Verification of Access Control Policies in Software Architectures

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. Mai 2018 – 14. September 2018

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

1 Introduction

Nowadays, systems are more complex and connected. Beacuse of the connectivity of systems, security become more important. For software engineers this means, that non-functional requirements become more important. Some of these non-functional requirements are transparency of data processing, confidentiality and data protection. User are concerned about their privacy, therefore modern system have to ensure this.
In my upcoming thesis, I will focus on data protection and transparency of data processing. Both non-functional requirements are well reflected on access control. Therefore, I will identify and analyse dataflows in the to make sure access control policies aren't violated.
I am going to analyse component based systems on an architectural level. As my modelling language, I am going to use the Palladio Component Model (PCM). PCM offers more features than only modelling the system. For example one can do predictions for cost, reliability, maintainbility and performance of a system, that is modeled with PCM. These additional features may be used for trade-off analysis during my thesis. Currently, security requirements aren't well covered in the architecture model. Security rather is examined directly in the written code than the architecture. Also, dataflows, which can be used to examine security on an architectural level, are modeled and documented seperate from the architecture model. If the architecture is changed in the process, an inconsistency between the dataflow model and architecture model may exist. So security flaws in the architecture may be overseen. When the implementation of the system started and errors violating data protection are found it is very expensive to fix them. The team has commit manpower, time and money to fix the errors in the architecture. In the worst case the team has to go back to square one. It would be cheaper to identify erros on the architectural level and fix them there or have at least as little as possible errors before the implementation begins. So Seifermann [3] proposed a solution. He proposed to add dataflows in the PCM-language on an architectural level. These dataflows can be used to investigate security related requirements.
In my approach, I will take the basic idea of Seifermann to add dataflows in PCM as first level entities. Then I am going to use these dataflows to verify that access control requirements aren't violated in the system.
To validate my results, I will conduct a case study on the CoCoME system. CoCoME is system which abstracts the inventorymanagement and sales process of a big supermarket like Lidl. As a upside CoCoME is already modeled in the PCM-language. To this model, I will add dataflows. Then I am going to define constraints to assure only authorized roles in CoCoME may access sensible data. By using the constraints and the dataflows, I will setup constraint system. I will solve this constraint system by using logic programming. Then I am going to analyse the result to see if some flaws in CoCoME are present.

2 Basics

In this section, I will give an overview over the basic concepts I am going to use. This includes the CoCoME system, definitions of the basic security definitions, the PCM and the current state of the art.

2.1 CoCoME

In this section, I will give an overview over the CoCoME system. This covers the basic functionality, the architecture and the roles in the system.

2.1.1 Functionality

CoCoME was designed and implemented with the goal, to have a common ground on which different algorithms, scenarios and much can be tested. Currently CoCoME abstracts the inventory management of a big supermaketgroup like Lidl. CoCoME is currently in development, but predefined use-cases has been implemented. This use-cases cover the following functions:

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

2.1.2 Component based systems

Component based systems are the composition of different components. A component is defined as a unit, which may require or provide interfaces. Each component handles a task. For instance, such tasks are connecting to a database, serialization, deserialisation. The various components communicate with each other via predefined interfaces. Also it is possible to add or swap out different components easily, because of the predefined interfaces. Another upside of components are, that each component is enclosed in itself and different components may be developed simultaneously.

2.1.3 Evolutionary scenario: Hybrid cloud based variant with Pickup Shop

Over time different evolutions of CoCoME were developed. In my thesis I am going to use one of the latest evolutions, the Hybrid Cloud based variant with the addition of a Pickup Shop. In the following I will give a short overview over this evolution of CoCoME. For further reading on the different evolutions , please see [1].

