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 – 14. 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 connected and complex, security in general and privacy particularly become more important. First of all, the financial loss through possible leaks are immense. Also the betrayal of the customer might the greater issue. All in all, privacy is 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, Seifermann [11] published the idea of an approach, how privacy can be reviewed. The motivation for Seifermann's approach is to ensure compliance already in the design process. To ensure this compliance, the approach introduced a data based security analysis on an architectural level.
It was not the first approach in this direction. The to other approaches that should be mentioned are UMLSec [6] and SecureUML [7]. All three approaches are a lightweight extension of an architecture description language. The main problem with the other approaches, is that security relevant model elements are not stored in the system model. Changes in the model may not be documented in the other models which leads to inconsistency.
Seifermann's approach extends the meta model of Palladio Component Model (PCM) instead of UML. Seifermann's approach introduce data based privacy analysis on an architectural level. The approach is evaluated using a case study. Currently, the approach has not yet been applied to a realistic system. It is not that simple to find viable case study systems to evaluate the approach. The reason is that a lot exemplary systems are often designed for a specific approach. This thesis presents a method for creating a case study system from for reviewing data-based privacy analysis. The method will be applied on the Common Component Modeling Example (CoCoME) to create a suitable case study to perform a data-based privacy analysis. At the end, the case study will be evaluated to determine if it is suitable for data-based privacy analysis. As already mentioned, the meta model of PCM has been extended which will also be part of the evaluation. To evaluate my contributions, we will use the Goal-Question-Metric (GQM) approach [1]. We will show that our procedure is applicable for a large system. Also we show that the created case study is usable to check that no illegal information flow is happening. The last result will be that the chosen modeling language is able to express data flows in the component model of the surveyed system.

1.2. Contributions

The contribution of the thesis is to present a method to create usable case studies that may be used to evaluate data-based privacy on an architectural. The mentioned approach from Seifermann [11] may use the resulting study for its evaluation. Further, the thesis introduces a finer grained, high level form of access rights. This form was motivated by Everend and Bögeholz [3]. Another contribution is a sample case study for the CoCoME system. A positive side effect is that we defined some incomplete elements in CoCoME.

1.3. Outline of the thesis

The thesis is organized as follow. The first chapter defines the foundations that are needed for the thesis. The second chapter presents the procedure to create a case study that can be used for data-based privacy analysis. The third and fourth chapter applies the procedure to CoCoME and create a case study system that can be used for data-based security analysis. In the fifth chapter evaluates the created case study. In the sixth chapter, we present related work that is similar to our contributions. The thesis closes with a conclusion of the contributions and an outlook for 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.

2.1.2. Data flow

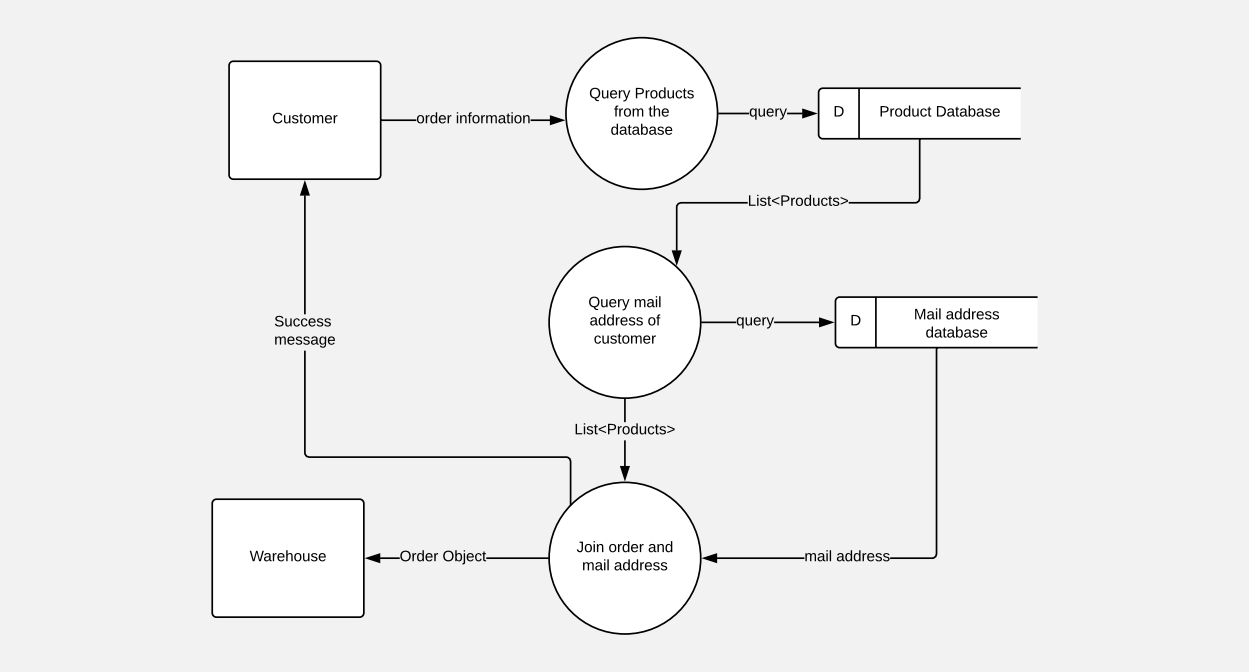

Figure 2.1.: Simple data flow diagram for linking an order of different products to a mail address

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 ur 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

The Palladio Component Model (PCM) is an ADL, which models component based systems

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. 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.

2.3. Data Centric PCM

2.3.1. Concept

Component based systems are software systems which consist of different components. A component is defined as unit in the system which provide and/or require interfaces. Each component in systems is designed for a specific task, like serialization, connection to a database, handling the user input and many more. These different component communicates via interfaces. Also a component can consist of several smaller components. A simple example is shown Component based systems are software systems which consist of different components. A component is defined as unit in the system which provide and/or require interfaces. Each component in systems is designed for a specific task, like serialization, connection to a database, handling the user input and many more. These

different component communicates via interfaces. Also a component can consist of several smaller components. A simple example is shown The Palladio Component Model (PCM) is an architecture description language (ADL) , that allows more than only modeling a system (. The resulting 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.

After all the different models are created, a prediction may be computed and without a line of code is written the architects may 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 [12]. 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 my advisor Seifermann created a meta model extension and provided it to me (it can be found here). 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. 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. It uses data containers to store information over a specific action in the SEFFs. Inside the data container one can specify different operations to model the data processing. The meta model has different operations that can be specified to model manipulation of data. It exists store, load, selection and projection operations mainly designed for databases but may be used in different contexts as well. Also operation for data creation, data transmission, union of data and resulting data are also included. These are used when in the program flow data is manipulated. There are also a third category of operations that describe mostly meta informations like characteristics for different data types.
The meta model allows to attach data processing directly to the components in the repository 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

