Access Control Verification in Software Systems

Bachelor's Thesis of

Julian Hinrichs

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

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

14. May 2018 – 14. September 2018

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

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

PLACE, DATE

. . . . . .

(Julian Hinrichs)

Abstract

English abstract.

Zusammenfassung

Deutsche Zusammenfassung

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motiviation

As Software systems become more connected and complex, security in general and privacy particularly become more important. First of all, the financial loss through possible leaks are immense. Also the trust loss from the customer is maybe the greater issue. All in all, privacy is a primary design goal for all commercially used systems. Privacy, on the other hand, is a non-functional requirement and it is difficult to ensure compliance. For this reason, Seifermann published the idea of an approach, how privacy can be reviewed. The motivation for Seifermann's paper was to introduce a data flow analysis to ensure privacy on an architectural level. It was not the first approach in this direction. Two main approaches in this direction should be. First SecureUML, which is a lightweight extension of the UML meta model. SecureUML goal is to model a already secure system at the end of the design process. The other approach is SecureUML. SecureUML is also a lightweight extension of the UML meta model. The goal of SecureUML is to model all security relevant data and attach these data at the fitting model elements in a concrete UML model. To verify security criteria UMLsec uses an attacker model to test the modeled system. Seifermann's approach should be settled between the two approaches. It is also uses a lightweight extension of an architecture description language (ADL). Further, the approach provides also a security analysis on an architectural level.The problem all three approaches tackle, is that currently security analysis is either conducted directly in the written code or the security relevant model elements are stored in a separate model, which leads to inconsistency. Seifermann's approach extends the meta model of Palladio Component Model (PCM) instead of UML. Another difference is that, the approach also includes runtime configuration and access control. To evaluate security criteria constraint logic programming is used. All in all, the Seifermann's approach introduce data based privacy analysis on an architectural level. Currently, the approach has not yet been applied to a realistic system. The approach will be evaluated using a case study. In this thesis, a case study will be carried out, which will later also validate the approach. The case study will be conducted on the Common Component Modelling Example (CoCoME). I describe a procedure how to conduct a viable case study and also apply this procedure on CoCoME. At the end, the case study will be evaluated to determine if it is suitable for the approach. As already mentioned, the meta model has been extended by PCM, which will also be part of the evaluation. To evaluate my contributions, I will use the Goal-Question-Metric (GQM) approach.

1.2. Contributions

~~My thesis contributes to Seifermann's approach~~. I describe a procedure to create a viable case study. The case study is later used for the evaluation of Seifermann's approach. Further, I applied this procedure to the already existing system CoCoME. For this, I added or completed messing or incomplete definitions. For the case study it was necessary, that I extended the current models for CoCoME with the meta model extension provided by Seifermann.

1.3. Outline of the thesis

~~to be used~~ The thesis is organized as follows. ~~All in all, the thesis consists of six chapters.~~ The first chapter defines the foundations. The second chapter presents the procedure to create a viable case study. The third chapter applies the procedure to CoCoME. The fourth chapter evaluates the created case study. The fifth chapter gives an outlook on future work.Further,t he chapter compares my contribution to related works. The six and last chapter draws a conclusion of the thesis.

2. Foundations

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

2.1. Terminology

2.1.1. Critical data

The term *critical data* as used in this thesis, describes data which should be protected by the system. Mostly these data are personal related informations like name, address, credit card and so on.

