Access Control Verification in Software Systems

Bachelor's Thesis of

Julian Hinrichs

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Dr. Ralf H. Reussner
Second reviewer:	Prof. Jun.-Prof. Dr.-Ing. Anne Koziolek
Advisor:	M.Sc. Stephan Seifermann

14. May 2018 – 13. September 2018

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Julian Hinrichs)

Abstract

Security in software systems becomes more important as systems becomes more complex and connected. Therefore it is desirable to to conduct security analysis on an architectural level. A possible approach in this direction are data-based privacy analysis. Such an approach is evaluated on case studies. Most exemplary systems for case studies are develop specially for the approach under investigation. Therefore it is not that simple to find a fitting a case study. The thesis introduces a method to create usable case studies for data-based privacy analysis. The method is applied on the Community Component Modeling Example (CoCoME). The evaluation is based on a GQM plan and shows that the method is applicable. Also it is shown that the created case study is able to check if illegal information flow is present in CoCoME. At last it is shown that the provided meta model extension is able to express the case study.

Zusammenfassung

Deutsche Zusammenfassung

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

As software systems become more interconnected and complex, it becomes increasingly difficult to ensure security in general and data protection in particular. First of all, the financial loss through possible leaks are immense . But the betrayal of the customer might be the greater issue. All in all, the guarantee of privacy should be a primary design goal for all commercially used systems. Privacy, on the other hand, is a non-functional requirement and it is difficult to ensure compliance. For this reason, there are three exemplary lightweight approaches to allow a privacy analysis on an architectural level. First the approach of Seifermann[13], uses data flows to check privacy on an architectural level. UMLsec[7] uses an attacker model to verify security on an architectural level. At last SecureUML[8], that defines an extension to already existing models to allow the modeling of a secure state of a system. To evaluate these kind of approaches, case studies are commonly used.
It is not simple to create suitable case studies for a data-based architectural security analysis . Often these systems are modeled exactly for one approach. We have used Seifermann's approach[13] to get an approximate idea of how data based privacy analyses work. Based on this, we developed a process for creating a case study that can be used to validate data-based data privacy analysis approaches in general. To verify the applicability of our methodology we applied it to the Community Component Modeling Example (CoCoME)[5] and created a case study. Further, we provide evaluation criteria to measure the quality of the defined access rights. The access rights were defined during the process.Also we provided criteria to verify which information flow classes are covered by the created case study. On this basis, it is possible to decide how appropriate the created case study is for validating an data-based privacy analysis approach.

1.2. Contributions

First of all, we defined a method to create case studies that may be used to conduct and validate data-based privacy analyses on an architectural level. Also we defined an evaluation for the created case study. The evaluation allows to validate the quality of defined access rights and the covered information flow classes. We created a sample case study from CoCoME[5] and evaluated it. In procedure of creating the case study from CoCoME, we discovered some vagueness in CoCoME. Some data types were not defined at all, for example. To fix this, we provided definitions for all vague elements of CoCoME that we used in our case study. The definitions mainly include roles, data and access rights. Further, we used a meta model extension for the Palladio Component Model (PCM)[12]

provided by Seifermann[14] to model data flows in the resulting case study. We have evaluated the state of PCM with the metamodel extension to see if it is possible to use it in the case study.

1.3. Outline of the thesis

The thesis is organized as follows. In chapter 2, we introduce the foundations. In chapter 3, the method we are using to create a case study is presented. Next, we apply the introduced method over the course of two chapters. First, we analyze CoCoME (chapter 4) and define the missing elements and add it to the system, secondly, we create the case study in chapter 5. In Evaluation we evaluate the method, case study and the used modeling language and discuss the evaluation. In chapter 7, we put the contributions in the context of related work. In the chapter 8, we conclude the thesis and discuss the future work.

2. Foundations

In this chapter, I will introduce the foundations that I am using in my thesis. For each, I will give a short definition and an explanatory example.

2.1. Terminology

2.1.1. security relevant data

The term *security relevant data* as used in this thesis, describes data which should be protected by the system. Security relevant data describes the data that causes harm for user or the system. As an example the security relevant data for a software system may consist of personal related informations, like name, address, credit card, etc. But the security relevant data is from system to system different. The term security relevant data is used in this thesis to describe data that is worth protecting in a system.

2.1.2. Data flow

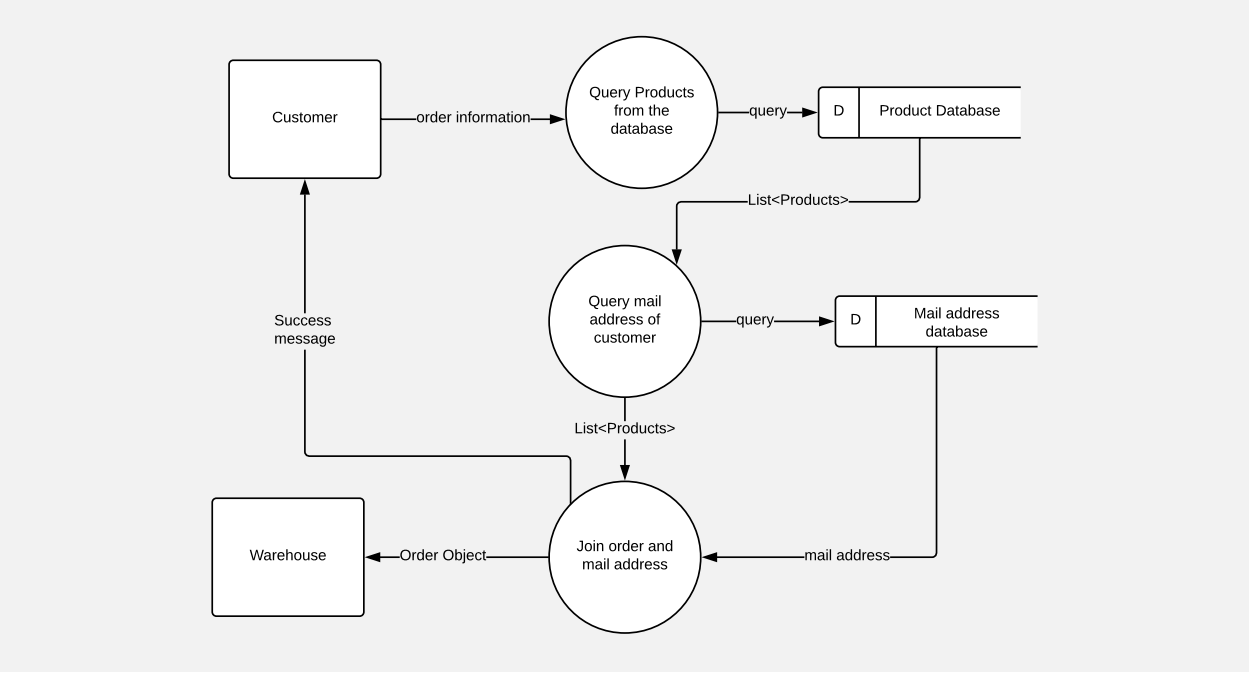

Figure 2.1.: Simple data flow diagram for linking an order of different products to a mail address

The term data flow is used in different fields of computer science. To name a few examples, data flow is used in software architecture, hardware architecture, networking, etc. We mainly use data flow in our architectural models
Data flow describes the way of data through a system. One can imagine data flows like an activity diagram. Each node in the diagram is either a process or an external entity. Processes are operations which processes the data. External entities are users or other systems which request or submit data. The edges between the nodes containing the type of the data passed. Also different data stores (most likely databases) are part of the diagram. These data stores holds different information, like the credentials, and are accessed via a process.
Figure 2.1 shows the simple process of ordering products and linking them to a mail address. The customer is already authenticated and inserts the order informations. Then the system queries all the necessary informations for the products from the database and adds them to a list. After that, the mail address of the customer is queried and packed in an object together with the product list. This object is then send to the warehouse and the customer is presented a success message. Later, the warehouse will collect the products and pack them in a delivery, but this is not a part of the diagram. For the sake of simplicity, eventual errors aren't modeled.

2.2. Palladio Component Model

2.2.1. Component based systems

Component based system are systems that consist out of components. Each component in a system is usually designed to fulfill one functionality, like connecting to a database, handling the user input, etc. It is possible that a component consists of multiply component. In such a composite component, each component also handles a single task. This allows to split larger tasks into smaller ones and assign each of them to a single component. This allows to reuse components that are designed to handle general tasks, like formatting data to a String.The definition of a component is that a component provides or requires an interface. The provided or required interfaces are used for communication between the components. Each of the interfaces is clearly defined. The components are chained together to a system using the interfaces. The main strength of the component based systems are the modularity by design. As simple example is shown in Figure 2.2. In the model a simple oline shop is shown. As shown, there is a component for managing the warehouse, a component for handling the business logic e.g. selling products and a component for the user interface. In this component based system the user interface is used by both customers and employees. Usually each of the components consists of different components for th different tasks.

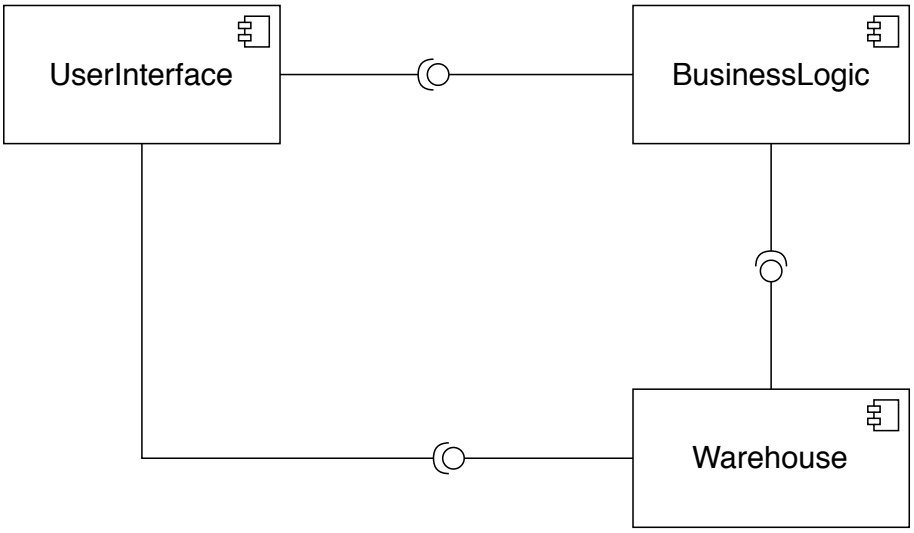

Figure 2.2.: A simple component based system

2.3. Data Centric PCM

2.3.1. Concept

The Palladio Component Model (PCM) is an architecture description language (ADL) , that allows more than only modeling a system. The strength of PCM is compute performance predictions for the modeled system [12]. A PCM model consists of four different models, each of them encapsulate their specific design knowledge and the models build on each other. After the system is complete, an architect may compute different predictions how the system will perform when it is deployed. The predictions are for cost, reliability, performance and maintainability. In the following, I will explain the individual models and their encapsulated design knowledge in detail.
First of all, for each model a role inside the PCM is assigned, which will be explained in detail later. Each of this roles conserve their design knowledge in a specific model which is used by the next role. The hierarchy of the different roles , from top to bottom, is:

- **Component Developer**
  The component developer specifies and implements the component. Furthermore, s/he has to provide additional information, that will be used for the predictions, PCM is able to compute. The resulting model is called *repository model* and is used by the system architect.

- **Software Architect**
  The system architect builds the system using components. S/He connects the different components provided by component developers to a fully functional system. The resulting model is called *system model* and is used by the system deployer.

- **System Deployer**
  The system deployer decides how the system is distributed between the available

resources. This role creates two resulting models. First, the *resource model*, in which the resources characteristics, for example transfer rate. Second, the *allocation model*, in which the allocation of the different resources to the different parts of the system is conserved. These models are used by the domain expert.

- **Domain Expert**
  Domain experts creates the *usage model* for the system. This describes user behavior. Also they specify the user workload. This workloads can be open or closed. In the closed case, a finite number of user interact with the system, in the open case the domain expert specifies the user arrival per time slice.
  As the domain expert is the last role in the chain, the models aren't used by another role. With the addition of the usage model the creation of the system model is complete.

After all the different models are created, a prediction may be computed and without a line of code is written and the architects can see if there are flaws in the architecture. The PCM models are more than just a system model. A lot of domain knowledge is encapsulated in the resulting models. The result are more a simulation for the upcoming system than a system model.

