Requirements Engineering of an Access Protection

Sharon FRIEDRICH[a] and Barbara PAECH[b]
[a] *Institute for Computer Science, University of Heidelberg, Germany*
sh.friedrich@urz.uni-hd.de
[b] *Institute for Computer Science, University of Heidelberg, Germany*
paech@informatik.uni-heidelberg.de

Abstract. Access protection is an important requirement for systems, which handle confidential data. This paper describes an approach for the requirements engineering of an access protection using the example of an open system. A major problem of open systems is that many users with different roles access it. Moreover, the open system is connected to the Internet and has ports for connecting hardware like an external storage medium. Therefore, it is easy to steal or misuse confidential data from open systems if access protection is not existent. First, we used Task and Object-Oriented Requirements Engineering (TORE) in order to specify functional requirements on the access protection. For the elicitation of non-functional requirements, we applied Misuse-Oriented Quality Requirements Engineering (MOQARE), on which this paper is focused. Furthermore, we used the German IT-Safety and Security Standard Handbook in order to ensure the completeness of the solution requirements. For consideration of architectural requirements, we used Integrated Conflict Resolution and Architectural Design (ICRAD). It allows to analyze which design can realize which requirements and therefore to identify the most suitable one. Combining these three requirements engineering methods ensured a complete and appropriate solution.

Keywords. Access Protection, Open Systems, Requirements, Quality, Design

Introduction

In many cases, users with different roles and therefore rights have access to a system. This characteristic becomes a major challenge if the system has to handle confidential data like research results or personal data from employees or customers. Some of the users might be allowed to access the confidential data, some only partly and some are not allowed to access them at all. The system has to ensure that only authorized users can access the confidential data. This feature is called *access protection*.

Confidentiality violations can have big consequences for a company. Affected customers or employees could sue for damages. A business rival could misuse the data and gain an advantage over the company.

Misuse can be conducted in many different ways. Generally, everyone who can get access to the confidential data could violate the confidentiality. For example, a so far trustworthy employee might want to harm the company because of a disagreement. This possibility is often underestimated so that less or even no protection against internal misuses exists whereas in most cases firewalls are installed to protect against attacks coming from the Internet [1]. Another example of misuse could be that someone steals the confidential data in order to blackmail the company. The context of the system, for instance the number of different user roles or the importance of confidentiality of the data, determines the requirements on the access protection. In an environment where the users are not aware of security, so that they choose too easy passwords or let their workplace unattended, access protection has to be realized differently than for an environment where the users can be easily motivated to adhere to strict security guidelines.

As for every other software system, requirements on the access protection have to be elicited. However, requirements can easily be incorrect, incomplete or inconsistent. This paper describes how requirements engineering methods can be applied for systematically developing adequate context-specific access protection requirements using an open system as an example.

Our approach uses three different requirements engineering methods. The elicitation of functional requirements on the access protection is supported by Task and Object-Oriented Requirements Engineering (TORE) [2][3][4]. TORE uses different levels of abstraction in order to specify functional requirements. Therefore, requirements can firstly be elicited focusing on the context and not the system to be developed and then gradually be refined. Using TORE, we specified actors, user tasks, system responsibilities and use cases of the future system.

This paper is focused on the second method Misuse-Oriented Quality Requirements Engineering (MOQARE) [3][5]. It focuses on the non-functional requirements and provides an algorithm for identifying misuse cases and adequate countermeasures. Misuse cases similarly to use cases describe scenarios. They describe how the future system could be misused by a specific person. Therefore, identifying the misuse cases of a system provides a good overview over possible threats. That makes it easier to analyze and specify possible countermeasures. We used the German Safety and Security Standard Handbook "IT-Grundschutz Katalog" (abbreviated IGK in the following) [6], written by the Federal Office for Information Security (abbreviated BSI) [7] in order to verify that the results obtained through MOQARE are complete. The IGK describes possible threats to an IT system and standard security measures against them. Through our approach, we could also analyze how the requirements engineering method complements the checklists provided by the IGK.

However, specifying functional and non-functional requirements is not sufficient for systematically realizing access protection. Many detailed protection requirements depend on the architecture and design of the system. Therefore, we used Integrated Conflict Resolution and Architectural Design (ICRAD) [8]. ICRAD provides the possibility to connect requirements decisions such as prioritization of requirements or inconsistency detection with architectural decisions. Thus, this method allows us to analyze which requirements are jointly realizable and which architecture would be the most suitable one. The focus of this paper is on the benefits of systematic requirements engineering to derive complete and appropriate requirements. We therefore do not treat the implementation and the test of the system.

We investigated our approach in a case study where we applied it to a system used in a notary's office for the management of their documents, an open system without any access protection for the customer's confidential documents. We worked on this case study with a company who provides software for notary's offices and wanted to explore the feasibility and complexity of such an access protection.

The business process of a notary's office consists amongst others of the notarization of its clients. Hereby personal information like value of a property or inheritors are written down in a document and signed by both the client and the notary. Concerning the content of these documents, the notary is sworn to secrecy (Federal Notary Order §§18 [9]). Furthermore, these documents consist of personal data. That means they can be clearly assigned to a specific person. Since 1983, a right of informational self-determination exists for the German citizens. That means that everyone can determine who is allowed to collect or use their personal data, when and for which purpose [10]. This is the reason why several laws for data protection exist. Therefore, it is very important for the notary that the documents are treated confidentially.

For the management of the documents like creation, editing or exchange most notary's offices use software systems. That means the documents are created and saved on the computer. Thus, the confidential documents are available in digital form. However, the operation of IT systems increases the risk of misuses of the confidential data. For instance, digital documents can easily be copied onto external storage mediums or sent over the Internet. The misuser could steal the confidential documents and publish them.

In our case, the notary's office has an open IT system. Open systems have the characteristic that the included systems like client and server are networked with each other, often with a connection to the Internet, and that they have ports for plugging in hardware like external storage media [11]. For the creation, editing and view of the documents standard software like Microsoft Word is used.

Furthermore, most employees in a notary's office only have the most basic computer skills. They are often not aware of security so that they choose too simple passwords or even write their passwords down so that they are easily accessible by a misuser. Sometimes the computer can even be started without any login.

Moreover, with the IT system external people can get access to the confidential documents. For the installation and maintenance of the IT system, a notary's office often has a system administrator who is hired from another company. Besides, the software for the management of confidential documents is maintained by an employee of the software company that develops it. Actually, an employee of the notary's office always has to be present during the work of the aforementioned external employees. However, the system administrator often works during lunch break or in the evening after the employees of the notary's office finished their work and went home.

Often, the misuser only needs access to the computer in order to gain access to the confidential data because access protection does not exist. The aforementioned employees have access to the computer anyway so that the confidential data is not protected against them. That is the reason why it is necessary to decrease the risk of misuse and to protect adequately the confidential documents.

This paper is organized as follows: Section 1 introduces the three requirements engineering methods TORE, MOQARE and ICRAD and illustrates their artifacts through the case study. Section 2 gives a short overview over the IGK and its benefit. Section 3 describes our approach of systematically analyzing and realizing access protection requirements. Section 4 discusses our approach and describes our lessons learned. Section 5 introduces related work. Finally, section 6 gives a conclusion and outlook of the paper.

1. Our Requirements Engineering Methods

This section introduces our requirements engineering methods and illustrates their artifacts through the case study. The specifics of our integrated approach are described in Section 3.

