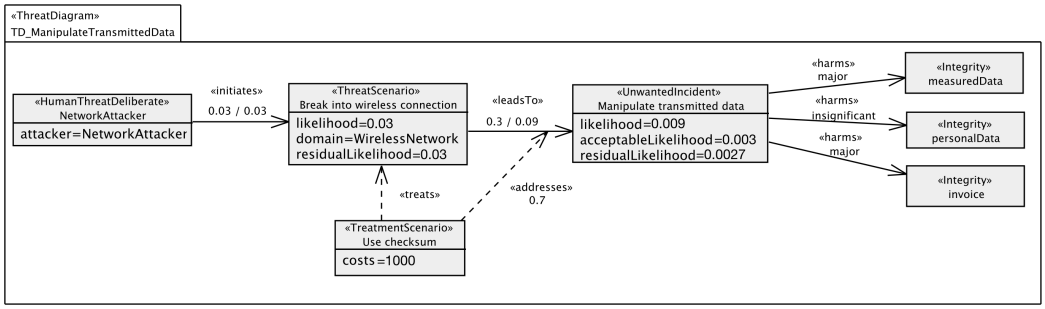


Figure 10.4.: Case Study: Threat Diagram with Treatment Scenarios for HL-SR2

HL-SR3 Figure 10.5 shows the augmented threat diagram for HL-SR3. There is one treatment scenario called *Independent definition of test cases and test algorithm* with expected costs of 10.000 €. It treats the threat scenario *Incorrect implementation of algorithms* and addresses the likelihood that the threat scenario leads to the unwanted incident *Wrong calculations for invoice*. The likelihood reduction is stated with 0.5 so there is a residual likelihood of 0.45 for the *leadsTo* relation. There are no changes to the *initiates* likelihood and the threat scenario. The residual likelihood for the unwanted incident is 0.00225. This is lower than the acceptable likelihood. Hence, applying the treatment leads to an acceptable risk and the high-level security requirement can be fulfilled using the chosen treatment.

HL-SR4 Figure 10.6 shows the augmented threat diagram for HL-SR4. There are two treatment scenarios. The first one is called *Contract with high punishments for manipu-*

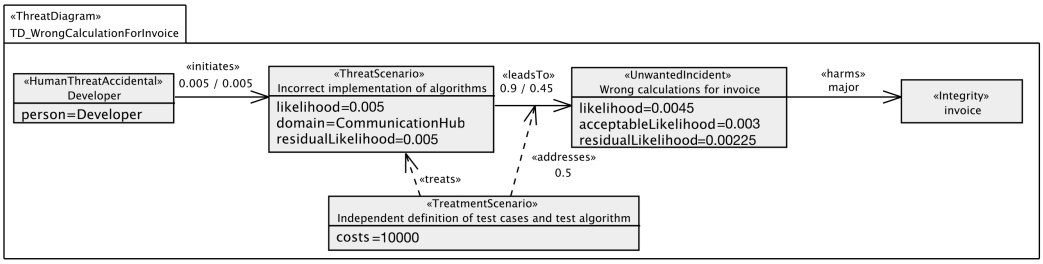


Figure 10.5.: Case Study: Threat Diagram with Treatment Scenarios for HL-SR3

lation with expected costs of 100 €. It treats the human-threat deliberate *EndCustomer* and addresses the likelihood that the threat leads to the threat scenario *Manipulate sensor*. The likelihood reduction is stated with 0.5 so there is a residual likelihood of 0.0135 for the *initiates* relation. The residual likelihood for the threat scenario is then 0.0135. The likelihood reduction of the first treatment does not suffice to fulfill the HL-SR. Therefore, it is necessary to introduce a second treatment scenario. It is called *Diverse sensors* with expected costs of 100.000 €. It treats the threat scenario and addresses the likelihood that the threat scenario leads to the unwanted incident *Incorrect measuring of power consumption*. The likelihood reduction is stated with 0.95. There is a residual likelihood of 0.04 for the *leadsTo* relation. The residual likelihood for the unwanted incident is 0.0005. This is lower than the acceptable likelihood. Hence, applying the treatments leads to an acceptable risk and the high-level security requirement can be fulfilled using the chosen treatments.

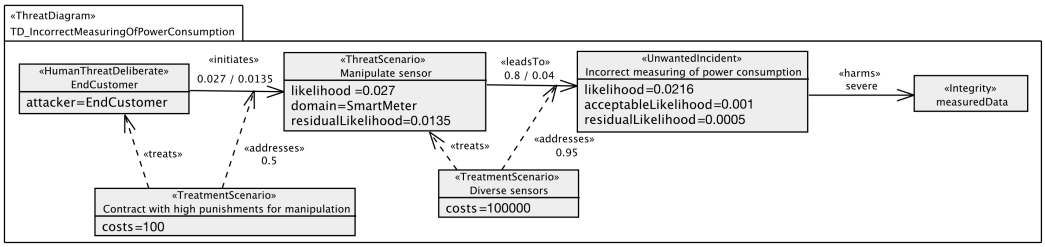


Figure 10.6.: Case Study: Threat Diagram with Treatment Scenarios for HL-SR4

HL-SR5 Figure 10.7 shows the augmented threat diagram for HL-SR5. There is one treatment scenario called *Use 5GHz band* with expected costs of 100 €. It treats the threat scenario *Disrupt frequency of own wireless network* and addresses the likelihood that the threat scenario leads to the unwanted incident *Connection unavailable*. The likelihood reduction is stated with 0.99 so there is a residual likelihood of 0.005 for the *leadsTo* relation. There are no changes to the *initiates* likelihood and the threat scenario. The residual likelihood for the unwanted incident is 0.005. This is lower than the acceptable likelihood. Hence, applying the treatment leads to an acceptable risk and the high-level security requirement can be fulfilled using the chosen treatment.

10.4.1.2. Treatment Consolidation

There is one domain that is treated by more than one treatment scenario. For the domain *WirelessNetwork* there are three treatment scenarios.

Let $\mathcal{T}_{WirelessNetwork} = \{TLS, Checksum, 5GHz\}$ be the set of all treatment scenarios for this domain. The treatment scenarios are abbreviated. There are three pairs to be investigated:

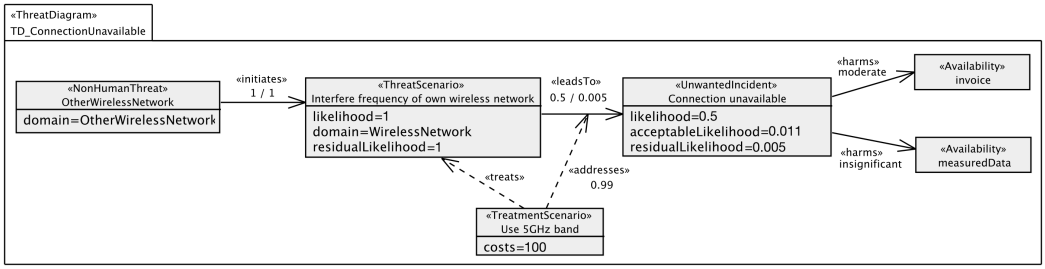


Figure 10.7.: Case Study: Threat Diagram with Treatment Scenarios for HL-SR5

{**TLS, Checksum**} The treatment scenarios do not address *initiates* relations but treat the same threat scenario. The treatment scenario *Use checksum* addresses the likelihood that the threat scenario *Break into wireless connection* leads to the unwanted incident *Manipulate transmitted data*. The encryption with TLS cannot address this relation because a checksum provides error recognition and correction. Hence, a replacement is not possible.

{**TLS, 5Ghz**} The treatment scenarios do not address *initiates* relations and they treat different threat scenarios. It is not possible to treat the threat scenario *Break into wireless connection* with this treatment scenario. Hence, a replacement is not possible.

{**Checksum, 5GHz**} The treatment scenarios do not address *initiates* relations and they treat different threat scenarios. It is not possible to treat the threat scenario *Break into wireless connection* with this treatment scenario. Hence, a replacement is not possible.

Hence, no updates are necessary for the threat diagrams. All initial treatments have to be considered.

11. Step 7: Treatment Problem Diagrams Creation

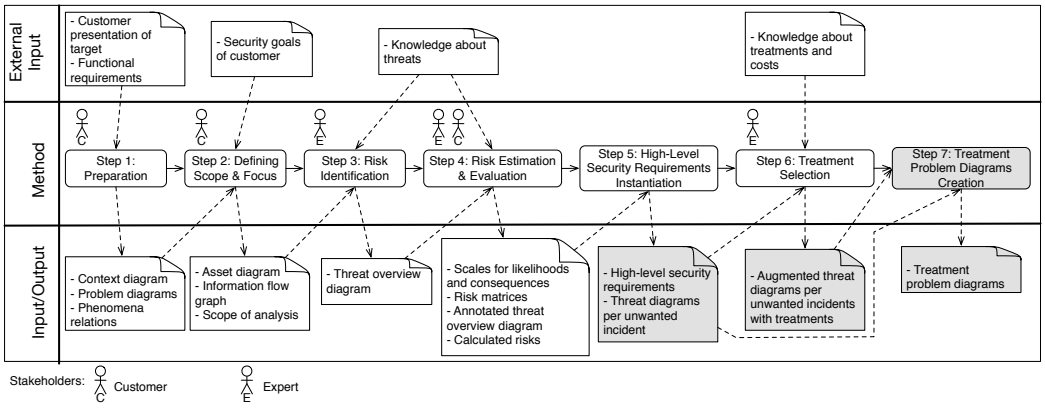


Figure 11.1.: ProCOR - Step 7

In Step 6 of the method, we described a way to identify treatments to fulfill the high-level security requirements. Currently, the treatments are only contained in the threat diagrams. To further describe the high-level security requirements, we introduce a new diagram type. This diagram type is called *Treatment Problem Diagram*. It has some similarities with problem diagrams, as indicated by the name. There is no machine in this type of diagram, but there are treatments that have to be implemented to fulfill the high-level security requirement in the same way as a machine fulfills a functional requirement. An overview of the step is given in Table 11.1.