2.1.2. Data flow

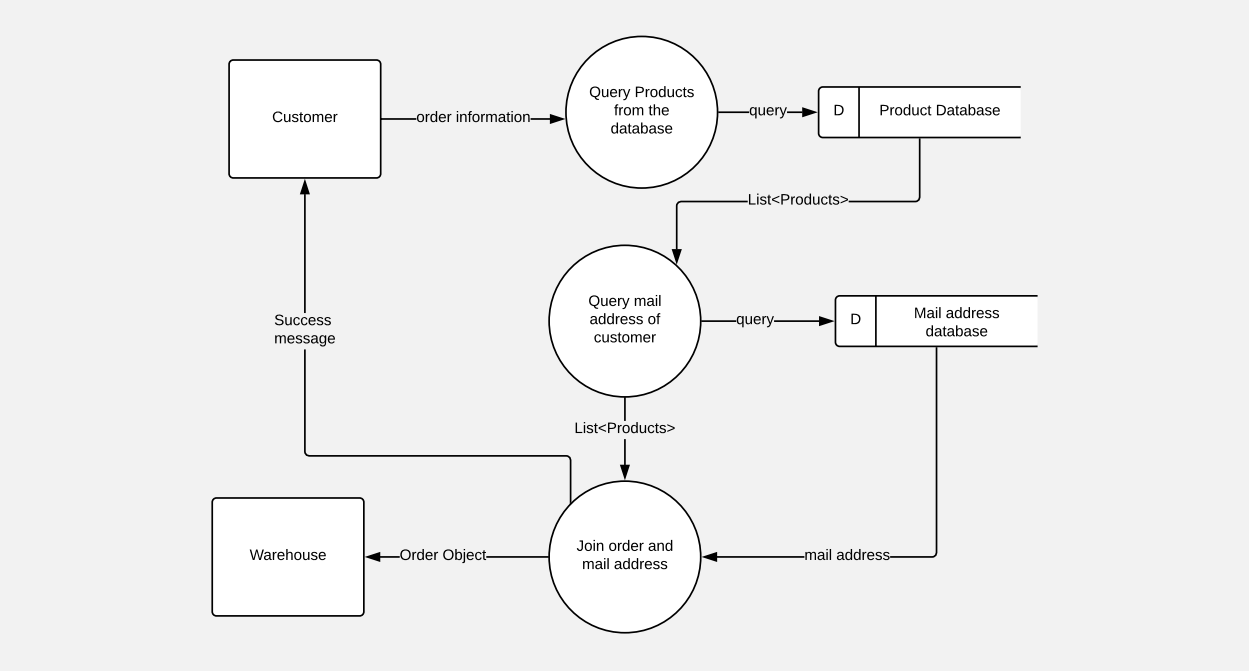


Figure 2.1.: Simple data flow diagram for linking an order of different products to a mail address

There are many definitions for data flows, even in the field of computer science. So I will give a short definition, how the term *data flow* is used in this thesis.

	File A	File B
User 1	r,w,x,a	r
User 2		r,x

Figure 2.2.: A simple table for a software system with two roles and two files. The elements of the matrix shows the different permissions for each user

Data flow describes the way of a piece 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 alters or passes 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.
Figure 2.1 shows the simple process of ordering products and liking 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.

2.1.3. Role Based Access control

Role based access control (RBAC) is an approach in computer systems security. This approach defines roles, data and permissions between role and data. The permissions defines under what circumstances a role may access data. The subsection 2.1.3 will clarify the basic idea of the approach. The subsection 2.1.3 defines two files and user and 4 permissions. The letter **r** stands for a read operation, the letter**w** stands for a write operation, the letter **x** stands for an execute operation and the letter **a** stands for an append operation. As you can see in the first row of subsection 2.1.3, user 1 has all rights to change the File A and can only read the File B. In the second row, you can see that user 2 has no rights to access File A and may read or execute the file B.

2.2. Palladio Component Model

2.2.1. Component based systems

2.3. Data Centric PCM

2.3.1. Concept

Component based systems are software systems which consist of different components. A component is defined as unit in the system which provide and/or require interfaces. Each

component in systems is designed for a specific task, like serialization, connection to a database, handling the user input and many more. These different component communicates via interfaces. Also a component can consist of several smaller components. A simple example is shown Component based systems are software systems which consist of different components. A component is defined as unit in the system which provide and/or require interfaces. Each component in systems is designed for a specific task, like serialization, connection to a database, handling the user input and many more. These different component communicates via interfaces. Also a component can consist of several smaller components. A simple example is shown The Palladio Component Model (PCM) is an architecture description language (ADL) , that allows more than only modeling a system (. The resulting model consists of four different models, each of them encapsulate their specific design knowledge and the models build on each other. After the system is complete, an architect may compute different predictions how the system will perform when it is deployed. The predictions are for cost, reliability, performance and maintainability. In the following, I will explain the individual models and their encapsulated design knowledge in detail.
First of all, for each model a role inside the PCM is assigned, which will be explained in detail later. Each of this roles conserve their design knowledge in a specific model which is used by the next role. The hierarchy of the different roles , from top to bottom, is:

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

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

- **System Deployer**
  The system deployer decides how the system is distributed between the available resources. This role creates two resulting models. First, the *resource model*, in which the resources characteristics, for example transfer rate. Second, the *allocation model*, in which the allocation of the different resources to the different parts of the system is conserved. These models are used by the domain expert.

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

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

2.3.2. Meta model extension for data centric PCM

3. Methods

in this chapter I am going to explain my methods for the creation of a viable case study, that may be used to perform an data based privacy analysis. In this chapter, I am going to describe the procedure and with it all necessary steps. Further, I will present an evaluation to verify the general goal if the case study is sufficient to perform a data based privacy analysis. The layout of this chapter is as follows.
The chapter consists of six sections. The first section gives a brief overview over procedure of creating a viable case study. This procedure consists of five concussive steps. In the four sections, each step is explained in detail and in the final section the evaluation is explained in detail.

3.1. Creation of the case study

In this section, I describe which steps has to be taken to construct a viable case study. This involves five successive steps. In the first step, I will look into the basic requirements for a case study. I defined five basic requirements for my case study. A system which will be studied with a case study, have to fulfill these requirements. This fulfillment is needed so a case study can be built from the system. The first requirement is that the system is modeled as a component based system. The second requirements is the definition of user roles for the different types of users. The third requirement is that critical data is present and transferred in the system. The fourth requirement is that are access rights are defined between the roles and the (critical) data. The fifth requirement is , that it is possible to model data flows in the modeling language. In the second step, I analyze the system. I am looking for critical parts. This critical parts are components that process or transmit critical data. In the third step, I take a step backwards and take a look at the current state of the case study. It may happen, that some requirements or parts of requirements are not well defined or even missing. In the fourth step, all missing or partly defined requirements are defined and added to CoCoME. Also, in the fourth step This is done to build a viable case study. In the fifth and last step the created case study is evaluated based on two aspects. The two aspects are:

- Is it possible to create a case study with the chosen modeling language ?
- Is the created case study suitable for data-based privacy analysis?