1.1. TORE

For the identification and specification of functional requirements (FR) from a task- and object-oriented view, the Institute of Software Engineering of the University of Heidelberg co-developed a method called Task and Object-Oriented Requirements Engineering (TORE) [2][3][4]. This method comprises different levels of abstraction which make it possible to describe requirements first focusing on the context and not the system to be developed and then gradually to refine them. The different levels are called task level, domain level, interaction level and system level (see Figure 1). In the case study, we did not specify the UI-structure and the system level because our focus was on creating an access protection concept without the determination of the user and graphical user interface. In the following, we briefly describe the relevant levels of abstraction.

1.1.1. Task Level

On the task level, the actors, who will work with the future system, are described concerning their responsibilities and skills. Besides, their user tasks are defined completely abstract from the future system. Therefore, it is not yet fixed on this level for which tasks the actor and for which ones the future system will be responsible.

In the case study, we identified four different actors: the notary, employee of the notary's office, the system administrator and the software engineer. The notary owns the documents. Thus, s/he should always be allowed to access them. S/he has a high interest that these documents are treated confidentially. The notary often only has basic skills in IT security. In most cases, the notary does not manage the confidential documents but controls the correctness of them. For the management of the confidential documents an employee of the notary's office is responsible. Therefore,

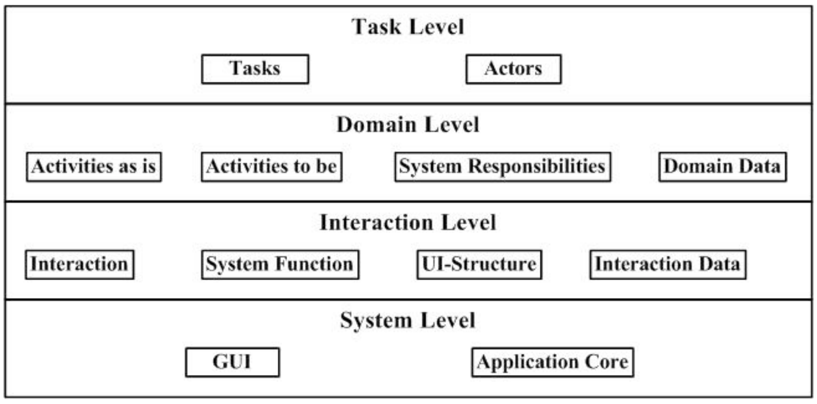

Figure 1. Levels of Abstraction in TORE.

s/he needs access to them as well. This employee often has only basic computer skills as well. In addition to the employees of the notary's office, there are also external employees. One is the system administrator who installs and maintains the IT system. For her/his work, s/he needs administration rights within the IT system. However, s/he does not need any access to the confidential documents. The other external employee is the software engineer from the company that developed the software for managing the documents. S/he has to install and to maintain the software. For corrective maintenance, s/he often gets remote access to the IT system of the notary's office. It can be that s/he needs access to a confidential document in order to find the bug. The computer skills of the external employees are advanced.

Secondly, we specified the following two user tasks: "UT 1: Management of confidential documents" and "UT 2: Installation and maintenance of the IT system".

The management of confidential documents is the user tasks of the employees of the notary's office. This includes the creation, editing, deletion and copying of confidential documents. The notary is involved in this user task because s/he at least reviews the confidential documents.

The installation and maintenance of the IT system is the user task of the external employees. For this user task, they need access to the computer and in some cases to the confidential documents. UT 2 is not as often performed as UT 1.

1.1.2. Domain Level

On the domain level, the user tasks are treated in more detail. For that purpose, the activities, which are necessary to perform the user tasks, are identified. Current activities (as is) and future activities based on the new system support (to be) are distinguished. To be activities, which shall be supported by the system to be developed, are called system responsibilities.

Besides, domain data are described as an entry in the glossary or by means of an entity-relationship diagram. Domain data are entities, which are important in this context. These are not necessarily all managed by the system. Therefore, this level of abstraction reveals how the work process will change by the system. However, on the domain level, it is not fixed in detail how the actor and the system will work together.

In the case study, we did not detail the user tasks into activities because the two user tasks are not that complex. However, we identified some (not all) future activities because the implementation of an access protection necessitates an authorization activity before any management activity or software test activity with use of confidential documents. As for the system responsibilities (SR), the system has to support "the management of confidential documents" (SR 1). Especially, it has to ensure that an access protection is created for every document and that it stays active. Besides, the system is responsible for the "authorization" (SR 2). That means it has to control whether an actor is allowed to access a confidential document. From the task description we identified the domain data (see Figure 2).

1.1.3. Interaction Level

The interaction between actor and system is defined on the interaction level. For this purpose, use cases describe how the actors perform their user tasks with the support of the system. Here, the steps of the actor are clearly separated from the ones of the system (the latter define the system functions). Moreover, not only is the successful scenario described in the use case but also possible exceptional cases. However, the use

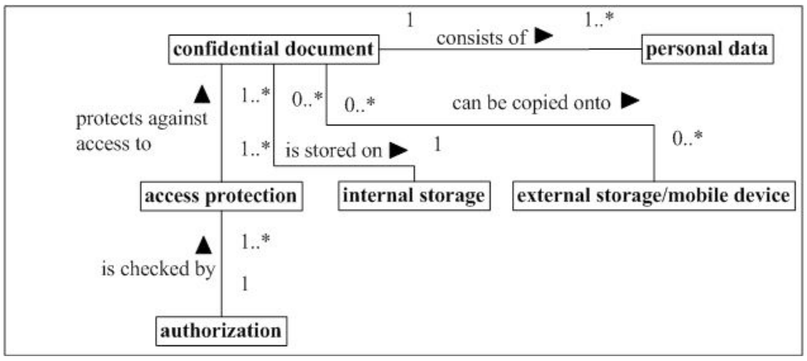

Figure 2. Domain Data Diagram.

case does not include details of the user interface (UI) (views or other UI data). That means the description still abstracts from system details.

For complex system functions, the input and output parameters are identified and the algorithm of the system function with possible exceptional cases is described.

Besides, the domain data, which were specified on domain level, are expanded with interaction data, that means all data used in the use case descriptions.

All descriptions and decisions on this level of abstraction are defined from the user's point of view.

Figure 3 shows a use case diagram, which summarizes the results obtained through TORE on interaction level in our case study. Here, UC is the abbreviation for use case and SF for system function. The system administrator is not part of the use case diagram because s/he does not need to interact with the access protection system. That means s/he is not allowed to access the confidential documents. Especially, s/he does not need to access them in order to complete her/his user tasks. Concerning the system functions, authentication is the process where the user names her/his identity and proves it, e.g. with a password. Authorization is the process where the system checks the input of the user and verifies that the user is really allowed to access.