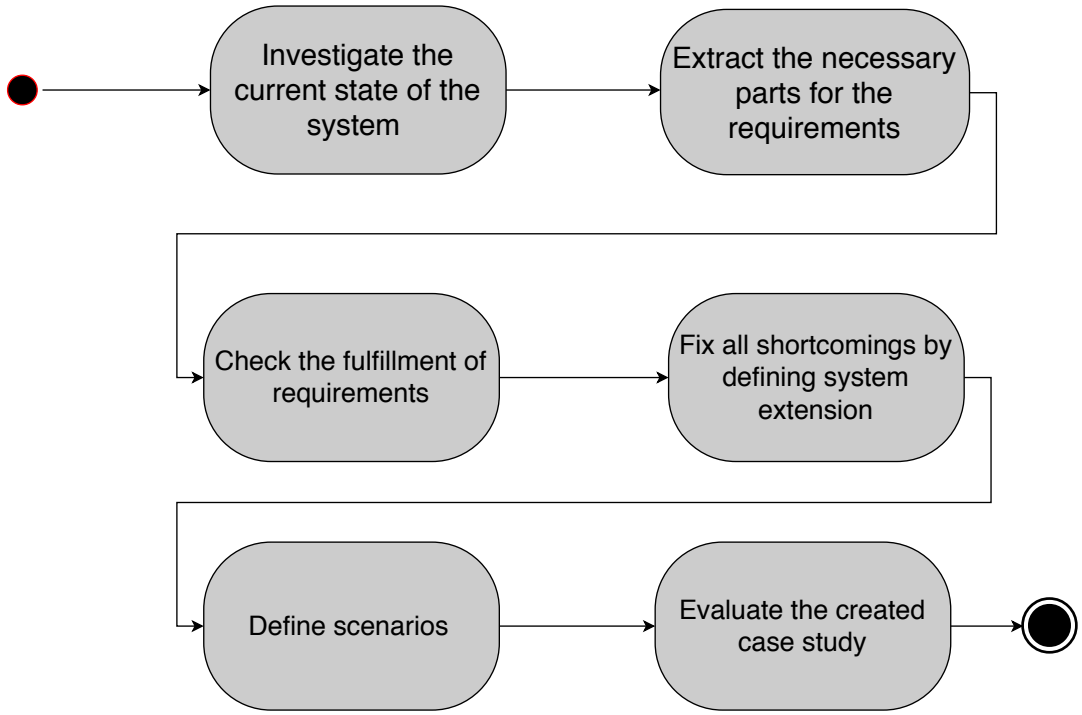

Figure 3.1.: General overview over the procedure for creating a viable case study

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 [9] 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

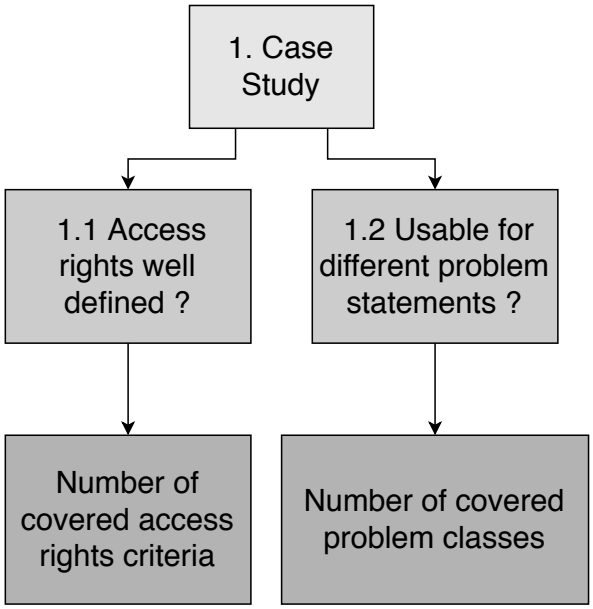

Figure 3.4.: The GQM plan for the evaluation.

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 section 6.1

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 problem statements ?

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 all seven criteria. 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

In the next two chapters, we will apply the procedure described in chapter 3 to a concrete system. The system we chose is CoCoME. The procedure described in chapter 3 is applied over two chapters . A brief overview over the procedure are shown in chapter 4. This section only performs the first three steps. We analyze the system first if all requirements are met and after that all shortcomings are fixed, then we create a viable case study in the next chapter.
First, we give a brief overview of CoCoME and then examine the current state of the requirements for creating a viable case study. Then, components are defined in which security-relevant data is processed. After that, the fulfillment of the requirements are checked and the chapter is concluded with a summary of all shortcomings.

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 first introduced as 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:

requirements	R1-R5
security relevant components	S1-S3
analyze current state	
fix shortcomings	
Define scenarios	

Figure 4.1.: General procedure for creating a case study as described in chapter 3

Plain Java Variant
Service oriented variant
Hybrid cloud based variant

Figure 4.2.: short overview of all CoCoME variants

- 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. The three milestone variants that emerged are shown in section 4.1. 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 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 4.3. 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

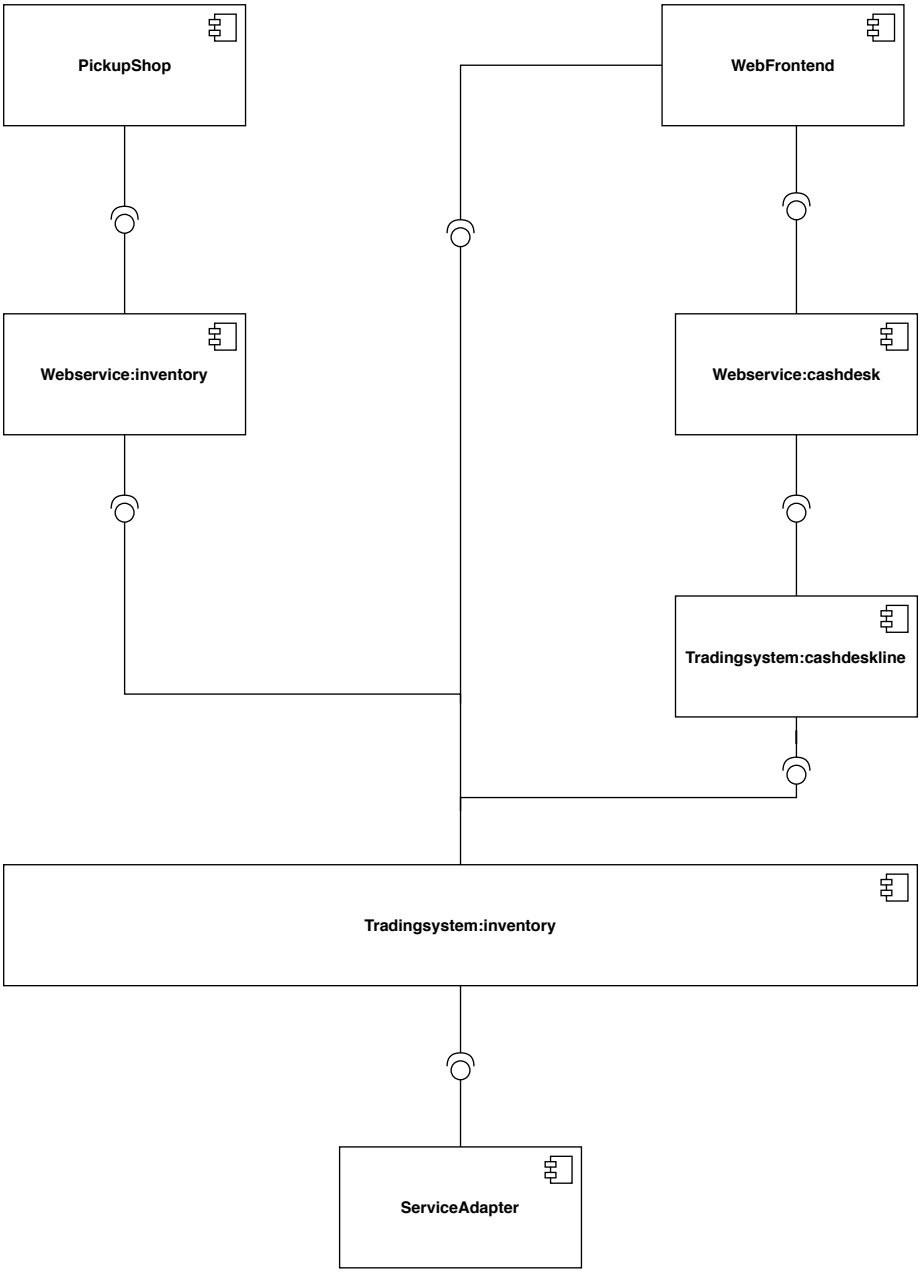

Figure 4.3.: The figure shows a simplified overview over the hybrid cloud based variant

requirements	R1-R5
security relevant components	S1-S3
check current state of the case study	

Figure 4.4.: Current steps performed from the procedure described in chapter 3

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 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

After a brief overview of CoCoME was given in section 4.1, we now start to apply the method described in chapter 3 to CoCoME. In this section, we will check the requirements and define the components that are processing critical data. section 4.2 shows the three steps that are performed in this section.