3.2. Requirements for a case study

In this section, I will take a closer look onto the first step for creating a viable case study in CoCoME. The first step is the definition of all necessary requirements to create a viable

case study. All in all, I defined five necessary requirements. The first two requirements (the system is modeled as a component based system and the modeling language is able to model data flows) assure that the system and modeling language have the required elements to model and store necessary parts for the case study. The last three requirements (definition of data types, definition of roles and the definition of access rights) defines the rules on how the different users may interact with the different data types in the system. This is necessary to enable a security evaluation of the system.
The first requirement is that the system is modeled as a component based system. 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.
The fifth and last requirement for the case study is an extension to the meta model of the architecture description language (ADL). This meta model extension have to enable two aspects. Firstly, the modeling of data flows for a given system. Secondly, that modeled data flows are attached in the component model. The second aspect is necessary to reduce inconsistency between the modeled system and the modeled data flows for the system. If the data flows and architecture aren't attached it may happen that the data flows in the case study are outdated. The first aspects is a fundamental part of the case study, because it uses the data flows to express the processing of data on an architectural level. This is later used in an evaluation to verify if the modeled system fulfill the access rights.
The third requirements for a viable case study are the existence of critical data in the system. If there is no critical data in the system, the entire case study that was created to verify the protection of critical data is pointless. Critical data in the context of the case study mostly consists out of personal informations about a customer. These personal informations includes mainly names, addresses and bank details(e.g credit card number). Sometimes one can argue, that the purchases of a customer may also considered critical data. From the critical data a malicious user/attacker may derive valuable information about the personal life of a customer e.g. marital status, income, living situation, just to name a few. Also, it is possible to harm the customer in different ways. For example, if an attacker acquired the credit card details from a customer, there is possible financial harm for the customer. The goal of a system should be to protect this critical data from malicious use. To verify the presence or absence of this protection the case study is created.
The second 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 one type of user. Roles are used to model the different users that are going to interact with the system. Different roles uses on different data inside the system. Mainly customer role enters critical data (name, address, credit card details, ...) in the system, that have to be protected. To protect the critical data it is necessary, to assign each role a privilege level.Privileges describe which data may be accessed by the assigned role.

Another reason for roles, is that the case study is going to model role based access control. In this approach roles are an fundamental part. I decided to use role based access control, because a lot of security relevant problem statements can be mapped to it. Also, roles are used to define scenarios in the system. These scenarios model exemplary data flows in the system that can later be evaluated by a different approach.
The fourth requirement to create a viable case study is the definition of access rights for the system. These access rights control access from the single roles to different types of data. The defined access rights contains various rights to access data in the system. To clarify this, I will give two examples. For example, the access rights determine which role ~~data~~ can read or write on critical data. Likewise, it is defined which roles may execute a query to request the database. Access rights also define which combination of data a role is allowed to access. The access rights are stored in a matrix, the so called access control matrix(ACM). The ACM defines for each role the respective rights and is an important part of the case study. The ACM is the source of informations in terms off for security details. In the later evaluation, the ACM is used to decide if a specific data flow contains errors.

3.3. Relevant components that include critical data transmission

In this section, I will take a closer look on the second step for creating a viable case study. In this section, the step-by-step process for identifying 'relevant' components is described and explained. The goal of this step is to provide an in-depth understanding an excerpt or the whole system. Further, in this step the data flows are added to the architecture model.
The process is separated in four steps. First of all, components are identified in which and how security-relevant data is processed. In the second step, for each role it is identified, which data is used in which (security relevant) context. In the third step, scenarios are defined, which reflect a representative interaction with the system. Int the last step, data flows for each scenario are added to the architecture model.
In the first step, the component diagram is analyzed. For each component it is identified which data type is processed by the component, especially predefined security relevant data. For each component, it is also defined with which operation the data is processed. The operations are heavily dependent on the meta model extension. Some of the most common operations are load, store, create and transmit. Each operation has input data and output data to model the impact of the processing and to model the changes throughout the system.
In the second step, for each role it will be defined which data is used by this specific role. This is done by analyzing the use cases and respective sequence diagrams for the specific role. use cases are used to model the interaction of a specific role with the system. Sequence diagrams models the interaction for a specific use case and specific circumstances. Also the different data objects and the communication between them is modeled in sequence diagrams. After the analysis for both diagrams is done, the result is a 1:n mapping for each role to one or more components. For each relation the used data is added.

In third step enabled with the information gained from the previous ones, it is possible to create scenarios. Scenarios are often a more precise description of a use case (but this is not a must). Scenarios are more precise description of an interaction with the system. Also the flow of data is described within the scenario. Fitting scenarios are created from use cases that process critical data. This means, when a use case processes security relevant data it is a good hint to model this use case as an scenario for a later evaluation.

3.4. Current state of requirements

~~In this section, I will take a closer look on the third step for the creation of a viable case study. The description for this step states that it now the time to~~ take a step back and review the current state of the requirements.The best case is, that all requirements are well defined, the use case and sequence diagrams are clear and well modeled. But this is not the usual case for the most systems (especially CoCoME).The usual state after the first two steps is that not all requirements are met. In the usual case it happens that some of the requirements not well or fully defined. It may even happen that some requirements aren't defined at all. The review of the current state is an important to outline which parts of the system are 'blurry'. Blurry in the context of the thesis means that some requirements (e.g roles, access rights) are not clear defined. To give an example, it may happen that roles are just mentioned and a definition is missing. A throwback to the missing may be that the jurisdiction of the role is not clear and therefore it is hard to define the tasks for the role in the system. this review of the current state may also show parts where the model is in the current not well defined. This may lead to an improvement of the model. The case study is usually conducted in an early stage of the development process and may show the next working points for the system. In the following enumeration, I show some important aspects for each requirement.