1.2. MOQARE

The second method, which the Institute of Software Engineering of the University of Heidelberg developed, makes it possible to identify detailed non-functional requirements (NFR) and additional FR by means of misuse cases. Accordingly, the method is called Misuse-Oriented Quality Requirements Engineering (MOQARE) [3][5]. MOQARE generalizes the idea of threats (used typically in the security and safety domain) and derives every kind of NFR (e.g. also usability, maintainability or performance) by considering misuse cases. This method consists of four main steps:

- Step 1: Identification of quality goals
- Step 2: Description of misuse cases
- Step 3: Identification of countermeasures
- Step 4: Iteration of step 2 and 3 if necessary

However, the order of the steps can be changed if needed. This is only a guideline for deriving detailed NFR and FR from misuse cases. In the following, the method is

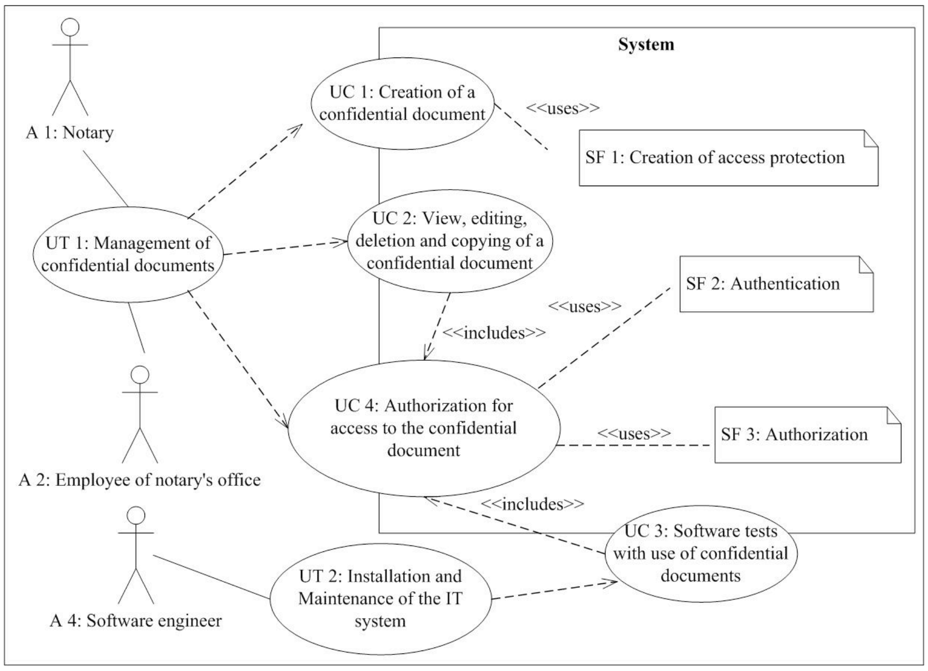
System
A 1: Notary
UC 1: Creation of a confidential document
<<uses>>
SF 1: Creation of access protection
UT 1: Management of confidential documents
UC 2: View, editing, deletion and copying of a confidential document
SF 2: Authentication
<<uses>>
<<includes>>
UC 4: Authorization for access to the confidential document
<<uses>>
SF 3: Authorization
A 2: Employee of notary's office
<<includes>>
UC 3: Software tests with use of confidential documents
UT 2: Installation and Maintenance of the IT system
A 4: Software engineer

Figure 3. Use Case Diagram.

introduced in detail. Figure 4 shows the relationships between the conceptual elements of the MOQARE method.

1.2.1. Step 1: Identification of Quality Goals

In the first step, the objective is to specify quality goals of the future system. First, *business goals (BG)* are defined. They give information about what shall be achieved with the implementation of the software system. In the case study, we identified three business goals:

1. Compliance with legal regulations and contractual provisions
2. Expansion of the notary's office (concerning clients and income)
3. Satisfaction of the employees

Naturally, the notary has to make sure that the law and contracts with his clients are observed. He also wants to achieve a financial benefit. Moreover, unsatisfied employees often work more slowly and make more mistakes like letting their work place unobserved.

A *quality goal (QG)* consists of a *quality attribute (QA)* and an *asset*. Hereby, an asset is a part of the system and its environment, for instance actors, tasks or domain data. For a list of possible quality attributes, we refer to section "A.3 List of QAs" in [3], which are mostly derived from the ISO/IEC 9126 [12]. Naturally, not all quality attributes have to be considered but only those ones, which are relevant for the satisfaction of the defined business goals.

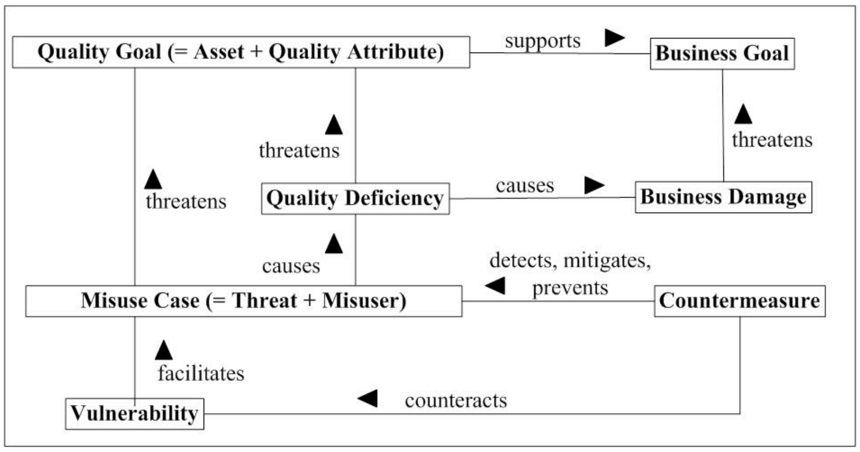

Figure 4. Relationships between the MOQARE Elements.

In the case study, we identified the personal data, the user tasks and the access protection as assets. We determined the quality attributes confidentiality, efficiency and usability as relevant because their violation would have an impact on the defined business goals. In combination, we obtained three quality goals:

1. Observance of the confidentiality of the personal data
2. Efficient performance of user tasks
3. Good usability of the access protection

The first quality goal is important in order to observe data laws and contracts with clients. It specifies that the system shall ensure the protection of the confidential documents and shall only allow authorized people access to them.

Besides, ensuring the efficiency of the performance of user tasks will prevent that clients and actors get unsatisfied. Clients usually do not want to wait too long. Actors would be stressed if the work would accumulate.

For a good usability of the access protection, the activities required for authorization like authentication need to be as convenient as possible for the actors. Especially this means that the number of necessary actions is as low as possible and that they are understandable, easy-to-learn as well as practicable. Otherwise, the actors would tend to make mistakes more easily and to work more slowly. Besides, the third quality goal shall also ensure that the notary is always able to access the confidential data without much effort at any place. However, only the notary has this privilege.

As a first step towards misuse cases, quality deficiencies (QD) and business damages (BD) are considered. *Quality deficiencies* describe system-related quality deficiencies, which prevent the achievement of the quality goals. *Business damages* prevent the achievement of the business goals. Often specific quality deficiencies are related to specific business damages. Often there are different deficiencies with respect to one goal. In the case study, we had only one deficiency for each goal, but several relationships. Figure 5 shows the relationships between the identified goals and deficiencies in the case study.