2.3.2. Meta model extension for data centric PCM

In this section, I am going to explain the meta model extension to the PCM provided by Seifermann [14]. The extension is lightweight. It aims to embed data flows in an already existing PCM model.
In the current state of PCM, it is not possible to model data flows. Therefore Seifermann created a meta model extension. The meta model extension is called data processing.
The meta models extends the Service Effect Specification (SEFF) for a component in the repository model. SEFFs are similar to UML activity diagrams. Like Activity diagrams, SEFFs specify the observable behavior for a system, in the case of PCM SEFFs specify the observable behavior for a component. SEFFS are used to model different types of actions. These actions holds various information about the systems performance and are chained together to create a sequence of events to model the (observable) behavior of a specific component.
The meta model extension is specially for these SEFFs. The main goal for this meta model extension was to model data processing. For each action in a SEFF an operation may be defined that specifies the data processing for this action. The operations model different types of manipulation of data. All SEFFs chained together in model already create a model similar to an activity diagram. For each activity a operation that processes data is specified. With this operations chained together it is possible to create a data flow for the model.

3. Method

In this chapter I am going to present a procedure to create a case study that can be used to conduct a data-based privacy analysis. The general applicability of the method is verified by an application to a relaistic system. We are going to describe the procedure and with it all necessary steps that have to be performed. Finally, I will present an evaluation approach to verify the general objective of whether the case study meets the requirements to perform a data-based security analysis.
The chapter consists of seven sections. The first section gives a brief overview over procedure of creating a viable case study. The proposed procedure consists of five concussive steps. In the next five sections, each step is explained in detail. The last section presents an approach for evaluating the case study system created. It concentrates on tow aspects, the defined access rights and the defined data flows.

3.1. General procedure for the creation of a viable case study

In this section, I describe the six consecutive steps that has to be taken to construct a viable case study. After the six steps are performed, the case study is evaluated. An overview over the six steps is shown in Figure 3.1.
The first step is to decide whether it is worthwhile to prepare a case study based on the system under investigation. The remaining procedure is divided in three main parts. First a brief survey of the system is conducted, then all necessary elements for the requirements are extracted from the system. Then current state of the requirements is reviewed, then all shortcomings are fixed and at last the scenarios are defined. The resulting case study is then evaluated.

In a second step, six requirements for creating a case study are introduced. The first requirement is that the system is modeled as a component based system or components can be derived from the current documentation of the system. The second requirements is the definition of use cases. The third requirement is that security relevant data is present and transferred in the system. The fourth requirement is the definition of roles. The fifth requirement is definition of access rights between (security relevant) data and the defined roles. The six requirement is the definition of the type of data processing within each component. Then the system is analyzed to extract the corresponding parts for each requirement.
After the third step is done there are usually shortcomings in the fulfillment of the requirements. These shortcomings include not well defined elements, missing elements or partly defined elements in the surveyed system. If there are no shortcomings at all, skip this and the next step. If there are shortcomings a summary of the shortcomings is created.
In the fifth step, all shortcomings detected in the previous step are eliminated. After that

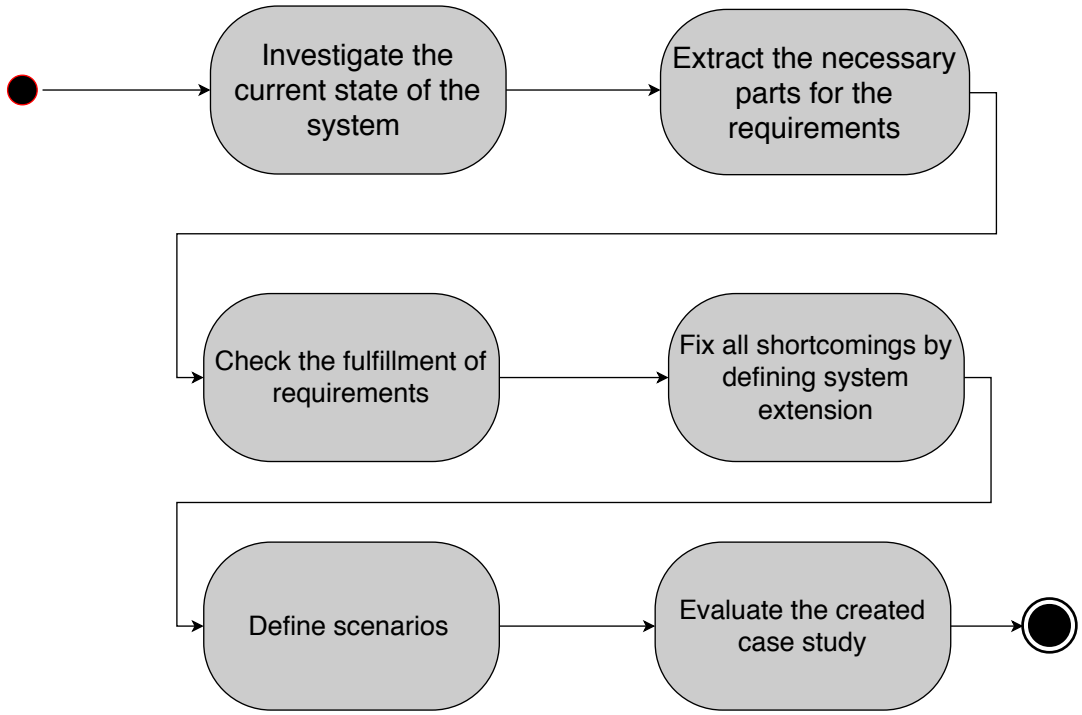

Figure 3.1.: General overview over the procedure for creating a viable case study

is done, the first milestone is reached. All necessary parts of the system are present, so that a case study can be created.
In the sixth step, scenarios are defined, from which data flows are derived. Finally, the data flows and the access rights are added to the system model, which concludes the creation of a case study .
In the seventh step, the created case study is evaluated based on two aspects. The two aspects are:

- the quality of the defined access rights through predefined criteria by Everend and Bögeholz [3].
- the covered problem classes of the created case study the quality of the defined access rights.

3.2. First look at the present system

Before the process is started, a first glance at the present system is taken. The outcome is heavily dependent on the system. This first glance shows whether it is worthwhile to create a case study. It is checked whether the system is sufficiently defined in width and depth. This check is basically a brief analysis of the available elements for six requirements. This analysis is dependent on the experience of the system architect and the system under investigation. The primary goal is to predict the amount of missing elements and therefore the necessary work. The idea for this step was to decide if the system is in a state where it

Requirements	
R1	component based system
R2	Definition of use cases
R3	Security relevant data
R4	Definition of user roles
R5	Definition of access rights
R6	Definition of the type of data processing in the components

Figure 3.2.: An overview over all the requirements for the creation of a case study.

is more than just a bare concept and the case study can be constructed with reasonable effort. The definition of reasonable effort has to be decide on a case-by-case basis.

3.3. Requirements for a case study

The second step in the procedure is to extract the corresponding elements or definitions from the system for each requirement. Altogether, five requirements are necessary for the creation of a viable case study.

R1: Modeled as component based system The first requirement is that the system is modeled as a component based system or it is possible to derive a component based system from the current state. The allows to use all benefits that a component based systems has and make it much easier to construct a case study. Firstly, in component based system there are well defined interfaces. These interfaces defining points in the architecture where data is transmitted. This ease up the creation of data flows, because one can directly identify where the data is processed. Secondly, component based systems are easy to extend. Therefore, it is not that complex to add missing elements to the model. Thirdly, the component based structure enables modularity. So it is relatively easy to just took an excerpt and take a close look on just this excerpt. All in all combined, component based systems excel in their modularity and their clear defined interfaces, that ease up the creation of a case study.

R2: Definition of use cases The second requirement is the definition of use cases in the system or it is possible to derive use cases from, for example, the documentation. This allows to get a good idea how the different users are going to interact with the system. Further, the use cases give a basic idea of the general interaction with the system.

R3: Existence of security relevant data The third requirement for a viable case study is the existence of security relevant data in the system. If there is no critical data in the system, the entire case study that is created to verify the protection of security relevant data is pointless. The definition of security-relevant data is highly dependent on the system under investigation. This means for each system and each context there are other data that is considered security relevant, therefore a lot of domain knowledge is required to identify

ACM	Usermanager
Employee	username: fullAccess password: noAccess

Figure 3.3.: Reduced ACM to the show the concept.

the data. Breier [2] introduced a basic approach to valuate assets. The basic idea of the approach is to value the various assets in terms of their priority within the system.

R4: Existence of different user roles The fourth requirement for the case study is the definition of user roles, in the following briefly roles, for the system. Roles are used to model different types of user for a system. Each role maps to one type of user. Roles are used to model the different users that are interacting with the system. Different roles uses on different data inside the system. For example the security relevant data mentioned in R3. The roles depend heavily on the system being studied. Each system defines different roles. For each system under investigation the use cases (R2) are a good source of informations. One can use business processes [10] to identify roles in a system.

R5: Definition of access rights The fifth requirement is the definition of access rights. For the definition of access rights, we use the fine-grained, higher level form proposed by Everend and Bögeholz [3] in contrast to the familiar used read/write semantics. For this step, we assume the system is in a correct state. The access rights are define on the basis of the current state of the system. As already mentioned, each component has a clearly defined interface. These interfaces are used for communication with other components. This structure is used to define a finer grained form of access control. For each component and each role, the access rights for the data types in the component are defined individually. The access rights are stored in a matrix, the so called access control matrix (ACM). In Figure 3.3 an minimal ACM is shown to get a better idea of the concept. The matrix shows that the role *Employee* in the component *Usermanager* has full access to username. Also it is shown, that the same role in the same component has no access to the password. It may happen that data is the combination of different data types. In this case, the more restrictive access rights applies. To clarify this, if the employee tries to access a (*username, password*) tuple, s/he is granted no access to the tuple.

R6: Definition of type of data processing in the components In the sixth requirement the type of data processing for each data type in each component is defined. The different processing types are described with different operations. The operations describe how and which data is processed in the specific component. The concrete operation that is needed is highly system-dependent. Operations could describe the transmission of data, processing of user inputs or operations used in the relational algebra.

3.4. Summary of shortcomings

In this step the current state of requirements is reviewed. The best case is, that all requirements are fulfilled. If this is the case, skip this and the next step. If it is not the case, create a summary of all shortcomings for the fulfillment of requirements in the system. After the first two steps of the procedure are done the usual case is that some of the requirements are not fulfilled. It may even happen that for some requirements are no elements in the system under investigation. The review of the shortcomings is important to outline which parts of the system are vague. Vague in the context of the thesis means that some elements (e.g roles, access rights) are not well defined. To give an example, it may happen that roles are just mentioned but a clear definition of the role is missing. This summary of shortcomings may also show parts where the model is not well defined. This may lead to an improvement of the model in the next step. After the summary is complete, the procedure moves to the next step.

3.5. Extending the system to fulfill all requirements

Based on the previous step, in which a summary of the shortcomings was created, the extension for the system model must be defined in this step in order to correct all shortcomings. The current documentation is the main source of information when extending the system. In the case, the documentation isn't clear or it is not possible to derive extensions for the shortcomings, the option left is to design to the best of your knowledge. This is sometimes necessary, but there are some dangers involved. It may happen that the resulting case study is less realistic and therefore provides less satisfying results, when used in a later analysis.
After all system model extension are defined and added to the model, the first milestone is reached. The system from which a case study should be created, is in the correct state. At this point all the requirements are fulfilled and the data processing inside the components has been defined. So all conditions are met to create a case study.