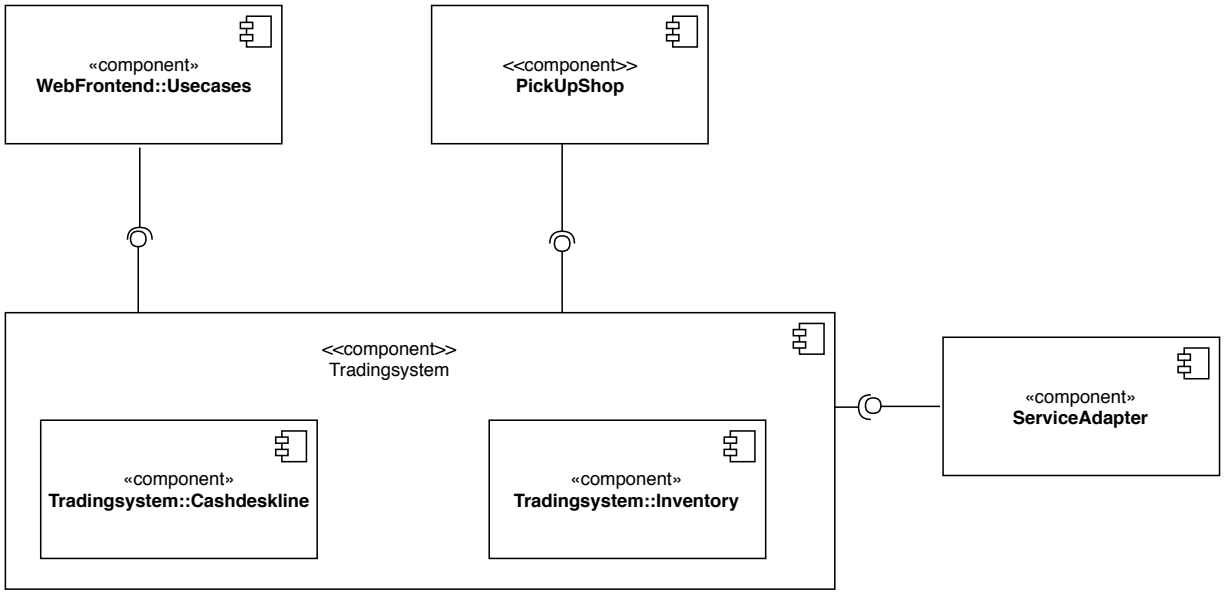

Figure 2.1: Architectural overview over CoCoME

2.1.4 Architectural overview

In this paragraph, I will give a quick architectural overview of CoCoME. A rough overview over the architecture can be seen in Figure 2.1. The fundamental layout of the CoCoME variant, which I am using, is deployed in different layers. On top there is a layer for the front end, an layer for web services and a final layer for the program logic. The front end layer is used to have different front ends for customer access. The front end forwards the request to the underlying structure and also shows the results.
The web service layer is used to add different services, like an online market, to the system. This layer encapsulates the access to the program logic.
The program logic layer holds the basic logic, the trading system, of CoCoME. This trading system divides into two components, the *Tradingsystem::cashdeskline* and the *Tradingsystem:inventory* component. The cashdeskline handles the various user inputs. This includes the different products an user purchased, the payment method and the displaying actions on the UI. The *Tradingsystem:inventory* component handles the connection to the underlying databases and the consistency of the stock items. The *Tradingsystem:inventory* component uses a service adapter. The service adapter manages the connection to the underlying database.

2.1.5 Addition of a PickUp-shop

Before the addition of the PickUp-shop, CoCoME has been a closed system. There was only a finite number of users, which interact with the system. Also no sensible personal information was stored in the system. I wasn't possible to connect a customer to his/her purchases. With the addition of the PickUp-shop, CoCoME changes to an open system. Customers create accounts in CoCoME. This accounts requires an address, first and last name and a password. Accounts are stored in the underlying database. With these accounts there are able to log into CoCoME and conduct an order. These orders are provided in the chosen shop, where the customer can

pick them up. The customer pays either via credit card in advance or directly with cash in the shop.

2.1.6 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.

2.2 Definitions

In this section I will give a definition of some important concepts which are used inside this proposal and the upcoming thesis.

2.2.1 Confidentiality

Confidentiality, in the context of computer systems, allows authorized users to access sensitive and protected data. This implies, that is impossible for unauthorized users to access sensitive and protected data.

2.2.2 Data protection

Data security refers to protective digital privacy measures that are applied to prevent unauthorized access to computers, databases and websites. Data security also protects data from corruption.

