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 – 12. October 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.

Karlsruhe, October 10, 2018

. . . .

(Julian Hinrichs)

Abstract

There approaches that are aiming to ensure compliance on an architectural level. This is accomplished by conducting a data-based privacy analysis. Most commonly these approaches uses case studies for an evaluation.
In this thesis, we introduced a method to create case studies. The created case studies may be used to evaluate data-based privacy analysis. We also provided evaluation criteria to verify if the created case study is usable. Further, we applied our method on a realistic system and evaluated the resulting case study. In the end, we verified that for the excerpt we were investigating the method is applicable. Also the created case study was evaluated. The findings were that the case study doe snot cover all the investigated information flow classes. At last, we also evaluated the used modeling language.

Zusammenfassung

Die Arbeit stellt eine Methode vor, die in der datenbasierte Datenschutzanalyse verwendet werden kann. Dazu wurde ein bekanntes System benutzt, auf das die Mehtode angewendet wurde. Als Ergebnis der Anwendung der Methode wurde eine Fallstudie erstellt. Zum Schluss haben wir die Methode, die Fallstudie sowie die verwendere Modellierungssprachen evaluiert. Die Ergebnisse der Evaluation sind die Folgenden: die Anwendbarkeit für ein spezielles System ist gegeben. Die erstellte Fallstudie deckt nicht alle überprüften Informationsflussklassen ab. Die verwendete Modellierungssprache ist bis auf eine kleine Einschränkung nutzbar. Zu guter letzt, die Ergbenisse der Arbeit können für datenbasierte Datenschutzanalysen zur Evaluation verwendet werden, um eine oder mehrere Fallstudien zu erstellen und deren Qualität zu überprüfen.

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 [15]. 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[5] uses an attacker model to verify security on an architectural level. At last SecureUML[6], 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)[3] 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[3] 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)[10] 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 4, 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 5) and define the missing elements and add it to the system, secondly, we create the case study in **??**. In Evaluation we evaluate the method, case study and the used modeling language and discuss the evaluation. In **??**, 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, we will introduce the foundations that we are using in my thesis. For each, we 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

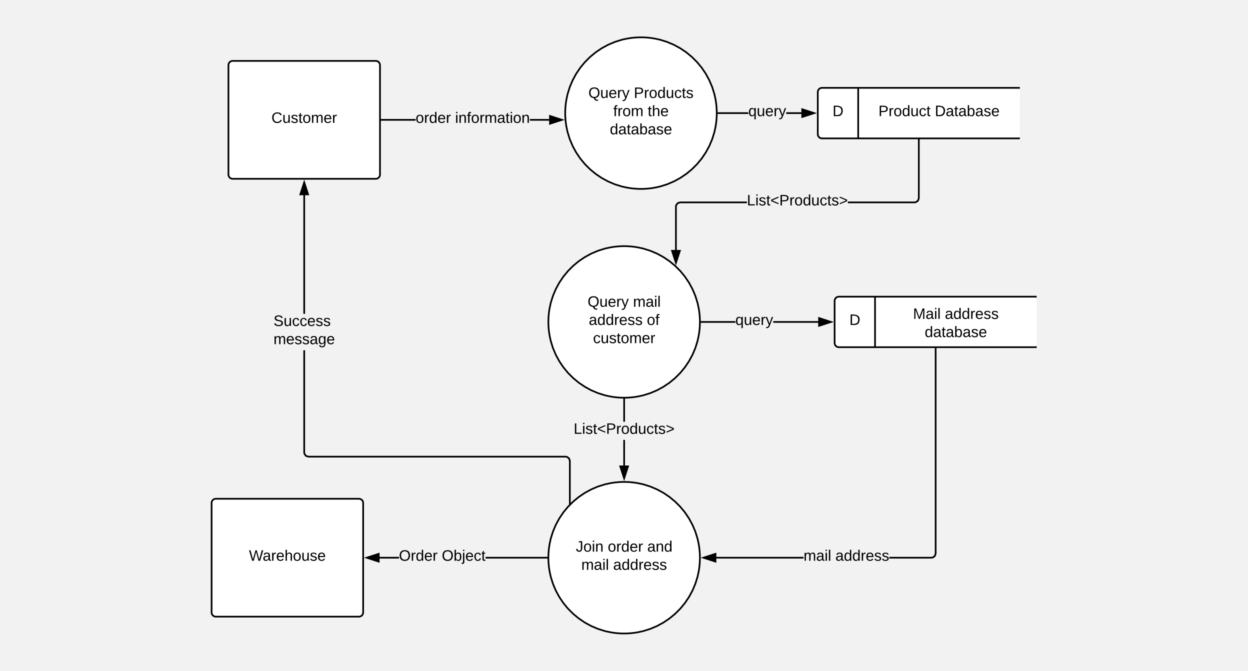

Figure 2.1.: Simple data flow diagram for linking an order of different products to a mail address

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.

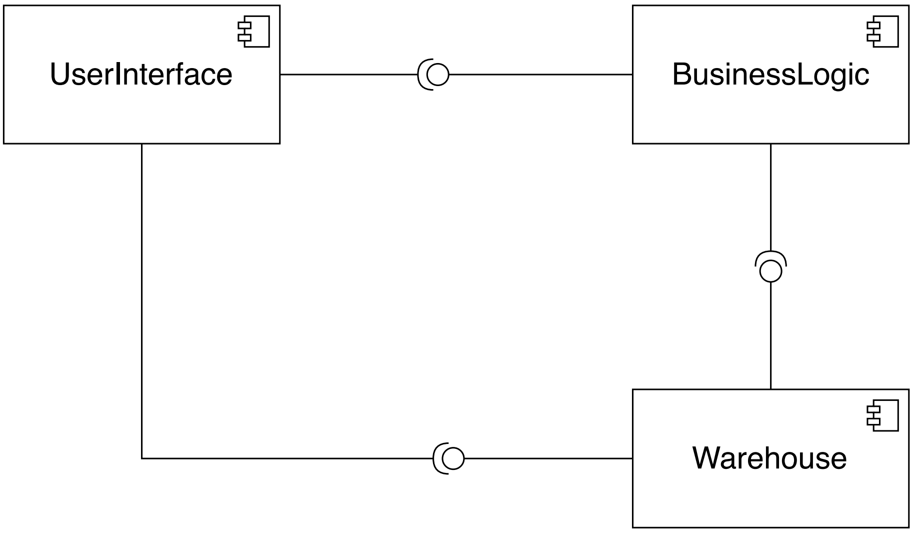

Figure 2.2.: A simple component based system

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 [10]. 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, we 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, we are 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.

2.4. Goal-Question-Metric approach

3. Related work

After the foundations were presented, we discuss here the related work. First, we present the work that was the basis for further work on this paper.

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 was 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 in the previously mentioned fields. Nevertheless, the discrepancy between computer science and the classical fields for case studies, is too great. Only fundamental methods could be adopted by the method. Such methods includes, for example, to use two sources of data [16]. 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 [4]. The only publication that roughly outlines the creation a case study for a software system, was the one from Evered and Bögeholz [2]. 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. The main goal of their publication was to define good access rights in component based systems by using their structure. We have adopted the introduced criteria of Evered and Bögeholz [2] 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 and then went right to evaluation. 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 running system. In contrast to Evered 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 Evered 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 Evered and Bögeholz have given us a good starting point for our method and we have expanded theirs.

We lloked further but could not find any other fitting publications that may address our matter. We found a lot publications that adresses different case studies but none with the same or similar aim as our method.

4. Procedure

In this chapter we are going to present a procedure to create a case study that can be used to evaluate a data-based privacy analysis. The privacy is defined by access rights. We will describe the procedure step by step and the places where the case study will be evaluated. During the process, the case study is evaluated based on two aspects,the quality of the access rights and the covered information flow classes. Overall, the resulting case study consists of an extended system model. The extensions include the access rights and the added data flows.
First, we give a brief overview over the procedure shown in Figure 4.1. After that each step will be explained in detail.
The procedure consists of seven consecutive steps. First the current state of the system is reviewed, secondly the necessary elements for the requirements are extracted. Thereafter the fulfillment of the requirements is reviewed and the required system extensions are defined in the next step. Afterwards, the access rights are evaluated and the results are stored in a milestone. Then scenarios for the system are defined. In the final step, the information flow classes covered by the scenarios are evaluated. Finally, it is decided whether the case study is sufficient for the intended use.

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

Here we describe the procedure for creating a case study. An overview is shown in Figure 4.1. We cover the steps P1-P7 that are necessary to create a case study. We further explain places when the evaluations are conducted.
The first step is to decide whether it is worthwhile to prepare a case study based on the system under investigation. This is decided by reviewing the current documentation. The goal is to In a second step, six requirements for creating a case study are introduced (see . 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 to identify the data in the system and identify which of the data is security relevant. The fourth requirement is the definition of user roles in the examined system. The fifth requirement is definition of access rights between (security relevant) data and the user roles. The access rights can either be generated from the previous requirements or extracted from the documentation. The sixth and last requirement is the identification or definition of the different types of data processing. Furthermore, for each component it is either defined or identified which data is processed as in the individual component.
In the third step, the current fulfillment of requirements are analyzed and collected in a summary. Usually there are shortcomings. If there are none, the next step can be skipped.