3.6. Creation of the case study

After the milestone is reached, scenarios for the case study are defined. Scenarios are often a more specialized description of a use case or a more specialized description of an interaction with the system. The scenarios are defined taking into account requirements R2, R3, R4 and R6. R5 is omitted in order to reduce the bias of the system architects that not only scenarios are created that are consistent with the access rights. The main part of this step is to transform the created scenarios into data flows that are later added to the model. I will provide a procedure how to add data flows to the model. This is done step-by-step. The defined scenarios serve as the basis for the transformation. To realize the transformation three successive steps are needed.
In the first step, the components that are needed to realize the specific scenario are identified. Then these components are collected and added to a reduced model. In the second step, the reduced model is analyzed. The aim of the analysis is to map data and

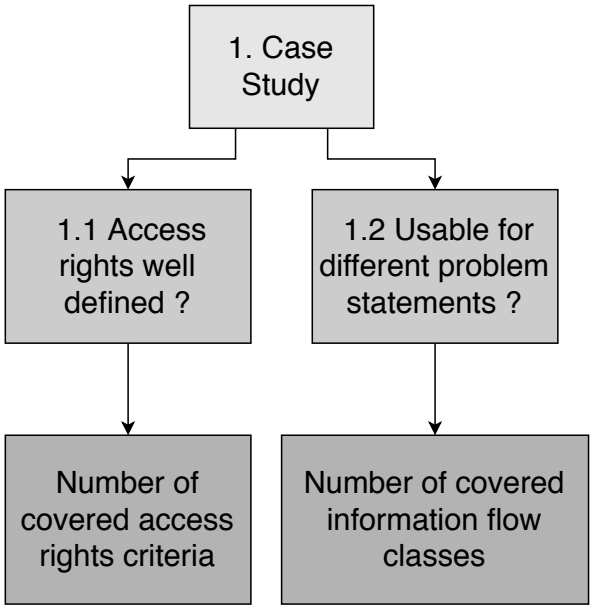

Figure 3.4.: The GQM plan for the evaluation.

the respective type of processing to components. Then the data flows can be created and added to the reduced model. In an last optional step, the reduced model is added back to the originally model. This can be done,so all data flows are available in the original model. This procedure is repeated for each scenario. After that, the access rights are also added to the component model. The structure of the access control matrix is used. For each component, the respective access rights of a role to the data types of a component are added to the respective components. After this is done, the creation of the case study is complete and can be evaluated.

3.7. Evaluation of the case study

In his step the evaluation the created case study is presented. The evaluation aims to verify if the created case study is usable for data-based privacy analysis. The evaluation follows a GQM plan [1]. The idea of a GQM plan is to split the evaluation into three parts. First, a goal for the evaluation is defined. In the second step, questions are defined. The answer to this question indicates if the defined goal is reached. In the last step, metrics are defined. These metrics allows to answer the previously asked question.
The evaluation validates the created case study. It is evaluated whether the case study is suitable for a data-based privacy analysis. The evaluation is embedded in an GQM plan. The plan is shown in Figure 3.4

The goal is to verify if the created case study is usable for a data-based privacy analysis. To verify this, we defined two questions.

- Are the access rights well defined ?

- Is the case study usable for different information flow classes ?

As a metric for the first question we use a checklist. Everand and Bögeholz [3] defined seven criteria for good access rights. Not all surveyed systems allows to check the seven criteria, concise, clear, fundamental, need-to-know, efficient, aspect-oriented and positive. Therefore it is case dependent which criteria are applicable. For each selected criterion, the system then checks whether the access rights meet these criteria. After that, based on the fulfilled criteria, it can than be decided if access rights are defined well.
The metric for the second question is also a checklist. To verify the question different classes of (illegal) information flows are defined. Possible information flows classes are that no information flows from higher security levels to lower ones, no direct information flow between different users, etc. Then it is checked for each information flow class whether the case study created expresses it. After that, based on the included information flows, it than can be decided, if the case study covers the desired number of different problem statements.

4. Analysis of CoCoME

To check the applicability of the method presented in chapter 3, we apply it to a concrete system. We chose CoCoME [5] to apply the method to, because CoCoME is a relative realistic system. In this chapter all steps up to the milestone are performed. When this steps are performed, it is possible to generate a case study from the the system.An overview over the performed steps is shown in Figure 4.1. First of all, we give a brief overview over the CoCoME system.

4.1. CoCoME overview

CoCoME is short for Common Component Modeling Example. For the sake of simplicity, a relative relatable scenario was chosen as a base for the system when it first was introduced.CoCoME was the result of a seminar at the Technische Universität Clausthal in 2006.
The basic setting for CoCoME is to abstract the products management of a big supermarket group like Lidl or Aldi. The goal of CoCoME is to provide an open source environment, in which different paradigms for component based software development can be tested. CoCoME tries to be as close as possible to the reality to provide a realistic environment to test new modeling approaches. CoCome is structured as follows. In CoCoME there are various enterprises, each of them models an individual supermarket group. Each enterprise may has various stores, with each of them selling products . Each of these stores has various cash desk, where products are sold. Furthermore CoCoME also handles the stock of each enterprise in the different stores. When a product is sold at a cashdesk or a delivery from a supplier is accepted, the current stock is automatically updated. This modeling of CoCoME provides three core functionalities:

- Operating of a register cash system
- Processing of inventory and order process of goods
- Providing of an web fronted for the services

CoCoME has been in development ever since. Over the years different variants emerged. Each variant was introduced to either react to new model pattern or solve problems that arose in the development or the deployment process. To keep this short, we will explain in detail only the hybrid cloud-based variant, as this is the variant we are using to apply the procedure.

Th hybrid cloud based variant emerged from the plain java variant, first introduced in [4]. We used the description given in [5]. The second reference is my main source for the

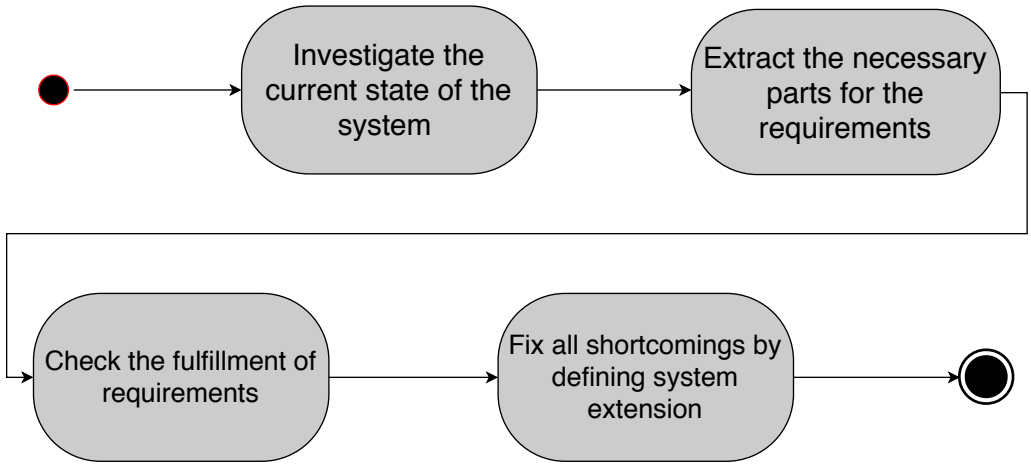

Figure 4.1.: Performed steps of the procedure described in Method

CoCoME system. A secondary reference is the source code located in git repositories. The hybrid cloud based variant was created by adding four evolutionary scenarios to CoCoME. These for evolutionary scenarios are : Platform Migration, adding a Pickup shop, database migration and addition of a service adapter. First, we will give a quick overview over the general architecture, then we will describe each of the evolutionary scenarios briefly. The general architecture for the hybrid cloud based variant are shown in Figure 4.2. As shwon, the architecture is divided in three different layers plus one layer to abstract the database. The first layer is called the Web Frontend. It consists out the Pickup shop and the webfrontned. The Webfrontend component acts as an interface to the user. Here the user interfaces are located to access the CoCoME system. It is also used to transfer the requests from a user to their respective underlying components.
The second layer are the Webservices. This component only transmit the requests from the Webfrontend. It is used to add other dynamic functions to the system.
The third layer is called Tradingssystem. This component is the heart and soul of CoCoME, because all the business logic is located here. All the different request from the users are handled here.
At last, the Service adapter is the extra layer to abstract the database. I t handles all requests issued to the underlying database. This abstraction is done to use different database types as the database for CoCoME.
The first evolutionary scenario is platform migration. This is done to reduce costs. The CoCoME system is moved to a cloud environment for easier scalability.
The next evolutionary scenario is the addition of a pickup shop. This was done for the enterprises to stay competitive. All integrated enterprises concurrents with big online shops that ships their products.To use Pickup shop customer create an account. Now they can conduct their orders online and pick them up at their chosen store. The payment is either via credit card in advance o directly in the Pickup shop.
Another evolutionary scenario is the Database Migration. This was necessary, because after the addition of the pickup shop they are an increased amount of sales thus a higher performance for the database is needed. As a solution the database was migrated to another

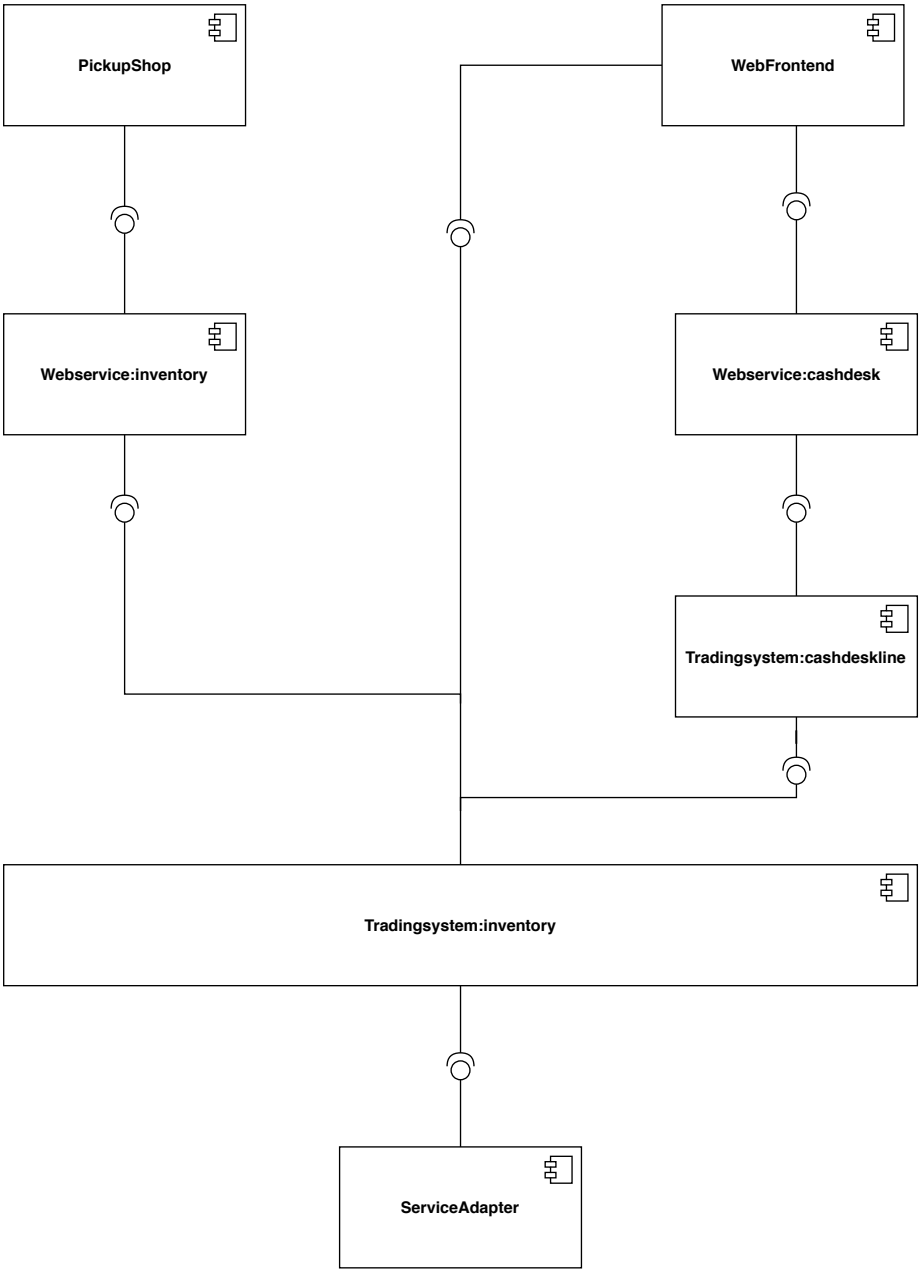

Figure 4.2.: The figure shows a simplified overview over the hybrid cloud based variant

cloud service.
The last evolution is the addition of the service adapter, which manages the connection to the migrated database and abstract this connection for the remaining system.

Why we chose CoCoME as the system to apply the procedure on The decision to use CoCoME as the system for applying the method was based on many reasons. The first is that CoCoME has already been modeled as a component-based system. Another reason is that CoCoME is described in detail in [5], which served as a good starting point. On the other hand, CoCoME is more of a minimal solution to what is actually planned to achieve with CoCoME. One can say, CoCoMe is rather a proof of concept than a functional system.

