Problem-based Elicitation of Security Requirements
The ProCOR Method

Roman Wirtz[1], Maritta Heisel[1], Rene Meis[1], Aida Omerovic[2] and Ketil Stølen[2]
[1]*University of Duisburg-Essen, Duisburg, Germany*
[2]*SINTEF Institute, Oslo, Norway*

Keywords: Risk Management, Security, Requirements Engineering, Problem-based, Model-based.

Abstract: Security is of great importance for many software systems. The security of a software system can be compromised by threats, which may harm assets with a certain likelihood, thus constituting a risk. All such risks should be identified, and unacceptable risks should be reduced, which gives rise to security requirements. The relevant security requirements should be known right from the beginning of the software development process. Eliciting security requirements should be done in a systematic way. We propse a method to elicit security requirements that address unacceptable risks. They require a reduction of the risk to an acceptable level. Our method combines the CORAS risk management method with Jackson's problem-based requirements analysis approach. Based on the functional requirements for a software system, security risks are identified and evaluated. Unacceptable risks give rise to high-level security requirements. To reduce the risk, treatments are selected. Based on the selected treatments, concretized security requirements are set up and represented in a similar way as functional requirements. Thus, both functional and security requirements can then drive the software development process.

1 INTRODUCTION

In a connected world, almost every piece of software may be subject to attacks. Such attacks can cause great harm to enterprises and individuals. Almost every week, media report on attacks against public or private organizations. Therefore, software should be developed with security issues in mind. Nevertheless, organizations can only spend a limited amount of resources on security. These resources should be spent in a way that maximizes return on investment, i.e., that provides the best possible protection against attacks.

In this paper, we describe a risk-based method to elicit security requirements. Given the functional requirements for a software system, possible threats to security and the corresponding risks are identified and evaluated. A risk is determined by two factors: the likelihood of the threat leading to harm of an asset, and the severity of the harm, i.e., the consequence. For each identified risk, it has to be determined if it is acceptable or not. Only unacceptable risks need to be treated, thus leading to corresponding security requirements for the software system to be developed. Moreover, our method supports the selection of treatments that are suitable to achieve the necessary risk reduction. The result of our method is a set of functional and corresponding security requirements that form the basis for the subsequent software development process.

Our method is model-based, so that the results of the risk analysis can be smoothly integrated in a model-based software development process. The name of the method is *ProCOR – Problem-based CORAS*, because it combines parts of the model-based risk management method CORAS (Lund et al., 2010) with Jackson's problem frames approach (Jackson, 2001). Functional requirements expressed using problem diagrams (according to the problem frames approach) form the starting point of the risk analysis. The elicited security requirements are expressed in a similar way, using a new kind of diagram, called treatment problem diagram. The so enhanced requirement model forms the starting point for a software development that adequately balances functionality and security.

The paper is structured as follows: In Section 2, we explain problem frames and CORAS in more detail before introducing our running example in Section 3. In Section 4, we discuss all steps of the

Wirtz, R., Heisel, M., Meis, R., Omerovic, A. and Stølen, K.
Problem-based Elicitation of Security Requirements.
DOI: 10.5220/0006669400260038
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 26-38
ISBN: 978-989-758-300-1

ProCOR method in detail. The paper concludes with a discussion of related work in Section 5 and an outlook on an evaluation we plan for our method in Section 6.

2 BACKGROUND

In this section, we introduce necessary background knowledge. Our proposed method is mainly based on the two concepts of *Problem Frames* and *CORAS*.

2.1 Problem Frames

To model requirements, we make use of Michael Jackson's problem frames (Jackson, 2001) that can be expressed using diagrams as shown in Figures 3 and 4. Problem frames are patterns to describe subproblems of a complex software development problem in the early stages of the software development lifecycle. An instance of such a pattern is called problem diagram and contains a functional requirement (dashed ovals) for the system-to-be. A requirement is an optative statement which describes how the environment of the software should behave when the software is in action. The entities related to a requirement are represented as domains (rectangles). There are different types of domains: biddable domains (e.g., persons), causal domains (e.g., technical equipment), machine domains (representing the software to be developed, rectangle with vertical bars) and lexical domains (data representations). There are symbolic phenomena, representing some kind of information or a state, and causal phenomena, representing events, actions, operations and so on. Each phenomenon is controlled by exactly one domain and can be observed by other domains. A phenomenon controlled by one domain and observed by another is called a shared phenomenon between these two domains. Interfaces (solid lines) contain sets of shared phenomena. Such a set contains phenomena controlled by the same domain (indicated by $A!\{...\}$, where A is an abbreviation for the controlling domain). Some domains are *referred to* by a requirement (dashed line) via some phenomena, and at least one domain is *constrained* by a requirement (dashed lines with arrowhead) via some phenomena. The domains and their phenomena that are referred to by a requirement are not influenced by the machine, whereas we build the machine to influence the constrained domain's phenomena in such a way that the requirement is fulfilled.

Faßbender et al. describe a way to draw problem diagrams with regard to security, so that no domain is left out which might be relevant for the analysis (Faßbender et al., 2014).

2.2 CORAS

CORAS (Lund et al., 2010) is a model-based method for risk management. It consists of a step-wise process and different kinds of diagrams. The method follows the ISO 31000 risk-management standard (ISO, 2009). Each step provides guidelines for the interaction with the customer on whose behalf the risk management activities are carried out and how to add the results to the model using the CORAS language. The method starts with the establishment of the context and ends up with the suggestion of treatments to address the risk.