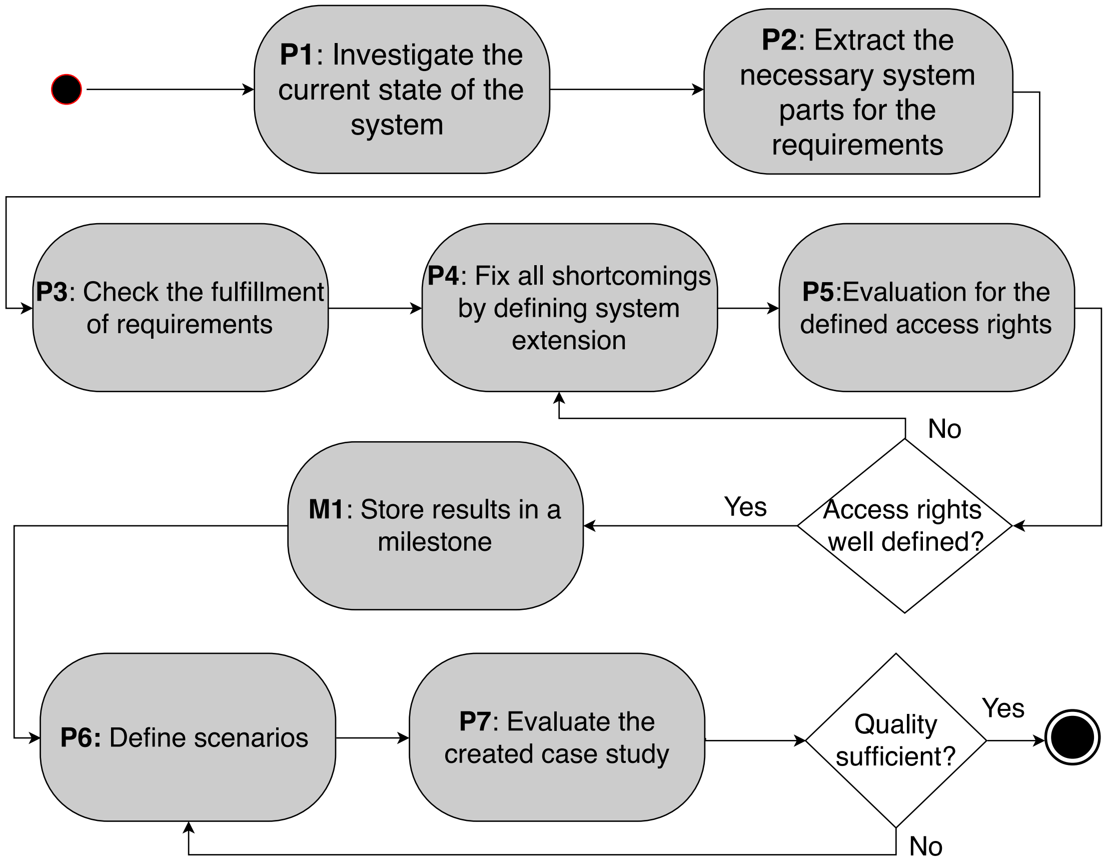
P1: Investigate the current state of the system
P2: Extract the necessary system parts for the requirements
P3: Check the fulfillment of requirements
P4: Fix all shortcomings by defining system extension
P5:Evaluation for the defined access rights
No
Yes
Access rights well defined?
M1: Store results in a milestone
P6: Define scenarios
P7: Evaluate the created case study
Quality sufficient?
Yes
No

Figure 4.1.: General overview over the procedure for creating a viable case study

In the fourth system extensions for the missing parts of the requirements are defined.
In the next step, the evaluation for the the access rights is conducted.
After the evaluation is done and the results are sufficient, the milestone is reached. All necessary parts of the system are present, so that a case study can be created. If the results are not sufficient the procedure iterates back to the fourth step. 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.
In the seventh step the covered information flow classes are evaluated. Then it is decided if enough information flow classes are covered for the planned use of the case study. If not, the procedure iterates back to the sixth step. Otherwise, the procedure is concluded and the case study is created. The case study finally consists of the system model extended by data flows and the access rights.

4.2. P1: Investigate the current state of the system

Before the process is started, a first glance at the present system is taken. This first glance is taken to decide whether it is worthwhile to start the process. It is checked if the system is sufficiently defined in *width* and *depth*. *Width* checks how many different system models are already defined. *Depth* checks how detailed each model is defined. The outcome is system dependent. For example, it could be important how much work one is willing to do before starting the process. The idea behind this step is to decide beforehand if the system is in a state where it is more than just a bare concept and the case study can be constructed with reasonable effort.

4.3. P2: Extract necessary system parts for the requirements

The second step in the procedure is to extract the corresponding elements or definitions from the system for each requirement. Altogether, six 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.

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

Table 4.1.: An overview over all the requirements for 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 the data. Breier [1] 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 [8] 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 Evered and Bögeholz [2] 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 higher level, 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 4.2 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

ACM	Usermanager
Employee	username: fullAccess password: noAccess

Figure 4.2.: Reduced ACM to the show the concept.

	Login data
User interface	transmit

Table 4.2.: A simple example to showcase an OpM.

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. As the the requirement R5, the necessary elements for this requirement may be derived from previous ones, namely R1, R2 and R3. In R1 ensures that the system is modeled as a component-based system. With R2 (definition of use cases) and R3 (security relevant data) it is possible to identify how which data is processed. The different processing types are described through different operations. The various operations describe how and which data is processed in a specific component. Each operation has one more inputs and produces one or more outputs. The operations needed are highly system-dependent. Operations could describe the transmission of data, processing of user inputs or operations used in the relational algebra. The identified operations are stored in an operations matrix (OpM). In this matrix, for each component it is stored with which operations data can be processed. In Table 4.2 an excerpt of an OpM is shown. As one can see, in the component *User interface* it is only possible to transmit the login data to other components.

4.4. P3: Check fulfillment of requirements

In this step the current state of requirements is reviewed. The best case is, that all requirements are fulfilled. If this is the case, one can skip 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 present in the system. 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. The summary is done due to two reasons. Firstly, to get a comprehensive overview of the current state of the system and secondly, to enable a division of tasks. The second reason is that you get an overview of the big picture and define possible system extensions not as soon as a shortcoming is encountered but with the big picture in mind. This ensures that system extensions are defined in the context of the whole system.

4.5. P4: Fix all shortcomings by defining system extensions

In the previous step, a summary of the shortcomings in the system was created. In this summary, all system components that stand in the way of fulfilling the requirements are stored. In this step, the system extensions are defined to fulfill all requirements. When defining system extensions, the first step is to derive the needed extensions from the documentation. In the case, the documentation isn't clear or it is not possible to derive extensions for the shortcomings, the option left is to define the system extensions 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.

4.6. P5: Evaluation of the access rights

In this part of the procedure, the access rights are evaluated. The evaluation follows a GQM-plan, shown in Figure 4.3. The access rights are evaluated based on seven criteria presented by Evered and Bögeholz [2]. We divided the seven criteria in three categories: *specification*, *comprehensibility* and *implementation. Specification* how the access rights are defined. *Comprehensibility* evaluates how well access rights can be understood by others. *Implementation* evaluates the way the access rights are embedded in the code. It may not be possible for all systems to evaluate all seven criteria. The seven criteria are:

- Concise : Access rights should be precise and simple.
- Clear: At first glance one can see what the access rights state.
- Aspect-oriented: The access rights may be used in different contexts
- Need-to-know: each role only should have the absolute necessary informations
- Positive: Each right to data has to be given specifically to each role.
- Fundamental: The access rights are realized inside the code
- Efficient: The overhead caused by the checks of the access rights is reasonable.

The evaluation is done in a checklist manner. For each criterion it is checked whether the system fulfills the criterion or not.
After the evaluation of the access rights is done, the results are analyzed. If sufficient

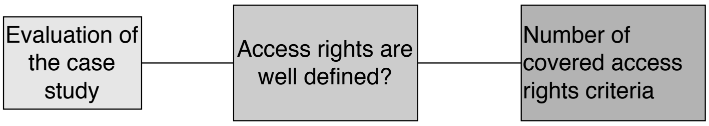
Evaluation of the case study
Access rights are well defined?
Number of covered access rights criteria

Figure 4.3.: Overview of the evaluation of the access rights.

criteria for the purpose of the case study are fulfilled, the current state of the system is considered correct. If the results are unsatisfying, the procedure moves back to the previous step P4. Otherwise the milestone is reached. The system is sufficiently defined so that is possible to add data flows and create a case study.

4.7. M1: Store the results in a milestone

After the evaluation results are sufficient, the current state of the system is stored in a milestone. This includes all definitions made for the each requirement.

- R1: The system model.
- R2: Use case diagram, definition for each use case.
- R3: The data types and the examination, which data is security relevant.
- R4: The different user roles.
- R5: The created ACM.
- R6: The created OpM.

4.8. P6: Definition of scenarios

After the milestone is reached, scenarios for the case study are defined. Scenarios are a concrete characteristic of a use case or a concrete characteristic 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. We 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 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.

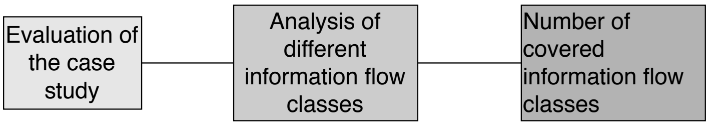
Evaluation of the case study
Analysis of different information flow classes
Number of covered information flow classes

Figure 4.4.: Overview over the GQM plan for the evaluation of the added data flows.

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 definition of the scenarios is complete and the added data flows can be evaluated.

4.9. P7:Evaluation of the covered information flow classes

The last step, is the evaluation of the defined scenarios for the system. The scenarios are evaluated on the basis of the information flow classes covered, which may differ according to the objective or use of the case study. The evaluation follows a GQM-plan. The plan is shown in Figure 4.4. We propose to use Non-influence [7] and the included information flow classes to evaluate the scenarios. Non-influence consist of the two part: non-interference and non-leakage. Non-interference describes, that in a program flow inputs of high users do not influence the output for low users. Non-leakage describes that it is not possible to observe if a specific action in the system has taken place.
non-interference and non-leakage are covered by the following four information flow classes:

- No illegal information flow:
- No information flow from high to low:
- No direct information flow:
- No observable information flow:

4.10. Conclusion of the procedure

After the second evaluation is conducted, it is necessary to decide if the coverage of the information flow classes are sufficient for the goal of the cases study. If so, the procedure is finalized and the case study is created. Otherwise, the procedure moves back to the P6. The created case study in the end consist of the system model extended with data flows and the ACM.

5. Analysis of CoCoME

In this chapter we are starting to apply the procedure described in chapter 4 to CoCoME. The procedure is applied up until the milestone is reached, as seen in First we describe CoCoME briefly. Then the steps P1-P5 of the presented procedure are carried out. First we briefly examine the current state of CoCoME, then the system parts necessary for the requirements are identified and/or extracted from the documentation. After that, a summary of the shortcomings is created and in the next step the identified shortcomings are eliminated by defining system extensions. The defined access rights are then evaluated. Finally, the current state of the system is saved in the milestone.

5.1. CoCoME overview

CoCoME is short for Common Component Modelling Example. 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 modelling approaches. CoCoME abstract the selling process and the warehousing of supermarket and provides three core functionalities:

- Operating of a register cash system.
- Management of a warehouse.
- Management of different enterprises.