4.2.1. Requirements for a case study in CoCoME

First, we check the needed requirements for the creation of a case study. The requirements are listed here for a better overview :

1. **R0: State of the system**
   Analyze that the elements of the system are well modeled or that, at first glance, there are enough elements to predict whether it pays to start building the case study.

2. **R1: component based system**
   Make sure the studied system is represented in a component model.

3. **R2: definition of use cases**
   Make sure all uses cases are well defined.

4. **R3: define (security relevant) data**
   Define all the (security relevant) data in the system. It is important for this require-

ment to accurately define all security-related data and to define the remaining data as sufficient.

5. **R4: define the roles in the system**
   Define all the roles in the system. It is important to assure, that all roles are well defined.

6. **R5: define access rights**
   Define all the necessary access rights between the roles and the security relevant. The other data appears only if it necessary

In the following each CoCoME willbe checked for each requirement. For the requirements, we analyzed the CoCoME tech report [5] as well as the code

4.2.1.1. R0: System modeled well enough to apply

For the fact that CoCoME is described in a tech report, where the system is described in detail and there is running code for the latest version of CoCoME, we decided to give it a shot and created a case study.

4.2.1.2. R1: component based system

This requirement is fairly easy. CoCoME is already modeled as an component based system, because the model is a PCM model. This ADL was described in section 2.2.

4.2.1.3. R2: use cases

CoCoME switched from a closed system to an open system when the pickup shop was added. For this reason, only the application cases added are relevant for the case study. Five new use cases were defined in the extension of the pickup shop.

- UC 9: Process online sale
  This use case describes how an sale is conducted. The actor in this use case is the customer.

- UC 10: Manage Product information
  In this use case there are one actor, the stock manager. S/he alters the information of an product in the database.

- UC 11: Create Customer
  The creation of an user account for the Pickup shop is described, The only actor is the customer

- UC 12: Authenticate user
  This use case describes the login process for an user account to the Pickup shop. The only actor is the user. Note that , the use rand the customer are two different actors.

- UC 13: View customer report
  The stock manager requests a full report of the purchases of a customer.

4.2.1.4. R3: security relevant data

Since the amount of data is relatively large, we have grouped the data into different equivalence classes. The data in each class has the same security level. We defined four equivalence classes. The results are shown the the listing below.

- 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, we classified it as **security relevant**

- product and sales related data
  All the data that is related to the products and the sales. Examples for data types are price of a product, quantity of a product and so on. These data are only relevant for security if traceability to the customer is possible, that's why we defined them as not strictly security relevant

- user related data
  This data refers to the account and credentials of user in the system. For the reason the user related data is linked to the customer and may be used to jeopardize the customers security relevant data, it is also classified as not strictly security relevant.

- the query data type
  he query class is a special one. It abstracts all possible query objects that may be executed in CoCoME. For the reason, that this data type issues queries to the database and there could be al ot of harm to an enterprise, we classified the query also as **security relevant**.

4.2.1.5. 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*. Note that,there is no separate chapter in which the roles are described. All the responsibilities are derived from the use cases described in the document and the written code.

- The *Customer* role is reasonably well defined, because this type of role is well known. As you can see from the code, the customer is the role of all the customers of a company, be they private customers or business customers. The role buys products in the stores or pick them up at a Pickup shop.

- The *Cashier* role is also reasonably well defined. As seen in the Code, the role sells the products to the customer role in the different stores of an enterprise.

- The *StoreManager* role is in one aspect well defined, that it handles matters of a single store. It is not clear, which precise responsibility this role has. It may possible, that the role has managing tasks in the store and also handles stock, but this is not made clear in the documentation.

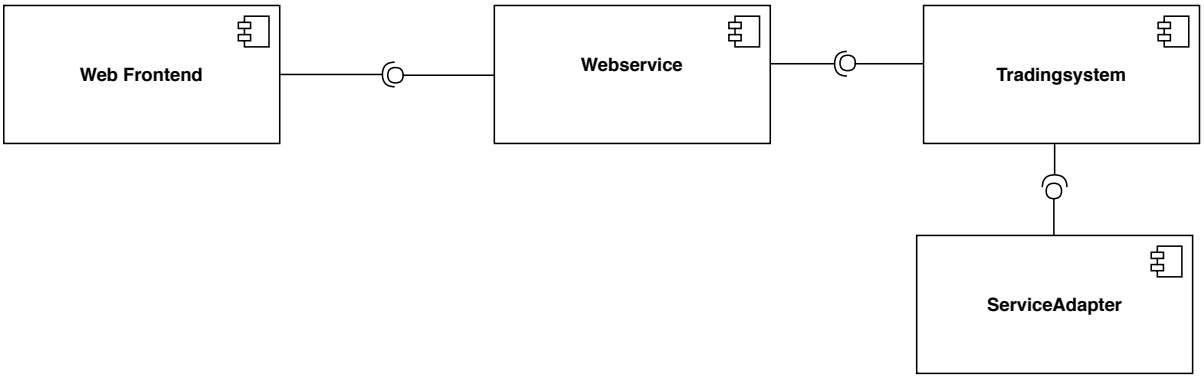

Figure 4.5.: Highly simplified view on the CoCoME system. One can see there are three different layer

- The *EnterpriseManager* role is in one aspect well defined, that it handles the matters of a whole enterprise. Which precise tasks the role handles, is not made clear, whether in the code nor in the CoCoME tech report .
- The *Stockmanager* role is not clear defined. Neither the code nor the CoCoME Technology Report indicate whether the role is for the stock of an entire enterprise or for the stock of a single store responsible.

4.2.1.6. R5: access rights

Since it is the case that no access rights are defined in the documentation, we used R2 to R4 to derived access rights from roles and their specific responsibilities as well as the security-relevant data. Another important factor for the definition of access rights is the component model of CoCoME. In subsubsection 4.2.1.6, the three layers are shown. The service adapter and the tradingssystem represent the system layer, where the business logic is. So we defined the data of this layer as *system protected*. We also defined for the two top layer, Webfrontend and Webservice, two additonal access rights. All the access rights can be seen in the listing below.

- *Access right: allowed*
  This access right is pretty straight forward. It defines if the role is allowed to access the respective data.
- *Access right: denied*
  This access right defines whether the role is denied access to the appropriate data.
- *Access right : system protected*
  This access right emerged from the component model. As described in section 4.1 the systems layout is in three layers. The top layer is the layer for the users, the bottom layer is pure for the system. Therefore, all processed data in this lowest layer should be accessible only by the system.

ACM	customer	sales and products	user	query
customer	allowed	allowed*	allowed*	system protected
cashier	allowed*	allowed	denied	system protected
stockmanager	allowed*	allowed*	allowed*	system protected
enterprisemanager	denied	allowed	denied	system protected
storemanager	denied	allowed*	denied	system protected

Figure 4.6.: The ACM at the current state of the case study

After the access rights are defined, the requirements R2 - R5 are used to build an ACM that stores the access right in one model element. The created ACM is shown in Figure 4.2.1.6. Note that, we defined the 'allowed' and 'denied' on an enterprise level. To clarify the 'allowed*' entries in the ACM. This entries signalizes that a role has partly access to the respective data type.

4.2.2. Identification of security relevant components in CoCoME

The identification of security relevant components divides in two different aspects S1 and S2. For a better overview the aspects are listed below:

- S1: Identify security relevant data in the components
  The first step is to identify in which component(s) security relevant data is present.
- S2: Identify the type of processing
  For each considered component, the type or types of data processign is defined. This means, one have to define which operations are used. For each operation the input data type and the resulting data type is defined.