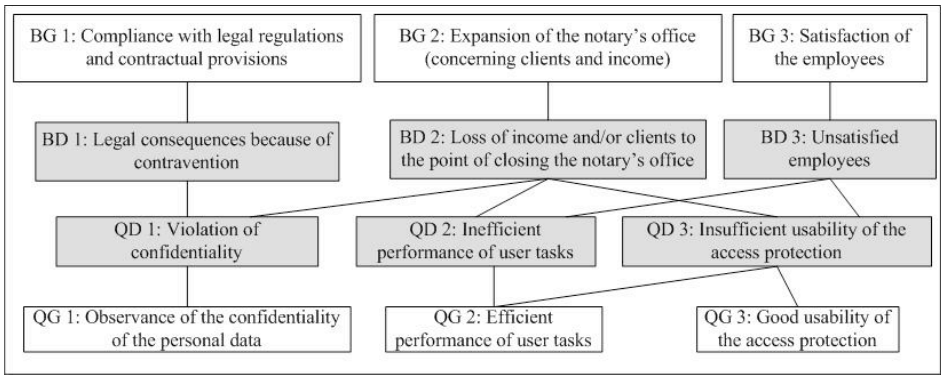
BG 1: Compliance with legal regulations and contractual provisions
BG 2: Expansion of the notary's office (concerning clients and income)
BG 3: Satisfaction of the employees
BD 1: Legal consequences because of contravention
BD 2: Loss of income and/or clients to the point of closing the notary's office
BD 3: Unsatisfied employees
QD 1: Violation of confidentiality
QD 2: Inefficient performance of user tasks
QD 3: Insufficient usability of the access protection
QG 1: Observance of the confidentiality of the personal data
QG 2: Efficient performance of user tasks
QG 3: Good usability of the access protection

Figure 5. Misuse Tree after First Step of MOQARE.

1.2.2. Step 2: Description of Misuse Cases

In this step, possible misuse cases are revealed and described. A *misuse case* describes a whole misuse scenario, including misuser, threat and its consequences (e.g., quality deficiency).

A *threat* is an action, which would cause the quality deficiency and therefore threaten the quality goal. In the case study, we continued applying the MOQARE method for the quality goal "QG 1: Observance of the confidentiality of the personal data" because it is the most important one for realizing a concept of access protection. We derived four threats from our domain data diagram obtained through TORE (see Figure 2):

1. Not active access protection
2. False authorization
3. Redundant internal storage medium
4. Unauthorized access to an external storage medium or mobile device

They were derived by looking at each domain data and its relationships to other domain data. Concerning personal data, they are included in the confidential document so that a threat for the confidential document is also one for the personal data.

We started with the relationship between access protection and confidential document. In general, the access protection has the responsibility to protect a confidential document for unauthorized access. Thus, "T 1: Not active access protection" is a possible threat for the quality goal.

Furthermore, the system has to ensure that only authorized actors are allowed to access the documents. This is realized through authorization. Therefore, another threat is "T 2: False authorization", which means that a misuser convinces the system in some way that s/he is authorized to access the confidential documents although s/he is it not.

Another potential threat exists because the confidential documents are stored digitally. The domain data diagram shows that a confidential document should always exist on only one internal storage medium. That means the confidential document may only be temporarily on additional internal storage mediums. That results in the following threat "T 3: Redundant internal storage medium". Hereby, "redundant" means that a confidential document is existent on an internal storage medium on which it should not or only for a short time be existent. An example is a hard drive, which is dismounted for reparation or sale.

Moreover, a confidential document can be stored on an external storage medium or mobile device. In this case, the threat potential is that the external storage medium or mobile device gets lost, stolen, is inappropriately disposed or used for purpose other than intended. For that, we specified the threat "T 4: Unauthorized access to an external storage medium or mobile device".

The threat is usually performed by a *misuser*, who can be a person (hackers, end-users, administrators, maintainers, developers, etc.), other systems or forces of nature like fire and thunderstorm. Not only can a misuser be an attacker who consciously harms the system but also s/he could be a normal user of the system, an administrator or maintainer who performs a misuse case for instance because of carelessness.

In the case study, we decided that the misusers are the same as the actors (see section 1.1.1) because they have the biggest threat potential for the system. Especially, even the notary is a potential misuser although s/he is not interested to violate the confidentiality of the documents. Nevertheless, s/he can initiate a misuse case because of carelessness.

Often, the threat is facilitated, made possible or even provoked by a *vulnerability*. A vulnerability is any – even wanted – property of the system, if it can be misused with respect to a quality goal.

In our case study, we identified three vulnerabilities:

1. Characteristics and personal situation of a misuser
2. The openness of the IT system
3. Use of confidential documents during software tests

The first vulnerability is not directly one of the system but highly important for the misuse analysis. This summarizes all motivations of a misuser to perform a misuse case. A misuser can have the following characteristics. In order to impress someone s/he might show confidential documents to another person, e.g. from a well-known person in the community. Another possibility is that s/he hopes for a substantial advantage for instance through selling a confidential document to the press or through blackmailing the notary's office. Moreover, curiosity is probably one of the most common characteristics and motivations for not abiding by a regulation. Especially, curious persons do not think that they harm anyone or do anything bad. Furthermore, curiosity can easily lead to other motivations. For example, if a misuser discovers the value of a confidential document s/he might want to sell it or to show it to other persons. Another characteristic could be an insufficient loyalty to the employer, another reason for not abiding by regulations. Concerning forgetfulness and carelessness, even the notary can become a misuser. Especially, forgetting a mobile device, for instance in a cab, is very common. Another example is the carelessness of employees of the notary's office like keeping their credentials at an unsafe place. Concerning the personal situation, a dismissal can be the reason why a misuser wants to take revenge on her/his employer and thus to harm her/him by publishing confidential documents. Finally, a stressful workday can encourage carelessness. As an example, an employee of the notary's office could ask the system administrator to wait until s/he finished her/his work. Afterwards, s/he leaves the computer unobserved to the system administrator.

The second vulnerability facilitates the theft of confidential documents by copying it to an external storage medium or by sending it over the Internet. Besides, the connection to the Internet makes the remote access for the software engineer possible.

The last detected vulnerability exists because the original confidential documents might be used for the reproduction of software bugs and testing in general instead of test data. Thus, the software engineer gets access to them.

The *misuse cases* are specified in detail, similarly to use cases. Like use cases, misuse cases have a flow of actions and preconditions, which comply with the identified vulnerabilities. However, these vulnerabilities are only necessary but not sufficient conditions. Furthermore, the postconditions comply with the quality deficiencies and business damages caused by the misuse case. In general, misuse cases specify what the system should better not allow.

In the case study, we identified 10 misuse cases:

1. Unauthorized access by using an inactive technique of the access protection
2. Unauthorized access by showing the confidential document
3. Unauthorized access to confidential documents on an unobserved computer
4. Unauthorized access during installation/maintenance of the IT system
5. Unauthorized access to confidential documents during remote access
6. Data theft by copying and pasting in unprotected data object
7. Unauthorized access to confidential documents because of insufficient data destruction
8. Unauthorized access to confidential documents stored on a forgotten/lost mobile device/external storage medium
9. Unauthorized access by stealing an external storage medium or mobile device
10. Unauthorized access through insufficient disposal of an external storage medium

The identification of misuse cases was supported by the IGK. It also ensured the completeness of our set of misuse cases (see section 3).

In the following, we sketch a short example of one of these misuse cases. The threat "T 3: Redundant internal storage medium" can be combined with the misuser system administrator. However, the fact that a confidential document is existent on a redundant internal storage medium is only a threat if the misuser has a certain motivation. That means, "V 1: Characteristics and personal situation of a misuser" is a precondition for the misuse case. As mentioned before, the case of redundant internal storage medium can occur for instance if it was previously used for storing the confidential documents and then replaced by another internal storage medium. A reason for that could be that an internal storage medium with more capacity was needed or that the first one has to be repaired. The misuse case "MUC 7: Unauthorized access to confidential documents because of insufficient data destruction" describes how the redundancy could happen and why it is a threat for a confidential document. The initiating misuser is the system administrator because s/he is responsible for maintaining hardware. Here it is not only her/his task to dismount and replace an internal storage medium but also to ensure that all data on it is completely destroyed after copying it to the new one. This misuse case describes what can happen if the system administrator forgets to destroy the data on the internal storage medium. After the replacement of the internal storage medium, it could be sold or given away. The receiver could restore the insufficiently deleted, confidential documents and misuse them so that the confidentiality is violated. In general, deleting files is not sufficient because it is possible to restore them. A complete data destruction is possible through overwriting the internal storage multiple times. Furthermore, not only forgetfulness could be the reason for missing data destruction but also the deceiving belief that nothing bad will happen because nobody would try to restore possibly deleted data.