The CORAS language consists of several elements. In this work, we make use of the following ones: *Direct Assets* are items of value. There are *Human-threats deliberate*, e.g. a network attacker, as well as *Human-threats accidental*, e.g. an employee pressing a wrong button accidentially. To describe technical issues *Non-human threats* are used, e.g. power loss of a server. A *Threat scenario* describes a state, which may possibly lead to an unwanted incident, where an *Unwanted incident* describes the action that harms an asset. Risk is defined as the combination of a likelihood and a consequence according to ISO 31000 (ISO, 2009). In CORAS, the likelihood of an unwanted incident which harms an asset and the consequence on the asset is used to derive a risk. *Treatment scenarios* are used to describe countermeasures to reduce the risk. The symbols we make use of in CORAS diagrams are shown in Figure 1.

The relations between CORAS and problem frames we developed for this work are described in Section 4.1, using a conceptual model.

3 CASE STUDY

To exemplify out method, we use a running example for which we apply the different steps of the ProCOR method.

The running example is a subsystem of a smart grid system inspired by the OPEN meter project (OPEN meter Consortium, 2009). A smart grid is an intelligent power supply network, in which different participants are able to interact and control the

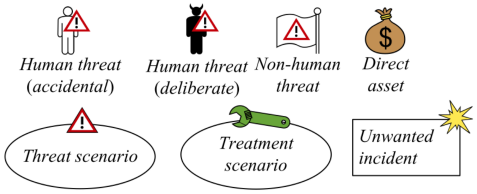

Figure 1: CORAS Language (Lund et al., 2010).

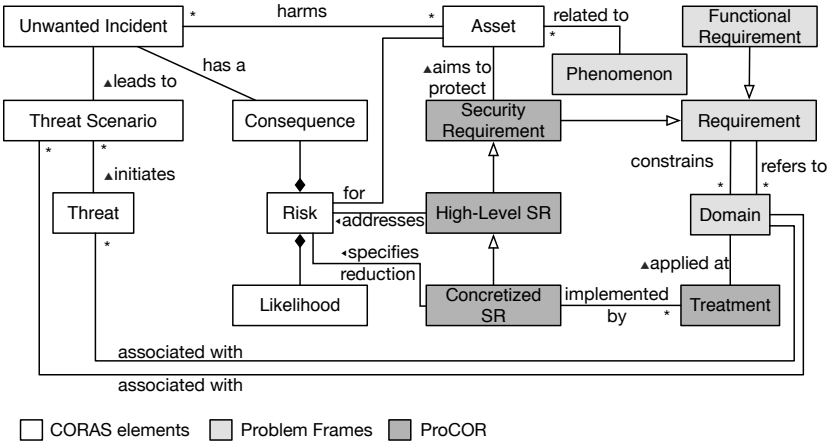

Figure 2: ProCOR Conceptual Model.

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. The system-to-be (i.e., the machine) is the *Communication Hub*. It serves as the connection between all other components and actors and is used to perform some calculations, e.g. to provide invoices to the customer for consumed energy. *Smart Meters* measure the consumption of energy with sensors. They are connected to the Communication Hub using a local metrological network (LMN) which might be wired or realized with a wireless connection. The *Energy supplier* is the provider of the smart grid. It is able to do an initial setup locally on the communication hub to be able to initiate a remote connection for later interaction. The *End customer* is the one who pays the energy supplier's invoices and in whose home the communication hub is installed.

In this paper, we focus on two functional requirements:

Setup: The energy supplier performs an initial setup for the communication hub. The personal data of the client and tariff parameters are stored in a configuration. Figure 3 shows the problem diagram for *Setup*. The requirement refers to a phenomenon of *EnergySupplier* and constrains a phenomenon of *Configuration*.

Measuring: In given intervals, the communication hub receives measured data from smart meters and stores it persistently. Figure 4 shows the problem diagram for *Measuring*. The requirement refers to a phenomenon of *SmartMeter* and constrains a phenomenon *MeterData*.

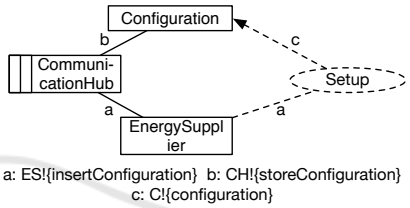

Figure 3: Case study: Problem diagram for Setup.

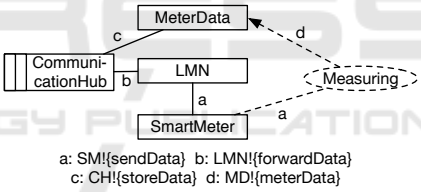

Figure 4: Case study: Problem diagram for Measuring.

4 ProCOR METHOD

The ProCOR method consists of five steps. We first introduce a conceptual model which describes the different terms used in the method and their relations. Next, we give an overview of the method which is followed by a detailed description of each step. For each step, we illustrate the method on the case study introduced in Section 3.

4.1 Conceptual Model

The conceptual model is shown in Figure 2 as a UML class diagram. All multiplicities not specifically mentioned are considered as *1*. The elements contained in CORAS are drawn in white. The meaning of the elements has already been described in Section 2.2. A *Threat* initiates some *Threat Scenario*s. Both ele-

ments are associated with a *Domain*, which is part of Jackson's problem frames terminology. That terminology is shown in light gray. The association is used to identify risks based on the functional requirements. A *Threat Scenario* leads to some *Unwanted Incident*s. An *Unwanted Incident* harms some *Asset*s which is related to a *phenomenon* with a specific *Consequence*, which is part of a *Risk*. A *Risk* exists for an *Asset*. A *Likelihood* is also part of a *Risk*. A *Requirement* can be of type *Functional Requirement* or *Security Requirement (SR)* and refers to and constrains *Domain*s. A *Security Requirement* aims to protect an *Asset*. A *High-Level SR* is a specialization and addresses a *Risk*, whereas a *Concretized SR* specifies how the risk is reduced. A *Concretized SR* is implemented by some *Treatments*, which are applied at a *Domain*. Thus, a *Treatment* is used to avoid or mitigate a *Threat Scenario* or *Threat* contained in this *Domain*.