Step 7: Treatment Problem Diagrams Creation	
CORAS Steps	Step 8
Input	- High-level security requirements - Threat diagrams per unwanted incident with treatments
Output	- Treatment problem diagram for each high-level security requirement
Validation Conditions	- There is exactly one treatment problem diagram for each high-level security requirement. - Each treatment scenario is represented as a treatment in the corresponding treatment problem diagram. - There is a connection between the treatment and the treated domain in the treatment problem diagram. - The high-level security requirement constrains domains that are treated by a treatment scenario. - The high-level security requirement refers to domains that are related to the requirement but that is not treated.

Table 11.1.: ProCOR: Step 7

11.1. Profile Extension

First, we extend the ProCOR profile with new stereotypes. The extension is shown in Figure 11.2. The profile ProCOR contains a stereotype for a domain which is called ≪`Treatment`≫. The stereotype is a specialization of the stereotype ≪`Domain`≫ as contained in the ProblemFrames profile. There is a another new stereotype called ≪`High-Level Security-Requirement`≫ which contains an attribute *unwantedIncident* to establish the link with the related unwanted incident. The stereotype is a specialization of the general term requirement contained in the ProblemFrames profile. To indicate a treatment problem diagram, the stereotype ≪`TreatmentProblemDiagram`≫ is introduced. This stereotype is a specialization of the stereotype ≪`ProblemDiagram`≫. The specialization means that a treatment problem diagram is a special kind of a problem diagram.

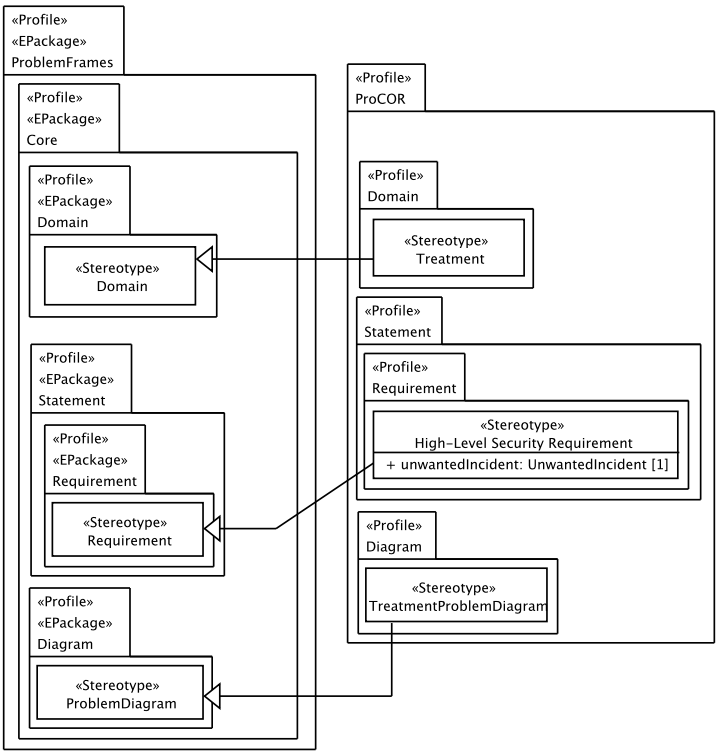


Figure 11.2.: ProCOR Profile: Treatment Problem Diagram

11.2. Creation of Treatment Problem Diagrams

The new stereotypes are used to create treatment problem diagrams. To retrieve the elements of a diagram, we consider a threat diagram *TD_UI* together with the corresponding high-level security requirement. There is one diagram for each high-level security requirement. It provides an overview of all treatments that are needed to fulfill the high-level security requirement.

The following steps have to be performed to create a treatment problem diagram:

1. Instantiate a class for the high-level security requirement on which the stereotype ≪`High-Level Security Requirement`≫ is applied.

2. Add all domains to the diagram that are related to a treatable item in the threat diagram. The related domains can be retrieved using the attribute *domain* of a threat

scenario or a non-human threat, the attribute *person* of a human-threat accidental or the attribute *attacker* of a human-threat deliberate.

3. For all treatment scenarios of the threat diagram, add a class with the stereotype ≪`Treatment`≫ to the diagram. The name of the class is derived from the description of the treatment scenario.

4. Add an interface between the treatment and the treated domain. This is done with an association and the stereotype ≪`connection`≫. The phenomena of this interface are controlled by the treatment and describe the way of how the domain is treated.

5. Add a dependency with the stereotype ≪`constrains`≫ between the high-level security requirement and the domains that are treated. Annotate the phenomena that are related to the treatable element. The behavior of the domain is changed by the requirement.

6. Add a dependency with the stereotype ≪`refersTo`≫ between the high-level security requirement and the domains that are not treated. Annotate the phenomena that are related to the treatable element. As the requirement only refers to the domain, the behavior of the domain remains unchanged.

The diagram can then be read in the following way: The high-level security requirement constrains the domains on which changes are made to fulfill it. A treatment is the needed element to fulfill the requirement. It has to ensure that the likelihood of the unwanted incident related to the high-level security requirement is indeed reduced as specified in Step 6. Domains that are referred by the requirement are somehow related to it, but their behavior and state remains unchanged.

11.3. Validation Conditions

There are five validation conditions.

1. There is exactly one treatment problem diagram for each high-level security requirement.

   To be able to identify the treatments that have to be applied to fulfill a high-level security requirement, a treatment problem diagram is given. All treatments that are related to the high-level security requirement are contained in one diagram.

2. Each treatment scenario is represented as a treatment in the corresponding treatment problem diagram.

   To fulfill a high-level it is necessary to apply all treatments. The treatment problem diagram has to contain all treatment scenarios in form of a class with the stereotype ≪`Treatment`≫.

3. There is a connection between the treatment and the treated domain in the treatment problem diagram.

   To explain how a treatment treats a domain, a connection between the treatment and the domain is introduced.

4. The high-level security requirement constrains domains that are treated by a treatment scenario.

   The *constrains* dependency indicates that a treatment has to be applied to the domain to fulfill the high-level security requirement.

5. The high-level security requirement refers to domains that are related to the requirement but that is not treated.

   The *refersTo* dependency indicates that the domain is related to the requirement but there is no treatment which treats the domain.

11.4. Next Steps

The treatment problem diagrams need to be considered in the next steps of the software development process. We distinguish three types of treatments:

- The first type are treatments which are only related to changes in the software. For example, there is a treatment that describes an encryption for a wireless connection. This treatment has to be implemented by the software engineers.
- The second type are treatments which cannot be assured by the software. For example, there is a treatment describing a special education for the employees. This treatment is then considered as domain knowledge, in particular, as an assumption.
- The third type are treatments that are related to some technical equipment used by the software. For example, there is a treatment that describes the use of diverse sensors. This treatment cannot be completely ensured by the software. Adjustments might be needed in the software to make use of the technical equipment. In this example, we need domain knowledge about the sensors and adjustments for the software, e.g. the replacement of the measuring component.

The topic of the thesis is the elicitation of the security requirements. Hence, it is not elaborated how they are forwarded to the following steps of the software development process. All information about necessary treatments are stored in the model. There are links to the domains on which the treatment is applied. Using this information, it is easy to select all treatments for a given domain that have to be considered in the following phases.

11.5. Case Study

The analysts have identified five high-level security requirements. Considering the related threat diagrams and treatment scenarios, they instantiate five treatment problem diagrams.

11.5.1. HLSR-1

The treatment problem diagram for the high-level security requirement HL-SR1 is shown in Figure 11.3. The diagram is called *DiscloseTransmittedData*. There are three domains contained in the diagram. The biddable domain *NetworkAttacker* is referred by the requirement because the domain is not treated. The connection domain *WirelessNetwork* is constrained by the requirement. The last domain is a treatment called *TLS*. It treats the domain *WirelessNetwork*. Hence, there is a connection between the treatment and the connection domain that describes that the treatment is used to encrypt the transmitted data.

11.5.2. HLSR-2

The treatment problem diagram for the high-level security requirement HL-SR2 is shown in Figure 11.4. The diagram is called *DiscloseTransmittedData*. It contains three domains. The biddable domain *NetworkAttacker* is referred by the high-level security requirement. There are no treatments for this domain. The connection domain *WirelessNetwork* is constrained by the high-level security requirement. There is a treatment in the diagram that treats the connection domain. A checksum is used to recognize unwanted changes of the transmitted data.