Relevant components for the case study can mainly be found in the newly added components when the PickupShop was added to CoCoME. Another important component is the Webfrontend. Due to the fact, that the customer interacts with CoCoME via the PickupShop component, the other roles still interact with CoCoME via the Webfrontend component. In subsection 4.2.2 an excerpt of the system is shown. This excerpt shows how the PickupShop is integrated in the system. As a reminder, the pickup shop was designed to sell products without using the cashdesks. Therefore, the PickupShop component is connected to the inventory managing component, namely *Tradingssystem:inventory*. Another reason, we narrowed the relevant compoents down to shown excerpt (subsection 4.2.2) is, that in the variant of CoCoME used, the personal data are stored for the first time as part of evolution for CoCoME. The external bank component is present, because it is allowed for customemrs to pay their products via credit card.

4.2.2.1. S1: identify security relevant data in single components

In this section, we are going to define the components in which security relevant data is available. We identified the PickupShop as a relevant component, because the customers interacts with it and insert his/her personal data. After we analyzed the access rights,

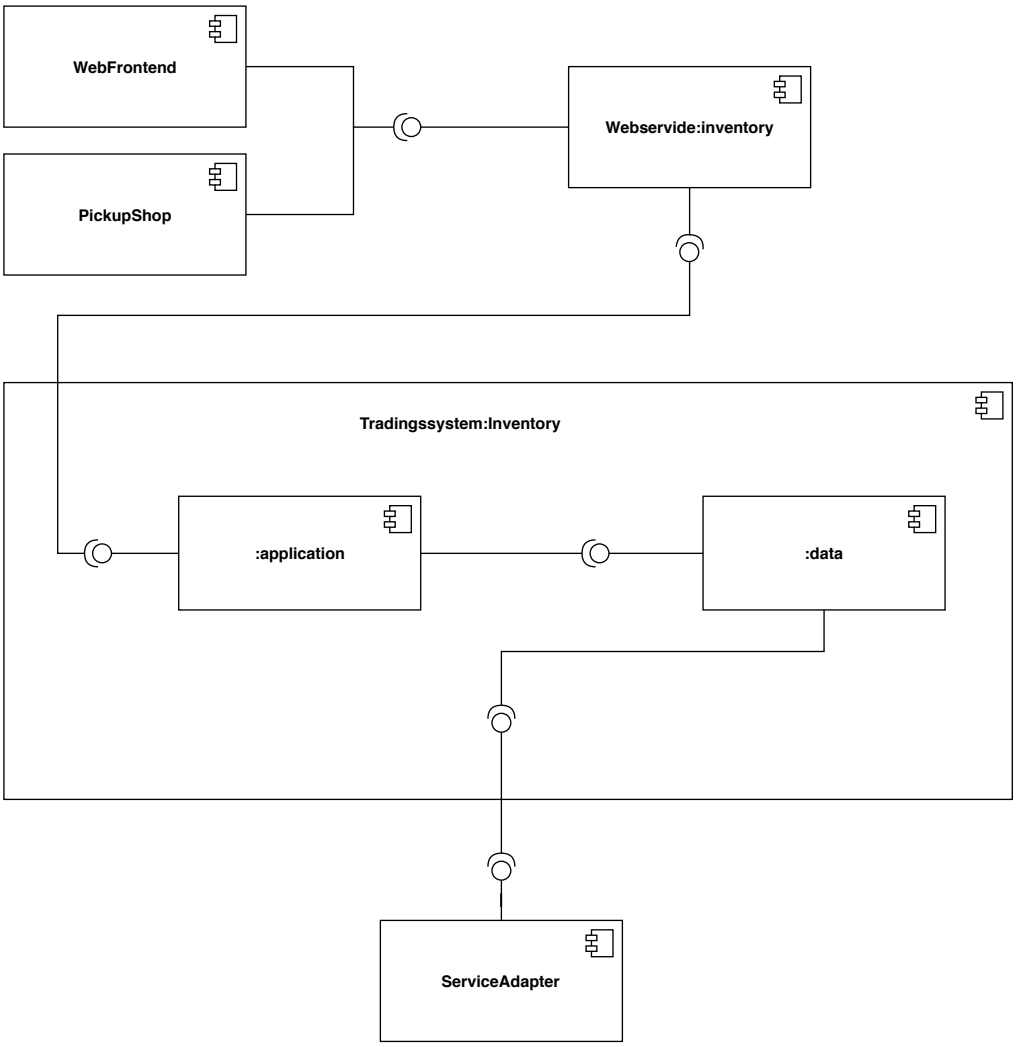

Figure 4.7.: This figure shows the excerpt where the relevant components are

Data processing	customer	user	sales and product	query
Webfrontend	transmit	transmit	transmit	non-existent
PickupShop	transmit	transmit	process/transmit	non-existent
Tradingsystem:inventory:application	process/transmit	transmit	process	non-existent
Tradingsystem:inventory:data	process	process	process	process

Figure 4.8.: a graphical representation of the processed data in th system.

we mentioned the stock manager is able to access the ID of a customer. S/he interacts with CoCoME through the Webfrontend component. So the Webfrontend component is also a relevant component. After we further analyzed the security relevant data, the query type is present in the Tradingsystem:inventory:data, which we identified as another relevant component. The report data type which is a part of the product and sales class, is used by the Tradingsystem:inventory:application component. A full list of the relevant components is shown below:

- Webfrontend
- PickupShop
- Tradingsystem:inventory:*
  This component includes the :application and :data components.

The list contains all components, except the Webservice component. We found that the Webbservice component in this scenario just transmit the data, so we omitted it.

4.2.2.2. S2: Define the type of data processing

we defined two types of operations that may process data in the system. First, the *transmit data operation*, which transmits the data through the component without any changes. Secondly, *the process data operation*, which can alter the data in a lot of ways. This alteration includes create new data from the given one, join the data with other data, and many more.
In Figure 4.8, it shows which component how which data is processed.

4.2.3. Analysis of the current state for CoCoME

With everything in the lead, it is time to step back and analyze the shortcomings of a viable case study. First a short disclaimer for CoCoME.

- **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.

Before the introduction of the hybrid cloud based variant, CoCoME was a closed system with no user management. After the inclusion of the hybrid cloud based variant in the system, CoCoME is converted to an open system. Another fact is that the Pickup shop was designed to introduce another point of sale to CoCoME. It also added user accounts to CoCoME. So the Pick up shop addition is the only relevant component for a security analysis
This means for the case study that we could only draw from a limited pool of model elements. Another factor was that the variant was quite new and not yet well studied.
To come to a brief analysis of the case study. All the shortcomings are explained in detail in section 4.3.

- The modeling as a component based system is accomplished.
- In CoCoMe there are many data that are not well defined. For example, for the report type (sales and product related data) the definition is unclear. For some functionalities there is no data defined at all.
- Various roles are mentioned, but there is no clear definition of their responsibilities.
- In the entire documentation of CoCoME there is no clear definition of the access rights to certain data in the system. We tried our best to derive these access rights from the use cases and the written code
- Since all components in the excerpt are already regarded as security relevant, there are no changes here. For the

All in all, CoCoME is not yet ready to create a case study from the system. A lot of work is needed to ready for the system for the creation a case study.

4.3. Summary of shortcoming in CoCoME

4.3.1. use cases

4.3.2. Security relevant data

As previously said, definition of data is 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 use case 13 (UC13) described in the CoCoME tech report [5]. A case of not well defined data, 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 also some shortcomings with the defined roles, especially with the roles of the enterprise manager, stock manager and store manager. The responsibilities of the enterprise manager are not well defined, also nearly no responsibilities may be derived by the use case diagram. For the role of the stock manager it is not clear if the role is

responsible for a whole enterprise or only for a store. for the fact that this role also effects the role of the store manager, it needs to be defined clearly. The store manager also misses a clear definition, because of the stock manager. There are two different cases. In the case the stockmanager is responsible for a store. The store manager and the stock manager share the tasks for a whole store. The store manager in this case, handles the selling and the work with customers and the stock manager provides the products. The other case is, that the stock manager is responsible for a whole enterprise. In this case the store manager also handles stock.
All in all, these three roles needs clearer definitions of their responsibilities inside from CoCoME. The roles of the customer and the cashier are clearly defined, as are the respective responsibilities.