- data
  ~~For the data, there are some aspects one should look out for.~~ All data is already defined or missing some important data types ?

- roles
  ~~When it comes to roles there are a lot of different aspects that are important.~~ First it is important to know the exact jurisdiction of each role to comprehend the access rights stored in the ACM. The jurisdiction also defines the tasks of a role in the system which is also important for the understanding the defined access rights.

- component model

- meta model extension
  For the meta model extension it is important, that all the model elements for the addition of the data flows are present. Another important aspect is, that all desired operations are provided by the meta model

- ~~ACM~~
  ~~For the ACM some possible problems arise.~~ First, it may happen that the ACM doesn't fit the uses cases of the system. Second, It also may possible that the use cases and the defined ACM are complement each other.

3.5. definition of CoCoME extensions

~~In this section I am going to take a closer look on the fourth step for the creation of a viable case study.~~ Based on the previous step in which the current state of the requirements is checked, in this step the concrete extension for the system have to be defined. After the previous step in section 3.4, where the fulfillment of all requirements for a viable case study was checked, it it always depends on the specific case study, which requirements are missing. It has to be decided on a case-by-case basis which concrete extension have to be defined. Therefore, here are just a short overview of the most common extension for missing requirements. Note that this isn't a full overview for each and every case, just an overview for the cases I encountered during the thesis. If the system architect unavailable or it is not fully clear yet, how the missing requirements are modeled in the final design, the only option left is to define them to the best of the current knowledge. An important point is to keep always the current documentation/goals for the system in mind. The documentation is a good source to figure out the missing definitions. Keep in mind that the definitions dependent on the case study.

The main part of this step is to transform the created scenarios, defined in the second step (see section 3.3), into data flows that are defined and later added to the model. To realize this transformation, the meta model extension is used that was defined in section 3.2. I will provide a procedure how to add data flows to the model. This is done step-by-step. The defined scenarios serve as the basis for the transformation. This scenarios were defined in section 3.3. 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 reduce model. In the second step, the reduced model is analyzed. The aim of the analysis is to find out which interfaces are used for communication between the components. After that the used data types are identified. The third and final step is to define operations for the processing of data in the different components. This includes to determine when data is just transmitted, when data is modified and so on. These definitions added to the reduced model. this addition is done by using the meta model extension.
This procedure is repeated for each scenario. In the end all scenarios transformed to data flows. In a final step each reduced model is added back to the originally component model.

3.6. Evaluation of the case study

~~In this section I am going to take a closer look on the fifth and last step step for the creation of a viable case study.~~ In his step the evaluation for the case study is presented. The evaluation aims to verify if the created case study is viable. The evaluation follows a GQM

plan. ~~The approach is was introduced in~~ The idea of a GQM plan is to split the evaluation into three parts. First, a goal for the evaluation. In the second step, questions are defined. The answer to this question indicates if the defined goal is reached. In the las step, metrics are defined. These metrics indicate if the question was answered.
I am applying now the the GQM plan to my case study for verifying it. The evaluation will be divided in two parts. Each part aims to verify a different aspect of the case study. The first part is relatively straight forward. It aims for the chosen modeling language. The goal is to evaluate if the case study can be realized with the chosen modeling language. The second part aims to verify if the case study may be used for an evaluation of data based privacy analysis. Two questions arise in this context and each will be measured with a metric. In the following each part will embedded in a GQM plan and described in detail. The first part of the evaluation aims to check whether the case study can be used with the chosen modeling language. A GQM plan for this part of the evaluation looks like this:

1. **Goal**: Verify if the case study is usable with the chosen modleing language
2. **Question**: Are the elements of the case study expressible with the chosen modeling language ?
3. **Metrics**: (i) Numbe rof meta model elements present to describe the case study
   (ii) Number of missing meta model elements and number of misused model elements

To clarify the term 'misused'. A meta model extension is created with a goal for each meta model element. It may happen that for a specific definition in the case study no meta model element is avaible and another element is used, which wasn't designed for this case. This element is counted as an 'misused' element in the metric. After the second part is embedded in a GQM plan, it looks like this:

1. **Goal**: Verify if the case study can be used for an data based privacy analysis
2. **Question**: Are predefined benchmark criteria are met ?
3. **Metric**: *curently emoty will be filled in at a later stage*
4. **Question**: Is the case study usable for different problem statements ?
5. **Metric**: Number of covered problem classes.

In order to arrive at a conclusive result, the case study is evaluated for each of the three parts.

4. Analysis of CoCoME

4.1. CoCoME overview

4.2. Application of the method

4.2.1. Requirements for a case study in CoCoME

4.2.2. Identification of relevant parts in CoCoME

4.2.3. Analysis of the current for CoCoME

4.2.4. analysisCoCoME

4.3. Summary of shortcoming in CoCoME

5. Case study system

5.1. Introduction to the case study system

5.2. Description of Scenarios

5.3. Definition of use case

5.4. Access control

5.5. Behavior of the single components

5.6. Description of the usage model

6. Deprecated

In this chapter I am going to present my case study from which I derived my approach. The approach is discussed in detail in **??**.
My case study is conducted in the CoCoME system. In this chapter, I will give a detailed overview over CoCoME and my case study.