CoCoME allows to abstract various enterprises, where each one models a supermarket group like Lidl or Aldi. Each enterprise may consist of one more store. Each enterprise may has various stores, with each of them selling products . Each of these stores has various cash desks, where products are sold. Furthermore, CoCoME also handles the management of the warehouses for each store. When a product is sold at a cash desk or a delivery from a supplier is accepted, the current stock is automatically updated.
CoCoME was introduced as a result of a seminar more than 10 years ago at the university Clausthal. 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.
The hybrid cloud based variant emerged from an earlier version. We use the description of the system provided in the CoCoME tech report [3]. The hybrid cloud based variant

was created by adding four evolutionary scenarios . These four evolutionary scenarios are : Platform Migration, addition of 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 5.1. As shown, the architecture is divided in three different layers plus one layer to abstract the database.
The first layer is the user interface. This layer includes the *PickupShop* and the *Webfrontend* components. The user interface is the point where the different users interact with CoCoME. Their request are transmitted to the underlying components and the results are displayed in this layer.
The second layer is called web service. This layer transmit the requests from the user interface to the business logic. Further this layer is used to add dynamically functionality to CoCoME.
The third layer is the business logic, namely *Tradingssystem*. The *Tradingssystem* component is split up in two main parts. First the *Tradingssystem:cashdeskline* and secondly the *Tradingssystem:inventory. Tradingssystem:cashdeskline* represents the register cash system. The *Tradingssystem:inventory* represents the management of a warehouse. Each request from customer or employees is processed in the *Tradingssystem.*
At last, the *ServiceAdapter* component doesn't match to any of the previously mentioned layer. This component abstracts the access to the database.

5.1.1. Platform migration

The first evolutionary scenario is platform migration. This is done to reduce the costs. The CoCoME system is moved to a cloud environment for easier scalability. If, for example, the pickup shop is heavily used due to a large advertisement campaign, additional resources in order to compensate for bottlenecks, can easily be purchased.

5.1.2. Addition of a Pickup shop

The next evolutionary scenario is the addition of a Pickup shop to CoCoME. The Pickup shop offers a second Point-of-Sale for CoCoME to concurrent with web shops like Amazon. Customer may order products online and pick them up in their chosen store. Therefore an account in the CoCoME system is required. With the Pick up shop, CoCoME changes from a closed to an open system. As shown in Figure 5.1, the *PickupShop* bypasses the *Tradingssystem:cashdeskline* and directly accesses the*Tradingssystem:inventory.* The payment process is carried out in the *PickupShop* component.

5.1.3. Database migration

In this evolutionary scenario the database is migrated to a cloud service. Due to the fact, that the PickupShop and the stores both operate on the inventory, a higher amount of

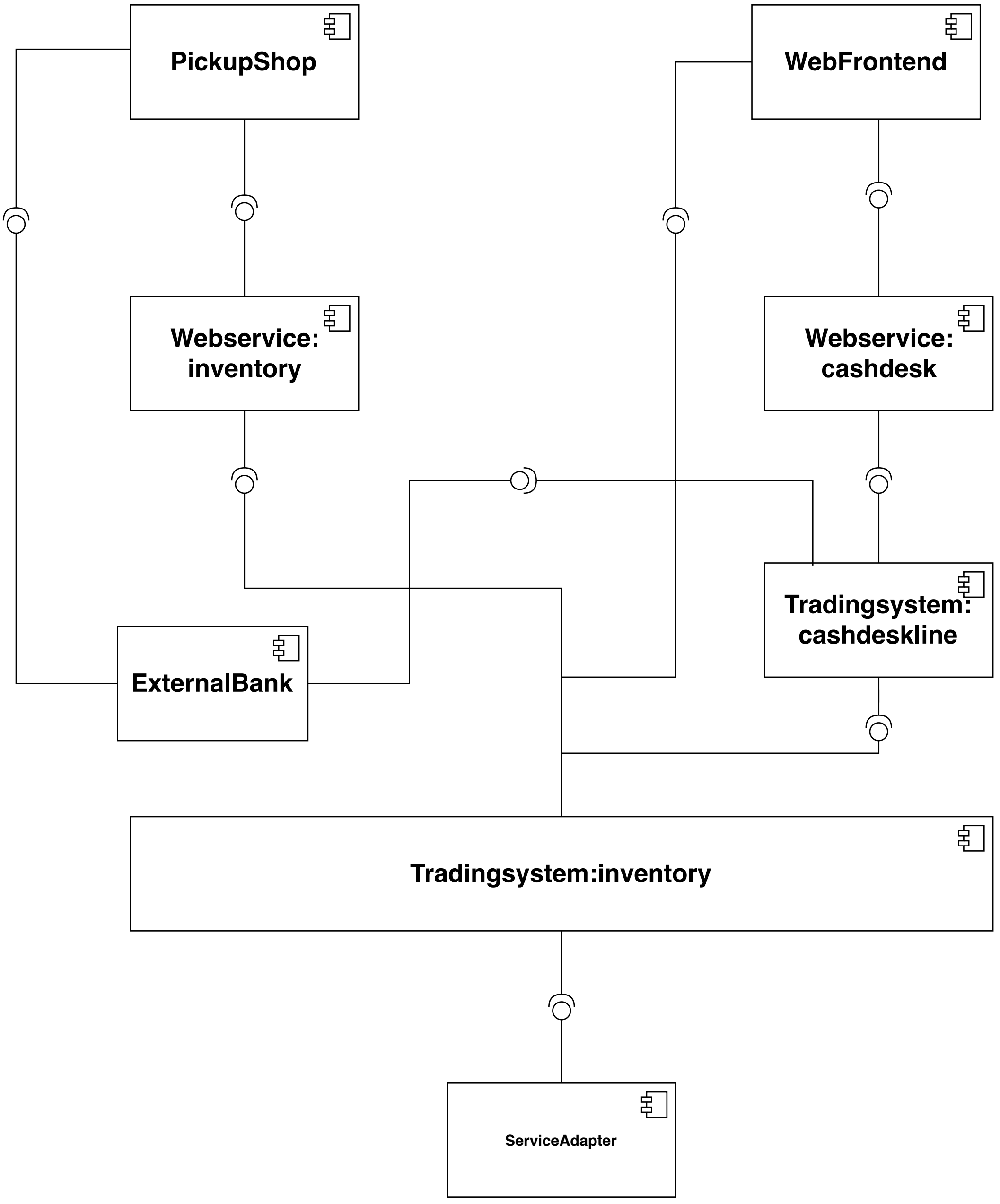

Figure 5.1.: The figure shows a simplified overview over the hybrid cloud based variant

database request is expected. In order to avoid possible shortcomings in the performance, the database was moved to a cloud environment. This allows to easily purchase additional bandwidth or space to solve upcoming scalability issues. With the migration of the database a new issue arise. It may happen, for example, that data from enterprises that are located in the European Union (EU) are stored outside the EU, which does not comply with the current law.

5.1.4. Service adapter

Since CoCoME was always in development, CoCoME currently consists of a relatively large code base. Since this does not have to be understood by every single developer to work with CoCoME, a service adapter has been added. The service adapter allows CoCoME to dynamically add functionality.

Reasons for CoCoME to apply our 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 [3] and the CoCoME book [9]. 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 fully functional system.

5.2. Application of the method

After an short overview over CoCoME in the previous section, we are starting with the procedure for the creation of a case study. The presented paper from Runeson and Höst [11], which is could be seen as a fundamental source for the creation of case studies, states that one should use more than one source of information for the investigated case. We used as our main source of information the CoCoME tech report [3] and the CoCoMe book [9], were the system is described in detail. Further we used the code of the implementation. . If there are differences between documentation and code, the definitions from the documentation applies. We favour the documentation over the code because CoCoME is still in development. The documentation represents a stable version of CoCoME. In the implementation it may possible that at the moment new features are tested that are later quashed.

We do not apply the process to the entire CoCoME system, but concentrate on only part of the system. The selected part of CoCoME is the PickupShop (**??**).

5.2.1. P1: Investigation of the current state of CoCoME

In this first step we take a first look at CoCoME. We are looking for two main aspects here. First, if the system is extendable. Secondly, how wide spread the system model already is defined. The CoCoME include a book [9], a tech report [3] and an implementation. In the documentation CoCoME various system models for CoCoME are available and CoCoME is modeled as an component-based system and therefore extendable. So we decided that

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

Table 5.1.: Overview for the requirements for a case study

the system is in viable state.In the case of CoCoME, a tech report [3], a book [9] and an implementation is available. For the fact, that CoCoME is modelled as a component-based system, the system is also extendable.

5.2.2. P2: Extract the necessary system parts for the requirements

In the next step, we identify the necessary system parts in CoCoME for the six requirements listed in Table 5.1. In the following, for each requirement the related system parts are extracted.

5.2.2.1. R1: Component based system

This requirement is met because a component-based system model is defined in the CoCoME tech report[3].

5.2.2.2. R2: Existence of use cases

The CoCoME documentation defines thirteen use cases, so this requirement is also met. The use cases are defined in detail either in the CoCoME book [9] in the CoCoME tech report [3].

5.2.2.3. R3: Security relevant data

Since the amount of data is relatively large in the investigated excerpt of CoCoME, we have grouped the data into different equivalence classes. The data type in each class has the same security level. Composite data types inherit the security levels from most restrictive class. We defined four classes: **customer data**, **account data**, **product&sales data** and **system data**.

- Customer data
  This class describes all data types that are provided by the customer role. This includes names, addresses and credit card details, to name the most important ones.

- Product&Sales 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.

- Account data
  Account related data refers to the introduced accounts with he addition of the PickupShop. It is needed that each role is registered with an account in the system. The account data type holds the *username* and the credentials for an account.

- System data
  In this class mainly summarizes the created queries for access to the underlying database.

The security relevance for each class is defined according to Breier's approach [1]. The approach is based on the possible damage the loss of an asset could cause. So for each asset the level of damage a loss of the asset could cause is defined. Further, for each asset the dependencies to other assets is defined. Based on this, it is possible to calculate the value of an asset. This approach is not one-to-one applicable to our case, but we can use the basic idea. We evaluated for each data class the possible impact a leak of data from this class may have in CoCoME. Keep in mind that the primary goal is to protect persons that are using CoCoME. Therefore, the customer data is the most relevant and defined as **security relevant**.
The account data allows to log in as the specific role, which may lead to leaks of the customer's personal data. So the account data is also **security relevant**. The system data mostly consists of queries. If one may gain access to the query, it is to alter the database. So this data is especially **security relevant**, because access to it may cause damage to the customer and the enterprise in the same scale.
At last, the product&sales data is not considered strictly security relevant, because in the sales process the payment process as we identify it is part of the customer data. This data becomes relevant in composite data types or if it allows traceability to customer data. Nonetheless, leak in this class may cause heavy damages to the respective enterprise, which is not in the scope of the thesis.