4.2 Overview

Figure 5 provides an overview of the five steps of the ProCOR method. For each step, we define the necessary external inputs, as well as the outputs. The generated output serves as an input for the following steps and can be used for documentation purposes. We distinguish three stakeholder roles: *analysts* take part in each step of the method. Therefore, we do not show them explicitly in Figure 5. *Customers* are the ones on whose behalf the analysts perform the risk analysis. *Experts* are persons with specific knowledge, e.g., about threats.

The method is structured as follows: (1) First, the scope and focus of the risk analysis are defined, based on the security goals of the customer and the functional requirements, which are expressed as problem diagrams. (2) The risks for the assets are identified and are documented with a CORAS threat diagram. (3) To evaluate the risks, likelihoods and consequences are estimated and annotated in the threat diagram. For each unacceptable risk, a high-level security requirement is set up. (4) To fulfill the high-level security requirements, treatments that reduce the risks are selected. These treatments are evaluated with respect to their costs, which should not be higher than the value of the asset they protect. The selected treatments are then added to the threat diagram. (5) For each high-level security requirement, a concretized security requirement is set up, which reflects the chosen treatments. Finally, for each concretized security requirement, a treatment problem diagram is set up. Such a diagram is similar to a problem diagram. It states which domains are referred to and constrained when fulfilling the concretized security requirements using the selected treatment. Thus, security requirements can be incorporated in the subsequent software development process in a similar way as functional requirements.

In addition, we identified *validation conditions* for each step that check the coherence of the results of each step, thus helping to identify errors in the application of the method as early as possible. We give examples of such validation conditions in the following. For reasons of space, we cannot present all of them.

We now describe each of the above steps in detail.

4.3 Step 1: Definition of Scope & Focus

In the first step, we define the focus (assets to be protected) and scope (domains which shall be considered) of the analysis.

4.3.1 Description

With ProCOR, we aim to support the protection of information in a software-based system with regard to the three security properties confidentiality, integrity and availability. In problem diagrams, information is represented by symbolic phenomena, whereas commands or events are represented by causal phenomena. Hence, we define an asset as a combination of a symbolic phenomenon and a security property. The results are documented in a table, such as Table 1. We also document the value of the asset for the customer to make it comparable with the costs of a treatment.

The scope of the analysis is defined as the set of domains where the information to be protected is available. "Available" means that the domain observes or controls the symbolic phenomenon representing the information to be protected, as specified in the set of problem diagrams that is part of the input for this step. However, it does not suffice to only consider symbolic phenomena, because valuable information can also be part of commands or events, i.e., causal phenomena. For example, there may be an interface with a causal phenomenon to store some user data. This phenomenon is a command, but contains information in form of user data. Hence, it is necessary to document that a phenomenon is contained in another to decide whether an interface contains some information or not.

To document the availability of information at a domain and the information flow between domains, we developed the concept of an *Information Flow Graph*, which is a directed graph. Its nodes are domains, and its edges denote the information flowing from one domain to another. To create this graph, we consider all interfaces contained in the set of problem

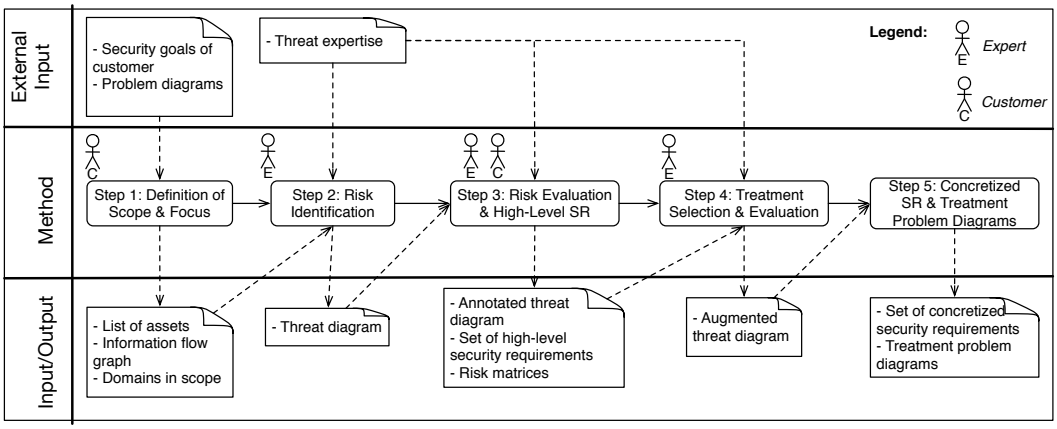

Figure 5: ProCOR Method Overview.

diagrams. A domain is contained in the graph iff it observes or controls a phenomenon that is related to an asset. Either, the phenomenon is a symbolic one, which contains the asset directly. Or the phenomenon is related to such a symbolic phenomenon, as explained earlier.

The directed edges of the graph indicate an asset-related information flow. The starting nodes of the edges are the domains which control the asset-related symbolic phenomenon or the one in which it is contained. The end nodes are the domains that observe the respective phenomenon.

As a result, the domains that are contained in the generated graph are considered to be in scope of the analysis, because the information to be protected is available there. In the next step, risks for the assets are identified at these domains.

4.3.2 Validation Conditions

We identified the following validation conditions (VCs) for the definition of focus and scope:

VC1 All interfaces contained in the set of problem diagrams are considered for the information flow graph.

VC2 Only domains on which an asset-related information is available are considered to be in scope.

VC3 For all information to be protected, the desired security goals (confidentiality, integrity, availability) are documented.