1.2.3. Step 3: Identification of Countermeasures

At this point, countermeasures for the misuse cases, which were described during the previous step, are identified. *Countermeasures* can detect, mitigate or prevent misuse cases. That means they try to protect the asset by counteracting threats or vulnerabilities. Countermeasures can be new FR or NFR like new quality goals.

In the case study, the following types of countermeasures could be derived from the misuse cases.

1. Adequate application of authentication
2. Accurate distribution of rights
3. Adequate application of cryptography
4. Adequate application of monitoring
5. Regular maintenance of the security technique
6. Organizational security measures

For each misuse case, we considered security measures known from literature and the IGK. We decided whether a security measure could be classified as detecting, preventing or mitigating. Indeed the six identified countermeasures correspond to standard security measures common in literature. Besides, the identification of countermeasures was supported by the IGK in order to ensure completeness (see section 3).

As an example, this paragraph briefly describes the countermeasure “CM 6: Organizational security measures”. Organizational security measures are for example the information of a new employee and training as well as sensitization of all employees with respect to security regulations. This countermeasure is classified to be mitigating for all misuse cases. It is not completely preventing because its effectiveness depends on the characteristics of the actors. For instance, there could be actors who easily can recall regulations but there could also be actors who have difficulties with this. It is not completely detecting either. There are situations where this countermeasure can detect a misuse case, for instance the use of social engineering of a misuser, but this is not always the case and depends again on the characteristics of the actors.

The best result can be achieved if all identified countermeasures are realized. For instance, the misuse case “MUC 1: Unauthorized access by using an inactive technique of the access protection” cannot be prevented by any misuse case because each assigned countermeasure could be the inactive technique. However, altogether they can prevent that the misuse case is performed with a very high likelihood. For example, if the technique for the authentication is inactive the confidential documents will still be protected by cryptography or monitoring. The probability that all techniques are inactive at the same time is much lower than that one of them is inactive. Thus, we decided that all countermeasures have to be realized and did not perform a detailed cost-benefit or risk analysis for the individual countermeasures.

1.2.4. Step 4: Iteration of Step 2 and 3

In case that countermeasures, found in step 3, are new quality goals it can be necessary to iterate step 2 and 3. However, the number of possible combinations of assets and quality attributes is finite so that found quality goals (countermeasures) in step 3 will repeat at some point. Besides, it is always possible to finish if a refinement with new NFR or FR is not considered as necessary anymore.

We did not need to iterate step 2 and 3 in our case study because the identified countermeasures were no new quality goals.

1.2.5. Misuse Tree

As a rationale for the detailed NFR and new FR, the MOQARE method captures the relationships between the goals and misuse cases in a so-called *misuse tree* (see Figure 5 for an extract of the misuse tree of our case study).

1.3. ICRAD

The previous two sections introduced methods for specifying FR and NFR. In this section, we describe a method for connecting the specification of requirements with architectural decisions. This method is called Integrated Conflict Resolution and Architectural Design (ICRAD) and has been developed at the Institute of Software Engineering of the University of Heidelberg [8]. The considerations of architectural requirements is necessary because prioritization and conflict resolution of requirements is often only possible in the context of a given architecture, e.g. the question of feasibility of requirements can only be answered relative to an architecture.

Typically, several different architectural alternatives exist for implementing a system. Not all of these architectural alternatives might be able to realize all specified requirements. It is even possible that none of them can realize them all together. ICRAD supports to determine which requirements can be realized by which architectural alternative to which degree.

In our case study, we had to make seven architectural decisions how to implement the countermeasures (see Table 1). For each countermeasure, several variants exist concerning a certain design question like place or time of operation. We researched different designs of the standard security measures in the literature and formed architectural decisions that shall decide about related variants in design. For each architectural decision, there are several possible solutions (architectural alternatives) that correspond to the related design variants. Overall, we identified 22 architectural alternatives (chosen architectural alternatives are bold in Table 1) by our research activities.

In order to choose the most suitable architectural alternatives they are rated relative to certain criteria. These criteria are benefit, residual risk, costs and complexity. In the ICRAD method, one can choose the scale freely, e.g. from 1 to 5. They are estimated by considering the different requirements on access protection. In the following, we briefly describe each of the criteria and their estimation.

The *benefit* of an architectural alternative tells how much advantage the implementation of it will provide to the future software. The *residual risk* is estimated by considering which misuse cases could not be completely prevented because of not covered requirements. For instance, an architectural alternative cannot realize a

Table 1. Architectural Decisions and Architectural Alternatives.

Architectural Decision	Architectural Alternative
AE 1.1: Kind of credential	**AA 1.1.1: Authentication with knowledge**
	AA 1.1.2: Authentication with ownership
	AA 1.1.3: Authentication with biometrical characteristic
	AA 1.1.4: Two-factor authentication
AE 1.2: Point of time of operation of the authentication	AA 1.2.1: Authentication before access to the system
	AA 1.2.2: Authentication before access to the confidential document
	AA 1.2.3: Combination of AA 1.2.1 and 1.2.2
AE 2.1: Cryptographic technique	AA 2.1.1: Symmetrical cryptosystem
	AA 2.1.2: Asymmetrical cryptosystem
	AA 2.1.3: Hybrid cryptosystem
AE 2.2: Object to be encrypted	AA 2.2.1: Encryption of the entire hard drive.
	AA 2.2.2: Encryption of the data flow
	AA 2.2.3: Encryption of the confidential documents
AE 2.3: Duration of encryption	AA 2.3.1: Encryption during user inactivity
	AA 2.3.2: Permanent encryption
AE 3.1: Point of time of operation of the monitoring	AA 3.1.1: Video monitoring
	AA 3.1.2: Monitoring of events within the computer without system support
	AA 3.1.3: Monitoring of events within the computer with system support
AE 3.2: Time of operation of the monitoring	AA 3.2.1: Monitoring of security critical actions and events.
	AA 3.2.2: Monitoring of every access to a confidential document
	AA 3.2.3: Monitoring of data flow
	AA 3.2.4: Combination of AA 3.2.1 - 3.2.3

countermeasure against a risky misuse case. In this case, the residual risk of that architectural alternative would be high.

Cost captures how expensive the implementation of the architectural alternative will be. *Complexity* captures costs additional to implementation costs, e.g. for maintenance.

In our case study, we used the specified requirements in particular the quality goals for the estimation of the criteria. For example, "QG 3: Good usability of the access protection" influences the residual risk of an architectural alternative since users tend to make more mistakes if a system has not a good usability. Besides, our objective was to find an access protection for an open system without changing the openness. If the openness is affected by an architectural alternative, we increased its costs.