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. We assume that there will be changes in the ACM due to the current status of the security-relevant data and the roles.

4.3.5. Data processing in security relevant components

The same applies to the safety-relevant components, since they are also dependent on the defined data. At present, not all data have been defined or precisely defined. We expect that there will be changes in the matrix in Figure 4.8.

5. Case study system

In this chapter, we apply steps 4 and 5 described in chapter 3. The first step is to fix all shortcomings. This includes the definition of data and roles. Then the remaining requirements are checked to see if they need an update. Once the system is fixed, the first milestone is reached. It is now possible to define scenarios. The scenarios created are used later to define and add data flows to the model. After that, the case study is complete. The complete case study can be evaluated and/or used to check access control in the system.

5.1. Introduction to the case study system

First of all, we give an overview how we fixed all the mentioned shortcomings (section 4.3). In the listing below all the shortcomings are shown. First R3:security relevant data and R4:roles are covered, because the have a direct effect on the remaining R5:access rights, S1: security relevant data in components and S2:type of data processing in the components.

- R3 security relevant data
  we defined new data, that is added to the customer. The customer object holds a list with all ordered products. With this addition, the shortcoming, that the connection between customer and the ordered products are 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.

- R4 user roles
  We defined the roles of the stock manager, store manager and the enterprise manager. After an analysis of the use cases, we chose that the stock manager handles the stock of an whole enterprise and the store manager(s) order products from the stock manager for their store(s). We also defined the tasks. The stock manager order the stock for an whole enterprise and manage the main warehouse. Another tasks is to calculate prediction for further orders. The store manager, with the definition of the stock manager, handles the tasks of a whole store on his own. The Enterprise Manager is the highest role of the respective company and has access to all data within the company.

The next list shows the changes to other requirements. We defined the roles and used data clearly. Since the different requirements are building on each other, the other requirements are reviewed and updated if necessary. The following list shows the result of the check.

- R5: access rights
  The roles of the stock manager, the store manager have been redefined and their tasks may have changed. So it is necessary to review and update the current ACM.

ACM	customer	sales and products	user	query
customer	allowed	allowed*	allowed*	system protected
cashier	allowed*	allowed	denied	system protected
stockmanager	**denied**	**allowed**	**denied**	system protected
enterprisemanager	**allowed**	**allowed**	**allowed**	system protected
storemanager	denied	allowed*	denied	system protected

Figure 5.1.: Reviewed and updated ACM for CoCoME

Data processing	customer	user	sales and product	query
Webfrontend	transmit	transmit	transmit	non-existent
PickupShop	transmit	transmit	process/transmit	non-existent
Tradingsystem:inventory:application	process/transmit	transmit	process	non-existent
Tradingsystem:inventory:data	process	process	process	process

Figure 5.2.: Reviewed and updated matrix that show show data is processed in CoCoME

The result is shown in Figure 5.1. The entries that have changed to the old version are marked bold

- S1: security relevant components
  Since all components are considered security relevant already, there is nothing to update in this requirement.

- S2: type of data processing in security relevant components
  The data processing in the components changed with the addition of the new data types. The updated processing matrix is shown in Figure 5.2. The entries that have changed to the old version are marked bold.

After all the short comings are fixed, the milestone is reached. The system is ready to be used to create a case study.
In the next step, scenarios are developed to show typical interactions with the system. From these scenarios data flows are derived to be added to the model.

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

5.2.1. 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 over 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

ACM	Webfrontend	TS:inv:application	TS:inv:data
stockmanager	user : 3 customer : 4 product & sales : 2	user : 3 customer : 4 product & sales : 2	user : 3 customer : 4 product & sales : 2

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

contains all purchases of the customer.

5.2.2. Access control

The access rights for this scenario are showcased in Figure 5.3. We omitted the Webservice component for the fact that it only transmit the data.
The lines contain the components. The roles are in the columns. The access is differentiated for each class of data in the entries. This display makes it possible to specify which role can access which data in which component. As mentioned beforehand, there are three different classes of data and a fourth modeling the query used to access the underlying database. The three classes are: customer related data, product & sales related data and user related data. In Figure 5.3 the relevant parts of the ACM for this scenario are shown. We defined four access level. Access to the data decreases with ascending numbers.

5.2.3. Component Behavior

In the shown excerpt from CoCoME (Figure 5.7) the scenario takes place. This is the basic model which was extended with SEFFs (subsection 2.3.2). For each component a SEFF is defined, which models the observable behavior. In the following the observable behavior of the scenario is described first, then the data types used, then the model extension with the described meta model. Finally, the resulting data flow is displayed.

5.2.3.1. Observable Behavior

First the ID is entered via the WebFrontend. The data type is *int*. 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:data in which it remains for the time being. Th the whole database is queried. In the next step, first the responding database entry for the ID is selected. Secondly, the database entry is used to project all purchases in forms of lists. This lists are merged in one by using concatenation. This data is transmitted back to the Tradingsystem:inventory:application, where the data is processed. The process takes the incoming list as the input and produces the output in form of the report, a list that is well formatted. Then the report is passed through the Webservice back to the Webfrontend where it is presented to the stock manager.

5.2.3.2. Used data types

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.

- user related data
  No data from this class is used.

- customer related data
  The customer object from this class is used. Mainly the ID of the customer is important.

- product & sales related data
  From this class the scenario uses the report type and the IOrderEntries. The report is a list of Strings, which depicts in detail all purchases of customer. The IOrderEntry represents an order. From a list of these IOrderEntries the report is created.

- query
  The query class is system protected. The used data type are the query data type. The query is used to encapsulate all the different operations in the database.

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 over 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.

5.2.3.3. Model extension

For each component a SEFF is added. Each SEFF has one action that showcases the , expect for the Tradingsysytem:inventory:application and the Tradingsystem:inventory:data. In the respective SEFFs two Actions are added. In the Tradingsystem:inventory:application, first the ID transmitted, then with the database' results the report is created. In the Tradingssystem:inventory:data the two data base operations are modeled.
Furthermore, a data processing container is defined for each action, in which the operation with which the data is processed is defined. Also a characteristic container is defined for each action, in which the respective access rights are defined. To complete the model, we added a database.
A data flow is defined by the SEFFs and the attached data processing containers. The SEFFs define the flow of the scenario, the containers documents the changes to th data. The resulting data flow is shown in Figure 5.5.