5.2.2.4. R4: roles

CoCoME defines six different roles. Five are defined in the CoCoME tech report [3]. These are the *Customer*, the *Cashier*, the *StoreManager*, the *Enterprise manager*, and the *Stockmanager*. The last role, the *Admin*, appears in the code. In the documentation, 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 information we are looking for, 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 analysis of the use cases, we are not sure what the concrete tasks are for the *EnterpriseManager* and the data requires requires or provide. There are only two uses cases that involves the *EnterpriseManager*, therefore a clear definition is not possible. Unfortunately, the code does not contain any further information. From 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.
- After analyzing the code we found a sixth role, the *Admin* role. This role is not mentioned in the use cases. From the name alone, we assume that this role keep a watch on the system. But use cases are missing and the implementation do not allow to define neither the tasks nor the provided/required data.

Since the hybrid cloud based variant is examined by us, all roles provide account data, because each role, in the hybrid cloud-based variant, needs an account in CoCoME to access the system.

5.2.2.5. R5: access rights

To define the access rights, we use the finer grained, high level from proposed by Evered and Bögeholz [2] in contrast to the well known read/write semantics for access control. Therefore, as described in chapter 4, it is defined for the data present in a component how which role can access it. We defined four different privilege level for the access to data, presented in the following enumeration. We used the elements from the previous requirements R2 (R2: Existence of use cases), R3 (R3: Security relevant data) and R4 (R4: roles) to derive the access rights for CoCoME. The access to data decreases with the ascending numbers.

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 in 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 account data or customer data. This access rights ensures that roles may access data that is provided by it but not necessarily needed for their tasks.

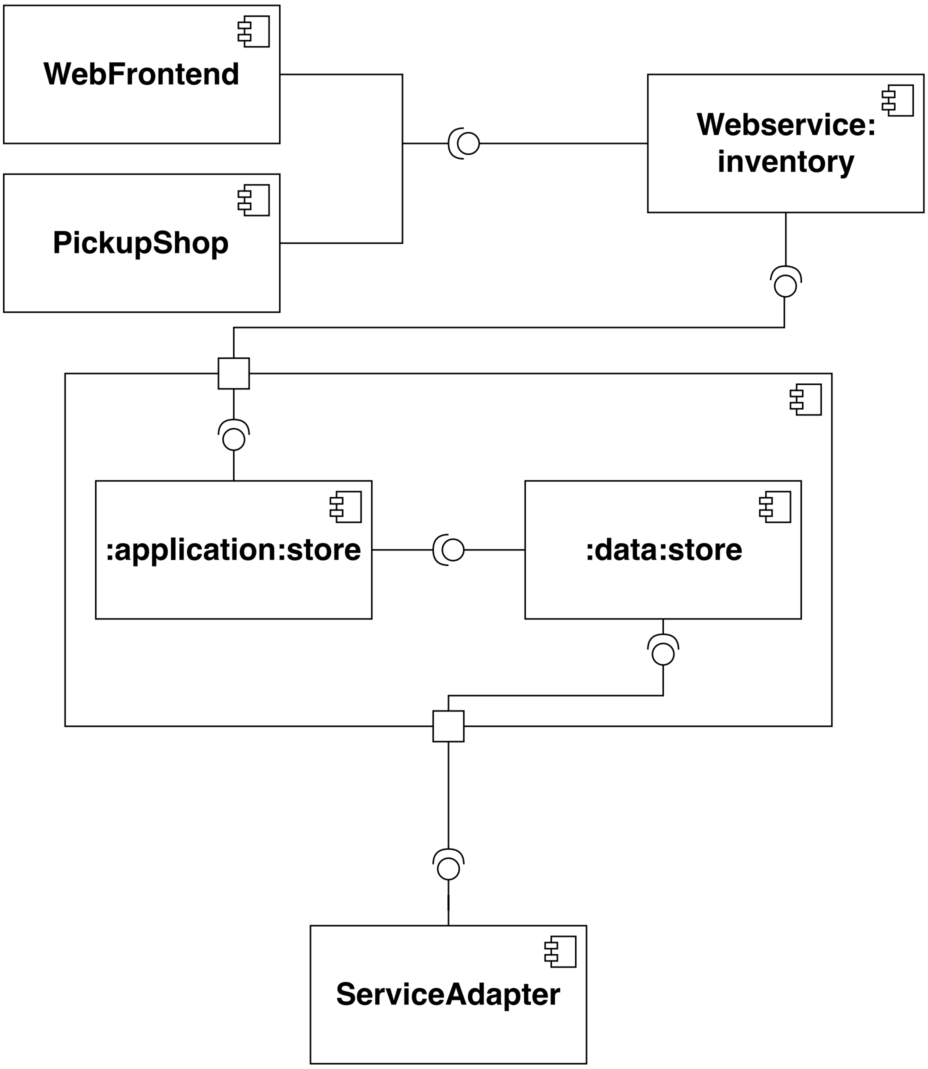

Figure 5.2.: The figure shows an simplified overview of the CoCoME. The Webservice component is omitted.

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. A reduced system model is displayed in Figure 5.2. As already mentioned in section 5.1, it is only possible for individual users to interact with CoCoME via the components *Webfrontend* and *PickupShop*. No role except the admin has direct access to the *Tradingsystem:inventory* component in which the data is processed. Therefore, all access rights to data are cascaded for the individual roles (except the admin). This ensures that it is always visible which role receives which data or has provided which data.
For the fact that the ACM in Table 5.2.2.5 is rather large and complex, we only point out

the most important points. We omitted the *Webservice* component, as it only transmits data between the user interface and the *Tradingsystem:inventory* and therefore has no effect on the data. As stated before, only the *customer* may access the *PickupShop*, therefore no other role may access the data in this component. For the *Admin* it is not clear if the role may access data. Also, the *Admin* it the only role that may access *system* data. The *StockManager* has access to customer data due to the use cases. In

ACM	Webfrontend		PickupShop		TS:inventory data		TS:inventory application	
customer	customer data	3	customer data	3	customer data	3	customer data	3
	account data	3	account data	3	account data	3	account 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
	account data	3	account data	4	account data	3	account 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
	account data	3	account data	4	account data	4	account 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
	account data	2	account data	2	account data	2	account 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
	account data	3	account data	4	account data	3	account 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
Admin	customer data	?	customer data	?	customer data	1	customer data	1
	account data	?	account data	?	account data	1	account data	1
	p&s data	?	p&s data	?	p&s data	1	p&s data	1
	system data	1	system data	1	system data	1	system data	1

Table 5.2.: 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*. The abbreviation *p&s data* stands for product&sales data.

5.2.2.6. R6: Type of data processing in components

The last requirement defines the different types how data is processed in CoCoME. For each component, it is defined how the data from the different data classes is processed in this specific component. To identify the different types of processing we analyzed the code and the use cases. All in all, we defined four different classes how data is processed in CoCoME. Each class may consists of various operations.

- relational algebra
  This type describes the operations of relational algebra on a database or similar construct. For example, if all elements of a list that have a particular ID are selected from this list.

- transmission of data
  This type describes the transmission of data between components.

- alternation of data
  This type includes the most operations. It describes each operation that changes the data class. Good examples are the creation of lists or the merging two or more data classes into one.

- I/O related processing
  This type describes the user interaction with CoCoME.

After the different classes of data processing are identified, we analyzed how which data is processed in each component. The respective type(s) of data processing are assigned to the respective components. The result is shown in Table 5.3.
System data is not available in any other component other than *Tradingssystem:inventory:data*, because in this component queries are created. The *Webservice* component only transmit data to other components. The *PickupShop* and *Webfrontend* transmits the *customer* and *account* data. In these to components the user interfaces are located and customer, for example, fill their shopping cart, therefore *products&sales* data is processed with the *I/O* processing.

5.2.3. P3: Check the fulfillment of the requirements

Once the system parts for each requirement have been identified, defined or derived, a brief check is made to ensure that each requirement has been met. For each requirement it is checked whether the individual system elements are complete. It is possible that some definitions are vague or missing. Vague means that there is room for interpretation or that different definitions contradict each other.
In the following, we analyzed the fulfillment for each individual requirement.

5.2.3.1. R1: component-based systems

This requirement is fulfilled, a very detailed component for CoCoME is provided [3].

Types of data processing	Customer data	Account data	Product&sales data	System data
Webfrontend	transmit	transmit	I/O-processing transmit	non-existent
PickupShop	transmit	transmit	I/O-processing, transmit	non-existent
Tradingsystem: inventory: application	change transmit	alter transmit	alter	non-existent
Tradingsystem: inventory:data	relational algebra operations	relational algebra operations	relational algebra operations	alter
Webservice	transmit	transmit	transmit	transmit

Table 5.3.: Overview of the different types of data processing in the components. For each component, it is defined how which data class can be processed.

5.2.3.2. R2: use cases

A detailed use case diagram with a brief description of each use case is provided in the CoCoME tech report [3]. All in all, 13 use cases are defined, which cover the main aspects of CoCoME.

5.2.3.3. R3: security relevant data

In the definition of the data, a lot of data is already defined by the implementation. Nonetheless, some data mentioned in the use cases is missing. We identified in the investigated excerpt two missing data types. First, there is no connection defined between the customer and the purchased products. Secondly, the *Report* data type mentioned in the use cases is also not defined.

5.2.3.4. R4: definition of roles

For some roles the actual tasks are not clear because some use cases contradict each other. Further, for the *EnterpriseManager* there are too few use cases to clearly define the tasks. For the *Admin* role use cases are completely missing, because this role is only present in the code.

5.2.3.5. R5: definition of access rights

In the case of the access rights we cannot say in general if they are fulfilled, because the access rights are derived from the previous requirements R1-R4. If there are changes in R1-R4, the defined access rights have to be reviewed and if necessary updated. Nonetheless, in the case *Admin* the tasks for this role are not clear, therefore it was not possible to derive access rights in some cases.

5.2.3.6. R6: types of data processing in the components

This requirement is adjacent to R5: definiton of the access rights. The different types of data processing are derived from the previous requirements R1-R4. If there are changes in R1-R4, the different types of data processing have to be reviewed and if necessary updated.

5.2.4. P4: Define system extensions

In the next step, we define system extensions for CoCoME to fix all shortcomings for the requirements. In the previous step all shortcomings were summarized. In the following, we give an overview how CoCoME was extended to meet all requirements.

R1: component-based system Here, no extensions was needed, because this requirement is met.

R2: definition of use cases Also, no need to define an extensions. This requirement is also met.

R3: security relevant data As identified previously in subsection 5.2.3, two data types are missing. First, there are no connection defined between a customer and its purchased products. Secondly, the *Report* data type is only mentioned but is not defined.

- Connection between customer and purchased products
  We defined that the connection is realized by adding a list of all orders to the customer object. Each order in CoCoME includes the involved products and other information like the total price.

- Report data type
  We have defined this data type as a string that can be read by humans.