We chose a scale from 1 to 5 with 1 very low, 2 low, 3 medium, 4 high and 5 very high. We assumed that none of the criteria would have a value of 0. For instance, there is always a certain residual risk or small effort. Thus, we did not include 0 in our scale. This scale makes it possible to analyze the architectural alternatives in detail and to make them better distinguishable from each other. A brief example of one architectural decision is given in section 3.

2. The German IT-Grundschutz Katalog

This section introduces the German safety and security standard we used for completeness checks.

The Federal Office for Information Security (in German: Bundesamt für Sicherheit in der Informationstechnik, abbreviated BSI) exists in Germany since 1991 [7]. It is engaged with questions concerning IT-security. This includes the detection and analysis of security risks and threats produced by IT as well as of possible security measures. Besides, it tracks the trends in information technology in order to identify new risks or to adjust existing ones.

The BSI publishes amongst others the IGK, which introduces and describes possible threats and typical security measures [6]. The IGK distinguishes between organizational, personal, infrastructural and technical standard security measures. These can arbitrarily be combined with each other in order to set up an appropriate security concept. Moreover, the IGK introduces several modules for typical business processes and areas where IT is used, which describe possible threats and recommend security measures. For instance, there are modules for client-server-networks, for certain institutions or rooms like server room or office or for typical communicational components. The BSI always keeps the IGK up-to-date.

3. Systematic Requirements Engineering of Access Protection

Each method TORE, MOQARE, ICRAD and the IGK has been used on their own before. The novelty of our approach is how we integrated them to ensure the suitability and completeness of the access protection.. This is described in the following (see Figure 6):

1. **Specification of FR:** We specified FR on each level of abstraction of TORE except of the system level. The system level as well as the UI structure on interaction level were not relevant in order to define a basic concept of access protection.
2. **Specification of quality goals:** In the second step, we applied the MOQARE method resulting in business goals, business damages, quality goals and quality damages.

3.a. **Listing of relevant IGK threats:** We studied the IGK and listed threats, which are related to access protection and confidentiality.

3.b. **Specification of misuse cases:** Simultaneously to step 3.a, we identified threats, misuser and vulnerabilities of the open system and combined them to misuse cases. For that, we used information about existing vulnerabilities and threats obtained from literature and the list of IGK threats and analyzed whether they are relevant for this specific open system.

4. **Completeness check of the misuse cases:** In this step, we created two tables, which show the assignment of each misuse case to related IGK threats and vice versa. For that, we analyzed which threats a misuse case or respectively which misuse cases a threat comprises. If an IGK threat is not covered by some misuse case, the list of misuse cases might be incomplete. If a misuse case is not covered by an IGK threat, it might show the incompleteness of the IGK. For example, the misuse case "MUC 7: Unauthorized access to confidential documents because of insufficient data destruction" (see section 1.2.2) corresponds to the IGK threat "G 2.48 Insufficient disposal of data mediums and documents". Table 2 shows an extract of the assignment of relevant threats drawn from the IGK to the misuse cases.

Table 2. Assignment of Relevant Threats from the IGK to Misuse Cases.

Misuse Cases	Threats from IGK
MUC 7: Unauthorized access to confidential documents because of insufficient data destruction	G 2.1 Missing or insufficient rules G 2.2 Insufficient knowledge of rules G 2.48 Insufficient disposal of data mediums and documents G 2.102 Insufficient sensitization for IT security G 2.103 Insufficient training of the employees G 3.3 Disregard of IT security measures G 3.44 Carelessness in the handling of information G 5.17 Threat through maintenance work done by external employees

5.a. **Listing of relevant IGK security measures:** We also prepared a list of IGK security measures, which are recommended against the threats listed in step 3.a or which are related to access protection or our misuse cases.

5.b. **Specification of high-level countermeasures:** For each misuse case, we identified possible high-level countermeasures classified as preventing, mitigating or detecting as we described it in section 1.2.3. We kept the countermeasures on a high level because we decided that ICRAD is more suitable for analyzing the implementation details (see section 4).

6. **Completeness check of the countermeasures:** We assigned the IGK security measures to the misuse cases. Since the countermeasures are connected to the misuse cases, we automatically obtained the assignment of security measures to countermeasures. If a security measure cannot be assigned to any countermeasure, the identified countermeasures might be incomplete. If a countermeasure is not related to a security measure, the IGK might be incomplete.

7. **Identification of architectural alternatives:** In this step, we identified the different architectural alternatives, which could implement a countermeasure as described in section 1.3.

8. **Benefit and risk analysis:** We estimated the benefit, residual risk, user effort and costs of an architectural alternative by analyzing the satisfaction of relevant FR and NFR. We calculated the benefit of an architectural alternative from a formula (see Eq. (1)).

$$\textit{benefit of countermeasure - residual risk of architectural alternative} \quad (1)$$

The formula facilitates the estimation of the benefit of an architectural alternative. If an architectural alternative implements a countermeasure, it provides the benefit of the countermeasure to the future system. However, it also has a residual risk, which lowers the overall benefit of the architectural alternative. Besides, we changed the criterion *complexity* into *user effort* because in the case study the users have a big impact on the success of the access protection. Furthermore, a connection between user effort and residual risk of a countermeasure exists because a higher effort can lead to higher carelessness or malpractice. The other criteria are estimated as described in section 1.3. As a brief example, we had to decide about "AE 1.1: Kind of credential", which has the following architectural alternatives:

1. Authentication with knowledge
2. Authentication with ownership
3. Authentication with biometrical characteristic
4. Two-factor authentication

The analysis with ICRAD has the results shown in Figure 7.

For instance, the estimations for the first architectural alternative could be obtained in the following way. For the costs, we considered that a password between user and system has to be agreed upon and that maybe additional software could be necessary. We assumed that the number of different passwords for authentication would be kept low. Otherwise, QG 2 and QG 3 would not be completely realized. Based on this assumption the costs can vary from very low to low. Especially, this architectural alternative does not need any additional hardware what keeps the costs low.