4.2. Application of the method

Once the system under investigation has been described in the previous section, we begin to perform the different steps for the creation of a case study. In various publications describing the creation of a case study it is pointed out that several sources of information should be used. We used as our main source of information the CoCoME tech report [5] and the CoCoMe book [11], were the system is described in detail. Further we used the actual code of the implementation . If there are differences between documentation and code, the definitions from the documentation applies.

4.2.1. Investigation of the current state of CoCoME

The aim of this step is to decide whether it is worth using the system for a case study. We check how complete the documentation is. This means to look for e.g. if the system models are defined or easy to generate. All in all, the step is to the decide if the systems documentation is far enough developed.
In the case of CoCoME, a tech report and an implementation is available. Therefore, we decided the sate of CoCoME is developed enough so that is worthwhile to apply the procedure and create a case study.

4.2.2. Identify relevant parts for the requirements

In the next step, we identify the necessary parts of CoCoME for the six requirements listed in Figure 4.3. In the following for each requirement we identify and extract the corresponding parts of the system.

4.2.2.1. R1: Component based system

This requirement is met because a component-based system model is defined in the CoCoME technology report.The requirement was defined to take advantage of all the strengths of a component-based system model.

Requirements	
R1	Component based system
R2	Use cases
R3	Security relevant data
R4	User roles
R5	Access rights
R6	Type of data processing in components

Figure 4.3.: Overview for the requirements for a case study

4.2.2.2. R2: Existence of use cases

The CoCoME documentation defines thirteen use cases, so this requirement is also met. The use cases are important to get an idea for using the system but also to understand how different users interact with the system.

4.2.2.3. R3: security relevant data

Since the amount of data is relatively large in CoCoME, we have grouped the data into different equivalence classes. The data type in each class has the same security level.

- customer related data
  This class describes all data types that are solely related to the customer role. This includes names, addresses and credit card details. Since the data is personal data of an entity , we classified it as **security relevant data**.

- product and sales related data
  All the data that is related to the products and the sales process. Examples for data types are price of a product, quantity of a product, date of shipping etc. data in this class may become security relevant if it enables traceability to a customer. For this fact product and sales related data is not defined security relevant. It is obvious that access to this data can cause damage to an enterprise. The case study aims to allow data-based privacy analysis in a system. The threat, access to product and sales related data poses comes from the outside and therefore is not in the scope of this procedure for creating a case study.

- user related data
  User related data refers to the introduced accounts with he addition of the Pickup shop. It is needed that each role is registered with an account in the system. The user data type holds the *username* and the credentials for an account. We classified this data as **security relevant**, because access to user data allows to log in as the role.

- System related data
  In this class mainly summarizes the created query for access to the underlying data. Access to the issued queries allows to cause damage in a lot of different ways.

It is possible to, for example, alternate the database, read user informations, etc. Therefore this data is classified **security relevant**.

4.2.2.4. R4: roles

CoCoME defines five different roles in the tech report [5]. These are the *Customer*, the *Cashier*, the *StoreManager*, the *Enterprise manager* and the *Stockmanager*. There is no separate description of the roles. All the responsibilities are derived from the use cases described in the documentation and/or the implementation. The key informations we were looking are the tasks that a role performs in the context of CoCoME and which data is provided/required by the role.

- After analyzing the implementation, the *Customer* role is the role of all the customers of an enterprise, be they private customers or business customers. The role buys products, either in the stores or in the Pickup shop. The data provided by the customer role are the customer related data, It requires product related data to, for example, decide which products to buy.

- For the *Cashier* role, we rely on the defined use cases in CoCoME tech report. As shown, the role takes over the sale of products and therefore requires product & sales data. The role provides product and sales-related data like the data that is generated when orders are placed.

- For the *StoreManager* role, we relied on the use cases. The *StoreManager* handles the tasks of a single store. The role provides and requires product& sales related data.

- After an anlysis of the use cases, we are not sure what the concrete tasks are for the *EnterpriseManager* and the data s/he requires or provide. Form the name it can be derived that the role is in charge of a whole enterprise.

- After analyzing the use cases, it is clear that the *StockManger* handles the stock of a single store. Therefore the role requires product&sales related data and provides the same type in the system.

Since the hybrid cloud based variant is examined by us. All roles provide user data, because each role needs an account in CoCoME.

4.2.2.5. R5: access rights

For the access rights, we used the finer grained, high level from proposed by Everend and Bögeholz [3]. We tried not to rely on a read/write semantic. Therefore,as described in Method, we defined for each component for each role the access to the different data classes. We defined four different privilege level for the access to data. In the following listing . We used the identified elements from the previous requirements R2-R4 to define fitting access rights for CoCoME. In the following list the access rights we defined are shown.The access to data decreases with the ascending numbers.

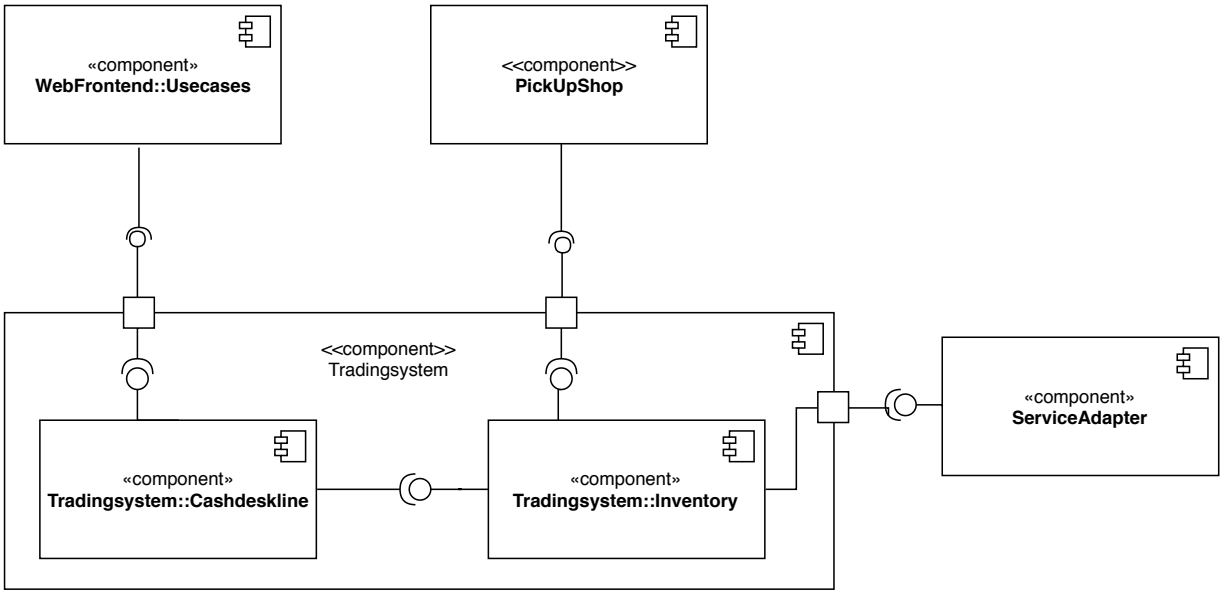

Figure 4.4.: The figure shows an simplified overview of the CoCoME. The Webservice component is omitted, because in the current state it only transmits the data.

1. **FullAccess**
   This access rights defines that the associated role has access to all data types for the respective class in the component

2. **AccessToUsedData**
   This access right defines that the role has accessed to it used data types for the respective class int the component. Used data in this context means the data types that are required by the role to perform its defined tasks.

3. **AccessToOwnedData**
   This access right defines that the associated role has access to its , for example, own user data or customer data. This access rights ensures that roles may access data that is provided by it but not needed for their tasks.

4. **Default** This access right forbids the complete access to all data in a class for this role and component.

Another part of this requirement is the identification of the present data types in each component. After analyzing the code and the CoCoME tech report, we identified data for each class of data in each component. Also we identified that no role except the customer may access the PickupShop component. In **??** a simplified overview over the architecture of CoCoMe is shown to better understand the complex matrix shown in **??**. We omitted the Webservice component for the fact that it transmits the data to the underlying components. For the fact that the ACM in **??** is rather large and complex, we point out the most important points. In the CoCoME tech report it is stated that only the customer may access the *PickupShop* component. From the role definition we identified the EnterpriseManagaer handles an whole enterprise, therefore the role is allowed to

access all components. For this reasons, no access rights defined for the other roles in this component. Further, the StockManager has no access to the Tradingsystem:cashdeskline component, because it is not needed to fulfill his tasks. The role handles the stock and has therefore no tasks that involve the component. Since no role can access CoCoME without the components Webfrontned and PickupShop, the access rights are inherited. This means that the access rights are the same in all underlying components. The idea behind this is that it is also important which role which data has entered into the system. No role has access to system related data. The StockManager has access to customer related data due to the use case 13 (UC13).

ACM	Webfrontend		PickupShop		TS:cashdeskline		TS:inventory	
customer	customer data	3	customer data	3	customer data	3	customer data	3
	user data	3	user data	3	user data	3	user data	3
	p&s data	3	p&s data	3	p&s data	3	p&s data	3
	system data	4	system data	4	system data	4	system data	4
cashier	customer data	3	customer data	4	customer data	3	customer data	3
	user data	3	user data	4	user data	3	user data	3
	p&s data	2	p&s data	4	p&s data	2	p&s data	2
	system data	4	system data	4	system data	4	system data	4
StockManager	customer data	2	customer data	4	customer data	4	customer data	2
	user data	3	user data	4	user data	4	user data	3
	p&s data	2	p&s data	4	p&s data	4	p&s data	2
	system data	4	system data	4	system data	4	system data	4
EnterpriseManager	customer data	2	customer data	2	customer data	2	customer data	2
	user data	2	user data	2	user data	2	user data	2
	p&s data	2	p&s data	2	p&s data	2	p&s data	2
	system data	2	system data	2	system data	2	system data	2
StoreManager	customer data	2	customer data	4	customer data	2	customer data	2
	user data	3	user data	4	user data	3	user data	3
	p&s data	2	p&s data	4	p&s data	2	p&s data	2
	system data	4	system data	4	system data	4	system data	4

Figure 4.5.: The ACM of CoCoME. The legend for the ACM is: 1 is the access right *FullAccess*, 2 is the access right *AccessToUsedData*, 3 is the access right *AccessToOwnedData*, 4 is the access right *Default*.

Types of data processing	customer	user	sales and product	system
Webfrontend	transmit	transmit	I/O-processing transmit	non-existent
PickupShop	transmit	transmit	I/O-processing, transmit	non-existent
Tradingsystem: inventory:app	alter transmit	alter transmit	alter	non-existent
Tradingsystem: inventory:data	relational algebra operations	relational algebra operations	relational algebra operations	alter
Tradingssystem: cashdeskline	alter transmit	non-existent	alter transmit	non-existent

Figure 4.6.: Overview of the different types of data processing in the components.

4.2.3. Type of data processing in components

In the last step of the procedure for each component the type of data processing is defined. we identified after an analysis of the code and the CoCoME tech report three different types of data processing.

- relational algebra
  This type describes the operations of the relational algebra on a database
- transmission of data
  This type describes the transmission of data through a component
- alternation of data
  This type includes the most. It describes each operation that alters the data type. Good examples the creation lists or merging two or more data types into one.
- I/O related processing
  This type describes the processing of the user and/or output.

Then we have assigned one or more types for each component and each role. We omitted the Webservice component because it only transmits the data. The result is shown in Figure 4.6. As shown, we divided the Trandingsystem:inventory component. This was done because the sub-components have different type of processing.

4.2.4. Analysis of the current state for CoCoME

With all necessary parts defined and identified, we analyze the current state of CoCoME. First of short disclaimer for the system.

- **Disclaimer to CoCoME** In the current state, CoCoME is more proof-of-concept than a working system. Due to the fact, that CoCoME is in constant development and originally created to test design paradigms on it. Therefore, CoCoME is relatively

limited in its functionalities. Before the introduction of the Pickup shop, there were not any use cases usable to conduct any security analysis.

In the following, for each requirement the current state is briefly explained.

- R1: Modeled as a component based system
  In the CoCoME tech report a detailed system model is defined. So this requirement is fully accomplished.

- R2: Existence of use cases
  Uses cases are defined in the CoCoME documentation. So this requirement is also fulfilled.

- R3: Existence of security relevant data
  In CoCoME is security relevant data present. But for some mentioned data a definition is missing.

- R4: definition of user roles
  For CoCoME the different user roles are mentioned and relatively clearly defined. Some roles are rather vague, but clear tasks may be derived.

- R5: Definition of access rights
  This definition is dependent on the requirements R2-R4. Therefore, with the shortcomings in the existence of elements/definitions, one can say that the access rights are not well defined.