R4: definition of roles For the fourth requirement, two roles need a clearer definition: the *EnterpriseManager* and the *Admin*.In the case of the *EnterpriseManager* there are too few use cases are defined to clearly derive tasks for this role. In the case of the *Admin* there are no use cases defined. because this role only exists in the code.

- EnterpriseManager
  Due to the fact, that only one use case is defined for the *EnterpriseManager*, the actual tasks for this role are not clear. So determined that the role is in charge of a whole enterprise. For the fact that a *StockManager* manages the stock for each store, we defined the *EnterpriseManager* works on a larger scale. The role monitors the workflows within an enterprise and calculates forecasts for, for example, the CPU capacity required. Therefore, the *EnterpriseManager* provides product&sales data and requires customer data, account data and product&sales data. For the customer and account data, the role only requires meta data, like the number of registered accounts and therefore no explicit data.

- Admin
  The *Admin* has no use cases associated, so we relied on the implementation. In the implementation, the role has access to all data, so we defined the *Admin* to be the master role in the system, that has access to all data. With access to all data any small or big problems may be reviewed and fixed. Therefore the role provides and requires all data types.

R5: definition of access rights Since the previous requirements are fulfilled with the defined system extensions, in this step the ACM is reviewed and updated. The updated ACM is shown in Table 5.4. The entries that have changed since the old version are shown in bold. As before, the *Webservice* component is omitted.

ACM	Webfrontend		PickupShop		TS:inventory: data		TS:inventory: application	
customer	customer data	3	customer data	3	customer data	3	customer data	3
	account data	3	account data	3	account data	3	account 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
	account data	3	account data	4	account data	3	account 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**
	account data	3	account data	4	account data	4	account 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**
	account data	**1**	**account data**	**1**	**account data**	**1**	**account 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
	account data	3	account data	4	account data	3	account 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
admin	customer data	**1**	customer data	**1**	customer data	**1**	customer data	**1**
	account data	**1**	account data	**1**	account data	**1**	account data	**1**
	p&s data	**1**	p&s data	**1**	p&s data	**1**	p&s data	**1**
	system data	**1**	system data	**1**	system data	**1**	system data	**1**

Table 5.4.: 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*. The bold entries changed since the old version.

Figure 5.3.: GQM-plan for the evaluation of the defined access rights. The evaluation focus on the highlighted parts.

Access rights	
Specification	Aspect-oriented
	Positive
	Need-to-know
Comprehensibility	Clear
	Concise
Implementation	Fundamental
	Efficient

Table 5.5.: Classification of the criteria for good access rights.

R6: types of data processing in the components The different types of data processing are also derived from the requirements R1-R4. After reviewing the different types of data processing that were defined, there are no need to extend anything.

5.2.5. P5: Evaluation of the defined access rights

After all system extensions are done, in this step the defined access rights are evaluated. The evaluation follows a GQM-plan The GQM-plan for the evaluation of the access rights is shown in

5.2.5.1. Goal: Evaluation of the case study

The goal is to evaluate the case study. A shown in Figure 5.3 the evaluation of the case study is split into two parts. In this we will only focus on the highlighted plan, the evaluation of the access rights.

5.2.5.2. Question: Are the access rights well defined ?

Well defined access rights means in this context that the defined access rights meet the criteria defined by Evered and Bögeholz[2].The two authors defined seven different criteria. In order to ensure well defined access rights, the seven criteria should be met. The seven criteria are : Concise, clear, aspect-oriented, fundamental, positive, need-to-know and efficient. We categorized the criteria into three classes: specification, comprehensibility and implementation. The division of the individual criterion into categories is shown in Table 5.5. The criteria in the class *Specification* evaluate the properties of the defined access and the general access levels that that well defined access rights cover.
The criteria in the class*Comprehensibility* evaluates the complexity of the notation.
The criteria in the class *Implementation* evaluate the embedding of the access rights in the implementation.
It was possible to only evaluate three of the seven criteria due to various reasons. *Comprehensibility* criteria ware not possible due to time constraint. The criteria in this class are

Access rights		fulfilled?
Specification	Aspect-oriented	✓
	Positive	✓
	Need-to-know	✓
Comprehensibility	Clear	?
	Concise	?
Implementation	Fundamental	n/a
	Efficient	n/a

Table 5.6.: Summary for the evaluation of the defined access rights.

checked by conducting a survey. This was not possible in the time frame of this thesis. For the *Implementation* criteria an integration of the access rights into the implementation of CoCoME is needed. This implementation is missing, so it was also not possible to check this class.
The remaining criteria *Aspect-oriented*, *Positive* and *Need-to-know* located in the *Specification* class.
Aspect-oriented deswcribes, that the implementation and the access rights are sperate stored and define. This alloows that the implementation and the access rights can be used in different contexts. Further, both are easier to understand , if they are separated.
Positive describes, that the access to data have to explicitly given to the entities. The default level of access is *no access*. This avoid that access is implicitly granted.
Need-to-know describes, that well defined access rights only allow access to the absolute minimum of data that are used by the entities.

5.2.5.3. Metric Number of satisfied criteria

The evaluation is done in a checklist manner. For each criterion the defined access rights are evaluated whether they fulfill this criterion. The results of the metric are summarized in Table 7.2.

Aspect-oriented The access rights are stored separately from the code and even from the defined 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 and each level represents fine grained access. 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 a role may access data in component A but not in component B. Further the differentiation in four levels allows, for example, to explicitly distinguish between access to data provided by that role and access to data that is not provided, but needed for the role's tasks. so this criteria is also met

Concise and clear Due to the time frame of the thesis, we could not make a qualified statement if these criteria are met.

Fundamental and Efficient Since access control is not yet part of the implementation, it was not possible to check it at all.

5.2.6. M1: Store the current state in a milestone

Before the current system state is saved in the milestone, one have to decide if the access rights are well defined. After that the milestone is created. The milestone stores for each requirement the necessary model elements.

Quality of access rights The defined access rights achieved all criteria that we were able to check. So we summarize that the defined access rights are for the moment well defined. So the procedure moves to the next step and the current system state is stored in a milestone.

Milestone For each requirement the associated system elements are stored.

- R1: The system model
  In this particular milestone, the system model from the CoCoME tech report [3] is stored.

- R2: Use case diagram, definition for each use case.
  In this particular milestone, the use case digram from the CoCoME tech report [3] is stored.

- R3: The data types and the examination, which data is security relevant.
  the four data classes ans the respective description of them are stored.

- R4: The different user roles
  The description of the tasks and the provided/required data for each role are stored. The *Admin* role is explicitly stored, because this role is not mentioned in the stored use case diagram.

- R5: The created ACM Table 5.4.

- R6: The created OpM Table 5.3.

6. Case study system

After we reached the milestone in the last chapter, we are going to define scenarios, evaluating the defined scenarios and concluding the procedure in this one.
A scenario is a actual characteristics from, for example, a use case. But scenarios in general describe a actual interaction by one or more users with the system. This scenarios are later converted in data flows that are added to the system model. The scenario created by using the ACM Table 5.4 and the OpM Table 5.3. We defined two scenarios for CoCoME. The basic structure for each scenario is: First, the scenario is briefly described. Then the specific data types for the scenario are identified. We then describe the behavior of the individual components in a scenario. Finally, the resulting data flow is shown.
After the generated data flows have been added to the system model, they are evaluated based on the number of covered information flow classes.
Finally, it is checked whether the number of covered information flow classes are sufficient. Then the process is concluded.

6.1. Excerpt of the system

First of al, the excerpt where both scenarios ar elocate dis brifely prsented. Both scenarios ar elocate din the *PickupShop* scenario from the CoCoME tech report [3]. In Figure 6.1

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

6.2.1. Description

This scenario is derived from the use cases described in the CoCoME tech report [3], more explicit the use case 13.
In this scenario, the *StockManager* is already authenticated and then requests a report for the purchased products of a customer. To identify the customer among all customers, the *StockManager* has access to the ID of the specific customer. The *StockManager* enters the ID in CoCoME, then the request is processed. At the end, the stock manager is presented with a full report for the customer. This report contains all purchased products of the customer.

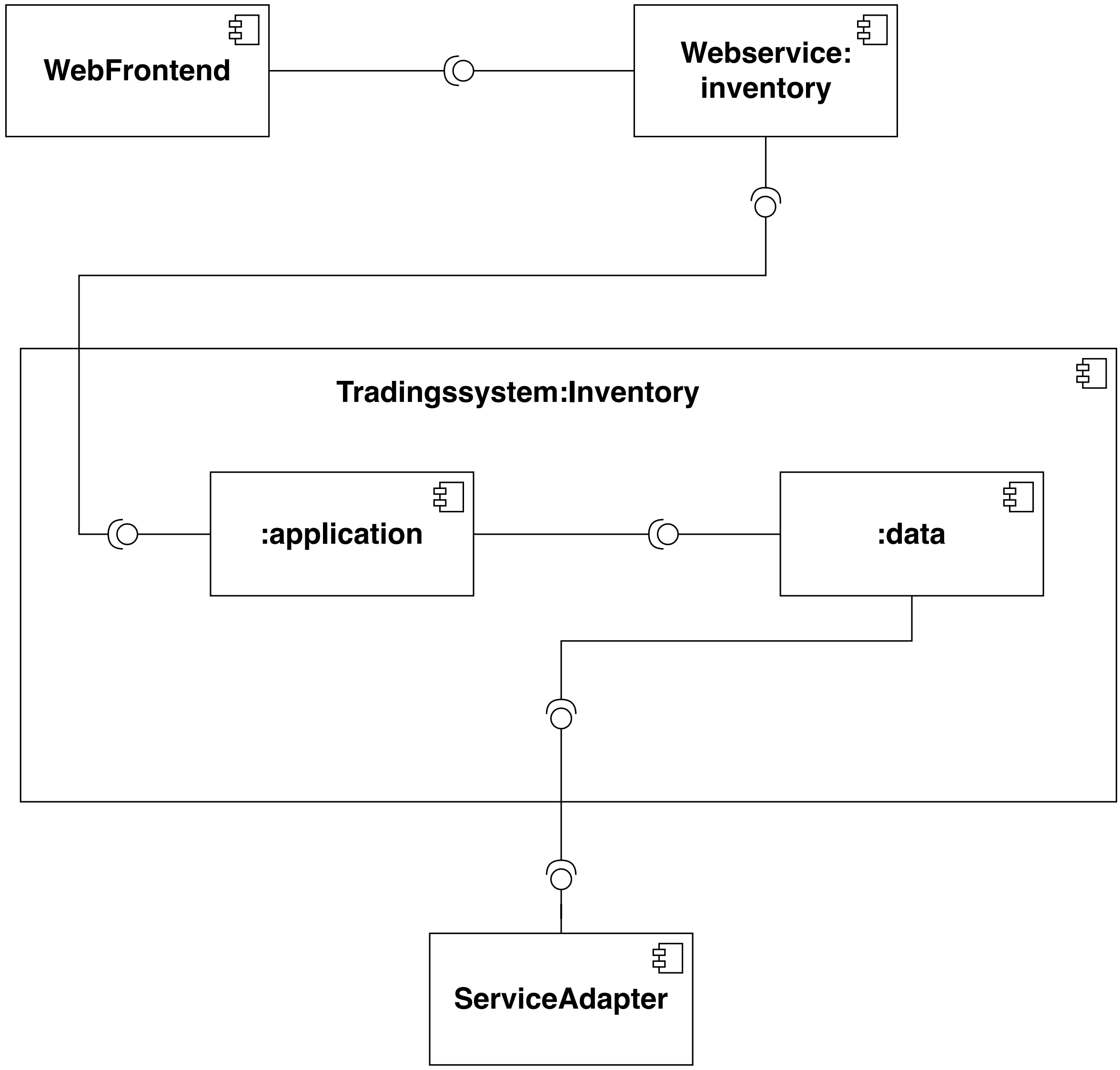

Figure 6.1.: Excerpt of CoCoME for which the described scenarios are defined.