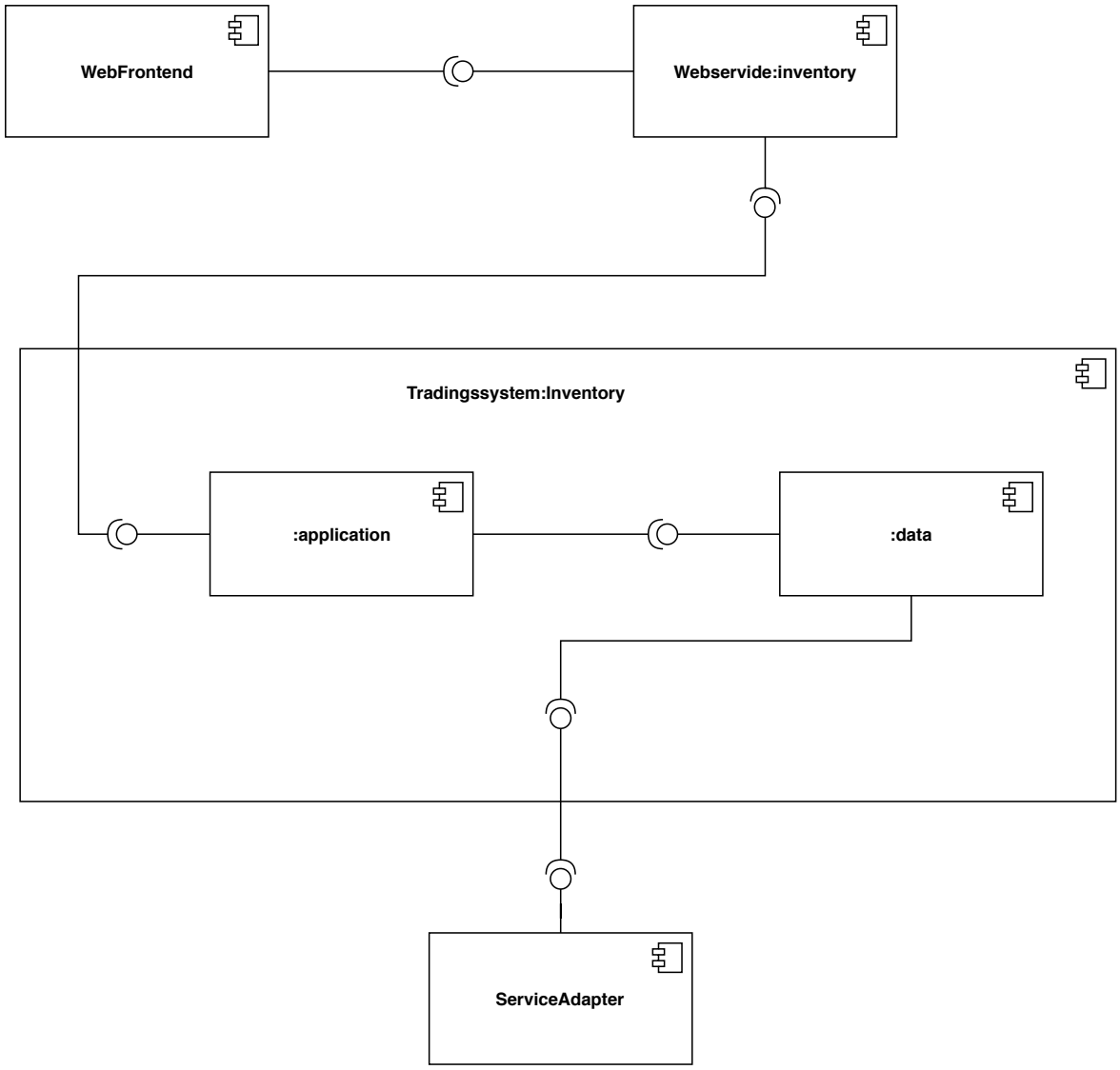

Figure 5.4.: Excerpt of CoCoME in which the described scenarios is processed

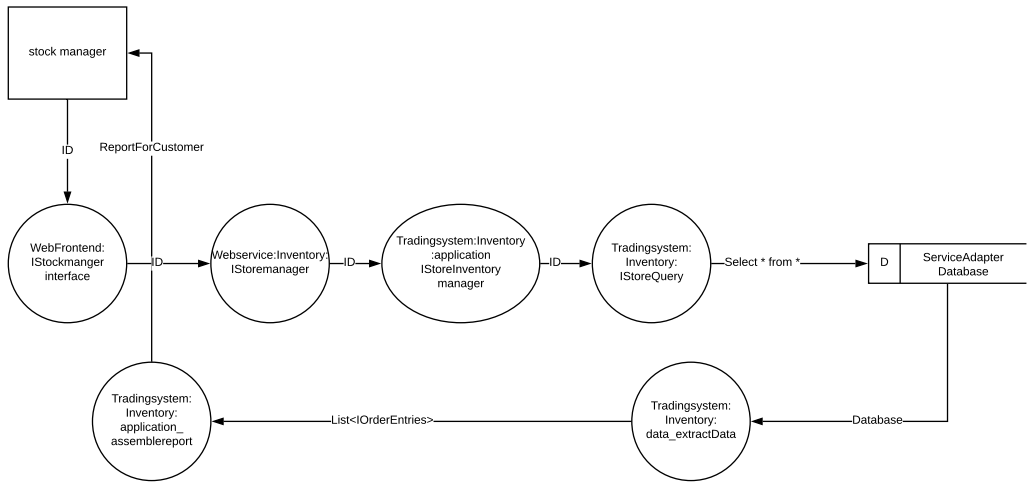

Figure 5.5.: Resulting data flow for the scenario

5.2.4. Usage Model

5.3. Scenario: Support employee requests information for an order

This scenario isn't on the basis of an use case. After we investigated CoCoME further, we couldn't find any fitting use cases that process security relevant data. We mostly looked for use cases, that process customer related data. So we created a new scenario.

5.3.1. 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 is concluded. With the report at hand, the support employee should be able to solve the customer's issue.

5.3.2. Identification of a new use case

We found the scenario described in subsection 5.3.1 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.

5.3.2.1. 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 the 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. The various actions are too broadly diversified and are not considered here.Each of the actions should be a separate use case.

5.3.2.2. Description of use case 14

- *Brief Description* The system provides access to the 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.

ACM	Webfrontend	TS:inv:application	TS:inv:data
support employee	user : 2 customer : 1 product & sales : 2	user : 2 customer : 1 product & sales : 2	user : 2 customer : 1 product & sales : 2

Figure 5.6.: An ACM showing the access rights for the case study system.

- *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

5.3.3. Access control

The access rights for this scenario are showcased in Figure 5.6. We omitted the Webservice component for the fact that it only transmit the data.
The lines contain the components. The roles are in the columns. The access is differentiated for each class of data in the entries. This display makes it possible to specify which role can access which data in which component. As mentioned beforehand, there are three different classes of data and a fourth modeling the query used to access the underlying database. The three classes are: customer related data, product & sales related data and user related data. In Figure 5.3 the relevant parts of the ACM for this scenario are shown. We defined four access level. Access to the data decreases with ascending numbers.

5.3.4. Component behavior

In this section, I present the behavior of the components that are part of the scenario. we split this section into three parts. First we describe the observable behavior, then we present the data types used. Finally, the model extension and the resulting data flow are shown.

5.3.4.1. Observable behavior

First the ID of the authenticated customer is entered in the Webfrontend by the support employee. Then the ID is transmitted through the Webservice to the Tradingsystem:inventory. The Tradingssystem:inventory consists out of two compoents. First, the tradingssystem:inventory:application transmits the ID to tradingssystem:inventory:data, where th ID stays for the time being. Then the whole database is selected. In the next step, a query is

created by using the ID. Then the query is used to select the corresponding customer. Then the customer is used to build another query and select the corresponding order. The order 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 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.

5.3.4.2. Used data types

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.

- customer related data
  In this scenario the complete customer object is used.

- user related data
  In this scenario, no user related data

- product & sales related data
  The support employee receives a complete report about the order. A large part of the product and sales based data is used for this purpose.

- query data
  This data is system protected. The data that is used to build queries are used to perform the two selections in the data base.

5.3.4.3. Model extension

For each component that is used in this scenario, a SEFF is defined. SEFFs define the observable behavior of components. Each SEFF consists of one more actions. We added only Internal action, except for the Webfrontend, because this component is called from the system user. For the components in the Tradingssystem:inventory component we defined two internal action. In the application component first the data is passed, than the report for the support employee is built. In the data component, two request to the data base are necessary to extract the data. For each of the actions a data processing container, in which the operation is stored. Such an operation takes an input and produces an output. In the operation the data processing is described.
The containers are linked by the SEFFs and the model. This creates a data flow. The data flow is shown in Figure Further, characteristic container are defined, in which the access rights are stored. A characteristics container is defined for each component in the scenario.