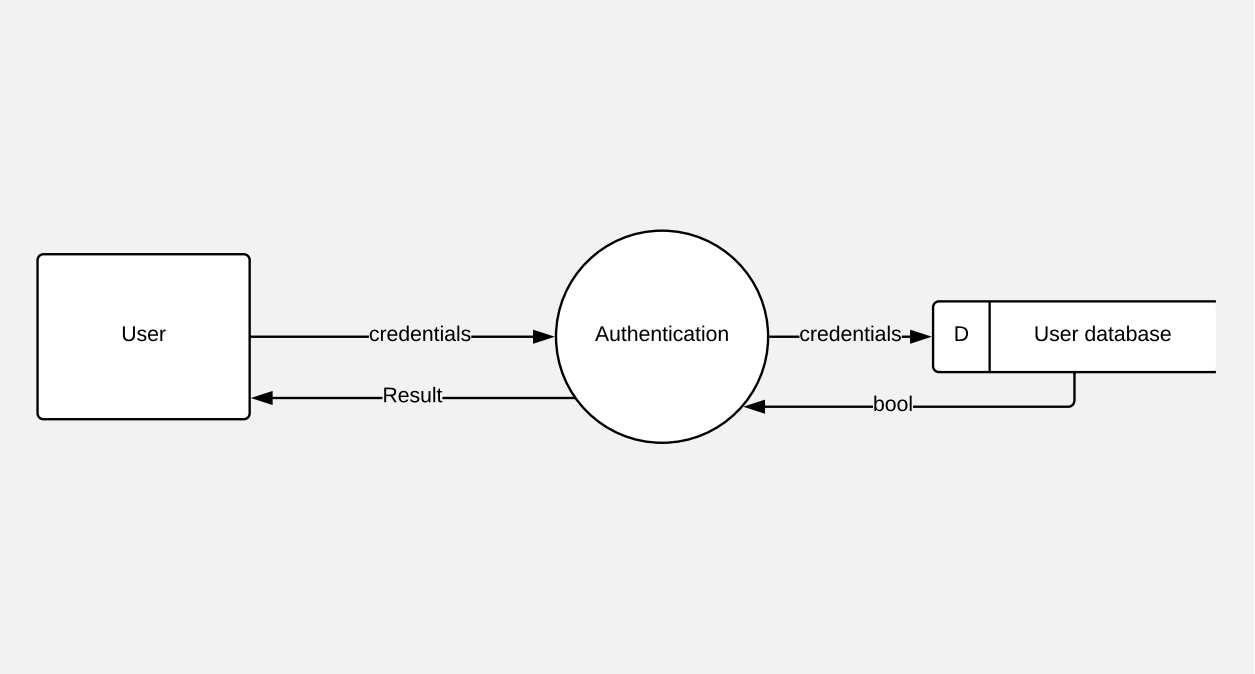

Figure 2.2: A simple data flow

2.2.3 Constraints

In the mathematical view, a constraint is part of an optimization problem. It is a condition of a solution, that has to be satisfied. In the context of the thesis, it is a condition, that may not be violated. Any viable solution has to make sure the constraints aren't violated.

2.2.4 Data flow

Data flow is the movement of data through a system comprised of software, hardware or a combination of both.
The Figure 2.2 is an example for clarification what a data flow is. As in Figure 2.2 shown, the credentials are provided by the customer. The authentication process then sends a query to the database to get the information, if the credentials are valid. If the credentials match, the customer is authenticated.

2.2.5 Role based access control

2.3 Palladio Component Model

The PCM allows more than only modeling the systems architecture. It's also possible to compute predictions for the modeled system. These predictions are predictions for cost, reliability, maintainability and performance. The PCM defines four roles in the development process. These are the component developer, software architect, system deployer and the domain expert. Each role creates and use different models to fulfill their task. In the following, I will give a

quick overview over the four roles, mainly which models are created by each specific role. This were all defined in PCM technical report [2].

2.3.1 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.

2.3.2 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.

2.3.3 System deployer

The system deployer decides on how the system is distributed between the available resources.

2.3.4 Domain expert

Domain experts creates 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.

2.4 Work flow

Te work flow in the PCM is the following. The component developer creates the component model and adds all necessary informations for th predictions analysis. Then the Software architect assembles the different components to a working system. After that the system deployer decides how the different components distributed on the available resources. At last, the domain expert defines the workload o each part. After all thesis done, PCM is able to compute the predictions for the modeled system.

2.5 State of the Art

Currently, in PCM the architecture and the data processing (modeled via data flows) aren't documented in the same model. Also analysis on the architectural level aren't added yet. Furthermore, data flows are not a first level entity yet. This means, they cannot exist on their own in a model. For all the mentioned aspects, that didn't exist, Seifermann [3] proposed a solution. My contribution is to create an prototype for the proposed solution.

3 Concept

In this section, I will present the concept of my upcoming thesis. This covers the goal I am trying to achieve, the questions I am going to answer in the process and the methods to verify my results.
The main goal of my work is to support an solicitous software architect. So, I am going to offer a tool to support the check the role-based access control in software systems. For this goal, some main questions arise:

- How do I have to extend the system model, in this case the PCM component model on a meta model level, to allow analyses for role based access control on the architectural level ? How do I have to extend the concrete models in a system like CoCoME to allow analyses of access control on an architectural level ?
- How does a possible transformation for an access control matrix and the models look like to be used as inputs for constraint solver ?
- How is a potential analysis of role based access control is carried out?

To verify my methods, I will apply them to CoCoME section 2.1 as a case study.