ACM	Webfrontend	TS:inv:application:store	TS:inv:data:store
stockmanager	customer data : 4 account data : 3 p& s data : 2 system data: 4	account : 3 customer : 4 product & sales : 2	account : 3 Figure 6.1) customer : 4 product & sales : 2

Table 6.1.: An ACM showing the access rights for the case study system.

6.2.2. Extract the data types, access rights and types of data processing for the scenario

In this section, we will first determine what actual data will be used in each component. Next, we obtain the corresponding parts of the two matrices for access rights (ACM) Table 5.4 and data processing (OpM) Table 5.3.
As stated in the milestone, the data is divided into four classes: customer data, account data, product&sales data and system data. In the following enumeration for each data class the used data in the scenario is described.

- account data
  No data from this class is used in this scenario.

- customer data
  The ID from the customer is part of the scenario.

- product&sales data
  In the scenario two data types are used. First, the *IOrderEntries* are used. The *IOrderEntries* are a collection of all data that is generated after an order successful passed. The second used data type is the *Report* data type, which is a human readable string.

- system data
  This class provides the data that is used to create the queries to the underlying database.

Int he next step, we extract the corresponding parts of the ACM and the OpM. The ACM for this scenario is shown in Table 6.1 and the OpM is shown in Table 6.2. We omitted the Webservice component for the fact that it only transmits the data.

6.2.3. Component behavior

The excerpt of the system model of CoCoME in which the scenario is located is extended with *operations*. For each component one or more operations are defined. These model the data processing in this component. An operation is associated with one of the data processing classes in the OpM Table 6.2. Each operation has one or more inputs and one more outputs. For each input and each output the data types are known.

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

Table 6.2.: A matrix showing the data processing for each component. the abbreviate p& s data stands for product and sales data.

First, we describe the observable behavior of the scenario across all components and the data processing for each component. All operations are then combined in a data flow.

6.2.3.1. Observable Behavior

In the scenario, the *StockManager* interacts with CoCoME by requesting a report for a customer. In the following, the behavior of the system is described.
The *StockManager* enters the *ID* of a customer through the *Webfrontend* into CoCoME. The *Webfrontend* processes the ID by transmitting it to the *Webservice* component, which also transmitting the *ID* to the *Tradingssystem:inventory* component. The *Tradingsystem:invetory* consists of two sub components. First, the *ID* is transmitted to the *Tradingsystem:inventory:application:store*, where it is further transmitted to the *Tradingsystem:inventory:data:store* component. In this component a *query* is created by using the *ID*. This query is then send to the *ServiceAdapter*, where the database is requested. The query first selects the respective customer for the *ID*. Then it projects all *IOrderEntries*. The *IOrderEntries* represents an order for a customer. In the *IOrderEntries*, beside other data, which occur with an order, the purchased products are included. The list of all *IOrderEntries* is then transmitted back to the *Tradingssystem:inventory:data:store* and then on to the *Tradingsystem:inventory:application:store*. In this component, the list is then processed in a *Report*. This *Report* is then transmitted via the *Webservice* back to the *Webfrontend*, where the result is shown to the *StockManager*. The resulting report contains a list of all purchased products. In addition, to each entry the date of purchase is added.

6.2.3.2. Data flow definition

Once the general behavior of the scenario within the system has been described, the data flow is created. For each component, the operations and processed data that are used are identified and added to the current component. The operations are then linked in the correct order to complete the creation of the data flow.
The overview which data and which operations is shown in Table 6.3. In Table 6.3 the association between the components, the used operations and the data is shown. To give a brief explanation of how to read this table. In line 1, the *Webfrontend* only uses the operation **transmit** and data from a total of 2 classes, **customer** data and the **product&sales** data. From these classes the concrete data *ID* and *Report* are used.
The resulting data flow is shown in **??**.The data flow notation has been extended to include the corresponding component and operations for the processes. The data is still on the

Components	Operations	Data
Webfrontend	transmit	I: ID II: Report
Webservice	transmit	I: ID II: Report
TS:inv:data:store	rel. Algebra	IV: query
TS:inv:app:store	transmit alter data	I: ID II: List<IOrderEntries>

Table 6.3.: Overview over the used operations and the used data for the Scenario: StockManager requests a report for a customer. **I**

edges. Also nothing has changed for the entities.
In this scenario a violation for the customer's privacy is displayed. According to the defined access rights tow violations are modeled. First, the *StockManager* should not have access to the *ID* of a customer. And secondly, the *Report* is in combination with the *ID* security relevant. This data flow should normally prohibited by the system.

6.3. Scenario: Support employee requests information for an order

This scenario was not defined on the basis of an use case. We defined this scenario for various reasons. First, we wanted to define a counterpart to the already existing scenario (section 6.2). The scenario is located in the same system part and therefore operates on the same components. The idea behind this is to provide a violation and a non-violation. An data-based privacy approach my be evaluated by checking if the violation and the non-violation is identified correctly. Due to the fact that CoCoME is a rather minimal system for the goal the system aims for, we couldn't find further use cases where multiple classes of data and/or multiple roles are involved.

6.3.1. Description

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

Figure 6.2.: Use case diagram for CoCoME with the addition of the new use case.

6.3.2. Description of a new use case

We decided to add the scenario described in subsection 6.3.1 as a new use case to CoCoME. The embedding in the already present use cases can be seen in Figure 6.2. First we briefly describe the new role added to CoCoME, then the new use case is described.

6.3.2.1. 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 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 Table 6.3.2.1. 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

6.3.2.2. 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*

ACM	Web-frontend	TS: inventory: app:store	TS: inventory data:store	Pickup Shop
support employee	account : 2 customer : 1 product & sales : 2 system : 4	account : 2 customer : 1 product & sales : 2 system : 4	account : 4 customer : 4 product & sales : 4 system : 4	account : 4 customer : 2 product & sales : 4 system : 4

Table 6.4.: 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*.

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.

6.3.3. Extract the data types, access rights and types of data processing for the scenario

In this section, we will first determine for the scenario what actual data will be used in each component. Next, we obtain the corresponding parts of the two matrices for access rights (ACM) Table 5.4 and data processing (OpM) Table 5.3.
As stated in the milestone, the data is divided into four classes: customer data, account data, product&sales data and system data. In the following enumeration for each data class the used data in the scenario is described.

- account data
  No data from this class is used in this scenario.

- customer data
  The ID from the customer is part of the scenario.

- product&sales data
  In the scenario two data types are used. First, the *IOrderEntries* are used. The *IOrderEntries* are a collection of all data that is generated after an order successful

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

Table 6.5.: A matrix showing the data processing for each component. the abbreviate p& s data stands for product and sales data.

passed. The second used data type is the *Report* data type, which is a human readable string.

- system data
  In this class the data is located provides the queries to the underlying database.

Int he next step, we extract the corresponding parts of the ACM and the OpM. The ACM for this scenario is the same as defined in the new scenario and shown in Table 6.3.2.1, and the OpM is shown in Table 6.2. We omitted the Webservice component for the fact that it only transmits the data.

6.3.4. Component behavior

The excerpt of CoCoME is extended with *operations*. For each component one or more operations are defined. These model the data processing in this component. An operation is associated with one of the data processing classes in the OpM Table 6.5. Each operation has one or more inputs and one more outputs. For each input and each output the data types are known.
First, we describe the observable behavior of the scenario across all components and the data processing for each component. All operations are then combined in a data flow.

6.3.4.1. Observable behavior

In the scenario the *SupportEmployee* requests and receive a report for a specific order of a customer. In the following, the behavior of the system is described.
The *SupportEmployee* enters the *ID* of the customer and the *ID* of the order. Then the tuple is transmitted via the *Webservice* component and the *Tradingsystem:inventory:application:store* component to the *Tradingsystem:inventory:data:store* component. In this component the tuple is utilizied to create a query that is transmitted to the *ServiceAdapter*. In this component the query is executed and the result is transmitted through the *Tradingsystem:inventory:data:store* component to the *Tradingsystem:inventory:application:store* component. There the result of the query is processed into a *Report*. This *Report* is then transmitted via the *Webservice* component to the *Webfrontend*, where it is shown to the *SupportEmployee*. The resulting report contains meta data in addition to the ordered products. This meta data includes, for example, delivery day, the used payment method, etc.

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.

6.3.4.2. 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 6.3.

6.4. Evaluation of the scenarios

After all data flow are defined and added to the system, the second part of the evalaution for the case study is conducted in this section. The overview is shown in Figure 6.4.

6.4.1. Goal: Evaluation for the created case study

The goal is to evaluate the case study. As shown in Figure 6.4 the evaluation of the case study is split into two parts. In this we will only focus on the highlighted plan, the evaluation of the defined scenarios.