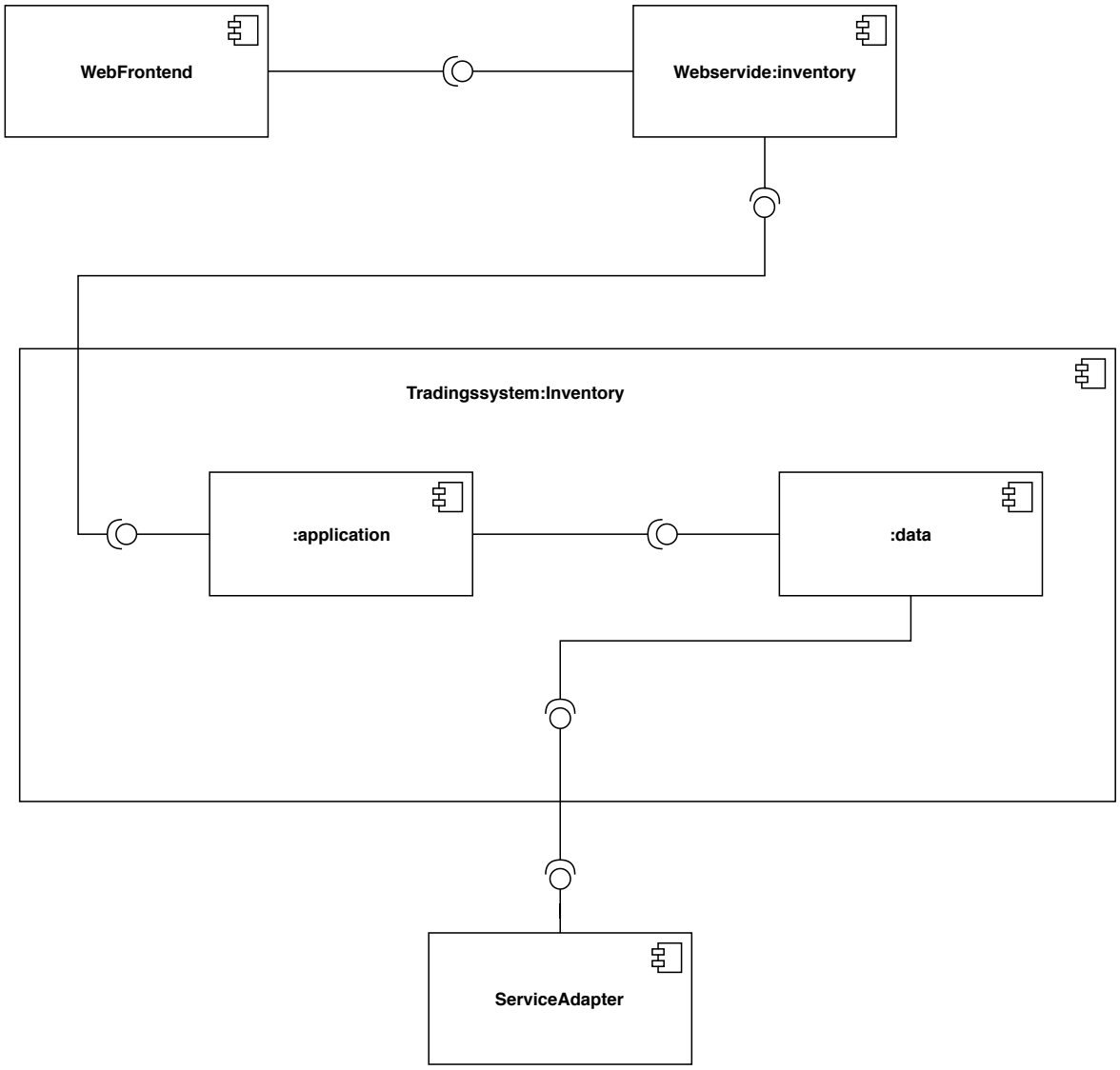

Figure 5.7.: Excerpt of CoCoME in which the described scenario is processed

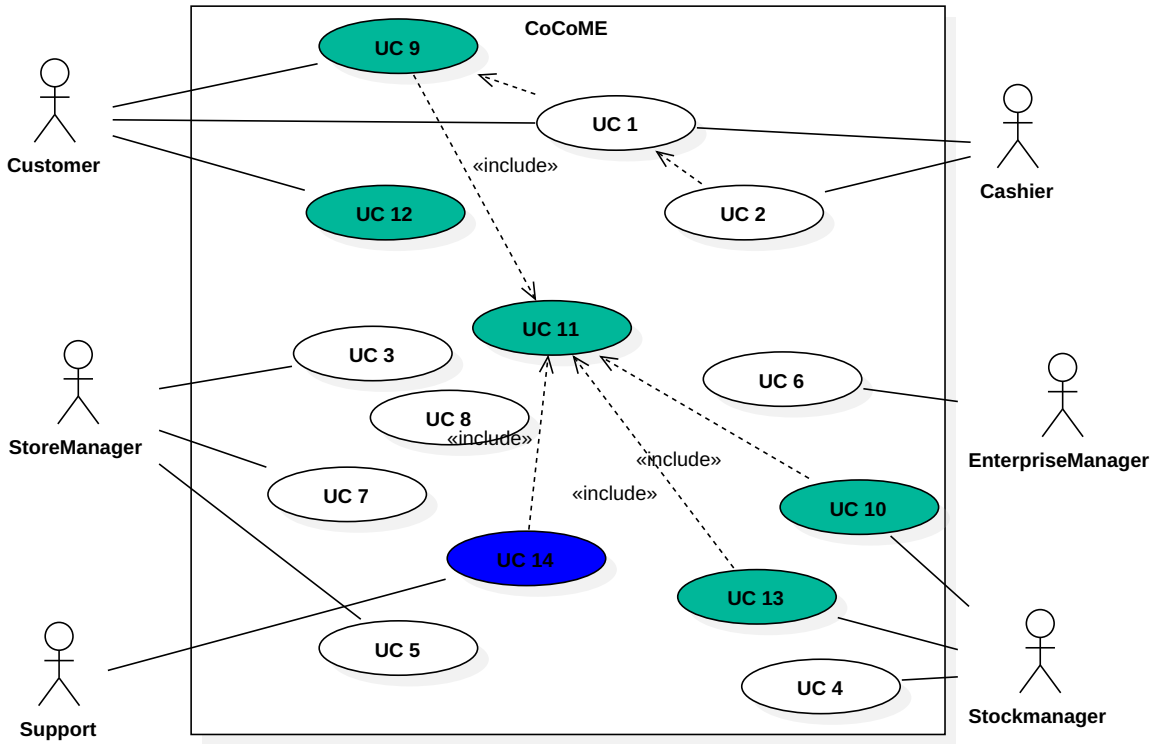

Figure 5.8.: Use case diagram with the new use case 14 in it.

5.3.5. Usage model

5.3.6. Resulting addition to CoCoME

The classification of the use case in CoCoME can be seen in Figure 5.8. The created use case is displayed in blue, the use cases we consider are displayed in cyan the light ones are the remaining use cases.

6. Evaluation

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. The three aspects are evaluated . Each evaluation follow a GQM plan [**GQMIntro**]

6.2. Applicability of the introduced method

The first aspect we want to evaluate is applicability for the introduced method for software systems. This is important to show that the method is applicable for software systems otherwise the goal for the creation of the method is missed. 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 concrete system?
As a metric to answer the question, we applied the described method to the CoCoME system.

6.2.1.1. Application to the CoCoME system

In the chapter four and five, 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 three and after all the steps were taken we created a case study that may be used to perform a data-based privacy analysis. Therefore, we concluded that the method is applicable.

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 some 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.