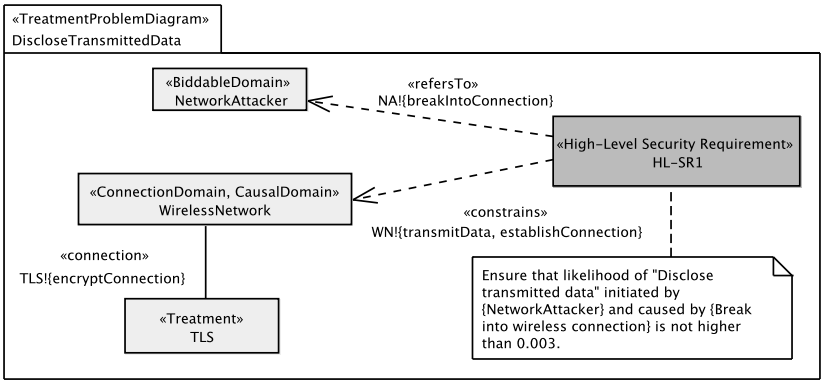


Figure 11.3.: Case Study: Treatment Problem Diagram for HL-SR1

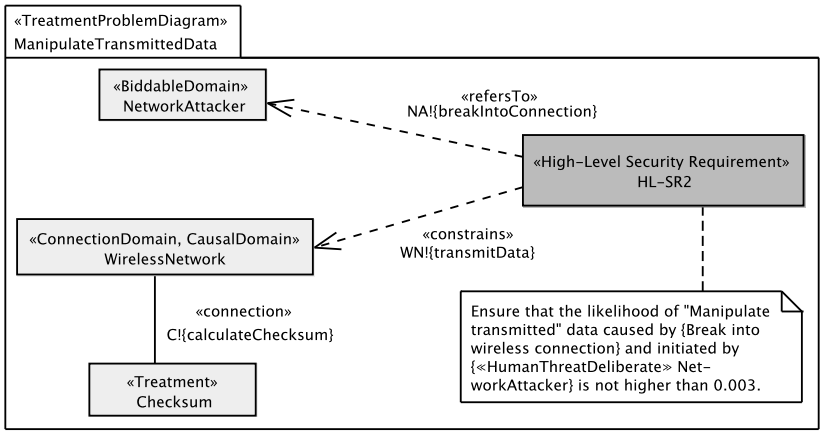


Figure 11.4.: Case Study: Treatment Problem Diagram for HL-SR2

11.5.3. HLSR-3

The treatment problem diagram for the high-level security requirement HL-SR3 is shown in Figure 11.5. The diagram is called *DiscloseTransmittedData*. It consists of three domains. The biddable domain *Developer* is referred by the requirement. Hence, there is no treatment for this domain. The machine *CommunicationHub* is connected to the treatment *Testing*. The algorithms of the software are tested with regard to their correctness. The machine is constrained by the high-level security requirement.

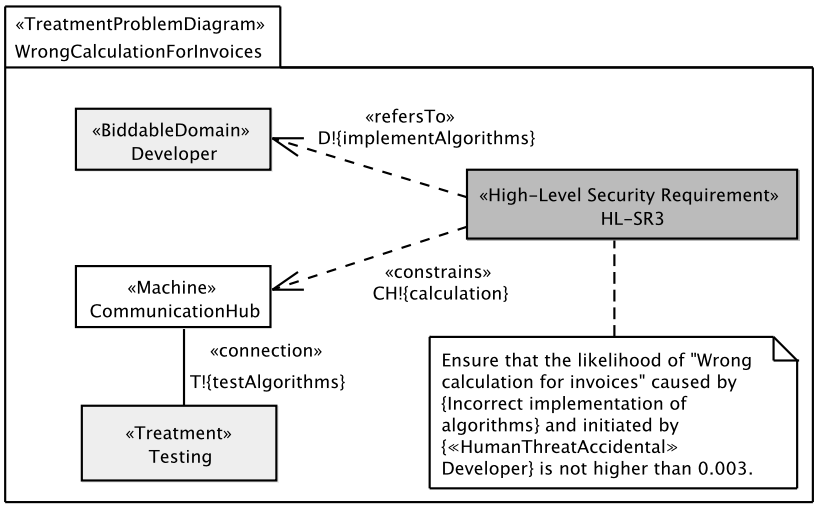


Figure 11.5.: Case Study: Treatment Problem Diagram for HL-SR3

11.5.4. HLSR-4

The treatment problem diagram for the high-level security requirement HL-SR4 is shown in Figure 11.6. The diagram is called *DiscloseTransmittedData*. This diagram contains four domains. There is a biddable domain *EndCusomter*, a causal domain *SmartMeter* and two treatments *Contract* and *DiverseSensor*. The domain *EndCustomer* is connected to the treatment *Contract* and constrained by the high-level security requirement. The treatment is a contract with high punishments in case that the customer manipulates the sensor. The domain *SmartMeter* is connected to the treatment *DiverseSensor* and is constrained by the high-level security requirement. The treatment describes the usage of diverse sensors to measure the power consumption. This makes it more difficult to manipulate all sensors at the same time.

11.5.5. HLSR-5

The treatment problem diagram for the high-level security requirement HL-SR5 is shown in Figure 11.7. The diagram is called *DiscloseTransmittedData*. It contains three domains. The causal domain *OtherWirelessNetwork* is referred by the high-level security requirement. Hence, there is no treatment for this domain. The connection domain *WirelessNetwork* is constrained by the high-level security requirement and connected to the treatment *5GHz Band*. Using this band for the transmission, there are less interferences for the wireless network.

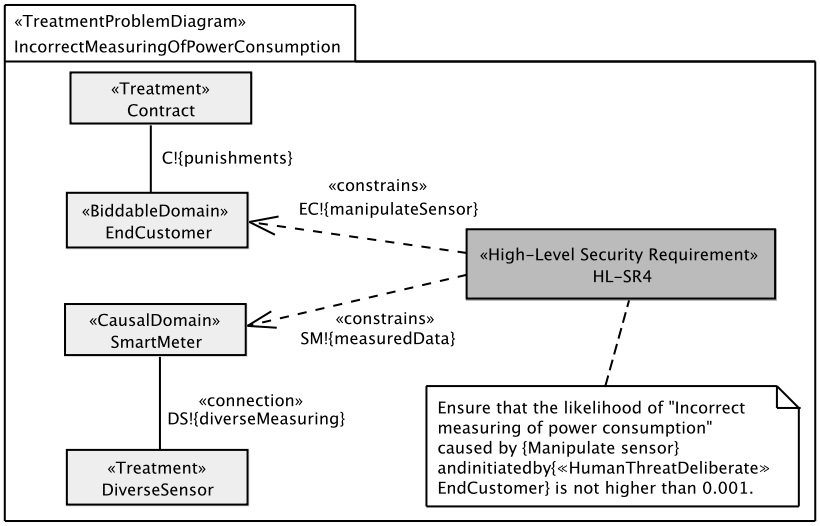


Figure 11.6.: Case Study: Treatment Problem Diagram for HL-SR4

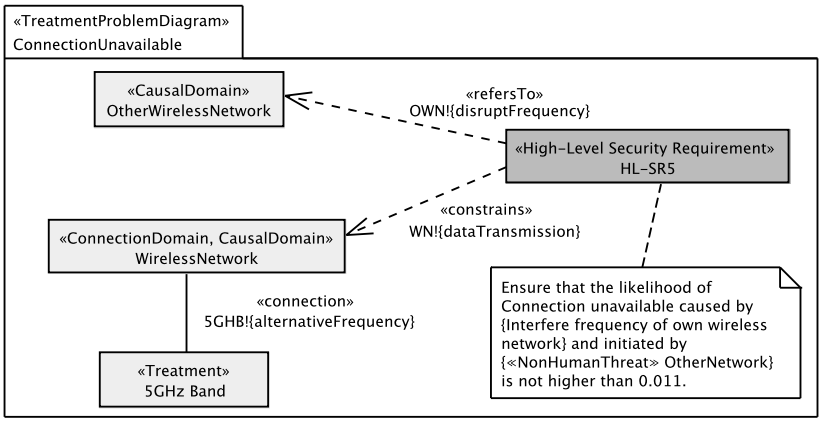


Figure 11.7.: Case Study: Treatment Problem Diagram for HL-SR5

12. Tool Support

In this chapter, we sketch up some ideas for a tool support that supports the user to apply the ProCOR method. There is one section for each step of the method. The mock-ups are only used to explain the main idea of the different modules of the tool. Hence, they are not meant as a final design.