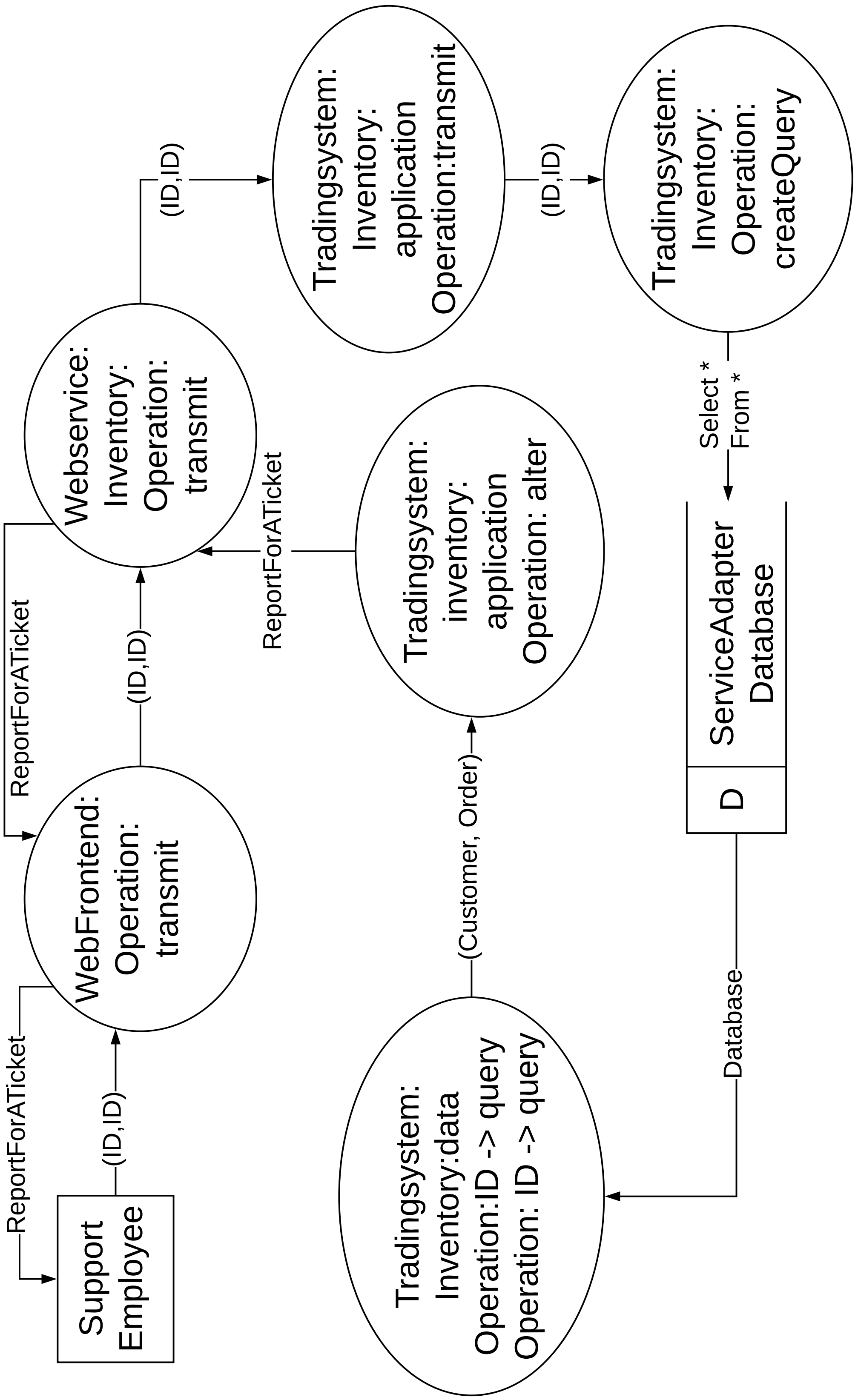
Support Employee
(ID,ID)
ReportForATicket
WebFrontend: Operation: transmit
(ID,ID)
ReportForATicket
Webservice: Inventory: Operation: transmit
(ID,ID)
Tradingsystem: Inventory: application Operation:transmit
(ID,ID)
Tradingsystem: Inventory: Operation: createQuery
Select * From *
D
ServiceAdapter Database
Database
Tradingsystem: Inventory:data Operation:ID -> query Operation: ID -> query
(Customer, Order)
Tradingsystem: inventory: application Operation: alter
ReportForATicket

Figure 6.3.: The resulting data flow for the scenario: Support employee requests information for an order.

Figure 6.4.: Overview for the evaluation for the Case study system

6.4.2. Question: Is the case study usable for a data-based privacy analysis?

To verify if the created case study are usable for the evaluation of a data-based privacy analysis, it is checked which information flow classes the created case study covers. We relied on the problem statement *Non-influence* [7]. Non-influence combines two sub problem statements, *Non-interference* and *Non-leakage*. Non-inference describes that no role may acquire information from a role that inherits a higher security level. In the course of a program flow, user with a high security level and user with low security level inserts data in the system and receive outputs from the system. If there is no interference in the system, inputs from high level user have no effect to the outputs of low level user. The effects of *Non-inteference* can be expressed by three infromtion flow classes: **Illegal information flow**, **Direct information flow between roles** and **Information flow form high to low**.
Illegal information flow describes that a role can receive information for which it is not authorized.
Direct information flow describes the flow of information between two roles for which is not cleared to either receive or transmit.
Information from high to low describes flow of information, even if not intended, from a higher security level to a lower security level. *Non-leakage* describes the issue that it is not observable what specific actions are taken place by a system. This can be expressed by the information flow class **Observable information flow**.
The metric is the number of covered information flow classes in the case study. The evaluation is conducted in a checklist manner.

6.4.2.1. Number of different covered information flow classes

We evaluate the variety of covered information flow classes for the defined scenarios in the case study. We defined four information flow classes to cover *Non-influence*(*non-interference + 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 information flow between roles. W

Illegal information flow The first scenario covers 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* 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.

Data flow	fulfilled?
Illegal information flow	✓
Information flow from high to low	✓
Direct information flow between roles	✗
No observable information flow	✗

Table 6.6.: Resulting checklist for the evaluation of the case study.

Information flow from high to low In the defined scenarios, for the role involved, data from different security levels is present. The data associated also flows to lower levels. This information flow class is also covered.

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.

Observable information flow This information flow class is not modeled in the scenarios.

6.5. Discussion

In this section, we are going to discuss the single design decision that were made during the creation of the case study. Further, we discuss we if the process can be concluded.

7. Evaluation

After we evaluated the created case study in chapter 5 and ??, in this chapter the procedure (chapter 4) and the used modeling language (PCM [10]) are evaluated. The evaluation are also based on a GQM-plan section 2.4. The plan for the whole evaluation can be seen in Figure 7.1.
First, we conduct the evaluation for the applicability of the presente dprocedure, then the expressibility of the modleing language. Afterwards, we summarize the results for the created case study. To conclude the chapter, we discuss the *threats to validity* ([12]).

7.1. Goal: Applicability of the introduced procedure

The first aspect we are evaluating is the applicability of the introduced procedure to create case studies from software systems. We evaluate the applicability to verify if its possible to create case studies that later may be used to evaluate approaches for a data-based privacy analysis. The examination if the quality of the created study is sufficient is a separate part and has to be done on a case-by-case basis.

7.1.1. Question: Is the introduced method applicable to an actual system?

To verify if the goal is achieved, we check if the introduced procedure is applicable to a software system that abstracts the selling process and warehousing of a supermarket group. The metric to answer the question is a successful application to an actual system.

7.1.1.1. Application to the CoCoME system

The system we chose to apply the method on is CoCoME. We applied the procedure not the whole CoCoME system but only to an excerpt of it. The application to the excerpt and resulting case studies is conducted in chapter 5 and ??.
We successfully applied the procedure to an actual system and created a case study. Therefore, we conclude that the procedure is applicable.

7.2. Goal: Expressiveness of data-centric PCM

In this part, we evaluate the modeling language that is used for the case study. To model the case study, technically speaking, it have to be possible to add two extensions to the system model. First the data flows and ,Secondly ,the defined ACM.
We chose to use data-centric PCM as the modeling language By default it is not possible with data-centric PCM to extend models by data flows and an ACM. Therefore we went

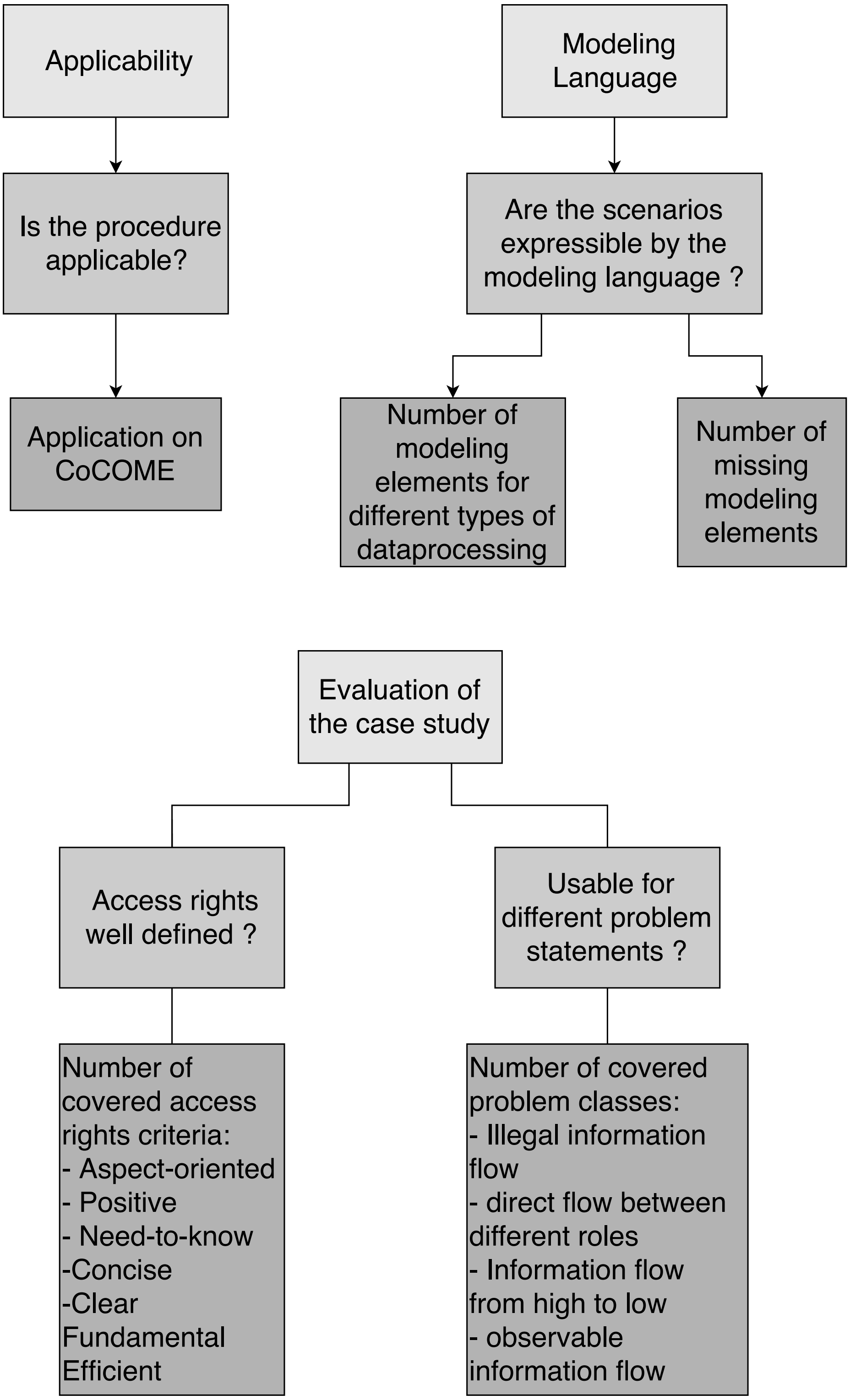
Applicability
Is the procedure applicable?
Application on CoCOME
Modeling Language
Are the scenarios expressible by the modeling language ?
Number of modeling elements for different types of dataprocessing
Number of missing modeling elements
Evaluation of the case study
Access rights well defined ?
Usable for different problem statements ?
Number of covered access rights criteria:
- Aspect-oriented
- Positive
- Need-to-know
-Concise
-Clear
Fundamental
Efficient
Number of covered problem classes:
- Illegal information flow
- direct flow between different roles
- Information flow from high to low
- observable information flow