6.1. CoCoME: Introduction

CoCoME is short for Common Component Modeling Example. For the sake of simplicity, a relative relatable scenario was chosen a base for the system. CoCoME depict a big supermarket like Lidl or Aldi. CoCoME was first introduced as the result of a seminar at the Technische Universität Clausthal in 2006. CoCome provides three core functionalities:

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

To be more precise and give a more complete overview, I am going to into more detail. In CoCoME there could be various enterprises. . Each enterprise may has various stores, each of them selling various products . Each of these stores has various chasdesks where products are sold. Furthermore CoCoME also handles the stock of each enterprise in the different stores. When a product is sold at a cashdesk or a delivery from a supplier is accepted, the current stock is automatically updated. 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 close to the reality to ensure that new paradigms are tested in a realistic environment. This has to be taken with a grain of salt. CoCoME is currently in development and new technologies or approaches are added incrementally.
I chose CoCoME as my case study, because it is already modeled in PCM. Also CoCoME is big enough to investigate some non-trivial cases of access control management.

6.2. CoCoME:overview

6.2.1. General Architecture

The general architecture of CoCoME variant that I am using is divided into three layers and an extra layer of abstraction. The item 6.2.1 shows the general layout of the architecture.

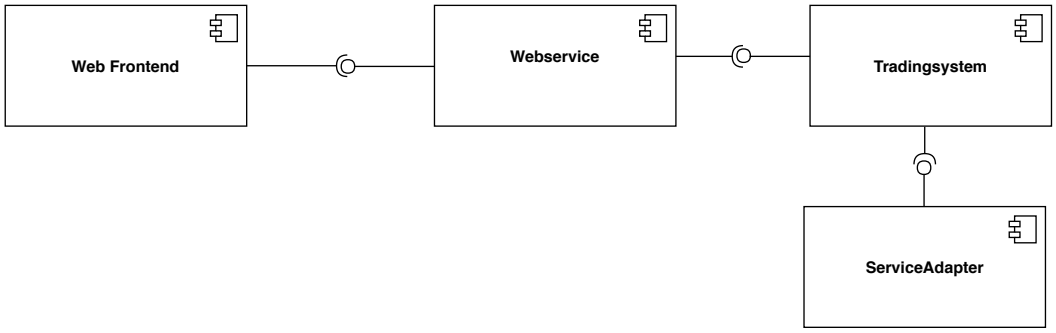


Figure 6.1.: Simpliefied view over the CoCoME system

1. **Client tier**
   In this layer all the client related code is placed. This includes mainly the graphical interface for customers and employees.

2. **Web services**
   In this layer, different services, which may be used by the clients and aren't a part of the underlying business logic are placed here.

3. **Tradingsystem**
   The heart and soul of CoCoME. In the tradingsystem is where all the work is done. In this tier all the request from the webservices and the users are handled. This includes queries to the backend, conducting sales (e.g. verifying credit card informations), updating the stock, handling user accounts and a lot more.

4. **Service Adapter**
   The service adapter is an extra layer of abstraction. The service adapter abstracts the database for the *tradingsystem*. With this addition it is now possible to have different database architecture for the underlying database(s). Another plus is that developer doesn't have to understand how the underlying database(s) are organized.

6.2.2. User roles in CoCoME

In CoCoME there are six different roles. Each role has different task and can perform different actions. The following list gives a short overview over each role.

- **Customer**
  The customer uses CoCoME for shopping and is the main role, which provides sensible (personal) informations.

- **Stockmanager**
  The stock manager handles the different product information, receive a report over all the ordered products and can view customer reports

- **Storemanager**
  The store manager orders Products, change their price and view the stock report

- **Cashier** The cashier sells products to the customer in the store via a cash desk
- **EnterpriseManager**
  The enterprise manager can only view the delivery reports
- **Admin**
  The system administrator provides a working environment, including the authentication of the user and the connection to the underlying database.

6.2.3. CoCoME technical details for the variant I am using

Since its first introduction in 2007 , CoCoME is under constant development. With time passing by and a lot of research project are conducted on CoCoME, a lot new variants of the basic CoCoME emerged. I am going to use the latest variant, the hybrid cloud based variant. I will go into more detail about this variant later. It would be too much to explain all variants , so I am only focusing on the one I am using in this thesis. In this thesis, I am going to use the hybrid cloud based variant. This variant emerged from the plain java variant by adding four evolutionary scenarios. These evolutionary scenarios are:

1. Platform Migration
2. Addition of a Pickup Shop
3. Database Migration
4. Addition of a Service Adapter

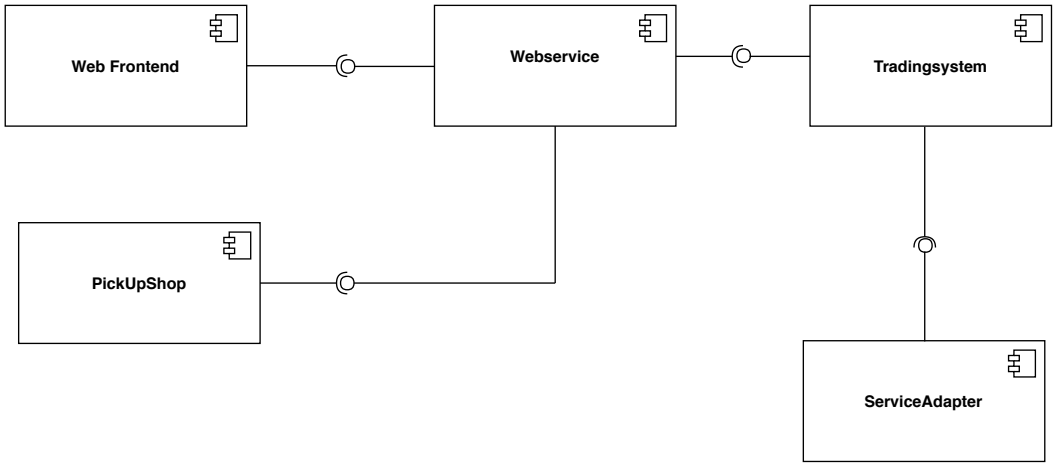
Also an extra abstraction layer was added for the communication between the *client tier* and the *tradingsystem*. This abstraction layer works as an wrapper for the *tradingsystem*. In the following I am going to go into detail for each evolutionary scenario

6.2.3.1. Platform Migration

The Platform Migration was performed to reduced used resources. The *tradingsystem* and the database runs in a cloud environment. This allows for flexible reconfiguration and easy adaptation. the implications to the architecture can be seen in The new component Webservice was added for the communication between tradingsystem and the client tier. The webservice only passes the

6.2.3.2. Addition of a Pickup Shop

The PickUp Shop is a new concept added to CoCoME. Customers can order products online and pick them up later in a store. They either pay via credit card or directly in the store of his/her choice. The implications to the architecture can be seen in . The most important implication is that each customer, who want to use the pick up shop now needs an account. Also, CoCoME now changes from a closed system to an open system.



6.2.3.3. Database Migration

The database was migrated , because with the addition of the pickup shop the numbers of requests are increased. Now from two sources requests to the databse are queried. First, from the stores and second from the pickup shop. Therefore scalability becomes an issue. Therefore the database is migrated into a cloud environment to solve the scalability issue.

6.2.3.4. Addition of a Service Adapter

The service adapter is a level of abstraction between the *tradingsystem* and the underlying database. The benefit is, that the *tradingsystem* doesn't have to bother with the actual database technology. It just send the queries to the Service Adapter, who does the rest of the connection. This was neccessary, because the database was migrated to a cloud environment. Within the cloud ennvironment, one cannot be sure which technology is used. Therefore the Service adapter solves this problem for the system.

6.2.3.5. Use case

In this paragraph, I will give a short summary over all the uses cases mentioned in In the current variant, described in subsection 6.2.3, there are currently 13 use cases. I will not describe all use cases in detail only the one that play a role in this thesis.
Use case 13: View CustomerReport
In this use case the stockmanager is able to generate a report about the purchased products of a specific customer. The stockmanager knows the id of the specific customer.
The standard process is as follows. The stockmanager is already authenticated in the system. Then the id is entered by the stockmanager. After that, the stockmanager is presented with a detailed report about the purchases of the specific customer.

6.3. Contributions

I described CoCoME in section 6.2 as the system was when I started my thesis. In this section I am going to explain my contributions and changes to prepare my case study out

of which I am going to develop an approach. My contributions could be divided in three parts. First, I had to extend the PCM language to fit my needs. Second, I had to define some blurryness in the CoCoME. This includes that roles used in the use cases aren't well defined. and Third, I had to develop scenarios on which I developed my approach. In all this Iam going to go in detail in the following.

6.3.1. Meta model extension

In the current state of PCM, it is not possible to model data flows. Therefore my advisor Seifermann created a meta model extension and provided it to me (it can be found here). The meta model extension is called data processing. I will give a quick overview over data processing.
The meta models extends the Service Effect Specification (SEFF) in the repository model. SEFFs are similar to UML activity diagrams. Like Activity diagrams, SEFFs sepcify the observable behavior for a system.SEFFS uses to model these different types of actions. These actions holds various inforamtion about the systems performance and are chained together to create a sequence of events to model the (observable) behavior of a specific method.
The meta model extension is specially for these SEFFs. It uses data containers to store information over a specific action in the SEFFs. Inside the data container one can specify different operations. Each of this operations store different informations about the data processed. The meta model has different operations that may manipulate data. It exists store, load, selection and projection operations mainly designed for databases but may be used in different contexts Also operation for data creation, data transmission, union of data and resulting data are included. These are used when in the program flow data is manipulated. There are also a third category of operations that describe mostly meta informations like characteristics. These are used, when additional informations are needed. With this meta model it is possible to model data manipulation in a system using the data containers and chain them together to create a data flow. The resulting dataflow can then be further investigated.

6.3.2. Model Transformation

6.3.3. role of the stockmanager

In CoCoME there is no clear specification for the role of the stockmanager mentioned in the use cases. It is not clear if the stockmanager manages the stock of one store or one whole enterprise. Nonetheless, the stockmanager is part of the use case diagram. Therefore I had to come up with a precise definition of the tasks and permissions of this role. I chose that a stockmanager is handling the stock of a whole enterprise. The following tasks arise from this assumption:

- Calculate predictions for future quantities of each product on the basis of the customer purchases.