- R6: definition of the different types of data processing
  The definitions for this requirement are dependet on R2-R4 and for the fact that some parts are missing, the types of data processing are not well defined.

4.3. Summary of shortcoming in CoCoME

4.3.1. use cases

In the use cases, there are shortcomings. While analyzing them, we found that only the newly added ones by the addition of the pickup shop processing security relevant data. Therefore the pool of possible use cases that later may be used to create scenarios from, are limited.

4.3.2. Security relevant data

As previously said, definitions for data are missing. We have found that there is currently no definition of how to connect a customer to their purchased products. This missing data is used in the UC13 described in the CoCoME tech report [5]. Another case of missing data definitions, happens in the same use case. A report data type is mentioned but never clear defined. So this data type is also missing.

4.3.3. Definition of roles

There are some vagueness in the roles, which possibly affects the ACM. But all in all, the tasks of the roles are clear.

4.3.4. Definition of access rights

The definition of access rights is strongly dependent on the roles and the security-relevant data. As things stand at present, we are not yet able to make any statement about access rights deficiencies. The made assumptions and definitions have to be revisited when the missing definitions for data and roles is added.

4.3.5. Data processing in security relevant components

The definitions made for the data processing have to be revisited as well, when the missing definitions for the data and the roles are made.

4.4. Add system extensions

In the next step, we extend CoCoME so that all requirements are. In the previous step all shortcomings are listed. We gibe a short overview how we extended the system to meet all the requirements. As previously said, the use cases are already well defined, so there is nothing to fix.

- **security relevant data**
  We added new data to the customer related data. The customer object holds a list with all ordered products. With this addition, the shortcoming, that there was no connection between customer and the ordered products is fixed. We also defined the *report* data type. This data type is used to display the purchased products of a customer to the stock manager. We defined it as a list and it is placed itn product& sales data class.

- **roles**
  In the roles there is some vagueness. Especially in the *SotckManager* role. The setting in which the *StockManager* operates is clear, but the inherited responsibility are not. So we defined, that the *StockManager* only handles the stock. This includes, managing the warehouse, commissioning of goods, etc. Another resposibility we added , is to predict the future needs of products. To perform any of this tasks, no customer related data is needed. Further, we defined the role of the *EnterpriseManager*. We defined it as the highest role of the respective company and therefore has access to all data within the company.

After all the definitions for data and roles is added, whe review the ACM and the types of data processing.
An updated version of the ACM is shown in **??**. As defined the EnterpriseManager got access to all data except the system related data and the Stockmanager has lost access to

customer related data.
For the types of data processing for the single components nothing changed.

ACM	Webfrontend		PickupShop		TS:cashdeskline		TS:inventory	
customer	customer data	3	customer data	3	customer data	3	customer data	3
	user data	3	user data	3	user data	3	user data	3
	p&s data	3	p&s data	3	p&s data	3	p&s data	3
	system data	4	system data	4	system data	4	system data	4
cashier	customer data	3	customer data	4	customer data	3	customer data	3
	user data	3	user data	4	user data	3	user data	3
	p&s data	2	p&s data	4	p&s data	2	p&s data	2
	system data	4	system data	4	system data	4	system data	4
StockManager	**customer data**	**4**	customer data	4	customer data	4	**customer data**	**4**
	user data	3	user data	4	user data	4	user data	3
	p&s data	2	p&s data	4	p&s data	4	p&s data	2
	system data	4	system data	4	system data	4	system data	4
EnterpriseManager	**customer data**	**1**	**customer data**	**1**	**customer data**	**1**	**customer data**	**1**
	user data	**1**	**user data**	**1**	**user data**	**1**	**user data**	**1**
	p&s data	**1**	**p&s data**	**1**	**p&s data**	**1**	**p&s data**	**1**
	system data	**4**	**system data**	**4**	**system data**	**4**	**system data**	**4**
StoreManager	customer data	2	customer data	4	customer data	2	customer data	2
	user data	3	user data	4	user data	3	user data	3
	p&s data	2	p&s data	4	p&s data	2	p&s data	2
	system data	4	system data	4	system data	4	system data	4

Figure 4.7.: The updated Version of the ACM for CoCoME after the shortcomings are fixed. The legend for the ACM is: 1 is the access right *FullAccess*, 2 is the access right *AccessToUsedData*, 3 is the access right *AccessToOwnedData*, 4 is the access right *NoAccess*.

With all the extensions added to the system, the mile stone is reached. In CoCoME all elements are available for creating a case study for databased privacy analysis.

5. Case study system

After we reached the milestone in the last chapter, we are going to create the case study in this one. Therefore we have to define scenarios. Scenarios describe a course of actions in the system. Also the roles and data involved are specified. Scenarios are created to describe the flow of data and the course of actions in a comprehensive way. For presented case study system, we defined two scenarios in CoCoME. All defined scenarios are later converted into data flows by using the matrices in Figure 4.7 and Figure 4.6. The data flows then added to the system model, which concludes the creation of the case study. After that, the created case study can be evaluated. The basic

5.1. Scenario: stock manager requests the report for a customer

5.1.1. Description

The scenario is derived from UC13. This use case is described in the CoCoME tech report [5]. In this scenario, the stock manager requests an report for the purchased goods of a customer. To identify the customer among all customers, the stock manager has access to the ID of the customer. S/he enters the ID in the system, then the request is processed by CoCoME. At the end, the stock manager is presented with a full report. This report contains all purchases of the customer.
The usage model is relatively simple. The StockManager accesses the system with an customers id and in the end is presented with a report of all purchased products for this customer.

5.1.2. CoCoME excerpt

After the usage model is done, we show the excerpt of CoCoME in which this scenario takes place. This is just an excerpt of the whole system. The scenario only uses four components, the Tradingssystem:inventory:application, the Webfrontned, the Tradingsystem:inventory:data and the Webservice.

Figure 5.1.: The usage model for the Scenario: StockManager requests a report for a customer.

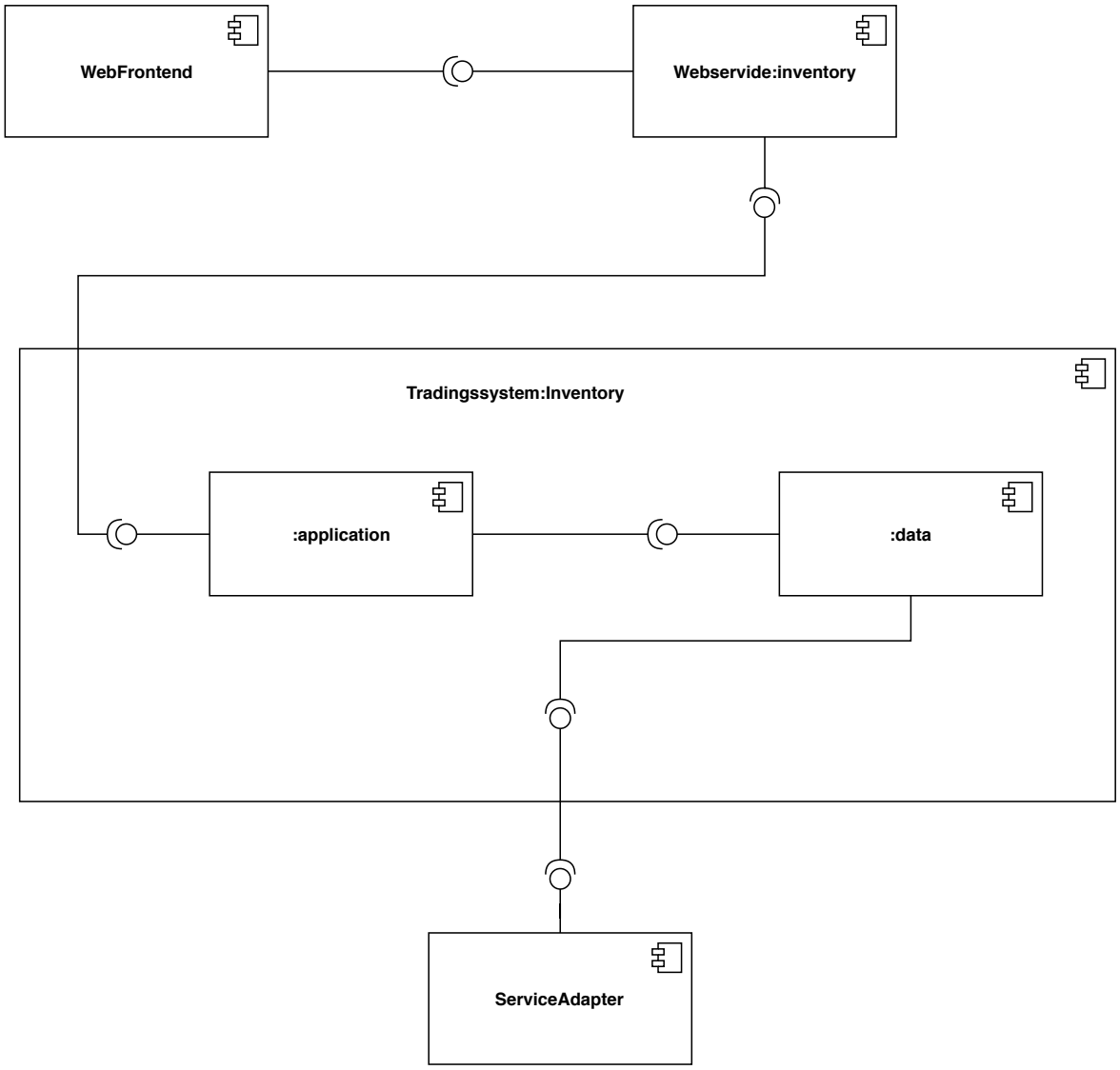

Figure 5.2.: Excerpt of CoCoME in which the described scenarios is processed

ACM	Webfrontend	TS:inv:application	TS:inv:data
stockmanager	customer data : 4 user data : 3 p& s data : 2 system data: 4	user : 3 customer : 4 product & sales : 2	user : 3 Figure 5.2) customer : 4 product & sales : 2

Figure 5.3.: An ACM showing the access rights for the case study system.

Types of data processing	customer data	p&s data	system data
Webfrontend	transmit	I/O operations, transmit	non-existent
TS:inv:application	alter, transmit	alter	non-existent
TS:inv:data	relational algebra	relational algebra	alter

Figure 5.4.: A matrix showing the data processing for each component

5.1.3. Data types and access rights

Previously, the data was divided into four classes.In the following listing the data classes and the concrete data for each class is shown.

- user related data
  No data from this class is used in this scenario
- customer related data
  The ID from the customer is part of the scenario
- product & sales related data
  In the scenario two concrete types are used. First, the IOrderEntries type.This type is a list of all purchased products for an order. This type is a collection of all products in an order. The second type is the report, which collects all IOrderEntries in one list and transform it to a human readable document.
- system data
  The system data are the queries used to request the IOrderEntries from the database.

Next, we extract the access rights and the types of data processing from the corresponding matrices. The access rights matrix for this scenario is shown in Figure 5.3. We omitted the Webservice component for the fact that it only transmits the data in this scenario. The matrix for the data processing is shown in Figure 5.4.

5.1.4. Component behavior

The shown excerpt is extended with SEFFs (subsection 2.3.2). For each component a SEFF is defined, which models the observable behavior for this component. In the following the observable behavior of the scenario is described first, then the model extension with the

described meta model. Finally, the resulting data flow is displayed.

5.1.4.1. Observable Behavior

The ID is entered via the WebFrontend component. After that, the ID is passed through the Webservice component to the Tradingsystem:inventory. The Tradingsystem:inventory has two inner components. First the ID is passed to the Tradingsystem:inventory:apllication in which it remains for the time being. Then the whole database is queried. In the next step, first the responding database entry for the ID is selected. Secondly, the received database entry is altered into a query used to select all IOrderEntries for the customer. The result is merged into a list. This data is transmitted back to the Tradingsystem:inventory:application, where the data is transformed in a report. A Repor tis a human readable document. Then the report is passed through the Webservice back to the Webfrontend where it is presented to the stock manager.

5.1.4.2. Data flow definition

For each SEFF one or more operations are added which describe the data processing for the component. For each operations it is defined which type of data processing is done for which data.

- Webfrontend
  Transmit the ID (customer data) and the report (product& sales data).
- Tradingsystem:inventory:application
  Transmits the ID (customer data), alters and transmits the report and IOrderEntries (product& sales related data).
- Tradingsystem:inventory:data
  Transmits the IOrderEntries (product& sales related data), take the ID (customer data) alter it to a query (system data) and creates a new query (system data).