4.3.3 Case Study

The ProCOR method is carried out based on the problem diagrams shown in Figures 3 and 4. We assume that a fictitious customer has specified the assets to be protected as given in Table 1. The values of the assets are estimated in Euros per year according to the following reasoning: (1) The integrity of tariff parameters has a value of 20.000 Euros because without correct parameters, the invoicing cannot be performed correctly. Because of periodical updates of the parameters, incorrect parameters will be overwritten at some time. Therefore, the incorrect values exist only for a limited amount of time. This results in a relatively small value of the asset. (2) The availability of measured data is important for invoicing, too. The absence of measured data leads to the case that an invoicing is not possible. In this case, an employee needs to collect the data manually. For this reason, we estimate the overall costs for all clients to 10.000 Euros. (3) A harm of the integrity of stored measured data leads to an incorrect invoicing. Due to the high number of clients of an energy supplier, the financial consequences are estimated with 50.000 Euros.

Next, we have to identify the phenomena in Figures 3 and 4 that are related to the assets given in Table 1. The results of that identification process are shown in Figure 6. The symbolic phenomena which are considered as an asset are shown in gray. An arrow pointing from one phenomenon to another means that it is contained in the other one. For example, *ES!*{*clientData*} representing the personal information of a client is contained in *ES!*{*insertConfiguration*}.

The corresponding information flow graph is shown in Figure 7. Below a domain name, the available asset-related information is given in gray. For

Table 1: Asset documentation.

Symbolic Phenomenon	Security Property	Value
tariffParameters	Integrity	20.000 Euros
measuredData	Availability	10.000 Euros
measuredData	Integrity	50.000 Euros

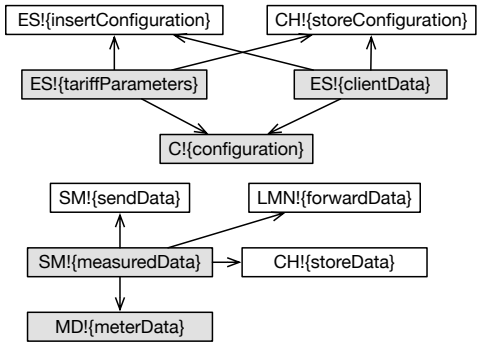

Figure 6: Case study: Phenomena relations.

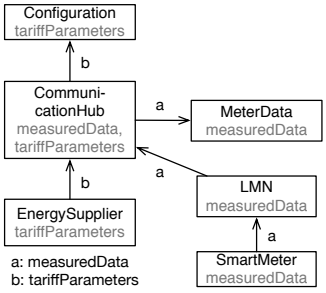

Figure 7: Case study: Information flow graph.

example, at the domain *Configuration* the information *tariffParameters* is available, and there is a corresponding information flow to this domain from the *CommunicationHub*.

4.4 Step 2: Risk Identification

Based on the definition of focus and scope, we now identify possible risks for the assets.

4.4.1 Description

To identify risks, we make use of a structured brainstorming process as proposed by CORAS (Lund et al., 2010). The analysts moderate a meeting with experts which have different expertises. The results of this brainstorming session are documented in a *threat diagram*. Such a diagram (see Figure 8) contains CORAS elements as described in Section 2.2, and relations between them, as described in the conceptual model given in Section 4.1.

The domains that are contained in the information flow graph are inspected in this meeting, and possible attacks on these domains have to be elicited. For example, a domain representing an employee needs investigation for social engineering attacks. However, it might be the case that additional domains have to be considered for risk identification. This is because the information flow graph has been created based on the functional requirements. These requirements only

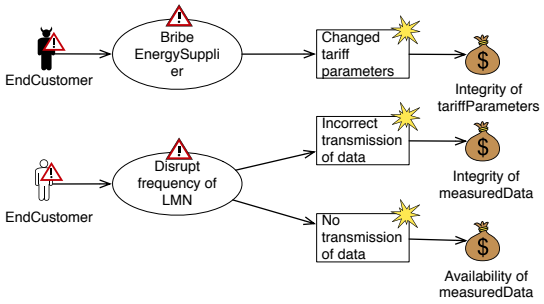

Figure 8: Case study: Threat diagram.

describe the desired behavior. For example, the attacker who performs a social engineering attack on the employee is not part of the desired behavior and is hence not contained in the information flow graph. This threat is then newly introduced in the threat diagram, even though it is not contained in any problem diagram.

4.4.2 Validation Conditions

We identified the following validation conditions for the risk identification:

VC1 All domains contained in the information flow graph have to be investigated for some threat or threat scenario.

VC2 Only domains on which an asset-related information is available can have a threat or threat scenario which can harm the asset.

4.4.3 Case Study cont'd

Figure 8 shows the outcome of the risk identification process for our case study. There is one human-threat deliberate *EndCustomer* who bribes the energy supplier (threat scenario) to change the tariff parameters (unwanted incident). This harms the integrity of the tariff parameters (asset). The end customer is also a human-threat accidental who disrupts the frequency of the local metrological network (threat scenario), for example by using the same frequency for other purposes. This leads to two unwanted incidents. First, the integrity of measured data (asset) is harmed by an incorrect transmission of data (unwanted incident). Second, the availability of measured data (asset) is harmed by no transmission of data (unwanted incident).

4.5 Step 3: Risk Evaluation & High-Level Security Requirements

In this step, we evaluate the identified risks and set up a high-level security requirement for each unaccepta-

ble risk, based on a given risk matrix.

4.5.1 Description

The risk evaluation process is performed as proposed by CORAS (Lund et al., 2010). Based on the threat diagram and the knowledge of experts, likelihoods and consequences are estimated. There is a likelihood for the following elements of the threat diagram: (1) for a threat to initiate a threat scenario, (2) for a threat scenario to occur, (3) for a threat scenario leading to an unwanted incident and (4) for an unwanted incident to occur. The likelihoods for (2) and (4) are derived from the likelihoods of the incoming edges (i.e., (1) and (3)) based on empirical knowledge about the dependencies between the different likelihoods. The consequences for an asset are annotated on the relation between an unwanted incident and an asset. Thus, it describes what consequence an unwanted incident has on an asset.