12.1. Environment

Based on Eclipse[1] and Papyrus UML[2], Côté et al. have developed a tool support for UML4PF [CHSH11]. The tool is based on UML class diagrams and provides OCL (Object Constraint Language) statements to check some of the validation conditions. Rene Meis[3] has developed a newer version of some parts of the tool. There are wizards for the diagram generation, but validation conditions are not yet implemented. The wizards are implemented using Epsilon which is an open source script language for the Eclipse environment. With this language, it is possible to create small graphical user interfaces for the wizards. The UML diagrams are well readable for computer scientists and OCL expressions can be easily implemented. The diagrams are stored in the Eclipse Modeling Framework (EMF). EMF is a common format for all modeling tools based on Eclipse. Eclipse, Papyrus and Epsilon have in common that they are Open Source. They can be distributed free of charge. The disadvantage is that the wizards are very slow when the models become bigger. The lack of performance is a problem we have to address in the development of a support tool for ProCOR.

The CORAS tool is based on Sirius[4] which is also based on Eclipse and EMF. With Sirius, it is possible to create one's own modeling tool. For example, in CORAS there is a graphical language and a meta model to describe the relations between the elements. When drawing a CORAS diagram, the meta model is used to check the relations between the models. In Papyrus one can draw wrong diagrams and the correctness is checked afterwards with the OCL conditions. In Sirius it is not possible to draw diagrams with a wrong syntax. This makes it easier for the user to make use of the language. To connect CORAS with UML4PF, we will make use of the CORAS meta model and extend this with new elements for UML4PF.

In the following, we propose a UML4PF graphical language as well as some basic ideas about tool support for the different steps of the ProCOR method.

[1] Eclipse IDE - `http://eclipse.org`
[2] Papyrus UML - `https://eclipse.org/papyrus/`
[3] Rene Meis - `rene.meis@uni-due.de`
[4] Eclipse Sirius - `https://www.eclipse.org/sirius/`

12.2. UML4PF Language

To represent the different domains of an UML4PF diagram, we propose a graphical notation. There is one symbol for each domain type. The graphical notation is shown in Figure 12.1. The name of the domain can be annotated in the ellipse.

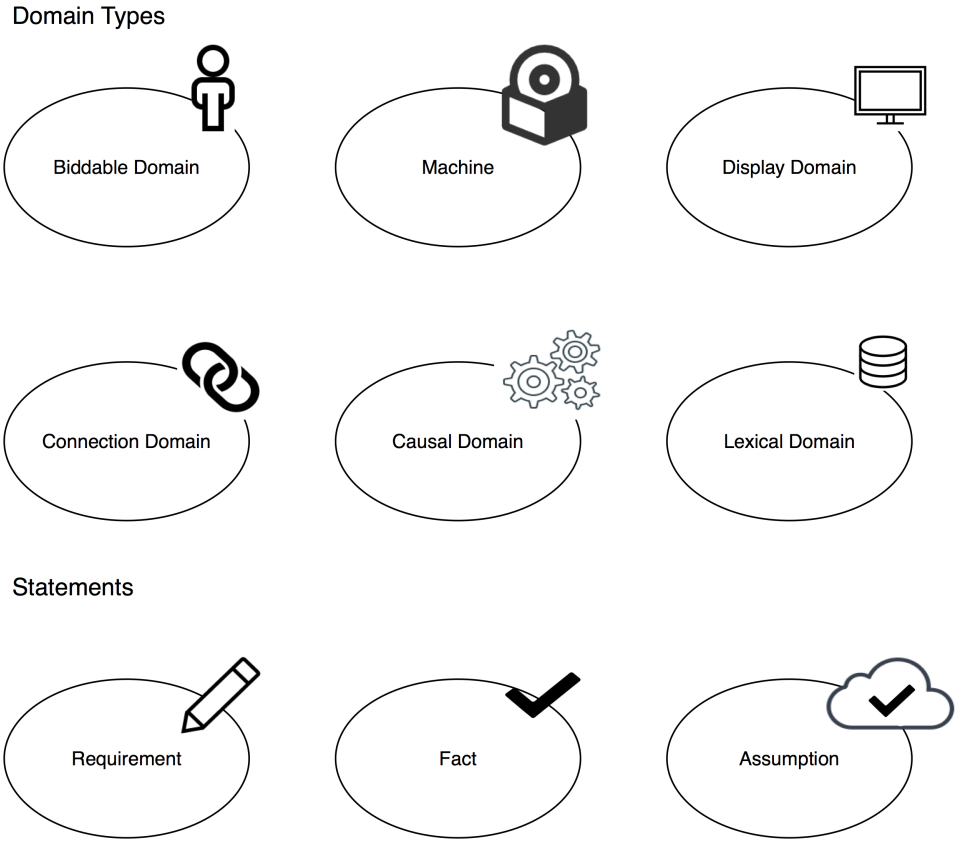


Figure 12.1.: UML4PF Language: Graphical Notation

There are also three icons for the different statements. A requirement is represented by a pencil as it is a task which has to be fulfilled by the software engineers. A fact is represented by a check mark as it is naturally given. An assumption is represented as check mark contained in a cloud as it should be true but cannot be checked by the software.

In the original Jackson notation [Jac01], the dependencies between a requirement and the domains are realized with dashed arrows. We will adapt this approach for our language. The interfaces between the domains are realized with a line and a letter. In a comment, the phenomena at the interfaces are annotated. This is very similar to Jackson's notation and the notation of the current UML4PF UML diagrams.

Figure 12.2 shows the problem diagram for the requirement *R1: Setup* of the case study. The UML diagram for this requirement is shown on page 35.

To ensure the syntactical correctness of the diagrams, we will develop a meta model to express all relations between the domains and statements.

12.3. Combining UML4PF and CORAS

Using Sirius, we do not have UML profiles any longer. Hence, it is necessary to introduce a new meta model. All relations which are already stated in the meta models of CORAS and

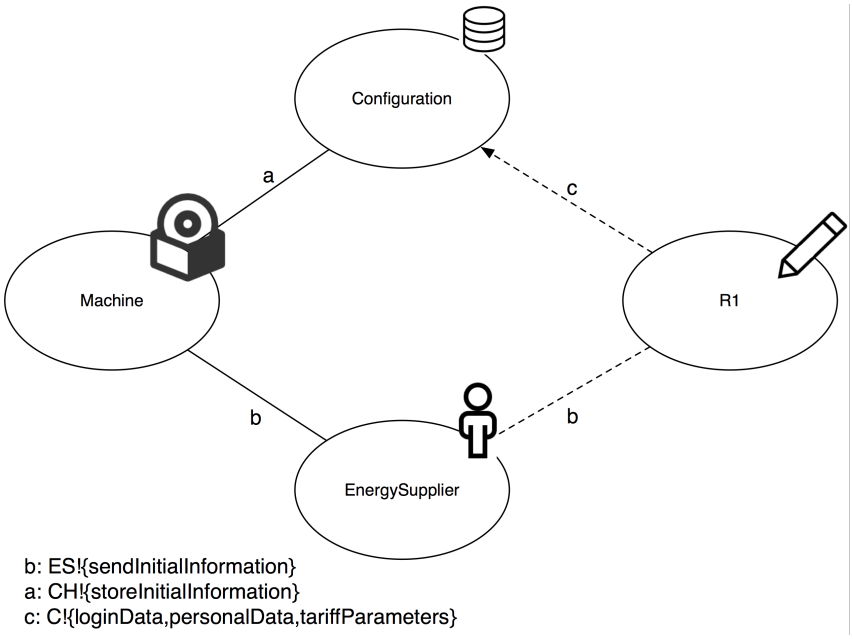


Figure 12.2.: UML4PF Language: Problem Diagram Example

UML4PF are still considered. Additionally, we introduce relations between the elements of both languages. For example, there is a relation between a threat scenario and a domain. All relations which are expressed in the CORAS UML profile on page 26 are contained in the meta model.

12.4. ProCOR Tool

In this section, we state some ideas for the tool support of our method. If possible, we provide a mock-up of how the interaction with a user of the tool might look like. The section is structured stepwise. There is one description for each step of the method.

12.4.1. Step 1: Preparation

In this step, it is necessary to create problem diagrams and the phenomena relations. To do so, we propose two wizards. The first one is for creating the problem diagrams. The diagrams are represented with the UML4PF language. The second one is used to express the phenomena relations.

Figure 12.3 shows a dialog to support the creation of a problem diagram. The user can enter the name of the problem diagram and the name of the machine. There is a list of domains that are already contained in the model. It is possible to select those domains that are constrained or referred by the requirement. It is also possible to add a new domain. On the right-hand side, the user can enter the name of the requirement together with a textual description. The dialog can be called from the top menu of the tool. It is possible to add phenomena on the interfaces by right-clicking on the connections between two domains.