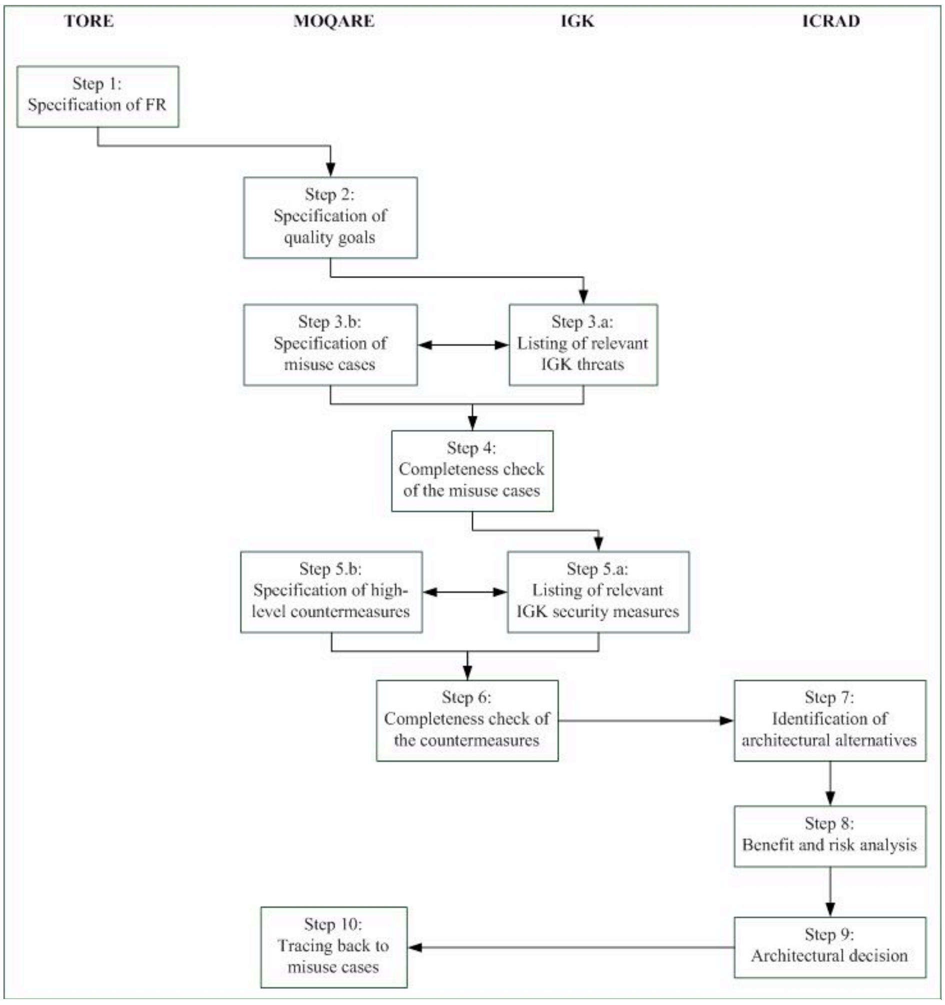
TORE
MOQARE
IGK
ICRAD
Step 1: Specification of FR
Step 2: Specification of quality goals
Step 3.b: Specification of misuse cases
Step 3.a: Listing of relevant IGK threats
Step 4: Completeness check of the misuse cases
Step 5.b: Specification of high-level countermeasures
Step 5.a: Listing of relevant IGK security measures
Step 6: Completeness check of the countermeasures
Step 7: Identification of architectural alternatives
Step 8: Benefit and risk analysis
Step 9: Architectural decision
Step 10: Tracing back to misuse cases

Figure 6. Overview of Steps of Our Approach.

Concerning the user effort, the user has to remember the password and to type it in during authentication. Considering the other architectural alternatives this results in a very low to low user effort depending on whether the user can easily remember difficult passwords or not.

For the estimation of the residual risk, we analyzed the following. Inappropriate handling of passwords exists if passwords are too easy or written down. Even organizational security measures cannot ensure that users do not write their passwords down. However, the system can enforce that the passwords have at least a certain complexity. Moreover, a misuser has many possibilities to acquire a password. If the user is not careless, the misuser can just try to guess the password. Hereby a certain complexity of the password can counteract this threat but today, many software tools, even freeware, exist for breaking a password. That is the reason why the residual risk is estimated to be medium. However, the software tool needs to test many possible combinations what can be too time-consuming for a complex password so that we did not consider the residual risk to be high.

The benefit of the countermeasure is high because through authentication the identity of a user can be verified what is necessary for the use of authorization. Without any authorization, the system cannot know who wants to access the confidential documents. Authentication is the first barrier for a misuser. However, the benefit cannot be categorized as very high because the countermeasure only mitigates the misuse cases instead of completely preventing them. For instance, in case of carelessness concerning logout or locking the system a misuse case could be performed successfully despite authentication. Therefore, the benefit of the architectural alternative is low as to Eq. (1).

9. **Architectural decision:** We considered the results of the benefit and risk analysis and decided for one of the architectural alternatives. We chose the architectural alternative with a good trade-off between benefit and residual risk. That means that the benefit is preferably high whereas the residual risk is as low as possible.

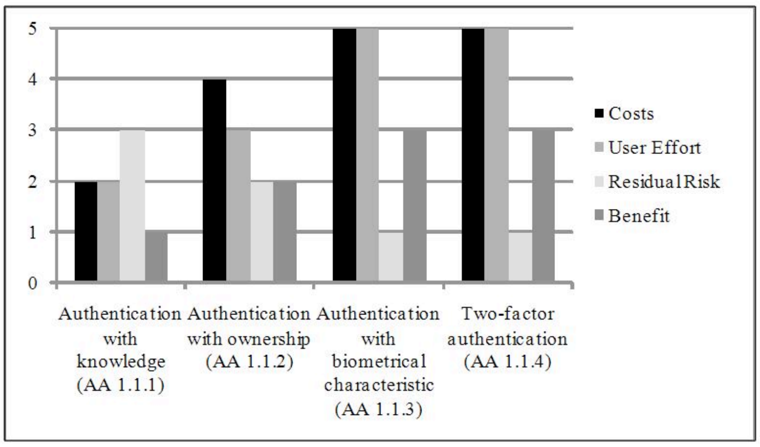

Figure 7. ICRAD Results for "AE 1.1: Kind of credential".

10. **Tracing back to misuse cases:** In this step, we traced the results of the ICRAD method back to the misuse cases in order to analyze whether they are completely prevented. In the case study, the solution obtained through ICRAD prevents or at least mitigates all misuse cases. For instance, misuse cases concerning the threat "T 2: False authorization" (MUC 2 - 5) can be prevented by this solution because the confidential documents are still protected by cryptography and data flows as well as access to confidential documents are monitored. All other misuse cases are counteracted adequately as well. The only exception is "MUC 6: Data theft by copying and pasting in unprotected data object", which is difficult to completely prevent because in some cases the employees of the notary's office need to provide the content of a confidential document decrypted (for instance for the clients) and thus have to create an unencrypted copy of it. In general, it is possible to provide the feature that content can only be copied encrypted to the clipboard. However, each notary's office has to think about in what way this would constrict their workflow.

4. Discussion and Lessons Learned

In this section, we discuss the rationale and the experiences of our approach of analyzing and realizing access protection for an open system.

Why is the IGK not sufficient? Confidentiality is widely discussed in the literature. Especially the IGK provides a compact list of all known typical security threats and recommended security measures. They are organized in modules for typical business processes and areas where IT is used rather than in sections for confidentiality, integrity and availability. Companies can combine the modules but they often need to adjust them. We discovered that modules, which do not seem to be relevant for a specific context, could include threats or security measures, which might be important. As an example, the IGK module "B 2.8 Work place at home" is not relevant since the open system operates within the notary's office. However, the module includes the threat "G 5.70: Manipulation by family members or visitors". This threat can be assigned e.g. to the misuse case "MUC 10: Unauthorized access through insufficient disposal of an external storage medium". For instance, a CD, which seems to be empty but was not sufficiently deleted, is found by a family member who gives it to another person. The recipient restores the data and misuses them.

The high amount of possible threats and security measures makes it difficult to analyze systematically what might be relevant for a given context. The IGK does not provide a systematic process to choose and to prioritize different security measures or to check whether chosen threats or security measures are complete, consistent with the system, relevant and feasible. For example, in our case, we only wanted to consider security measures that do not constrain the openness of the system. Thus, we needed a method to define the specific requirements of the case study, to compare them with the IGK and to choose the most suitable security measures.

Why is MOQARE not sufficient? MOQARE facilitates the identification of relevant security measures by analyzing gradually the threat potential of the system. The difference to the IGK is that the security measures or countermeasures are specific for the considered system since they are derived gradually from the business goals of that system. The whole MOQARE process is focused on the given system and its specific business and quality goals what ensures the relevance of threats, vulnerabilities, misuser, misuse cases and countermeasures. Another difference is that MOQARE provides a systematic process. This reduces the risk of missing, inconsistent or irrelevant requirements.