The likelihoods and consequences are expressed using scales. These scales can for example be defined based on intervals. To evaluate the risks, we make use of a risk matrix. For each asset, a risk matrix must be defined. On the x-axis of the matrix, the different values of the consequence scale and on the y-axis the values for the likelihood scale are annotated. Since a risk consists of a likelihood and a consequence, a cell in the matrix denotes a risk level. For each risk level, it must be decided whether the risk is acceptable or not. Each risk of the threat diagram, represented as the likelihood of an unwanted incident and the corresponding consequence on an asset, is added to the risk matrix. Using the matrix, it is now possible to decide whether a risk is acceptable or not. For each unacceptable risk, a *high-level security requirement (HL-SR)* is set up, which describes the necessary risk reduction. Such a requirement is expressed using the following textual pattern:

> Ensure that the risk for **A** due to **UI** caused by $\mathcal{TS}$ and initiated by $\mathcal{T}$ is acceptable according to the risk matrix of the asset.

A stands for the asset, *UI* for the unwanted incident, $\mathcal{TS}$ for the set of threat scenarios that lead to the unwanted incident and $\mathcal{T}$ for the set of threats that initiate the threat scenarios.

To achieve such a likelihood reduction and thereby fulfill the HL-SR, we select treatments in the next step.

4.5.2 Validation Conditions

We identified the following validation conditions for the risk evaluation and the instantiation of high-level security requirements:

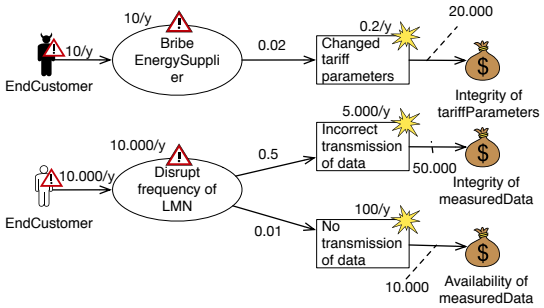

Figure 9: Case study: Threat diagram with annotated risks.

VC1 All likelihoods and consequences are documented in the threat diagram to ensure a correct estimation of the risks.

VC2 For each asset, a risk matrix is defined which specifies the acceptable and unacceptable risks for that asset.

VC3 The likelihood scales and consequence scales used for the documentation in the threat diagram are consistent with the ones used in the risk matrices.

VC4 For each risk that is considered as unacceptable according to the corresponding risk matrix, exactly one high-level security requirement is set up.

4.5.3 Case Study cont'd

In Figure 9, we show the threat diagram with annotated likelihoods and consequences. We express the likelihood of an event as the frequency of the event per year. We assume that ten times a year some end customer tries to bribe the energy supplier, which is successful in 2% of the cases and leads to a change of tariff parameters. The consequence is given in Euros as a value loss, here 20.000 Euros. There are 10.000 disruptions of the local metrological network by the end customer per year, because the local metrological network and most of the wireless equipment used in private areas make use of the 2.4GHz band. In 50% of the cases, this leads to an incorrect transmission of data and a value loss of 50.000 Euros for the integrity of measured data. In one percent of the cases it leads to no transmission of data, which harms the availability of measured data completely and hence, produces a value loss of 10.000 Euros.

Since there is only at most one incoming edge per element, the likelihood for each unwanted incident is calculated by multiplying both previous likelihoods. The risk for each pair of unwanted incident and asset is documented in Table 2. The risks are then added to the risk matrix as shown in Table 3. Here, we assume

that the values for the risk matrix are the same for each asset. The unacceptable risks are marked with gray. Hence, there are three unacceptable risks, and it is necessary to provide a high-level security requirement for each of them:

HL-SR1. Ensure that the risk for *Integrity of tariffParameters* due to *Changed tariff parameters* caused by {*Bribe EnergySupplier*} and initiated by {*EndCustomer*} is acceptable according to the risk matrix of the asset.

HL-SR2. Ensure that the risk for *Integrity of measuredData* due to *Incorrect transmission of data* caused by {*Disrupt frequency of LMN*} and initiated by {*EndCustomer*} is acceptable according to the risk matrix of the asset.

HL-SR3. Ensure that the risk for *Availability of measuredData* due to *No transmission of data* caused by {*Disrupt frequency of LMN*} and initiated by {*EndCustomer*} is acceptable according to the risk matrix of the asset.

Table 2: Case study: Risk calculation.

No.	Unwanted Incident	Asset	Risk ($frequ./y \times consequ.$)
(1)	Changed tariff parameters	Integrity of tariffParameters	(0.2, 20.000)
(2)	Incorrect transmission of data	Integrity of measuredData	(5.000, 50.000)
(3)	No transmission of data	Availability of measuredData	(100, 10.000)

Table 3: Case study: Risk matrix.

frequency / year	Consequence / Euros: [0,100[	[100,1000[	[1000,2000[	[2000,5000[	[5000,∞]
[0,0.1]					
]0.1,1]					(1)
]1,10]					
]10,100]					
]100,∞]					(2),(3)

4.6 Step 4: Treatment Selection & Evaluation

In this step, we first select treatments based on the threat diagram to fulfill the previously determined high-level security requirements. Then, the treatments are evaluated to decide whether an application is worthwhile or not, based on the overall costs compared to the value of the asset.

4.6.1 Description

For each high-level security requirement, treatments have to be selected to achieve the necessary risk reduction. The elements which a treatment can possibly address are described in the high-level security requirement (threats and threat scenarios) and can also be found in the threat diagram. As shown in the conceptual model (see Section 4.1) a threat or threat scenario, respectively, is associated with a domain. Therefore, a treatment is applied at that domain. A treatment either leads to a likelihood or a consequence reduction. To express this in a threat diagram, we extend the CORAS language with a new arrow type for a treatment scenario, called *addresses*, which points to the likelihood or consequence to be reduced, whereas the *treats* arrow points to the threat or threat scenario on whose domain the treatment is applied.