Figure 12.4 shows the dialog to state the phenomena relations. On the left-hand side, there is a list of all phenomena contained in a model. There is another list which only contains the symbolic phenomena. The user can select a phenomenon in the left list and is then able to select all symbolic phenomena that are contained in it. On the right-hand side, it is possible to add new symbolic phenomena. The information is then stored in the model. It is not yet investigated if we need a graphical representation of these relations. If

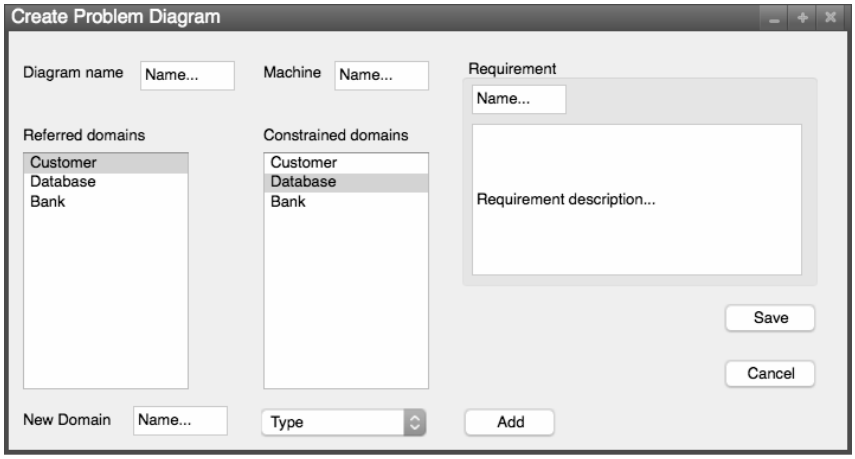


Figure 12.3.: Tool Support: Create Problem Diagram

we need such a representation, we will extend the UML4PF language with new elements for both types of the phenomena.

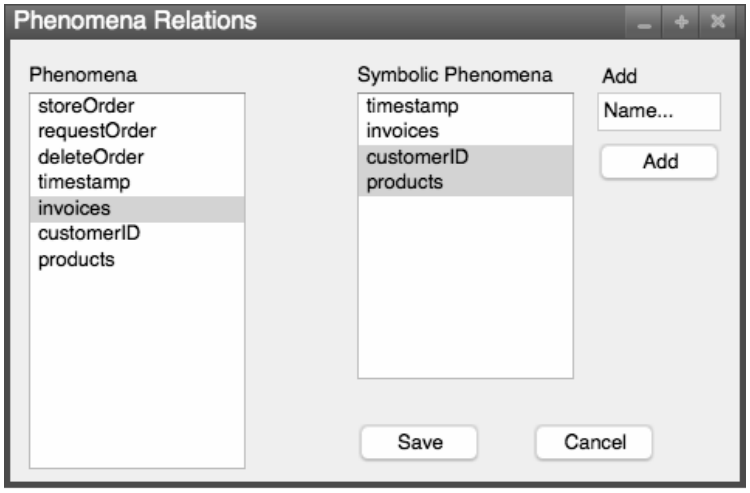


Figure 12.4.: Tool Support: Phenomena Relations

12.4.2. Step 2: Defining Focus & Scope

The asset selection is based on the phenomena relations. There is another dialog in which the user is able to select the security properties for a symbolic phenomenon. The related symbolic phenomena are also shown in this dialog. After saving the results, an asset diagram is created and shown to the user. To be able to identify indirect assets, the dialog has to be extended as the method itself.

The information flow graph is created automatically based on the information of the model. Hence, we do not need a dialog for this.

For each domain that is related to a node of the information flow graph, the user can decide if this domain shall be in the scope of the analysis. There is a list of all domains and the user can select the domains to be in the scope. This dialog is shown Figure 12.6. There might be a long list of domains depending on the size of the model. There will be some

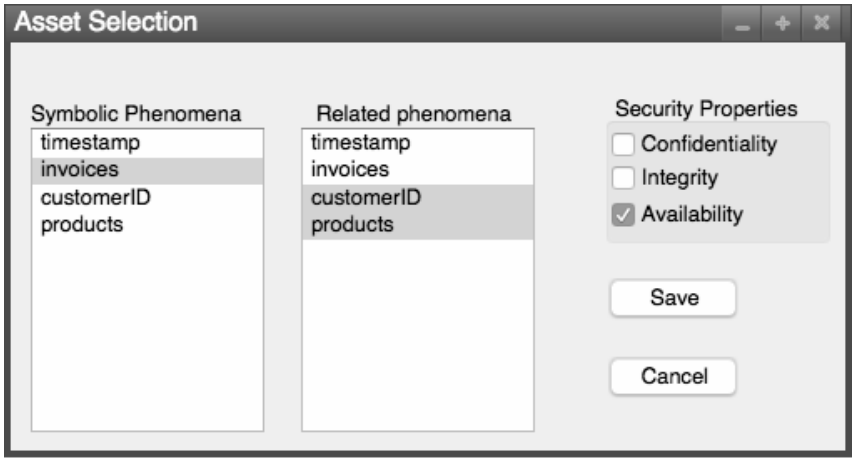


Figure 12.5.: Tool Support: Asset Selection

kind of selection criteria, e.g. based on the domain type, to enable the user to filter specific domains from the list.

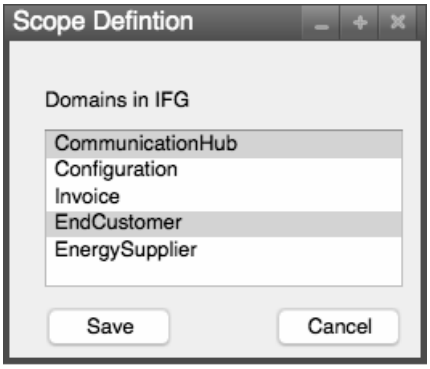


Figure 12.6.: Tool Support: Scope Definition

12.4.3. Step 3: Risk Identification

The risk identification is the most complex part of the tool support. The questionnaire has to be asked for each domain that is in scope. Hence, there will be a dialog that presents the user each question and adds the elements to the threat overview diagram.

As explained in Section 14.3.6 on page 114, our vision is to identify possible threats by analyzing the problem frames the subproblems fit to. In this case, there is a dialog that proposes threats and threat scenarios for a domain in scope. The user then can select the desired threat and they are added to the threat diagram automatically.

12.4.4. Step 4: Risk Estimation & Evaluation

To estimate the likelihoods and consequences, we provide two dialogs. The first one to estimate the likelihoods is shown in Figure 12.7. There is a list of all *initiates* relations. The user can select one relation and add the value of the likelihood. The same is available for all *leads to* relations. The likelihoods for threat scenarios and unwanted incidents are then calculated automatically.

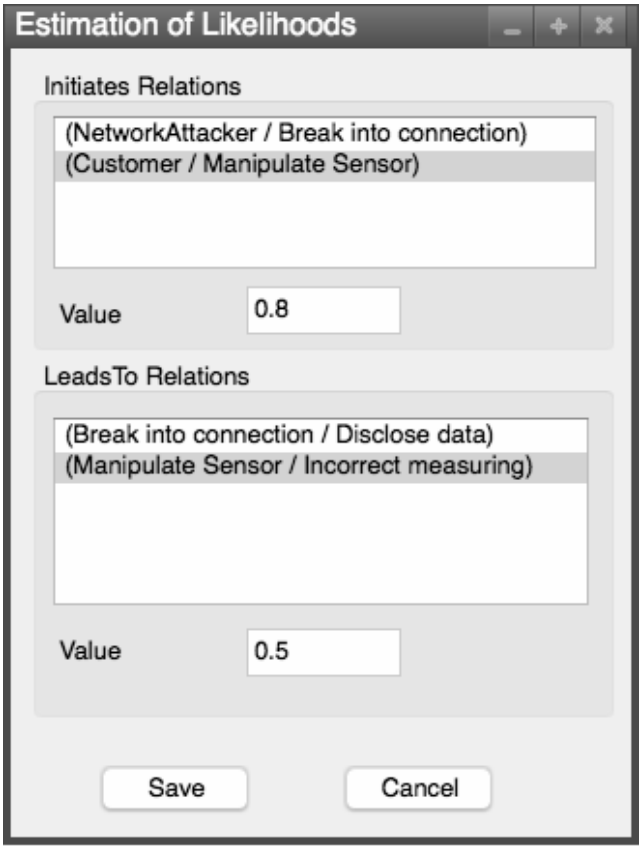


Figure 12.7.: Tool Support: Estimation of Likelihoods

The dialog to estimate the consequences is shown in Figure 12.8. There is a list of all combinations of unwanted incidents and assets. The value is chosen according to the consequence scale. For the dialog, we assume that the user does not express a consequence with a numerical value.

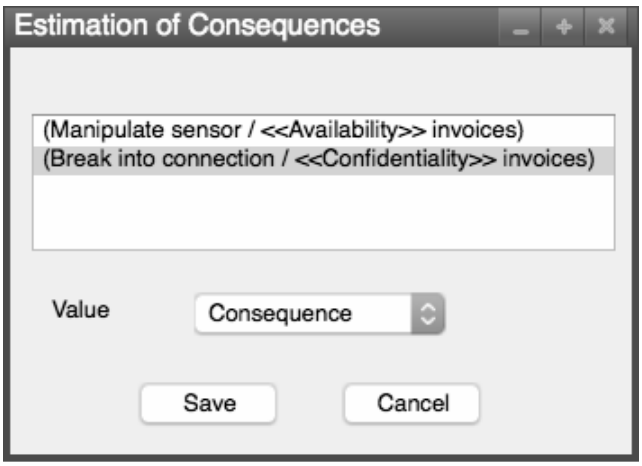


Figure 12.8.: Tool Support: Estimation of Consequences

To be able to evaluate the risk automatically, it is necessary to embed the risk matrix in the model. This is not done until now. We plan that the list of unwanted incidents with an unacceptable risk is created automatically.