The different operations are added to the respective components. The resulting data flow is shown in Figure 5.5.

The data flow is as follows. The ID from the customer is transmitted to the Tradingsystem:inventory:data. Then the respective IOrderEntries are selected and merged into a report. This report is then transmitted back to the StockManager.

5.2. Scenario: Support employee requests information for an order

This scenario was not defined on the basis of an use case. After we investigated CoCoME further, we could not find any fitting use cases in which security relevant data is present. We mostly looked for use cases, that process customer related data. So we created a new scenario.

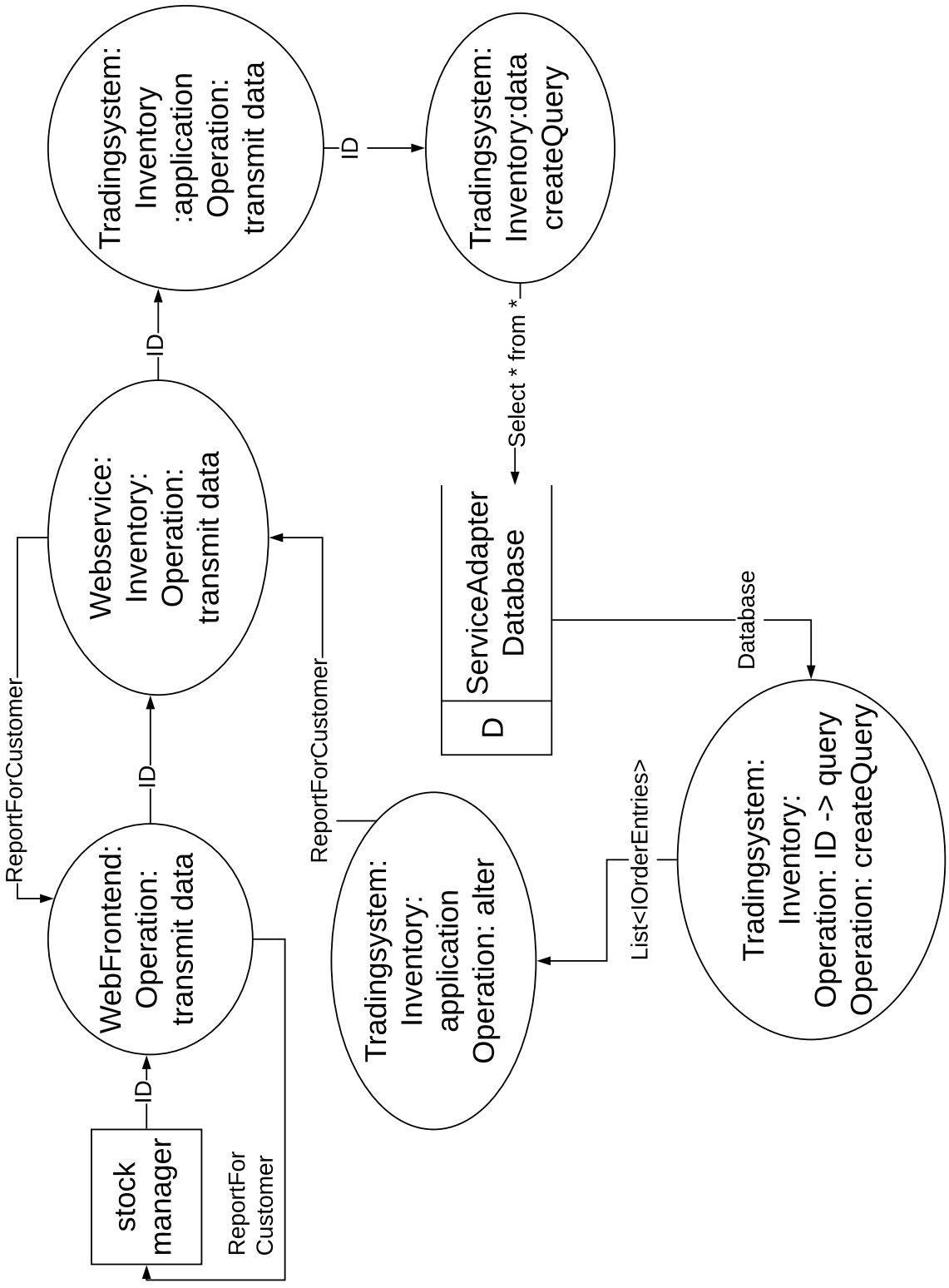

Figure 5.5.: Resulting data flow for the scenario: StockManager request the report for a customer

5.2.1. Description

The fundamental setting of this scenario is that a customer has a problem with an order s/he issued. To solve the problem, the customer authenticates her/himself in CoCoME and issues a ticket for the order. The support employee then request the particular order from CoCoME. After the report is received, the scenario has ended. With the report at hand, the support employee should be able to solve the customer's issue.
The usage model is relatively simple for the scenario. The support employee request with the customer and order ID a report of the current state of the order.

5.2.2. Identification of a new use case

We found the scenario described in so fitting for CoCoME, that we derived a new use case from it. First we introduce the new role of support employee before we move on to the description of the new use case.

5.2.2.1. Description the new role: Support employee

The support employee role processes the tickets issued by customers. A ticket is issued if there are problems with an order. The support employee receives access to the order and checks their current status. Then he takes an appropriate action to solve the issue if possible. The actions are dependent on the actual order. They can reach from resending the order to taking no action, because the customer tries to scam the enterprise.

Role description The support employee is a new role in the system, so we had to define the access rights for this role. It requires data beyond the data it owns itself, this includes customer and product & sales related data. The role may provide customer and product & sales related data. It may happen, for example, that an order is changed, because some products are not available and the customer decides to take a similar product. Also customer may used an invalid credit card in the order under investigation and the credit card details are changed.

access rights From the description we derived the access rights shown in Figure 5.2.2.1. To point out the important definitions. The support employee got no access to the Tradingsystem:cashdeskline, because for his tasks s/he does not need to. Also no access to system data as every role. At last no access to the customer related data in the PickupShop to verify , for example, the credit card details with the bank. We omitted the Tradingsystem:cashdeskline for space purposes

ACM	Web- frontend	TS: inventor:	TS:cash- deskline	Pickup Shop	Web- service
support employee	user : 2 customer : 1 product & : 2 sales system : 4	user : 2 customer : 1 product & : 2 sales system : 4	user : 4 customer : 4 product & : 4 sales system : 4	user : 4 customer : 2 product & : 4 sales system : 4	user : 2 customer : 1 product & : 2 sales system : 4

Figure 5.6.: Access right for the role *Support employee*. 1 refers to the access right *FullAccess*, 2 refers to the access right *AccessToUsedData*, 3 refers to the access right *AccessToOwnedData*, 4 refers to the access right *Default*.

5.2.2.2. Description of use case 14

- *Brief Description* The system provides the possibility to generate a report for the current status of a customer's order.
- *involved actors* customer, support employee
- *Precondition* the support employee is authenticated.
- *Trigger* the customer submits a ticket for a certain order.
- *Post condition* The report for the order was generated and is displayed to the support employee.
- *Standard process*
  1. The support employee enters the customers identifier and the report identifier to create the report.
  2. The report is generated and displayed
- Alternative or exceptional process
  - in step 2: the order doesn't exist
    The system sends an error message to the customer
  - in step 3: order is ready
    the support employee sends a reminder to the customer to pick up the order

5.2.3. CoCoME excerpt

The CoCoME excerpt is the same as in section 5.1. The excerpt is shown in Figure 5.2. The scenario is settled in the same region as the previous one. So the same components are used. This are the Tradingssystem:inventory:application, the Webfrontned, the Tradingsystem:inventory:data and the Webservice.

5.2.4. Data types and access rights

We divided the data in to four classes of data types. In the following listing, for each class the concrete data is summarized.

- user related data
  No user related data is used in the scenario
- customer related data
  The ID of the customer is used in this scenario
- product & sales related data
  Two data types are used in this scenario. First, the ID of the order, second the report type. The report which merges the an order and a customer in a human readable document.

- system data
  The system data used are queries to request data from the database.

The access rights matrix for this scenario is shown in Figure 5.2.2.1. We omitted the Webservice component for the fact that it only transmits the data in this scenario. The matrix for the data processing is shown in Figure 5.4. It is extracted from the corresponding matrix.

5.2.5. Component behavior

The shown excerpt is extended with SEFFs (subsection 2.3.2). For each component a SEFF is defined, which models the observable behavior for this component. In the following the observable behavior of the scenario is described first, then the model extension with the described meta model. Finally, the resulting data flow is displayed.

5.2.5.1. Observable behavior

First the ID of the authenticated customer and the ID of the order are entered in the Webfrontend by the support employee. Then the tuple is transmitted through the Webservice to the Tradingsystem:inventory. The Tradingssystem:inventory consists out of two compoents. First, the Tradingssystem:inventory:application transmits the tuple to Tradingssystem:inventory:data, where the tuple stays for the time being. Then the whole database is selected. In the next step, a query is created by using the customer ID. Then the query is used to select the corresponding customer. Then order ID is used to build another query and select the corresponding order. The order and the customer data is passed to the Tradingsystem:inventory:application, where it is processed to create an report. This report contains meta data in addition to the ordered products. This meta data includes,for example , the day of delivery, the payment method used, etc. Then the report is transmitted via the Webservice component back to the Webfrontend. There the report is shown to the support employee.

5.2.5.2. Data flow definition

Previously, the data was divided into equivalence classes. Now we present the concrete data to be used in this scenario and specify the corresponding classes.

- Webfrontend
  Transmit the (customer ID (customer data), order ID (product& sales related data)) tuple and the resulting report (product& sales related data).
- Tradingsystem:inventory:application
  Transmits the tuple and alters the Customer and Order object to a report and transmits it.
- Tradingsystem:inventory:data
  Transmit the Customer and Order object and takes the tuple to create two queries.

With these informations about the data processing, we are able to build the resulting data flow, which is shown in Figure 5.7.

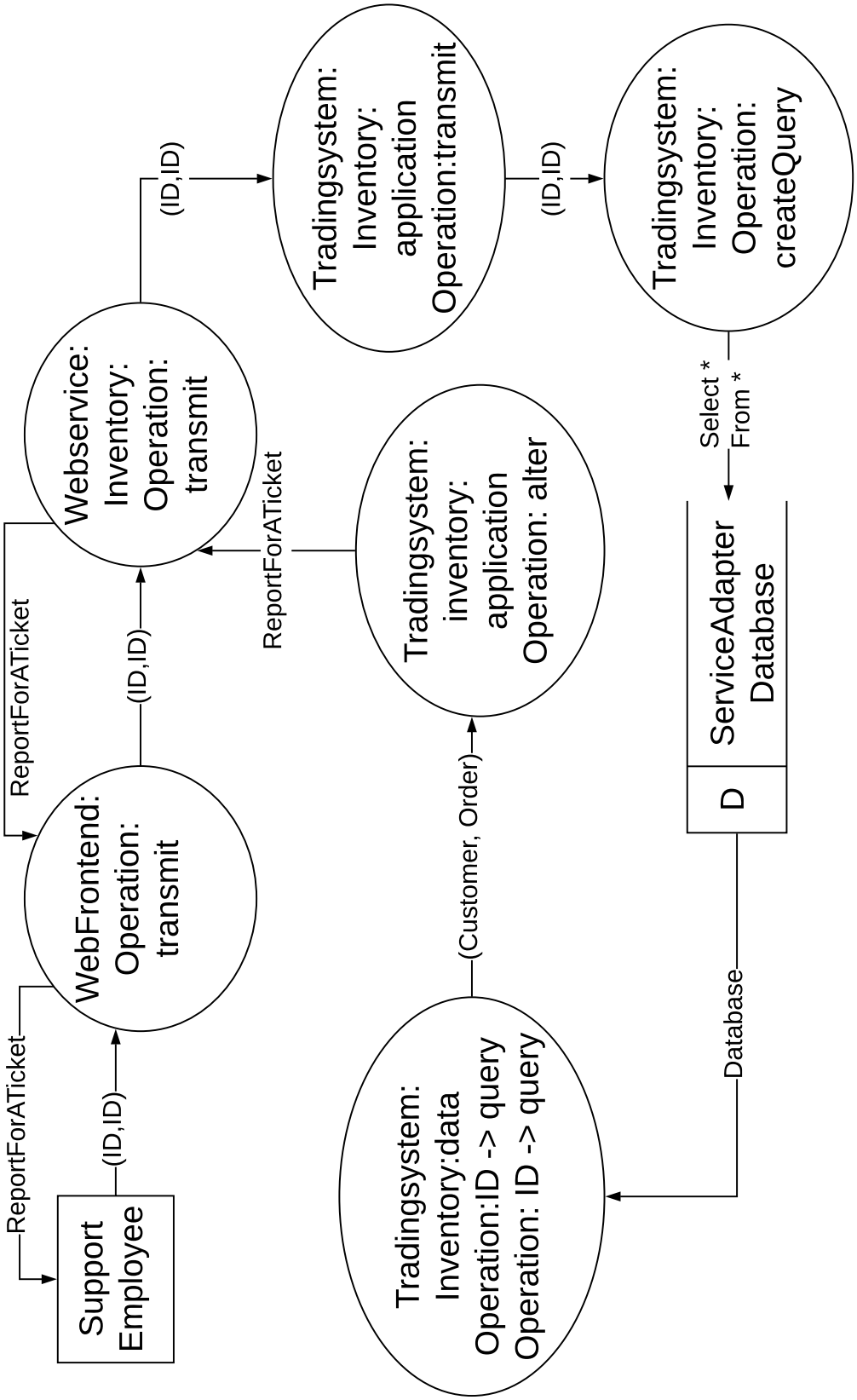

Figure 5.7.: The resulting data flow for the scenario: Support employee requests information for an order.