Figure 7.1.: The GQM plan for the three parts of the evaluation. Each part is divided into 01

for the meta-model extension for data-centric PCM from Seifermann [14]. So, this part of the evaluation will mainly focus on the meta model extension for data-centric PCM.

7.2.1. Is the created case study expressible with data-centric PCM ?

To verify, if the case study is expressible with data-centric PCM, we ar eusijgn two metrics. First we measure which elements are available and then which elements are missing.

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

The main focus is to verify, if the created case study is expressible with data-centric PCM. We check if it is possible to model all defined operations in the OpM. This is also done in a checklist manner. We defined four operations for the case study in the OpM: transmission of data, alternation of data, operations for relational algebra and operations to model user interaction.

Transmission data This operation describes if components transmit data. An operation for this available in the meta model extension.

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, the splitting of data, like lists, in the single data, etc. With the alternation of data it can happen that the class of data is changed and thus the applying access rights change. To describe the change of the applying access rights an operation is available in the meta model extension.

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 extension.

I/O- operations Operations to model the user interactions are also available.

7.2.1.2. Number of missing elements for the different

Currently it is not possible to store the ACM in the same system model. It is possible to define access rights for each data in each component for each role. First, for each data a new access right has to be defined and , secondly, it is not or only with disproportionate effort possible to swap out the access rights or generating the ACM form the model. For the fact, we created our case study *aspect-oriented* (see []) the current modeling contradicts the concept. Therefore, a model element for the ACM is missing.

Meta model	possible ?
relational algebra	✓
I/O operations	✓
Transmission of data	✓
Change of access rights	✓
Alternation of data	✓
ACM in system model	✗

Table 7.1.: Summary for the evaluation for the modeling language.

Access rights		fulfilled?
Specification	Aspect-oriented	✓
	Positive	✓
	Need-to-know	✓
Comprehensibility	Clear	?
	Concise	?
Implementation	Fundamental	n/a
	Efficient	n/a

Table 7.2.: Summary for the evaluation of the defined access rights.

7.2.1.3. Summary: expressiveness of data-centric PCM

Now that the two metrics have been collected, we summarize them. The summary of the both metrics is shown in Table 7.1. As it is shown, the only flaw is that the ACM cannot be included in the system model.

7.3. Evaluation of the created case study

The evaluation of the created case study is split into two parts: the quality of the access rights and the covered information flow classes. Each part is already conducted in Here we shortly summarize the results of the single evaluations.

7.3.1. Access rights

Further details on this evaluation can be seen in Evered and Bögeholz [2]defined seven criteria to review the quality of defined access rights. All in all, the criteria can be divided in three categories: Specification, Comprehensibility and Implementation.
Specification criteria review the definition of the access rights. *Comprehensibility* criteria mainly reviews the notation and ho clear each is defined. *Implementation* criteria reviews the embedding of the access rights in the implementation.
The result of the evaluation are displayed in Table 7.2.

Data flow	fulfilled?
Illegal information flow	✓
Information flow from high to low	✓
Direct information flow between roles	✗
No observable information flow	✗

Table 7.3.: Summary for the evaluation of the covered information flow classes of the case study.

7.3.2. Covered information flow classes

Further details on this evaluation can be seen in In teh evaluation we examine which different information flows are covered by the created case study and may later be checked in a data-based privacy analysis. To define different information flow classes, we relied the problem statement *Non-influence* [7]. This problem statement includes four different information flow classes: illegal information flow, information flow from high to low, direct information flow between roles and observable information flow. The summary of the covered information flow classes is displayed in

7.3.3. Summary for the evaluation of the case study

We achieved to cover three out of seven criteria (43%) for good access right. The evaluation of the criteria in the *Comprehensibilty* class would have exceeded the scope of the thesis. So we cannot make a qualified statement for these criteria. The criteria associated with *Implementaation* class may be omitted in a purely conceptual case study. For instance, this may happen when a complete system is modeled but not implemented yet. We achieved all of the *Specification* criteria with our defined access rights.

All in all, we say the number of applicable criteria are dependent on the goal for that the specific case study is created.

We also achieved to cover 50% of the information flow classes. We coul not achieve to cover all classes due to time constraints of the thesis and therefore missing scenarios.

7.4. Threats to validity

In this section we discuss possible threats to the validity of the evaluation. The general description for this threats are published by Runeson and Höst [12] The possible threats are divided into four categories: internal validity, external validity, construct validity and conclusion validity.

Internal validity Internal validity describes if the evaluation supports the conclusion.

External validity External validity describes that the results can be generalized.

Internal Validity	External Validity	Construct Validity	Conclusion Validity
II, III	I	II	III

Table 7.4.: Threats to validity for the thesis.

Construct validity Construct validity describes that the measurements for the construct under investigation are correct. In particular, this means that the measurement is not influenced by other factors.

Conclusion validity Conclusion validity describes that evaluation is based on *reasonable* data, for instance statistical data, to minimize or eliminate the subjective view of the researcher.

7.4.1. Analysis of the evaluation

We analyze the three parts for the evaluation carried out and identified aspects that could pose a threat to the validity of the results. The aspects are: the procedure, the defined access rights and the covered information flow classes. In Table 7.4 a categorization for the aspects is shown.

I: Procedure The introduced procedure has only been applied to an excerpt of a system. This endangers the *external validity.* The procedure works for CoCoME but this cannot be ensured in general.

II: Access rights Three out of seven criteria were evaluated for different reasons. This could be a threat to the *Internal validity.* A case study generates qualitative data and therefore unchecked categories may backfire. So we could not say with certainty that our evaluation supports the results. Further, the

III: Information flow classes The information flow classes were defined based on one problem statement. Due to that fact, the information flow classes are a threat to the *conclusion validity* because the impact of the researcher. Other information flow classes could be defined which probably give more insights. Further, the *internal validity* is endangered. As with access rights, qualitative data is used and missing categories can alter the evaluation.

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 analysis. To allow such analyses, we defined data flows for a system. Further, we created a process to create case studies from software systems and applied the method to CoCoME. While in the process of application to CoCoME, we found vagueness in CoCoME. This vagueness does not happen by accident. Another benefit of the method, beside the creation of a case study, is to detect vague definitions and add definitions for them . First of all, the roles were rather well defined, but missed the last bit of precision. For example, the role of StockManager was mentioned, but there were too few use cases to derive tasks and provided/required data. Also for some data used in CoCoME exact definitions 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 encounter problems with an order they can issue a ticket. The support employee handles the tickets to solve the problem.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 investigated information flow classes. We then validated the method described through the application to CoCoME. 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 on which types of systems the method should be applied. At last we evaluated the used modeling language PCM [10]. 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.
At last, CoCoME was not the perfect system for this thesis. First of all, CoCoME is in constant development and therefore is more a proof of concept than a full system. Further, before the addition of the PickupShop, CoCoME was a closed system with a finite number of users. With the addition, CoCoME became an open system, therefore we chose the PickupShop as our system to perform the procedure on.

8.2. Benefits of our work

The main goal of the thesis was to introduce a method to create case studies that may be used to evaluate approaches for 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. Ready-to-go means that the case study may not be perfect but may be used to evaluate data-based privacy analysis approaches in the current state. 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. The process benefits motivated software architects, who want to perform a data-based privacy analysis in an early stage of the design process to ensure compliance for the system.

8.3. Future work

This section presents possible approaches for future work. The discussion of future work will be divided into the three main aspects of the thesis: the method for creating a case study, the resulting case study and its evaluation, and at last the used modeling language PCM..

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 class, one should define more scenarios to allow a more deep analysis. Another possible approach for the future work is the analysis of access rights according to various criteria not yet covered. We could not cover, for example,fundamental and efficient. As future work, we propose to include access rights in the current implementation. This makes it possible to check whether the access rights have been added directly in the code and whether the checks of the access rights do not cause too much overhead. The remaining access rights were not checked due to time constraints and the lack of running code with built-in access rights.

As a long-term work, we propose to perform analysis for the created case study in this thesis. One should define more not yet covered information classes . These additional information classes are added to allow a wider spread more in depth evaluation. Another possibility is to analyze the case study created in this thesis by using it to validate a data-based approach privacy analysis approach. The cases for additions or developments are not dealt with in this thesis.

8.3.3. Meta model extension

As future work, we propose to extend the meta model of PCM even further to allow the storage of the ACM in the same model.

Bibliography

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[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.

[15] Bart van der Sloot. "Where is the Harm in a Privacy Violation? Calculating the Damages Afforded in Privacy Cases by the European Court of Human Rights". In: *JIPITEC* 8.4 (2017), pp. 322–351. ISSN: 2190-3387. URL: http://nbn-resolving.de/urn:nbn:de:0009-29-46414 (visited on 09/13/2018).

[16] Donna M. Zucker. "How to Do Case Study Research". In: *Teaching Research Methods in the Social Sciences* (2009).

A. Appendix

A.1. First Appendix Section

Github URL for PCM models for the defined scenarios.

- **stock manager requests the report for a customer**
  Gihub URL: `https://github.com/julhin/DataProcessing/tree/dataprocessing_withoutAssemble`

- **Support employee requests information for an order**
  Github URL: `https://github.com/julhin/ModelsUC14`