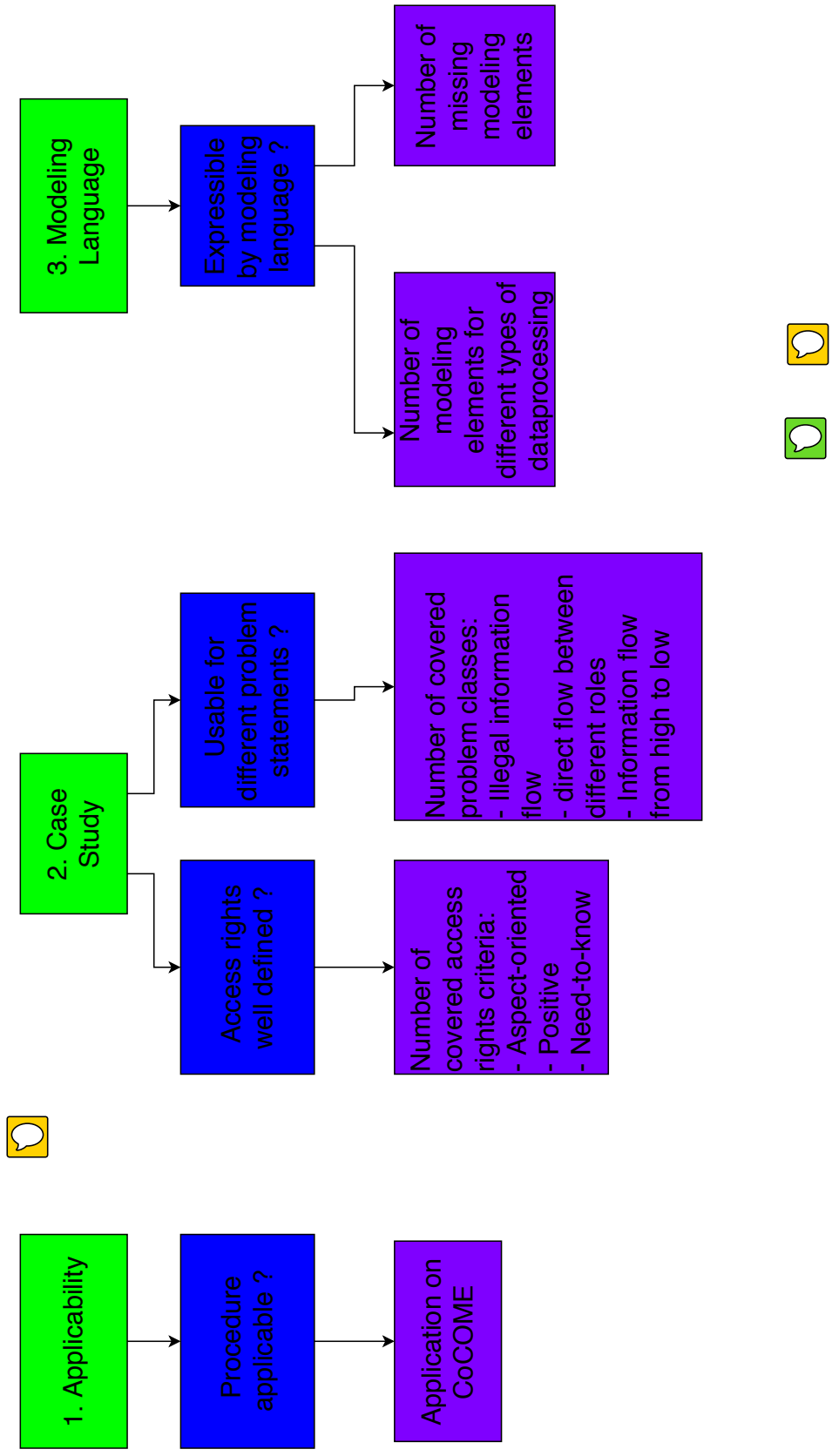

Figure 6.1.: The GQM plan for the evaluation.

6.3.1. Are the access rights well defined ?

~~To verify if the goal is achieved, we defined the following question:~~
~~Are the access rights well defined?~~
Well defined access rights means in this context that the defined access rights meet the criteria. In [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, so we selected three of them to evaluate the defined access rights. This is due the current state of CocoME and time constraints for this thesis. The three chosen criteria are 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 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 use din different contexts. This criterion is achieved by the created case study. 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.

Positive The criterion *positive* ensures that the access rights are well defined for each role. As benchmark it forbids an access right that simply say *no access*. The idea of this criterion to force that each access permission is listed. This ensures that no role and no data is left out, which leads to well defined access rights. The created case study achieves this criterion partly. First, granular access control rights have been defined. The access rights are differentiated in four different levels. But we defined the access right *noAccess* to forbid access for the stock manager to customer related data.

Need-to-know 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 is intended 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. This criterion is achieved by the created case study. The access levels *accessToOwnedData* and *accessToNecessaryData* are defined. The first describes access to data that belong to the role, the other includes and describes access to other data in the same class that is not owned by the role but necessary to perform its tasks.

6.3.2. Is the created case Study usable for different problem statements ?

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?
It is not trivial to which problem statements the question refers to. In the case of this evaluation we refer to the problem statement *Non-influence* [8]. 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

~~The different information flow classes are dependent on the created scenarios. This metrics measures therefore the quality of the defined scenarios.~~ We evaluate the variety of covered information flow classes and therefore the number of defined scenarios that model information flow for a specific class. We defined four classes of information flow to cover non-interference and non-leakage.

Illegal information flow This class is the most intuitive of all. This class covers information flow of data to roles that are not allowed to get access to this type data. This class is present in the case study. The first scenario model illegal data flow. The stock manager in this scenario has access to the ID of an customer. S/he 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*AccessToNecessaryData*. The report for one customer is no necessary data for the stock manager to perform his/her 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 This information flow describes the existence a sequence of inputs so that is possible for a role that inherit a certain security level to get access to data that are only accessible by a higher security levels. In the case study no such information flow is present, because our scenarios only involve a single role. There is no scenario in which two roles interact with the system. So there is no data flow that model this class.

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. We did not change the base system of CoCoME. In CoCoME the data flows are observable, so this class is not modeled in a scenario.

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. For the fact that no scenario with roles 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 [12] 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 ?

~~To verify the goal of the evaluation we defined the flowing question:~~
~~Is PCM able to express the case study ?~~
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.

Access Rights	fulfilled ?
Aspect-oriented	yes
Positive	no
Need-to-know	yes

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

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

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

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, a 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

6.5. Summary

After all three parts of the evaluation are now done, it is time to summarize. 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.
The second part evaluates 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 section 6.5, we achieved with our definition of the access rights two out of three criteria. There is surely some work to be done to get to a more solid state. The second part of the created case study are 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 flow sin the system. The result is shown in Figure 6.5. 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

Meta model	possible ?
relational algebra	yes
I/O operations	yes
Transmission of data	yes
Change of access rights	yes
Alternation of data	yes

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

PCM. The result is shown in Figure 6.5. As shown, all types of data processing are modeled. Therefore, we conclude it should be possible to model all created case studies with PCM. 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 would 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.

7. Related work

Nur eine kurze Idee, das wir schon mal drüber sprechen können. Wenn es dir lierber ist, können wir es auch auf Montag den 10.09.2018 verschieben. Da ich nun nicht mehr UMLsec und SecureUMl als RelatedWork angeben kann, stelle ich nun mein related work vor.

- [3] as the main reference. Here a case study of software system is created. The main focus of the publication are the access rights. Some criteria that define good access rights are defined. Also the creation of the case study is shown. Our methods are more fine grained, because we aim at larger systems. The system under investigation is more bare minimum than CoCoME.
- Case studies are mainly used in psychology, sociology, political science, social work, economy and community planning. In software system they are a relatively new concept, or how we implement them. With regard to case studies, we probably have to quote Yin [13]. But I found another publication here [10], which basically describes case studies, but not the way we do it. I even flew over it.
- The main statement here is that there are really few suitable publications. The idea is to

8. Conclusion and future work

Das würde ich erst schreiben wenn die Evaluation steht.

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. "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.

[7] 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.

[8] 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.

[9] 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.

[10] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164. DOI: 10.1007/s10664-008-9102-8. URL: https://doi.org/10.1007/s10664-008-9102-8.

[11] 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.

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

[13] Robert K. Yin. *Case Study Research: Design and Methods (Applied Social Research Methods)*. Fourth Edition. Sage Publications, 2008. ISBN: 1412960991. URL: http://www.amazon.de/Case-Study-Research-Methods-Applied/dp/1412960991%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1412960991.

A. Appendix

A.1. First Appendix Section