12.4.5. Step 5: High-Level Security Requirements Instantiation

The high-level security requirements are generated automatically based on the risk evaluation. To express them in the model, we introduce a new element for the UML4PF Language. This new element is shown in Figure 12.9. The textual description of the high-level security requirement is generated using a text pattern in which the related elements of the threat overview diagram are added. The corresponding threat diagram per unwanted incident is generated dynamically as a new view when the user selects a high-level security requirement from a list. This selection is also the starting point for the treatment selection.

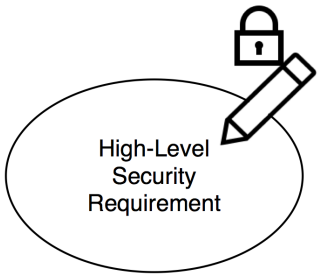


Figure 12.9.: UML4PF Language: Symbol for HL-SR

12.4.6. Step 6: Treatment Selection

There is a list of all high-level security requirements. When the user selects one, the corresponding threat diagram is shown. The user can right-click on a threat or threat scenario to add a treatment. The dialog to add a new treatment is shown in Figure 12.10. The user can enter the name of the treatment and the costs. All relations that can be addressed by the treatment are contained in the list. The user can select relations and can enter the values for the likelihood reduction. The treatment scenarios are then added to the model and are shown in the diagram.

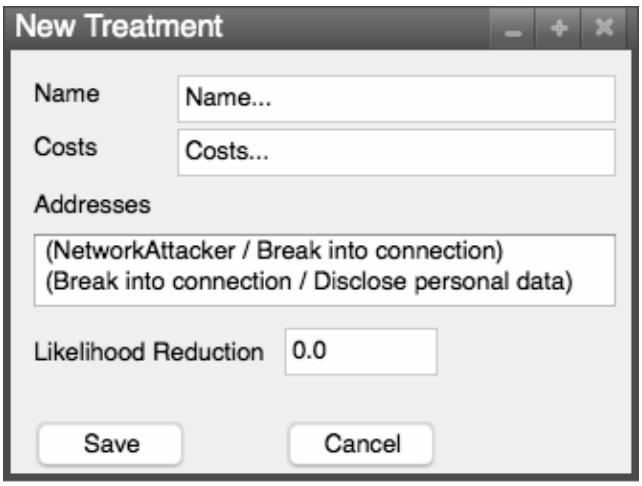


Figure 12.10.: Tool Support: New Treatment

12.4.7. Step 7: Treatment Problem Diagrams Creation

The treatment problem diagrams are again created automatically based on the information contained in the model. The user can select a command in the top menu to create all diagrams. They are represented with the UML4PF Language. For a treatment domain, there is a new symbol shown in Figure 12.11.

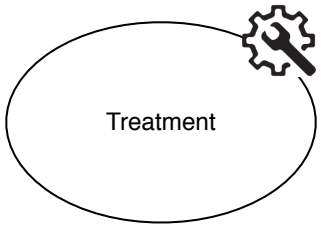


Figure 12.11.: UML4PF Language: Symbol for Treatment Domain

12.5. Plugins

As UML4PF can be used for different approaches, the tool will have the possibility to create plugins. The tool for the ProCOR method is one plugin. There might be also a plugin for a privacy analysis or something else. The vision is that parts of the ProCOR method can be used for other plugins. For example, a threat analysis is also interesting for a privacy analysis.

Therefore, it is necessary to define an API. We will investigate if Sirius already provides plugin support like in Eclipse.

Part III.

Conclusion

13. Related Work

In this chapter we provide an overview of related work. For this overview we make use of scientific papers and standards which address security risk management or the definition and elicitation of security requirements.

The ISO 31000 standard [ISO09] has already been mentioned before to explain the term *risk*. This standard also provides a description of risk management. Risk management is defined as "Coordinated activities to direct and control an organization with regard to risk". Figure 13.1 shows an overview of the steps for a risk management process. The names of the steps are similar to the steps of the ProCOR method. The first step *Establish the context* and the last step *Treat risk* are not part of the risk assessment process in which the risk is identified and evaluated. All steps of the risk management process have to be monitored and reviewed to ensure the up-to-dateness and that changes of the system have been considered. The original CORAS method is based on the description provided in the ISO 31000 as well.

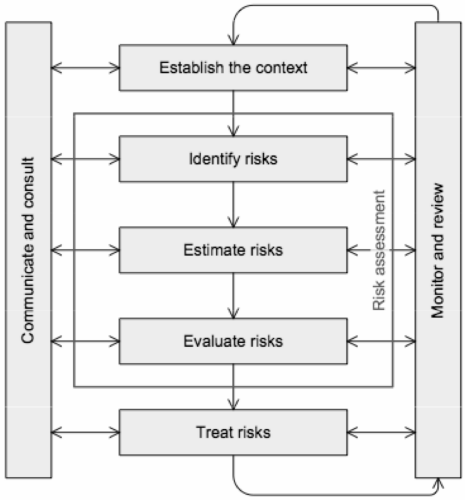


Figure 13.1.: Risk Management According to ISO 31000 [ISO09]

Alebrahim et al. [AFH+15] propose a pattern-based method to conduct a risk management for cloud computing systems. This method complies to the ISO 27001 standard [II05]. The ISO 27001 describes an Information Security Management System (ISMS). An ISMS is a set of rules and policies concerned with information security management. An organization should design, implement and apply such a set to ensure an acceptable level of risk.

In a first step of the method the cloud system is analyzed and the results are documented in a *Cloud System Analysis Pattern (CSAP)*. All elements contained in the CSAP are considered as assets and are refined further using compositions or specializations. To identify threats and vulnerabilities the method makes use of threat patterns that are based on the results of the research of the Cloud Security Alliance (CSA) and others. As the method is specialized for cloud systems, the catalogue of threats is more concrete than it is possible for the ProCOR method which is applicable to all kinds of systems. The catalogue is structured based on the CIA properties. Next, the risk is assessed. To do so, there are scales for the business impact and failure likelihoods. The risk is estimated for each asset and it has to

be decided whether is is acceptable or not. The security requirements are instantiated using *Security Requirements Patterns* which are textual and based on the risk estimation. The security requirements themselves do not talk about an acceptable likelihood or consequence. Treatments are called *controls* in the method as defined in the ISO 27001 standard. There are textual patterns to describe the controls and how they treat an asset. Last, a documentation about the risk analysis is generated. In contrast to the ProCOR method, the method proposed by Alebrahim et al. is applied to existing cloud systems.

Faßbender et al. propose the PresSuRE method [FHM14]. We make use of this work to adjust the problem diagrams. The method provides a process to elicit security requirements. In contrast to the ProCOR method where some kind of information represented as symbolic phenomena is considered as assets, PresSuRE aims to identify domains as assets. The starting point of the process are functional requirements. There is an elicitation process to identify entities which are allowed to access the assets as well as an elicitation of attackers and their abilities based on attacker templates. Security experts are involved in this step. The security requirements are derived from graphs that are created based on the information about the functional requirements and the elicited knowledge about attackers. The process does not cover a risk estimation or risk evaluation. The selection of treatments is not part of the method as well.

Beckers et al. [BHSS14] propose a method to establish an ISMS based on CORAS. Three steps of the CORAS method have been modified to comply with the ISO 27001 standard [II05]. The first step *Establish the context* is modified to focus on information security and to ensure that all parts of the description required by the standard are contained. For each asset, there is a prioritization and an asset owner. The high-level risk analysis of the original CORAS method is extended with focus on the CIA properties. There are attacker templates to describe attackers. The standard also requires the consideration of legal aspects such as laws. These aspects are also covered in the first step. The second step *Identify Risk* has also been modified. The attacker description of the first step is refined using threat diagrams. The CORAS notation is extended for specifying attacker types and to relate elements contained in the CORAS diagrams to the initial target description. The last step *Treat Risk* has been modified as well. Again, the original CORAS notation is extended to establish links to the initial target description. There is a mapping from the attacker types to the ISO 27001 controls. Controls that are described in the standard but that are not applied to the system are documented in a treatment overview table together with a reasoning. All the information which has been elicited with ISMS-CORAS is used to create a documentation that complies to the ISO 27001 standard.

The links between the initial target description and the diagrams of the later steps are similar to the ProCOR method. Our method does not comply to any standard. The provided description of ISMS-CORAS for the compliance to ISO 27001 can be used to make the ProCOR method compliant to the standard as well.

The ISO 27005 standard [II11] deals with risk management for information security. It supports the general concepts of the ISO 27001 standard. It does not describe a specific risk management process. There is a sequence of activities and recommendations how such a process shall be performed. The standard distinguishes qualitative and quantitative risk assessment methods. The CORAS method distinguishes between qualitative and quantitative scales as well. The standard describes a strong interaction with the customer. He/she shall always be informed about the progress. The risk management process shall be reviewed again after some time to ensure the up-to-dateness of the identified risks. The ProCOR method has been developed for non-existing systems. Therefore, it is important to apply all changes to the running system to the model. Otherwise, the ProCOR method cannot be applied to review the risk assessment.