The treatments can be selected in a brainstorming session. It is necessary to add treatments until all risks have been reduced to an acceptable level. If such a reduction is not possible, the high-level security requirement cannot be fulfilled. The customer then has to decide whether to accept a higher risk (changing the risk matrix), change the functional requirements, e.g. by not offering risky services any more, or search for other possibilities to mitigate the unacceptable risk.

Furthermore, the treatments should not cost more than the value of the asset they protect. Therefore, it is necessary to calculate the overall costs for all treatments that are related to a high-level security requirement. The overall costs are then compared to the value of the asset to be protected. If the costs are higher than the value, the customer has to decide how to proceed with the development, similarly as described above.

4.6.2 Validation Conditions

We identified the following validation conditions for the selection of treatments:

VC1 Treatments are selected in the way that they reduce a certain risk (consequence and/or likelihood) and are applied to a domain that is related to a threat or threat scenario which leads to the risk.

VC2 The costs for treatments are not higher than the value of the asset to be protected.

VC3 All risks are reduced to an acceptable level according to the risk matrices.

4.6.3 Case Study cont'd

We selected three treatments to illustrate the fourth step of our method, using fictitious values for their costs. (1) To reduce the likelihood that bribing the energy supplier will be successful, there is a treatment providing better working conditions, e.g. more money. There are only few employees who are able to change the tariff parameters. Hence, the costs of the treatment are 5.000 Euros. This is less than the value loss of 20.000 Euros. The likelihood reduction is estimated to be 80%. The residual likelihood for the unwanted incident is then $10 * (1 - 0.8) * 0.02 = 0.04$. According to the risk matrix (see Table 3), the risk is now acceptable. (2) To avoid disruption of the local metrological network (LMN), the frequency is replaced by a 5GHz band. As there are major changes to the LMN, the costs are estimated with 40.000 Euros. This treatment addresses two likelihoods, because the threat scenario leads to two unwanted incidents. Both likelihoods are reduced by 99,9%, because the 5GHz band allows much more concurrent connections. The value loss for an incorrect transmission of data is 50.000 Euros, and for no transmission of data it is 10.000 Euros. Hence, the costs of the treatment are less than the value loss. Recalculating the likelihoods for the unwanted incidents leads to $5/y$ for an incorrect transmission and $0.1/y$ for no transmission. According to the risk matrix, the likelihood for an incorrect transmission is still too high, so we need an additional treatment. (3) The implementation of a checksum can be realized by an existing component that can be applied at the LMN. The costs are 1.000 Euros. Since a checksum makes it possible to detect modifications of data, the consequences for the integrity of measured data are reduced by 99%. In fact, the incorrect data is still transmitted, but the communication hub as the receiver is able to detect the incorrectness. The incorrect data will not be used for the invoicing. There is still a value loss, because the data has to be retrieved in a different way. The residual consequence is $(1 - 0.99) * 50000\ Euros = 500\ Euros$. According to the risk matrix and the likelihood reduction by the second treatment, the risk is now acceptable.

There are no unacceptable risks left in our example, and hence all high-level security requirements can be fulfilled. The augmented threat diagram containing all treatment scenarios is shown in Figure 10.

4.7 Step 5: Concretized Security Requirements & Treatment Problem Diagrams

Based on the previously determined high-level security requirements and the proposed treatments, we concretize the security requirements and close the circle to the functional requirements by setting up treatment problem diagrams.

4.7.1 Description

A *Concretized Security Requirement (C-SR)* adds information about selected treatments to a high-level security requirement. For this reason, there is one C-SR for each HL-SR set up in step 4. A C-SR has the following form:

> The risk for **A** due to **UI** caused by $\mathcal{TS}$ and initiated by $\mathcal{T}$ is reduced to an acceptable level according to the risk matrix of the asset by applying $\mathcal{TR}$.

There, $\mathcal{TR}$ describes the required treatments as a set of tuples $domain \times treatment$ where *domain* describes the domain to which the *treatment* is applied. To align the concretized security requirements with the functional ones, we introduce a new type of problem diagram, called *Treatment Problem Diagram*. In contrast to a functional requirement, a C-SR is not implemented by a machine but by a treatment. This treatment can be technical. In this case, it could lead to a software development problem, similar to the functional requirements. But treatments can also be non-technical, for example training personnel to resist social engineering attacks. Therefore, a new type of problem diagrams is needed to specify the treatments to

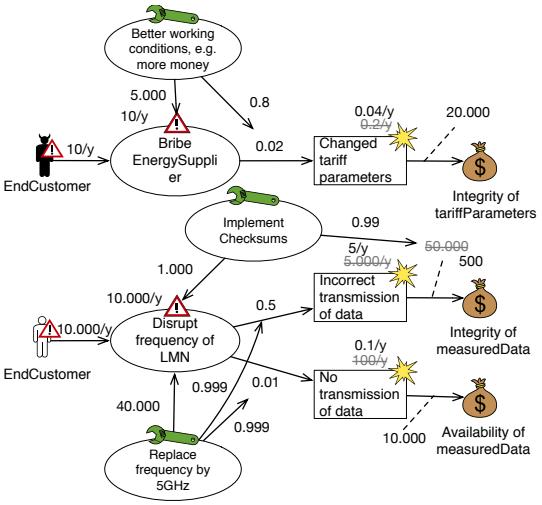

Figure 10: Case study: Threat diagram with treatments.

be implemented and their effects on the relevant domains.