Furthermore, the classification of countermeasures with respect to detection, mitigation and prevention supported us to evaluate which countermeasures are the most important ones and which would protect the system adequately from the misuse cases. This ensures the suitability of the solution.

However, MOQARE has the disadvantage that it does not include a mechanism to check whether the results are complete. That is the reason why we were forced to do the time-consuming listing of IGK threats and security measures relevant for confidentiality.

Conflict resolution between different architectures and the requirements as well as feasibility check show the limits of the MOQARE method. Indeed, it is possible to perform a risk analysis with MOQARE [3]. It estimates amongst others the benefit of a countermeasure. Nevertheless, it is also necessary to analyze complexity and costs in order to estimate the feasibility. These criteria are included in ICRAD. Besides, ICRAD is flexible enough to allow modification in them (e.g. our change from “complexity” to “user effort”).

However, the use of ICRAD has the disadvantage that the reference to the misuse cases is burrowed in other details. In ICRAD, the focus is on the different architectural alternatives not on the misuse cases, though they are considered for the estimation of the residual risk. ICRAD does not clarify how the architectural decisions altogether affect the misuse cases. Tracing the solution back to the misuse cases makes it possible to analyze whether it appropriately counteracts them.

Our experiences: It is time-consuming to list relevant threats and security measures although it is easy to identify them for confidentiality. However, it would be even more time-consuming and too risky to decide without a systematic process which of the security measures should be included in the solution. The risk would be too high that important security measures are overseen or that a more expensive or a less suitable solution is chosen. The assignment of IGK threats to misuse cases showed us that one misuse case usually comprises many IGK threats from different modules. Besides, each IGK threat can be assigned to more than one misuse case. This illustrates the complexity and thus the need for a systematic process of the threat analysis

It was easy to integrate the requirements engineering methods: TORE is flexible enough to leave out steps, which are not necessary (e.g. system level and UI structure in our case). Similarly, MOQARE is flexible about the iterations necessary for deriving the misuse tree. For example, for our purpose high-level countermeasures were sufficient. A refinement of a countermeasure would just consider the several possible alternatives to implement it. For instance, a refinement of “CM 1: Adequate application of authentication” could be “Password authentication”. However, we decided to save the time and effort of doing another MOQARE iteration and to proceed with ICRAD. ICRAD provides better means to analyze the different possible implementation alternatives and their feasibility.

Altogether our approach systematically integrates task analysis, misuse-case analysis and IGK-analysis for requirements elicitation and conflict resolution with respect to architectural options. This ensures completeness of the requirements and the architectural decisions and their rationale.

5. Related Work

There are many approaches for dealing with security requirements in the literature. Some of them apply logic-based frameworks such as [11] and [13]. As we were working with a medium-sized company, our stakeholders were not prepared to be involved in a formal method approach. However, it was also important for us to provide a rationale derivation of arguments from goals over threats and misuses to countermeasures. In addition, we wanted to support completeness by including the use of comprehensive checklists like the IGK. Many other approaches are based on misuse cases [14], which we have also adopted in MOQARE. The related work on misuse cases is discussed in [5]. None of these approaches includes prioritization of the countermeasures. A comprehensive survey on related work with respect to prioritization can be found in [15]. The novelty of our approach is the integrated elicitation and prioritization of functional and non-functional security requirements.

6. Conclusion and Outlook

This paper introduced an approach for systematically analyzing and realizing access protection using an open system as an example. The variety of possible threats and security measures makes it difficult to find a solution, which is consistent, complete, correct and feasible. We integrated three requirements engineering methods and the IGK to ensure the quality of the solution. First, TORE provided us with an overview of the characteristics of the current and future system and FR. With MOQARE, we obtained the threat potential of the system described by TORE as well as possible countermeasures, i.e. confidentiality requirements. Finally, ICRAD helped us to prioritize the variety of possible architectural alternatives for implementing a countermeasure and ensured that the chosen solution is the most suitable one.

In our case study, we defined an appropriate concept of access protection of confidential documents in the notary's office. The company was very satisfied with the concept but they have not yet decided to implement it. In our view our approach is also suitable for closed systems since TORE, MOQARE and ICRAD are general software engineering methods that do not depend on the characteristics of the considered system. We plan to apply our integrated approach to further security problems, but also to other software qualities such as performance or robustness.

References

[1] KPMG. Profile of a Fraudster - Studie. 2007 - URL http://www.kpmg.de/docs/070420_Profile_of_a_Fraudster.pdf. - Last accessed: 01/20/2009.

[2] A. Brandenburger. Durchführung einer Requirements Engineering Fallstudie mit den Methoden TORE, MOQARE und ICRAD. Diploma Thesis. University of Heidelberg, 2006.

[3] A. Herrmann and B. Paech. Software Quality by Misuse Analysis. Technical Report. Institute for Software Engineering of the University of Heidelberg, 2005 - URL http://www-swe.informatik.uni-heidelberg.de/research/publications/SIKOSA_WP2005-AH-1_V1.5_withoutcase.pdf. - Last accessed: 01/20/2009.

[4] B. Paech and K. Kohler. Task-Driven Requirements in Object-Oriented Development. Perspectives on Requirements Engineering 753 (2003), 45-67.

[5] A. Herrmann and B. Paech. MOQARE: Misuse-oriented Quality Requirements Engineering. Requirements Engineering Journal 13 (2008), No. 1, 73–86.

[6] Federal Office for Information Security (BSI). IT-Grundschutz Catalogues. - URL http://www.bsi.de/english/gshb/index.htm. - Last accessed: 01/20/2009

[7] Federal Office for Information Security (BSI). Homepage. – URL http://www.bsi.de/english/index.htm. - Last accessed: 01/20/2009

[8] A. Herrmann and B. Paech and D. Plaza. ICRAD: An Integrated Process for Requirements Conflict Solution and Architectural Design. Int. Journal of Software Engineering and Knowledge Engineering 16 (2006), No. 6, 917-950.

[9] Bundesnotarkammer. Bundesnotarordnung §§18. Pflicht zur Verschwiegenheit. - URL http://www.bnotk.de/Berufsrecht/BNotO/Bundesnotarordnung.erster.Teil.html#18 - Last accessed: 01/20/2009.

[10] M. Sonntag. IT-Sicherheit kritischer Infrastrukturen. Dissertation. Rechtswissenschaftliche Fakultät der Westfälischen Wilhelms-Universität Münster, 2003.

[11] C. Eckert. IT-Sicherheit. Oldenbourg, Munich, 2008.H. Mouratidis and P. Giorgini. Secure tropos: A security-oriented extension of the tropos methodology. Int. Journal of Software Engineering and Knowledge Engineering 17 (2007), No. 2, 285-309.

[12] International Organization for Standardization and International Electrotechnical Commission. International Standard ISO/IEC 9126. Information technology – Software product evaluation – Quality characteristics and guidelines for their use. 1991.

[13] C.B. Haley and R. Laney and J.D. Moffet and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. IEEE Transactions on Software Engineering 34 (2008), No. 1, 133-153.

[14] G. Sindre and A. L. Opdahl. Eliciting Security Requirements with Misuse Case, Requirements Engineering Journal 10 (2005), No. 1, 34-44.

[15] A. Herrmann and M. Daneva: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research, 16th IEEE Int. Requirements Engineering Conference (2008), 240-247.