The *Bundesamt für Sicherheit in der Informationstechnik (BSI)* provides some documents, so called *IT-Grundschutz-Kataloge* [Bun16]. There are different kinds of catalogues, each of which addresses different aspects of information security. General aspects are listed how to deal with security and what risks exist. The different risks are categorized, e.g. all risks

that are related to the infrastructure of a company or a system. A catalogue of possible threats can be used during the risk identification process. There is a high-level description of the threat followed by examples. The threat catalogue considers all three types of threats described in the CORAS language. Last, there is a catalogue of possible treatments. To be able to map a threat to a treatment, there are links in the documents.

The catalogues provided by the BSI can be used to identify threats and to automatically choose appropriate treatments. To do so, it is necessary to provide a mapping from the initial model used in the ProCOR method to the listed threats. For example, there is a category about possible threats for wireless and wired connections. Connections are described as connection domains in the model. Using the threats listed in the category, it is possible to elicit threats and threat scenarios and link them to the corresponding domain. For each domain that has been selected to be in scope of the risk analysis, the analysts and experts can look up which threats are listed in the catalogue.

The CORAL approach proposed by Erdogan et al. [ESA16] is a method for risk-driven testing. There are some similarities with the CORAS method. The method is model-based and the main language elements such as *threat scenario* are reused. Currently, our method only supports the analysis phase, but as mentioned in the beginning it is necessary to test the software after applying the treatments. The CORAL testing framework is not restricted to the implemented code. It is also possible to have some kind of model as input. The method itself consists of three phases: test planning, security risk assessment and security testing. In the first step, the system is modeled, the assets are selected, and the scales for likelihoods and consequences together with risk evaluation matrices are defined. In the second step, threat scenarios and corresponding likelihoods and consequences are elicited. Based on the likelihoods and consequences, the risk is evaluated and risks to be tested are selected. In the third step, the threat scenarios are tested. For each risk to be tested, the associated threat scenarios are chosen and the testing objectives are defined. The test objectives are added to the model and tests are carried out. The diagrams of the original CORAS method are replaced with sequence diagrams. This enables the user to define a model with regard to the time. In the ProCOR method, it is possible to specify messages between the domains but it is not possible to state the ordering of the messages.

The first and second step are already covered by the ProCOR method. To be able to test the model, it is necessary to extend the used profiles with stereotypes for the test cases. Additionally, it is necessary to specify how the treatments are considered for the security testing. Doing so, it is possible to make use of CORAL to test the results of the analysis with ProCOR.

Coles [Col15] addresses the issue that non-functional requirements such as security requirements are considered only after the analysis and design phases of a software development process. There is a need to bridge the gap between the business representatives and the software developers. The business experts decide about the effort that is spent for developing the software and to make it secure as well. Hence, it is necessary that software developers and security experts make security requirements comprehensible for the business representatives as well. To address this issue, Coles describes some security requirements elicitation techniques. With a checklist, the security needs of the customer are identified. Then, it is important to identify and consider policies, standards, laws and legislations that may be applied for the software-to-be. The elicited security requirements are then incorporated with the functional requirements documentation. Coles provides a list of known modeling techniques for security requirements. CORAS is listed as well. The contribution of Coles motivates to make use of the ProCOR method because it is suitable for the early stages of the software development process and combines functional requirements with security requirements.

14. Evaluation

In this chapter we first sum up all contributions of the thesis. Then, we review the research questions and last, we state some restrictions in the proposed method that will be addressed in future work.

14.1. Contributions in the Thesis

The main contribution of our work is the ProCOR method that enables the user to elicit security requirements based on risk considerations.

Step-wise method The ProCOR method consists of seven steps. For each step, we provide the necessary input, the output, a description how the step is performed, and validation conditions.

Validation conditions For each step of the ProCOR method, we defined validation conditions. These conditions have to be checked after the performance of the corresponding step to ensure that errors are detected as early as possible.

Case study We provide a case study to show how the ProCOR method can be applied in a software development process.

CORAS for non-existing systems The original CORAS is only applicable for existing systems. Our approach enables analysts to use the basic idea of CORAS for non-existing systems as well.

UML Profile for CORAS We developed a UML profile for CORAS to combine UML4PF models with CORAS models. The CORAS core profile contains all elements of the CORAS language and the diagram types which are used in the CORAS method. Newly introduced elements and diagram types are provided in the ProCOR profile.

Model-based approach All steps of the ProCOR method are model-based. We make use of UML class diagrams on which we apply the stereotypes defined by the UML4PF profile, the CORAS profile and the ProCOR profile. All results of the application of the method are stored in one model. This ensures consistency. Some of the validation conditions provided for each step can be automatically proved on the model using formal languages, for example OCL[1].

New elements & diagrams To support our method, we extended the CORAS language with three new elements. First, we added *High-Level Security Requirements*. Second, we added *Treatments*. Third, we added an *addresses* relation. In the new diagram type *Treatment Problem Diagram*, treatments are used to describe necessary implementations of treatment scenarios to fulfill the high-level security requirements. We defined an *Information Flow Graph* to detect domains on which some information is available.

Existing methods In our method, we make use of different existing methods. The method is mainly based on CORAS [LSS10] and problem frames by Michael Jackson [Jac01]. To identify left out domains, we make use of PresSuRE [FHM14]. The information flow graph is inspired by ProPAn [BFHM14].

[1] Object Constraint Language

Threat elicitation In the original CORAS method the threat elicitation is done in brainstorming sessions. We developed a questionnaire to elicit possible threats based on the initial model.

Tool support Last, we sketched some ideas for a support tool for the ProCOR method. There are mockups as well as textual descriptions how the user can use the tool to perform all steps of the method. The tool support is based on the model of the ProCOR method. In the corresponding chapter we also introduce new language elements to embed problem frames into the original CORAS language.

14.2. Contributions to Research Questions

In Chapter 1 on page 3, we defined several research questions that have been answered in this thesis. In this section we summarize the results of the work with regard to these research questions.

14.2.1. RQ1: How can we perform a risk analysis in the early phases of a software development process?

The ProCOR method presented in Part II is placed in the early phases of the software development process. We propose a structured method consisting of seven steps for a full security analysis. The result of the analysis is a set of high-level security requirements and a treatment selection to fulfill these requirements. All information is stored in one model to be available for the software development process.

14.2.2. RQ2: How can the initial information of the system be used to perform a sufficient analysis? What additional information has to be elicited?

The initial input of our proposed method is a set of functional requirements and a customer presentation of the software to be developed. To identify and evaluate risk we need external knowledge. In Section 7.1.1 on page 54, we stated some examples for external knowledge. All other information is derived from these inputs during the application of the method.

14.2.3. RQ3: How can existing methods be reused and combined?

In our method, we make use of several approaches. The main process of the risk analysis is based on CORAS [LSS10]. For the initial modeling, we make use of problem frames [Jac01] and UML4PF [HH10]. The phenomena relations are inspired by ProPAn [MH16b] as well as the basic idea of our information flow graph. To identify all domains that process the data, we make use of a problem diagram adjustment approach as described in PresSuRE [FHM14].

14.2.4. RQ4 How can the results of the analysis be used in order to suggest treatments to reduce risk?

The identification of risks is based on domains. All elements on which a treatment can be applied are linked to a domain. Hence, treatments can be selected with regard to the domain. The assets are linked to security properties. This information can also be used to suggest treatments. For example, a threat scenario leads to an unwanted incident that harms the availability of an asset. The treatment should then assure availability, e.g. with some kind of redundancy.

14.2.5. RQ5: How can the model be extended with the treatments so that they are available for the other phases of the software development lifecycle?

All results of the analysis are added to the same UML model. Treatment problem diagrams have been proposed in Chapter 11. These diagrams contain the relevant information about existing risks and what treatments are necessary to reduce the risks to be acceptable. A treatment domain has to be considered in the following phases of the software development process.

14.3. Restrictions & Future Work

We proposed a method to support a model-based risk management process. The original CORAS method has been enhanced to be applicable on non-existing systems. However, there are still some restrictions which are considered as future work.

14.3.1. Indirect Assets

We do not model indirect assets in our method. In future work, we will extend the asset identification process to identify indirect assets and the relations to the direct assets as well. As an indirect asset is not contained in the model, it is necessary to add it to the model. For example, the customer's trust in a company could be modeled as a domain knowledge diagram in which the customer and the company domain are contained. The connection between those domains describes the interaction between them. There is an assumption in this diagram which describes the trust relation between them. It refers to the company domain and constrains the customer domain because the behavior of the company influences the trust of the customer.

14.3.2. Vulnerabilities

The CORAS language supports the modeling of vulnerabilities. In the threat elicitation process introduced in Chapter 7 on page 53 we do not consider vulnerabilities. To support the full CORAS language, we will extend the proposed questionnaire. Vulnerabilities will then be collected in this step and are added to the threat overview diagram. To address threats and threat scenarios, it is necessary to have knowledge about the reasons for them. The vulnerabilities in the model can then be used to select treatments. It is also possible to select treatments to treat the vulnerabilities directly.