6. Evaluation

In this chapter, we evaluate the process to create a case on the basis of the application to the CoCoME system. Also, the created case study is evaluated. Then PCM modeling language is evaluated, if it si possible to express the created case study. At last, the results of the evaluation are discussed to conclude this chapter.

6.1. GQM plan

Three aspects of the thesis are evaluated. First, the applicability of the introduced method is evaluated, secondly the usability of the created case study for data based privacy analysis and and last but not least to what extent it is possible to model the created case study with PCM. Each evaluation follow a GQM plan [1], which is shown in Figure 6.1.

6.2. Applicability of the introduced method

The first aspect we want to evaluate is applicability of the introduced method for software systems. If a method is not applicable it is not possible to use it to create other case studies for a data-based privacy analysis.

6.2.1. Is the introduced method applicable to a concrete system

To verify if the goal is achieved, we defined the following question:
Is the method applicable for a system that models a basic shop?
Basic shop in this case means, that products are sold at a cash desk and orders for future pickup may be placed. For example, a warehouse management is missing or a possibility to ship the goods to a customer. As a metric for answering the question, we take the successful application of the method to CoCoME.

6.2.1.1. Application to the CoCoME system

In the chapters Analysis of CoCoME and Case study system, the application of the method to CoCoME is shown. It is possible to create case study for the CoCoME. We followed the steps described in chapter Method and after all the steps were taken we successfully created a case study. The use to carry out data-based data protection analyses is evaluated in section 6.3. Therefore, we concluded that the method is applicable.

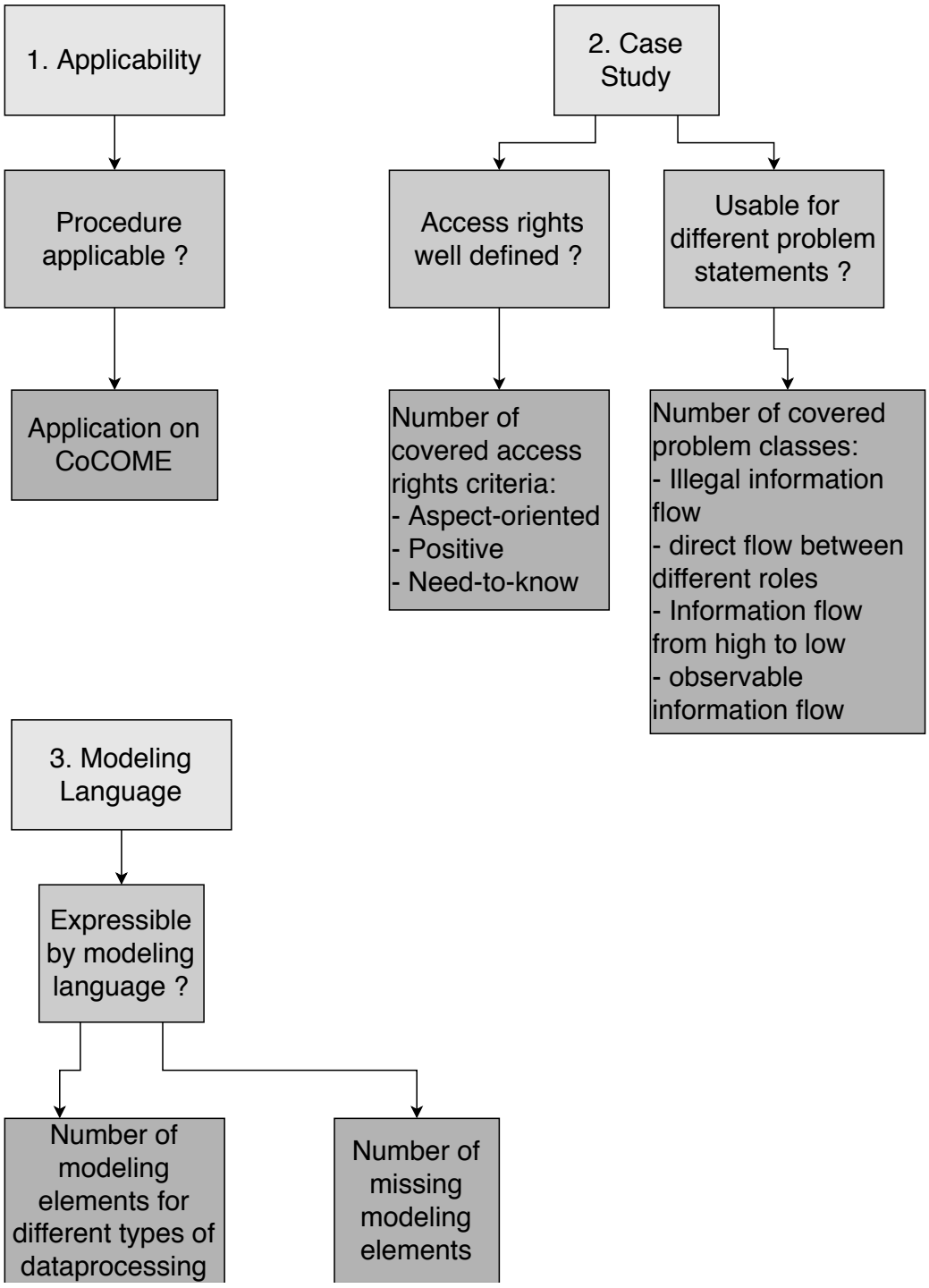

Figure 6.1.: The GQM plan for the evaluation.

6.3. Usability of the created case study for data-based privacy analysis

The next aspect we evaluated the usability of the created case study. A case study is not useful if it does not meet criteria. The criteria ensure that the case study is in a state where it can be used for data-based data privacy analysis. To ensure this, we evaluate two different parts of the case study. First, we evaluate the defined access rights, then the defined data flows.

6.3.1. Are the access rights well defined ?

Well defined access rights means in this context that the defined access rights meet the criteria defined by Everend and Bögeholz[3].The two authors defined seven different criteria. In order to ensure clearly defined access rights, the defined criteria should be met. In our case, it is not possible to evaluate all seven criteria. This seven criteria are : Concise, clear, aspect-oriented, fundamental, positive, need-to-know and efficient. We selected aspect-oriented, positive and need-to-know to evaluate the defined access rights. Clear and concise are omitted, because it was not possible to conduct a survey to verify if the access rights are clear and concise. It was also not possible to check the fundamental criterion, because it needs code to verify the access rights are embedded in it. At last the efficient criterion, there is no running code to measure the overhead for access rights checks. The three chosen criteria are aspect-oriented, positive and need to know. The criterion aspect-oriented describes that the access rights should be separated from the code. This allows a better understanding of the code and more importantly it allows that different code is used in different contexts. The criterion positive describes, that each access to data must be explicitly given to a subject. The default is value is no access. The criterion need-to-know describes the state of access rights where each role has just access to the absolute necessary data that is used to fulfill the tasks of the role.This criterion intends to ensure that access rights are so fine-grained that it does not occur that a role has access to data not required for its tasks. The number of met criteria is the metric to answer the question.

6.3.1.1. Number of satisfied criteria

The evaluation is done in a checklist manner. For each criterion the defined access rights are evaluated whether the fulfill this criterion.

Aspect-oriented The access rights are stored separately from the code and even from the data flows. The access rights are changeable in the context of the case study. This allows that a scenario may be evaluated in different security contexts. So this criterion is met

Positive The created case study achieves this criterion. First, granular access control rights have been defined. The access rights are differentiated in four different levels. We also defined a default value, which represents *no access* to data.

Need-to-know This criterion is achieved by the created case study. We split the data in four classes. For each component it is defined which role has access to which data. With this we can model, for example, that the support employee may access customer data in the Tradingsystem:inventory, but not in Tradingsystem:cashdeskline. The role doesn't need access to customer data in Tradingsystem:cashdeskline for its tasks. So it is possible that each role only get access to the absolutely necessary data. So this criteria is met.

6.3.2. Is the created case Study usable for different classes of information flow?

To verify if a case study system is created that may be used to for data-based privacy analysis, we evaluated as a second aspect, the defined data flows. The following question is defined to indicate whether the goal is reached:
Is the created case study usable for different problem statements?
In the case of this evaluation we refer to the problem statement *Non-influence* [9]. Non-influence combines two problem statements, *Non-interference* and *Non-leakage*. Non-inference describes that no role may acquire informations from a role that inherits a higher security level. Non-leakage describes the issue that it is not observable what specific actions are performed by a system. All in all the problem statements comes down to different classes of information flow. So we use the different information flow classes as a metric to verify the question. This is done in a checklist manner.

6.3.2.1. Number of different covered problem classes

We evaluate the variety of covered information flow classes for the defined scenarios int hte case study .We defined four classes of information flow to cover non-interference and non-leakage. The covered information flow classes are: Illegal information flow, Information flow from higher security levels to lower ones, observable information flow and direct informations flow between roles.

- Illegal information flow
  This class covers information flow of data to roles that are not allowed to get access to this type data.

- Information flow from higher security levels to lower ones
  This information flow describes the existence a sequence of actions, so that is possible for a role that inherit a certain security level to get access to data that are only accessible by higher security levels. The created case study uses data flows as the source for information, so we can use this definition to describe the class of information flow.

- Observable information flow
  This class describes if the data flow are observable from the outside and to determine, for example, if a security relevant operation is performed. The acquired informations may be used for an attack on the system.

- Direct information flow between roles
  This information flow class models data flow between two roles while both interact simultaneously with a system. This refers especially to the security relevant data that may be transferred between two roles with different security levels.

Illegal information flow The first scenario model illegal data flow. The stock manager in this scenario has access to the ID of an customer. the role has no rights to have access customer related data. Then, the stock manager request a report of this specific customer and receives a full report for this customer. The stock manager has the security level*AccessToUSedData*. The report for one customer is no necessary data for the stock manager to perform its tasks. In this scenario an illegal data flow is modeled. Therefore, the illegal data flow class is part of the created case study.

Information flow from higher security levels to lower ones In the case study such information flow is present, because in the scenarios data from different security levels is processed.

Observable information flow We did not change the base system of CoCoME. In the base system of CoCoME it is possible to observe when, for example, users are authenticated, because they are then allowed to perform different actions. So this class is not modeled in a scenario.

Direct information flow between roles For the fact that no scenario with more than one role is defined, this class is also not covered in the case study.

6.4. Expressiveness of data centric PCM

The last evaluation aspect of this thesis is to evaluate the chosen modeling language, in this case data-centric PCM. We want to check to what extent PCM is able to add the access rights and data flows in the already existing system model. To allow the extension, Seifermann [14] provided a meta model extension. This meta model extension is the central point of the evaluation. Without the meta model extension, PCM is not able to store data flows and access control rights directly in a system model.

6.4.1. Is the created case study expressible with PCM ?

A as metric to verify if the, we use the created case study and check if all operations for the different types of data are expressible by PCM. Also we check if elements for types of data processing are missing. The metric is done in a checklist manner.

6.4.1.1. Number of available elements to model the different types of data processing

First we measure in what current state the operations of the meta model extension are in. We identified five types of data processing that are needed to express the data flows.