As a counterpart of the machine domain in problem diagrams, we introduce a new domain type called *treatment* for treatment problem diagrams. For each C-SR, one treatment problem diagram must be set up. All domains and treatments given by the C-SR are added to the diagram. To indicate that a treatment treats a domain, we add an interface between the treatment domain and treated domain. These domains share a phenomenon controlled by the treatment domain, which describes how the treated domain is influenced. The C-SR constrains the treated domain and refers to all other domains related to the elements mentioned in the C-SR. The referring edges of the requirement are annotated with a phenomenon that is controlled by the domain and describes the harm on the asset.

The result of the last step is an extended requirement model consisting of a set of functional requirements, expressed as problem diagrams, and a set of security requirements, expressed as treatment problem diagrams. In subsequent phases of a software development lifecycle, it is now possible to consider both types of requirements directly and to ensure that the security requirements are considered right from the beginning of the software development process, instead of treating them as an add-on.

4.7.2 Validation Conditions

We identified the following validation conditions for the instantiation of concretized security requirements and the creation of treatment problem diagrams:

VC1 For each high-level security requirement, there is exactly one concretized security requirement.

VC2 Each concretized security requirement is represented by exactly one treatment problem diagram.

VC3 Only domains that are related to the concretized security requirement are contained in the treatment problem diagram.

VC4 A domain in a treatment problem diagram is constrained iff a treatment is applied to it.

VC5 For each applied treatment, a treatment domain is contained in the treatment problem diagram.

4.7.3 Case Study cont'd

We had identified three high-level security requirements. Hence, we have to set up three concretized security requirements (C-SR).

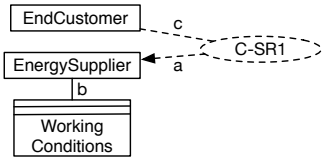

Figure 11: Case study: Treatment problem diagram for C-SR1.

C-SR1. The risk for *Integrity of tariffParameters* due to *Changed tariff parameters* caused by {*Bribe EnergySupplier*} and initiated by {*EndCustomer*} is reduced to an acceptable level according to the risk matrix of the asset by applying {*(EnergySupplier, Better working conditions e.g. more money)*}.

C-SR2. The risk for *Integrity of measuredData* due to *Incorrect transmission of data* caused by {*Disrupt frequency of LMN*} and initiated by {*EndCustomer*} is reduced to an acceptable level according to the risk matrix of the asset by applying {*(LMN, Implement checksums),(LMN, Replace frequency by 5GHz)*}.

C-SR3. The risk for *Availability of measuredData* due to *No transmission of data* caused by {*Disrupt frequency of LMN*} and initiated by {*EndCustomer*} is reduced to an acceptable level according to the risk matrix of the asset by applying {*(LMN, Replace frequency by 5GHz)*}.

We have to set up three treatment problem diagrams, one per concretized security requirement. A machine domain is represented by a rectangle with two vertical bars. Similarly, we represent the newly introduced treatment domain with two horizontal bars.

Figure 11 shows the treatment problem diagram for *C-SR1*. The domain *EnergySupplier* is constrained, because the application of the treatment *WorkingConditions*, describing a better payment for the employees, influences that domain. The domain *EndCustomer* is referred to by C-SR1, because the end customer is mentioned in the requirement.

Figure 12 shows the treatment problem diagram for *C-SR2*. Since there is no treatment for domain *EndCustomer*, the requirement only refers to it. For the domain *LMN*, there are two treatments. Thus, it is constrained by the requirement. The treatment *NewFrequency* enables the LMN to make use of a 5GHz band. The treatment *ImplementChecksums* allows the generation of message codes to verify the correctness of transmitted data.

Figure 13 shows the treatment problem diagram

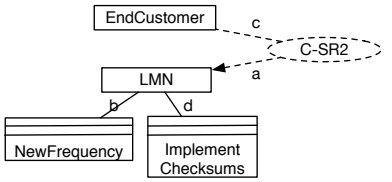

Figure 12: Case study: Treatment problem diagram for C-SR2.

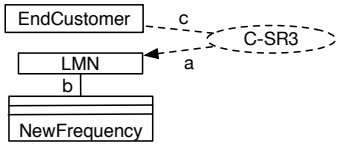

Figure 13: Case study: Treatment problem diagram for C-SR3.

for *C-SR3*. It is similar to the one for *C-SR2*, except that the implementation of a new frequency suffices to fulfill the requirement.

The final requirements model now consists of the functional requirements represented as problem diagrams and the security requirements represented as treatment problem diagrams.

5 RELATED WORK

Faßbender et al. (Faßbender et al., 2014) propose the PresSuRE method. The method provides a process to elicit security requirements. The starting point of the process are functional requirements, represented as problem diagrams. The authors define an elicitation process which consists of an identification of assets and an elicitation of attackers and their abilities based on attacker templates. 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.

The CORAS method (Lund et al., 2010) (see Section 2.2) is a model-based method for risk management. For each step, the input and output is defined as well as a language to describe the results in a model. The method is used for existing software. The security requirements are not explicitly stated, but there is a risk-based selection of treatments to achieve an acceptable risk level.

MAGERIT (Ministerio de Administraciones Publicas, 2014) is a methodology for Information Systems Risk Analysis and Management. It consists of several books. Book 1 describes the risk management method itself, which covers all steps of the risk management process and provides detailed mechanisms to evaluate the risk. There are no security requirements, and the method is not model-based.

Mayer et al. provide a risk-based security engineering framework (Mayer et al., 2005). The framework is used in the earliest stages of a software development life-cycle. It describes a way to extend existing requirements engineering methods with security aspects. The framework describes an iterative way to perform this extension. It is not model-based, and no security requirements are produced.

Herrmann et al. propose a method for managing IT risks (Herrmann et al., 2011). This method provides a risk identification with a corresponding risk prioritization and a selection of countermeasures to address the identified risk. The security requirements are elicited based on business goals. Business goals describe the expectations of different stakeholders for the software. For each business goal, one has to decide whether it is related to security. The method is not based on functional requirements, and is applied to an existing software. It is not model-based.