14.3.3. Treatments

All elements contained in a threat diagram are linked to domains of the initial model. The treatment selection is based on a threat diagram for a given unwanted incident with an unacceptable likelihood. Hence, we only consider domains that are linked to the specific threat or threat scenario. There might be treatments that treat domains which are not related to the threat diagram but can be used to reduce a risk. For example, there is a wireless network that is disrupted by another network. This is the threat scenario with the domain *Wireless Network*. The unwanted incident is that the data is not transmitted. One possible treatment is to retransmit the data. This treatment involves changes to the software. The software domain is neither related to the threat nor to the threat scenario but is contained in the problem diagram. Hence, also domains that are not contained in the threat diagram might be considered for the treatment selection. The proposed method has to be extended to support the identification of such treatments.

We do not support treatments which reduce a consequence. For example, a fire brigade does not avoid a fire, but reduces the consequences. As a future work, we will support this type of treatment.

Tran et al. propose a method to select treatments with regard to the costs and effects [TSS13]. In our method we already annotated the effects of treatments in form of the likelihood reduction. The method proposed by Tran et al. is divided into three steps. First, all countermeasures are documented in the risk model together with costs and effects. This documentation is also done in our approach. Tran et al. annotate the *treats* relations with an effect on the treated element. An effect can be a reduction of a likelihood or a consequence. In the examples provided in the paper, they make use of quantitative values. In our model, we only state likelihood reductions not reductions of consequences. In the second step of the method the risk reductions are re-evaluated for each combination of treatments. To do so, all effects and effect dependencies are considered and propagated along the paths of the threat diagram. In the last step the results are used to provide cost-benefit decisions for the treatment. The approach is not restricted to CORAS.

As future work, we will adapt the approach by Tran et al. for an enhancement of the treatment selection in the ProCOR method.

14.3.4. Combination of elements

In the original CORAS method some steps are performed using a structured brainstorming. The proposed method is more structured. This structuring leads to a less flexible identification process for threats. The creation of threat diagrams in the ProCOR approach does not allow a sequence of threat scenarios. There is one threat, one threat scenario, one unwanted incident and one asset. For example, it is not possible to describe that one threat scenario leads to another. The structured brainstorming of CORAS allows such a sequence of threat scenarios.

As future work, we will investigate how a structured brainstorming can be embedded into our method. To do so, the questionnaires and the way of documenting the results have to be more flexible.

14.3.5. Scalability

Currently, the model is very complex. For example, the threat overview diagram shown on page 66 is not well readable. To make the ProCOR method applicable for larger systems, it is necessary to provide different views on the diagrams. For the risk evaluation process, we need a global view on all risks. Hence, it is necessary to have everything in one model and we cannot split the information into different models. In Chapter 12 we sketched some ideas for a tool support for our method. We can address the problem of the scalability with this tool support. All information is contained in one model, but when using the tool the user is able to select different views. To create a view for a given threat, the tool selects only the relations and connected elements that are related to this threat. This view selection is possible for all kinds of elements. The different views make the information contained in the model comprehensible because each view focusses on relevant elements.

14.3.6. Risk Identification

Currently, it is only possible to elicit threats with the help of external knowledge. Our vision is to have a library of known threats as proposed by Uzunov and Fernandez [UF14]. To do so, we will make use of the problem frames that are contained in the initial model. As future work, we will work on a mapping process from problem frames to threat patterns. This mapping would allow to suggest possible threats. The user can then select what threats are relevant for the analysis.

15. Bibliography

[AFH+15] Azadeh Alebrahim, Stephan Faßbender, Denis Htebur, Ludger Goeke, and Isabelle Côté. A pattern-based and tool-supported risk analysis method compliant to iso 27001 for cloud systems. International Journal of Secure Software Engineering (IJSSE), 6(1):24–46, 2015.

[BFHM14] Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis. A problem-based approach for computer-aided privacy threat identification. In Bart Preneel and Demosthenes Ikonomou, editors, Privacy Technologies and Policy, volume 8319 of Lecture Notes in Computer Science, pages 1–16. Springer Berlin Heidelberg, 2014.

[BHSS14] Kristian Beckers, Maritta Heisel, Bjørnar Solhaug, and Ketil Stølen. ISMS-CORAS: A Structured Method for Establishing an ISO 27001 Compliant Information Security Management System, pages 315–344. Springer International Publishing, Cham, 2014.

[Bun16] Bundesamt für Sicherheit in der Informationsbrache (BSI). IT-Grundschutz-Kataloge. Catalogue, 2016.

[CH04] C. Choppy and M. Heisel. Une approache à base de patrons pour la spécification et le développement de systèmes d'information. Approches Formelles dans l'Assistance au Développement de Logiciels - AFADL, 2004.

[CHH+08] Isabelle Côté, Denis Hatebur, Maritta Heisel, Holger Schmidt, and Ina Wentzlaff. A systematic account of problem frames. In Proceedings of the European Conference on Pattern Languages of Programs (EuroPLoP), pages 749–767. Universitätsverlag Konstanz, 2008.

[CHSH11] I. Cote, M. Heisel, H. Schmidt, and D. Hatebur. Uml4pf – a tool for problem-oriented requirements analysis. 2014 IEEE 22nd International Requirements Engineering Conference (RE), 0:349–350, 2011.

[Clo16] Cloud Security Alliance. The treacherous 12 - cloud computing top threats 2016, 02 2016.

[Col15] Elena Simona Coles. Analyzing and Specifying Security Requirements in Early Stages of Software Development Life Cycle. Journal of Mobile, Embedded and Distributed Systems, 7(2):87–94, 2015.

[Cor12] Dan Cornell. Remediation statistics: What does fixing application vulnerabilities cost? https://www.rsaconference.com/writable/presentations/file_upload/asec-302.pdf, 2012.

[ESA16] Gencer Erdogan, Ketil Stølen, and Jan Øyvind Aagedal. Evaluation of the CORAL Approach for Risk-driven Security Testing based on an Industrial Case Study. 2016.

[FHM14] Stephan Faßbender, Maritta Heisel, and Rene Meis. Functional requirements under security pressure (best students paper award). In ICSOFT-PT 2014 - Proceedings of the 9th International Conference on Software Paradigm Trends, Vienna, Austria, 29-31 August, 2014, pages 5–16. SciTePress, 2014.

[FIR] FIRST.Org, Inc. (FIRST). Common vulnerability scoring system v3.0: Specification document.

[Hat12] Denis Hatebur. Pattern- and Component-based Development of Dependable Systems. Deutscher Wissenschafts-Verlag (DWV), 2012.

[Hei98] Maritta Heisel. Agendas - A Concept to Guide Software Development Activities. In Proceedings of the IFIP TC2 WG2.4 working Conference on Systems Implementation: Languages, Methods and Tools, pages 19–32. Chapman & Hall London, 1998.

[HH10] Denis Hatebur and Maritta Heisel. A UML profile for requirements analysis of dependable software. In Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP), LNCS 6351, pages 317–331. Springer, 2010.

[II05] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology - Security techniques - Information security management systems - Requirements. ISO/IEC 27001, 2005.

[II11] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC). Information technology – Security techniques – Information security risk management. ISO/IEC 27005, 2011.

[ISO09] ISO. ISO 31000 Risk management – Principles and guidelines. International Organization for Standardization, 2009.

[Jac01] Michael Jackson. Problem Frames: Analyzing and Structuring Software Development Problems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[Kas15] Kaspersky Lab. Kaspersky Security Bulletin 2015 - OVERALL STATISTICS FOR 2015. https://securelist.com/files/2015/12/KSB_2015_Statistics_FINAL_EN.pdf, 2015.

[LSS10] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. Model-Driven Risk Analysis. The CORAS Approach. Springer, 2010.

[MH16a] Rene Meis and Maritta Heisel. Computer-aided identification and validation of privacy requirements. Information, 7(2):28, 2016.

[MH16b] Rene Meis and Maritta Heisel. Supporting privacy impact assessments using problem-based privacy analysis. In Software Technologies - 10th International Joint Conference, ICSOFT 2015, Revised Selected Papers, volume 586 of Communications in Computer and Information Science, pages 79–98. Springer, 2016.

[Nat15] National Vulnerability Database. Cve-2015-8289. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-8289, 2015.

[OPE09] OPEN meter Consortium. Report on the identification and specification of functional, technical, economical and general requirements of advanced multi-metering infrasturcture, including security requirements, 2009.

[SJM08] Karen Scarfone, Wayne Jansen, and Tracy Miles. NIST Special Publication 800-123, Guide to General Server Security. 2008.

[Spi89] J. M. Spivey. The Z Notation: A Reference Manual. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[TSS13] Le Minh Sang Tran, Bjørnar Solhaug, and Ketil Stølen. An approach to select cost-effective risk countermeasures exemplified in CORAS. CoRR, abs/1302.4689, 2013.

[UF14] Anton V. Uzunov and Eduardo B. Fernandez. An extensible pattern-based library and taxonomy of security threats for distributed systems. Computer Standards & Interfaces, 36:734–747, 2014.

[vL09] A. van Lamsweerde. Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009.

Eidesstattliche Erklärung

Ich erkläre, dass ich meine Master-Arbeit "'Risk-based Elicitation of Security Requirements"' selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.

Duisburg, den 16.09.2016

(Roman Wirtz)