Currently twelve operations are available in the meta model extension. We measure which operation express the which type of data processing.

Operations relational algebra This type of data processing describes the manipulation of database or data requested from databases. The operations for this are available in the meta model.

I/O- operations This type of data processing describes an I/O operation in the data flow, where a user receives data and inputs the same or an other data type in the system. An operation for this is available.

Transmission data This data type describes if component transmit types of data. An operation for this available in the meta model extension.

Change Access rights In the course of a data flow it may happen, that the access right level of a data type changes. For this case an operation in the meta model extension is available. This mostly happens when an operation changes the type of data. This type of processing is closely related to the next one described.

Alternation of data This is a larger type of operations. All in all, this type describes the alternation of data. This includes creation of data, merging many points in , for example, into list or set. Also the splitting data, like lists, in the different data points. This type of data processing describes the contrasting operation to merging data in collections.

6.4.2. Number of missing elements for the different

An element for modeling the ACM in the system model is missing.

6.5. Discussion

After all three parts of the evaluation are now done, it is time to discuss the findings. For the last two parts an overview is shown which pats of the checklist are met, then the whole result is put into context to our contributions.
First, the evaluation of the applicability. We showed in the previous chapters ,that our method is applicable on a concrete system and it is possible to create a case study for data-based privacy analysis. After that we evaluated the two main aspects of the case study. First, the defined access rights in the case study. We evaluated based on predefined criteria by Everend and Bögeholz [3] the access rights. Not all criteria were applicable to the case study, because for some criteria one need running code which is not in the scope of the case study. Other criteria weren't applicable due to time constraints. All in all, as shown in Figure 6.2, we achieved with our definition of the access rights all three criteria. Ther eis surely some work to be done to allow to check more criteria, like conducting a survey for the criteria clear and concise. For the criteria fundamental and

Access Rights	fulfilled ?
Aspect-oriented	yes
Positive	yes
Need-to-know	yes

Figure 6.2.: Overview of the evaluation result for the access rights.

Data flow	fulfilled
Illegal information flow	yes
Information flow from high to low	yes
Observable information flow	no
Direct information flow between roles	no

Figure 6.3.: Overview over the evaluation result for the data flows.

efficient a deployment of CoCoME which includes the access rights is needed. The second part of the evaluation were the defined data flows. These data flows are later used to conduct a analysis of privacy shortcomings on an architectural level. We defined four information flow classes and evaluated if they are covered by the data flows in the system. The result is shown in Figure 6.3. Since we only cover one information flow class, we can say that the case study is just a proof of concept. We defined too few scenarios to cover more than that one class of information flow. At last, we evaluated PCM with the meta model extension. We defined five types of data processing and evaluated whether they are possible to model in PCM. The result is shown in Figure 6.4. As shown, all types of data processing are modeled. Therefore, we conclude it is be possible to model all created case studies with PCM. But it is not possible to store the ACM in the same model. To conclude the evaluation, one can say that we introduced an applicable method to create data flows for a data-based privacy analysis. For the sample case study, we say it is a proof-of-concept, because only one class of information is modeled. The defined access rights needs a bit more tuning then they fulfill the criteria. At last, the used modeling language is able to model all types of data processing and is therefore usable for data-based privacy analysis. The only throwback for the modeling language is that the ACM must be stored separately, this may lead to inconsistency and should be fixed soon.

Meta model	possible ?
relational algebra	yes
I/O operations	yes
Transmission of data	yes
Change of access rights	yes
Alternation of data	yes
ACM in system model	no

Figure 6.4.: Overview over the results for the evaluation of PCM.

7. Related work

After the contributions of the thesis have been evaluated, they are placed in the context of related works. First, it has to be said that related work for the kind of case study we created is hard to find. Case studies are an acknowledged procedure mainly in the fields of sociology, philosophy, law, etc. For this fields the case study method is documented well. The type of case study that we are going to create shows some similarities with case studies that were created about managers, for example. Nevertheless, the discrepancy between computer science and classical fields is too great to allow only fundamental methods to be adopted. Such methods includes, for example, to use two sources of data. In the field of computer science, case studies are mostly used to give an overview of certain topics, e.g. how different challenges in software engineering were tackled. Sometimes there are used to verify the applicability of different approaches, like UMLSec [6]. The only publication that roughly outlines the creation a case study for a software system, was the one from Everand and Bögeholz [3]. In this publication, the both authors describe briefly the creation of a case study on a much smaller system than we performed our case study on. We have adopted the criteria of Everend and Bögeholz [3] and have included them in our evaluation as a measurement for good access rights. Further, the author described briefly the creation of a case study. In their publication, the defined first the data than the access rights. In their work this was enough because the main aim of their work was to evaluate the different types how access rights may be added to a deployable system. In contrast to Everand and Bögeholz, the aim for our case study was to be used in a data-based privacy analysis. Therefore we used their approach as a basis for ours. We require a specific system model and the existence or the deductibility of use cases to ease up the later validation. Further, in contrast to Everand and Bögeholz, we want to use the created case study in another setting. Therefore we added the definition of scenarios and the definition of the type of data processing for each component and each role. All in all, we say that Everand and Bögeholz have given us a good starting point for our method and we have expanded their.
We could not find any other fitting publications that may address our matter. We could not find any other publications with the same or similar aim as our method.

8. Conclusion and future work

In this chapter, we first draw a conclusion for the thesis. Then, we discuss in which cases our work can be used and lastly, we discuss the future work for each aspect of the evaluation.

8.1. Conclusion

In this thesis, we presented a method to create case studies for a data-based privacy analyses. To allow such analyses, we introduced data flows that are added to the system model. Further, we created a case study by applying the method to CoCoME. While in the process of application to CoCoME, we found vagueness in CoCoME. First of all, the roles were rather well defined, but missed the last bit of precision. Also some used data in CoCoME were missing. After we added the definitions, data flows that describe the data processing were created. Further, we define a new use case (UC14) for CoCoME. The UC14 introduces a new role, the support employee. When a user have problems with an order they can issue a ticket. The support employee handles the tickets to solve the problem we then validated the method described. We have found that the method is applicable to CoCoME, but still needs to be applied to other component-based systems. section 8.3 goes into more detail. Then we evaluated the created case study based on two aspects. First, the quality of the defined access rights, secondly, the number of different information flow classes covered. The created case study covers 50% of the defined information flow classes. At last we evaluated the used modeling language PCM [12]. We used a meta model extension to add the data flows to the system model. The meta model extension allows the modeling of data flows, but misses an element to store the ACM in the same model.

8.2. Usage for our work

The main goal of the thesis was to introduce a method to create case studies that may be used to perform privacy analyses on an architectural level. More specific, we aimed to create case studies that can be used by data-based privacy analyses for validation. The created sample case study is a ready-to-go case study on which an data-based privacy analyses approach may be tested. Further, we provided evaluation criteria to measure the quality of the created case study. The main benefit of this work is to have a process for transforming software systems to perform a data-based data privacy analysis.

8.3. Future work

This section presents possible approaches for future work. The future work will be divided by the three main aspects of the thesis: the method for creating a case study, the resulting case study and its evaluation, and the extension of the meta model.

8.3.1. Method

In the case of the method, we validated the applicability for CoCoME. The next logical step is to apply the described method to other systems to verify the applicability. CoCoME sells products, either via the PickupShop or a cash desk. Possible system to apply the method are systems which also handles a warehouse and/or ships the product to the customer. It would also be conceivable to apply the method to systems away from the supermarket scenario, for example, a flight booking system.

8.3.2. Case study system

For the case study, we decided to divide the future work in a short term and a long term work.
As short term work, one should define more scenarios. First, to cover all defined information flow classes. Also for each information flow classes, one should define more scenarios to allow a more deep analyses. Another possible approach for the future work is the analysis of access rights according to various criteria not yet covered, which we could not carry out due to different restrictions.
As a long-term work, we propose to define more information classes in order to improve the evaluation. Another possible approach is the evaluation of the CoCoME system with a data-based approach privacy analysis using the exemplary case study or an extended version thereof.

8.3.3. Meta model extension

As future work, we propose to extend the metamodel to allow the storage of the ACM in the same model.

Bibliography

[1] Victor R. Basili and David M. Weiss. "A Methodology for Collecting Valid Software Engineering Data". In: *IEEE Trans. Software Eng.* 10.6 (1984), pp. 728–738. DOI: 10.1109/TSE.1984.5010301. URL: https://doi.org/10.1109/TSE.1984.5010301.

[2] Jakub Breier. "Asset Valuation Method for Dependent Entities". In: *J. Internet Serv. Inf. Secur.* 4.3 (2014), pp. 72–81. URL: http://isyou.info/jisis/vol4/no3/jisis-2014-vol4-no3-05.pdf.

[3] Mark Evered and Serge Bögeholz. "A Case Study in Access Control Requirements for a Health Information System". In: *ACSW Frontiers 2004, 2004 ACSW Workshops - the Australasian Information Security Workshop (AISW2004), the Australasian Workshop on Data Mining and Web Intelligence (DMWI2004), and the Australasian Workshop on Software Internationalisation (AWSI2004) . Dunedin, New Zealand, January 2004.* 2004, pp. 53–61. URL: http://crpit.com/confpapers/CRPITV32Evered.pdf.

[4] Ursula Goltz et al. "Design for future: managed software evolution". In: *Computer Science - R&D* 30.3-4 (2015), pp. 321–331. DOI: 10.1007/s00450-014-0273-9. URL: https://doi.org/10.1007/s00450-014-0273-9.

[5] Robert Heinrich, Kiana Rostami, and Ralf Reussner. "The CoCoME Platform for Collaborative Empirical Research on Information System Evolution". In: (2016).

[6] Jan Jürjens. "Model-based Security Testing Using UMLsec: A Case Study". In: *Electr. Notes Theor. Comput. Sci.* 220.1 (2008), pp. 93–104. DOI: 10.1016/j.entcs.2008.11.008. URL: https://doi.org/10.1016/j.entcs.2008.11.008.

[7] Jan Jürjens. "UMLsec: Extending UML for Secure Systems Development". In: *UML 2002 - The Unified Modeling Language, 5th International Conference, Dresden, Germany, September 30 - October 4, 2002, Proceedings.* 2002, pp. 412–425. DOI: 10.1007/3-540-45800-X_32. URL: https://doi.org/10.1007/3-540-45800-X_32.

[8] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. "SecureUML: A UML-Based Modeling Language for Model-Driven Security". In: *UML 2002 - The Unified Modeling Language, 5th International Conference, Dresden, Germany, September 30 - October 4, 2002, Proceedings.* 2002, pp. 426–441. DOI: 10.1007/3-540-45800-X_33. URL: https://doi.org/10.1007/3-540-45800-X_33.

[9] David von Oheimb. "Information Flow Control Revisited: Noninfluence = Noninterference + Nonleakage". In: *Computer Security - ESORICS 2004, 9th European Symposium on Research Computer Security, Sophia Antipolis, France, September 13-15, 2004, Proceedings.* 2004, pp. 225–243. DOI: 10.1007/978-3-540-30108-0_14. URL: https://doi.org/10.1007/978-3-540-30108-0%5C_14.

[10] Roman Pilipchuk, Stephan Seifermann, and Emre Taspolatoglu. "Defining a Security-Oriented Evolution Scenario for the CoCoME Case Study". In: *4nd Collaborative Workshop on Evolution and Maintenance of Long-Living Software Systems (EMLS'17)*. Vol. 37. Softwaretechnik Trends 2. 2017, pp. 60–77.

[11] Andreas Rausch et al., eds. *The Common Component Modeling Example: Comparing Software Component Models [result from the Dagstuhl research seminar for CoCoME, August 1-3, 2007]*. Vol. 5153. Lecture Notes in Computer Science. Springer, 2008, pp. 16–53. ISBN: 978-3-540-85288-9. DOI: 10.1007/978-3-540-85289-6. URL: https://doi.org/10.1007/978-3-540-85289-6.

[12] Ralf Reussner et al. *The Palladio Component Model*. Tech. rep. 14. 2011. 193 pp.

[13] Stephan Seifermann. "Architectural Data Flow Analysis". In: *13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, Venice, Italy, April 5-8, 2016*. 2016, pp. 270–271. DOI: 10.1109/WICSA.2016.49. URL: https://doi.org/10.1109/WICSA.2016.49.

[14] Stephan Seifermann. *PCM-DataProcessing-Metamodel*. https://github.com/seiferma/PCM-DataProcessing-MetaModel. 2018.

A. Appendix

A.1. First Appendix Section