In Section 6.1, we refer to Microsoft's STRIDE (Shostack, 2014). STRIDE is a popular security framework which is used to identify security threats. Using data flow diagrams for modeling the system and its behavior, threats are elicted based on existing threat categories. In the end, threats are documented as a basis for the instantiation of security requirements. The security requirements are not part of STRIDE.

6 CONCLUSION AND OUTLOOK

In this paper, we have described a step-wise method to derive security requirements from functional requirements based on a risk analysis to protect valuable information. We make use of models to document the results of each step and to express the relations between the functional requirements and the identified risks. The relations between the models are clearly defined using a conceptual model. The risk evaluation and the resulting high-level security requirements ensure that only unacceptable risks are treated. This is achieved by selecting appropriate treatments and setting up concretized security requirements that specify how the necessary risk reduction can be achieved. To ensure that the security requirements can be taken into account in the software development process in a si-

milar way as the functional requirements, we represent the elicited security requirements in a similar way as the functional ones. Finally, the validation conditions of our method assist analysts in detecting errors in the application of the method as early as possible.

6.1 Experimental Evaluation

We plan to evaluate the ProCOR method with an experiment. With this evaluation, we aim to measure the performance and success rate of our method. Our experimental setup is inspired by an evaluation of Microsoft's STRIDE (Scandariato et al., 2015). In this study, the authors evaluate Microsoft's method, e.g. with regard to its performance, but do not compare it with other methods. Since we introduce novel elements such as treatment problem diagrams, it is not possible to compare our ProCOR method directly with existing ones. The cited study can therefore serve as a blueprint for our experiment.

The experiment will be carried out as an application of the ProCOR method. As an initial input, we provide a document describing a case study. It consists of a detailed textual description of the scenario, the security goals of the customer and the set of initial problem diagrams. These elements represent the input of Step 1 of the ProCOR method. We plan to have several groups of five participants which perform the experiment independently. All groups start with the same amount of (virtual) money and time that is available for the analysis. The comparison of the different groups is used to avoid statistical flaws. During the experiment, there will be some experts available for the participants who can be asked for help. Asking an expert for help will cost money.

We define the following research questions (RQ) to be addressed during the experiment:

RQ1 *How many correct threats can be identified by applying the ProCOR method?*

Scandariato et al. define an expected rate of one threat per hour (Scandariato et al., 2015). We will adapt this assumption. The threats that are considered as correct have to be defined before executing the experiment.

RQ2 *How long does each phase of the method take? Does the method help to reduce the effort for the steps?*

We define a maximum time for the analysis in person hours. We will not assign a specific time slot to any step. Hence, the participants are free in their time management. We will record the time that is used for each step. Additionally, we will make use of camcorders to record the performance. Together with the measured time, we will be able to identify problems during the application of the method.

RQ3 *How much interaction between the experts and the participants takes place during the experiment?*

A certain amount of money is available for the analysis. For each consultation of an expert, the participants have to pay which limits the amount of expert help the participants can receive. Based on the money that is left in the end, we will be able to measure the contribution of the participants compared to the input that is provided by the experts.

RQ4 *How confident are the participants of their performance?*

After the experiment, we plan to make interviews with the participants and ask them about their feelings and their self-confidence about the results. The results of these interviews will be compared with the actual results as desribed in the other research questions.

6.2 Future Work

In the future, we will investigate in more detail how the treatments are considered in the subsequent design and implementation phases. Moreover, we intend to support some of the brainstorming processes with further methods, e.g., a systematic risk identification method. Currently, we assume that there are experts with a deeper background knowledge about threats. Our vision is to provide a library of possible threats and to assist the analysts in identifying the relevant threats by searching the library. Such a library has been proposed by Uzunov and Fernandez (Uzunov and Fernandez, 2014). The search process shall be based on problem diagrams or problem frames, as these are reusable patterns.

We intend to provide a web-based tool which assists analysts in enacting the method. The results of each step are documented in a model. This allows the analysts to generate a documentation automatically and to ensure consistency between the different diagrams. Some of the proposed validation conditions can be checked automatically. The documentation can be used to certify the developed software according to a standard. The tool will provide a walktrough of the steps described in Section 4. For adding new diagrams, we will provide a graphical diagram creation tool.

REFERENCES

Faßbender, S., Heisel, M., and Meis, R. (2014). Functional Requirements Under Security PresSuRE. In *ICSOFT-PT 2014 - Proceedings of the 9th International Conference on Software Paradigm Trends, Vienna, Austria, 29-31 August, 2014*. SciTePress.

Herrmann, A., Morali, A., Etalle, S., and Wieringa, R. (2011). Riskrep: Risk-based security requirements elicitation and prioritization. In *1st Intern. Workshop on Alignment of Business Process and Security Modelling, ABPSM 2011*, Lect. Notes in Business Inform. Processing. Springer Verlag.

ISO (2009). *ISO 31000 Risk management – Principles and guidelines*. International Organization for Standardization.

Jackson, M. (2001). *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc.

Lund, M. S., Solhaug, B., and Stølen, K. (2010). *Model-Driven Risk Analysis. The CORAS Approach*. Springer.

Mayer, N., Rifaut, A., and Dubois, E. (2005). Towards a risk-based security requirements engineering framework. In *In Proc. of REFSQ' 05*.

Ministerio de Administraciones Publicas (2014). *MAGERIT - version 3.0. Methodology for Information Systems Risk Analysis and Management. Book I - The Method*. Ministry of Finance and Public Administration.

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

Scandariato, R., Wuyts, K., and Joosen, W. (2015). A descriptive study of Microsoft's threat modeling technique. *Requirements Engineering*, 20(2):163–180.

Shostack, A. (2014). *Threat Modeling: Designing for Security*. Wiley.

Uzunov, A. and Fernandez, E. (2014). An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards & Interfaces*, 36.