- Storemanager may order products for their respective stores from the stockmanager. These supplies the stockmanager has to provide to the stores.
- Manage the different warehouses of an enterprise

To perform these tasks the stockmanager needs access to different data in the system. I will only focus on data that is relevant to the upcoming scenario and is in the Hybrid cloud based variant, because thats the variant I am using. The list shows which data exists and may be accessed by the stockmanager:

-

6.3.4. Definition of new data types in CoCoME

I defined two new datatypes, namely the connection between customer and his/her purchased products and the report type mentioned in . This was necessary, because these data types are missing and I need a precise definitions of these for case study. As I described in subsubsection 6.2.3.5, there have to some sort of connection between a customer and his/her purchased products. This connection is not defined nor implemented yet. I came up with different possibilities on how the connection can be made. In the end, I decided that the customer class holds a list of all orders. Each of these orders contains the pruchased goods. Second, I had to define the report mention in subsubsection 6.2.3.5. I defined the report as a list of Strings. To clariy this, the list contains strings, in which i saved the various informations. These informations includes product name, price, date of purchase and th store.

6.3.5. Definition of a new use case in CoCoME

I also defined a new Use case for the CoCoME system. This use case is the use case 14 as you can see in . For this use case, I first defined a new role. The new role is the support employee. This role has the following task:

- handle the problems of an customer with an order.

Now that the role is defined, I can define the use case 14. The use case is called: Handle customer problems. The formal description is as follows:

- Brief Description The system provides an opportunity to access the state of an order placed by a customer.
- involved actors customer, suppoer employee
- Precondition the support employee and the customer are authenticated
- trigger the customer submits a request
- post condition the problem is reviewed and an appropriate action is chosen.
- Standard process

1. The support employee reviews the request.
2. the support employee get access to the specifc order of the customer
3. the suppoer employee retrieves the status and sends a ticket to the respective store to re send the order

- 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

This use case is designed to help customers if there are problems with their order. Problems could be that the order is not avaible at the store, got lost in the system and many more.

6.3.6. Definition of access rights in CoCoME

In CoCoME there are different types f data. Not all data is as critical as other. So I singled the critical data out. Inside my case study, I only found personal data as critical data, namely name, address, e-mail address, credit card informations and account informations for the pickup shop. This account informations include credentials. These credentials are the username, passwords This critical personal data have to be protected by the system and only roles which have to know them for their tasks may access this personal data. Note that, there is a lot more critical data, but I only focused on the data described in subsubsection 6.2.3.5 and subsection 6.3.5, because these two use cases are relevant for my case study.
The following access control matrix I derived from the use cases:

6.3.7. Scenarios

Int his section I am going to present the two scenarios I defined for this thesis. I used scenarios to show realistic sequence of events inside the CoCoME system. To keep it realistic, my scenarios are derived from defined use cases in the CoCoME system. I decided to use the use cases, because the use cases represent real interactions with the CoCoME system. Also, use cases will be implemented. This guarantee that my scenarios are possible interactions in the CoCoME system and that I not just define scenarios that are a perfect fit for the upcoming approach. Furthermore, these scenarios were a good starting point for me to understand CoCoME. As my use cases, from which I derived my scenarios, I chose subsubsection 6.2.3.5 and subsection 6.3.5. The general structure of a scenario is as follows:

1. Define an Access control matrix (ACM) for each involved role and each involved data type.
2. Define the used part of the component diagram of the system. To clarify this, not all component of the system are part of a use case, so I cut out all 'unnecessary' parts.

3. Adding the data flows to the model.

With this scenarios defined, I can later showcase the basic idea of the approach I am going to present. In the following paragraph, I will give more details about my chosen scenarios. Before we go into the scenarios, a short disclaimer about the scenarios. The scenarios has to be taken with grain of salt, because CoCoME in the current implementation is rather a proof-of-concept and the implementation of a common modeling project than a complete system that is ready to be deployed. So it was relatively complex to find suitable scenarios, because the system is compared to the goal CoCoME is currently aiming for, bare minimum and not all use cases are implemented (at the current state of the implementation). This made the assumption about roles and data types necessary.

6.3.7.1. Scenario 1

My first Scenario is derived from the Use case 13 (UC13) described in subsubsection 6.2.3.5. First, I had to define an ACM for all the involved roles and used data types. The involved roles are only one, the stockmanager. The involved data types are (introduced in subsection 6.3.4):

1. customer id as long
2. query as String
3. List<IOrderEntries>
4. Report as a list of Strings

The ACM for this scenario is as follows:

	id	query	List<IOrderEntries>	Report
stockmanager	denied	denied	denied	allowed*/denied

A quick explanation to the presented ACM. In the introduction of the stockmanager-role (subsection 6.3.3) already explained, it is not needed that the stockmanager knows the ID of the customer. The query and the List<IOrderEntries> are system explicit data types , which must not be accessed by the stockmanager. The report type may be accessed by the stockmanger if there is no possibility to make a connection to be made, but by default I would argue that he is not allowed to access the report data type.
In the next step, I defined the part of the system that is used. In fact , UC 13 is exclusive to the PickupShop. So I only need components that belong to the PickupShop. A rough overview can be seen in Figure 6.2. The main goal of the PickupShop is to offer another alternative to purchase products in CoCoME aside from the typical supermarket scenario. So, it was designed to bypass the business logic and directly communicate with the component that handling the data. This is the reason, the business logic is missing in this diagram. Therefore, the component *PickUpShop* may communicate directly with the component *Tradingsystem:inventory*. There is no need for the component *WebFrontend* to be involved.
In the last step to define this scenario, I added the data flow to the model. This was

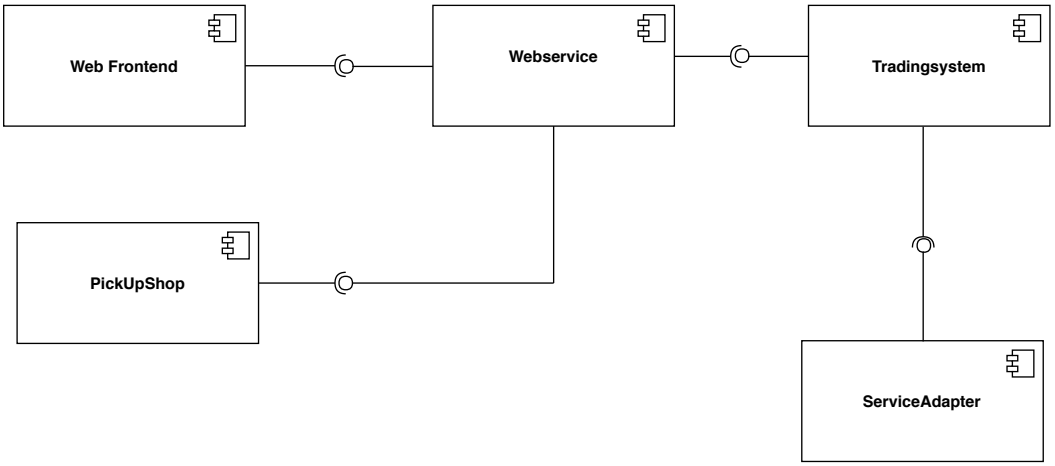


Figure 6.2.: Relevant Architecture for the UC13.

done in the PCM models with the meta modle extension provided by Seifermann (see subsection 6.3.1). To visualization can be seen in Figure 6.3. As one can see, the customer ID is passed through the system until it reaches *Tradingsystem:inventory:IStoreQuery*. Then the ID is used to build a query to select the customer. After that a list of all orders are created and send back to *Tradingsystem:inventory:application*. In this component the report is created and passed back to the stockmanger.

6.3.7.2. Scenario 2

For the scenarios, there are different evolutionary scenarios.

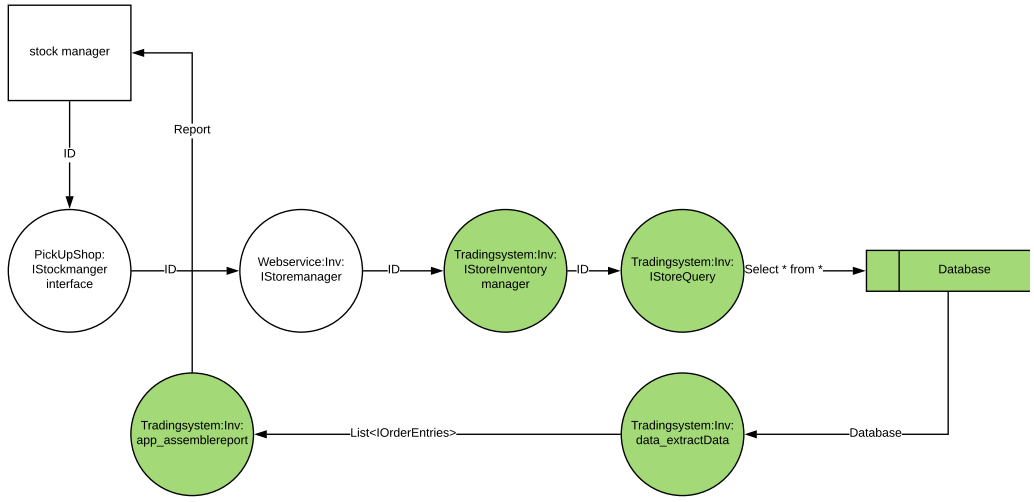


Figure 6.3.: Overview over the architecture relevant to the UC13. The grren processes are protected by the system.

7. Evaluation

7.1. Goal-Question-Metric plan

7.1.1. Question: Case study system for evaluation of data-based privacy analysis

7.1.1.1. Goal: Is the case study system usable for analysis of various problem statements?

Metric:

7.1.1.2. Goal: Are certain benchmark criteria met?

Metric:

7.1.2. Question: Case study system usable with data centric Palladio

7.1.2.1. Goal: Is the case study expressible by the modeling language?

Metric:

Metric:

8. Conclusion

Bibliography

[1] Steffen Becker, Heiko Koziolek, and Ralf Reussner. “The Palladio Component Model for Model-driven Performance Prediction”. In: *Journal of Systems and Software* 82 (2009), pp. 3–22. DOI: 10.1016/j.jss.2008.03.066. URL: http://dx.doi.org/10.1016/j.jss.2008.03.066.

A. Appendix

A.1. First Appendix Section

Figure A.1.: A figure

...