3.1 Preconditions, Limitations and Motivations

First of all, all the models, I am talking about, will be in the PCM modeling language. Since I am working on the architectural level, I will not consider the possible deployment of a system on different servers and therefore also not the possible cases of access control to that may come with such a deployment of a system. Also it is not possible to prevent abuse by different roles in the system. Such abuse may, if roles have access to sensible customer data (first name, last name, address, etc) they may store this information and sell them to unauthorized third parties to make a profit.
My main goal is to support a motivated software architect to identify errors in the architecture of a system in an early stage of the development process. The thesis will focus on altering the models of the component developer (subsection 2.3.1) and the software architect (subsection 2.3.2). I will use informations from the other models, namely the ones created by the system deployer() and domain expert, but beyond that, these model are no subject to this thesis. Note that, I can only detect flaws in the architecture, if they were modeled. Also it is only possible to detect flaws, that are specified in the the access control matrix.

3.2 Approach

The approach is divided in two different parts. The first part covers the modeling and transformation into a constraint system, which I can solve. The second part covers how to solve the constraint system and how to communicate the results to the system architect.

3.2.1 Defining goals, adding data flows and transforming into a constraint system

First of all, the access control matrix is created. This matrix describe how the system should work. This matrix contains the roles and the respective permissions. The entries in this matrix may be classified in to different categories. First, the predefined goals. These includes the entries in the access control matrix, which cover law regulations, like data protection. Second, the self-defined goals. This covers the entries, which are specific for the system. For instance, that in a a system sensible customer data cannot be accessed by a specific role. The population of the matrix is done with the help of the domain expert.
In the next step, the architecture models are extended. On a meta-model level in PCM, data flows will added. On the concrete architecture model level, the component developer adds the data flows to the model. This ensures that both the architecture and the respective data flows are in the same model. Note that only data flows will be modeled that are relevant in the context of the access control matrix. Then, after both the matrix is created and the models are extended, the architecture model is transformed into a constraint system. This constraint system can be solved to detect possible flaws in the architecture. The transformation is created by taking the access control matrix and the data flows as input and then generating a constraint system.

3.2.2 Solving the constraint system and interpretation

The resulting system is solved by using logic programming. The result then can be mapped back to the architecture to show possible flaws. For creating a solution, there are different approaches. One possible approach is the one from Seifermann , where he used the programing language Prolog to create a solution. another approach is the one from Jonas Kunz, which is currently developed. In the thesis, I will probably extend the approach from Seifermann. In the last step, the solution of th constraint system is mapped back to the architecture. This is done to visualize the results back to a software architect in a way s/he can understand them. This mapping back to the system is done by highlighting the unsatisfied constraints of the result. So the software architect directly identify the weak spot(s) and may address them.

3.3 Case study: CoCoME

In this section, I will apply the approach, described above on the CoCoME system (section 2.1). I will do this on an excerpt of the system. The main aspects I am going to cover are the creation of the access control matrix, an example data flow in the excerpt and an overview how the

Role	Access to customer personal data
stock manager	denied
cashier	granted
store manager	granted

Table 3.1: A simple access control matrix for the excerpt shown

evaluation with the constraint solver could work. This for better comprehensibility is done on an scenario in CoCoME system.

3.3.1 CoCoME: Scenario

The scenario I picked from CoCoME, is the Use Case 13 (UC13) (see the tech report for more informations [1]). This use case states that the the stock manager know the ID of an customer and then can request an full report on his/her orders purchased in CoCoME.
The description is very brief, because currently the use case is defined but not yet implemented in CoCoME. To provide an implementation is also a task of the upcoming thesis.

3.3.2 CoCoME: Approach

Here I am going to apply the approach described above to the CoCoME system. Therefore, I will apply th approach step-by-step on the scenario. The approach is done in four steps:

- Preconditions: Data flows in the model and defining the access control matrix
- Transformation to an constraint system by taking the data flows and the access control matrix as inputs.
- Solve the constraint system
- Transformation back to the model to visualize possible errors

For this scenario the access control matrix is the one shown below (Table 3.1). The cashier and the store manager are able to access the personal data to verify the right person stands in front of him, the stock manager is not, because he doesn't need the information for his tasks. To keep it simple, I left the missing roles out, because they are not part of the scenario. Then the data flow is added to the architecture model. Currently the architectural model is simplified and only the relevant components are shown. In this model, the components *PickupShop*, *Tradinsystem::inventory* and *SeviceAdapter* are involved. The stock manager access the system through the textitPickupShop, then the *Tradinsystem::inventory* handles the request. This means to build and execute the queries to the underlying database. The service adapter then answer these queries and the report is build in the *Tradinsystem::inventory*.
Currently, the both models are separated for better comprehensibility, but will be merged together in the final thesis. After the data flow is added and the access control matrix defined, the model is transformed. As input for this transformation the access control matrix and the data flow , for the excerpt, are taken and a constraint system is created. In the next step, the

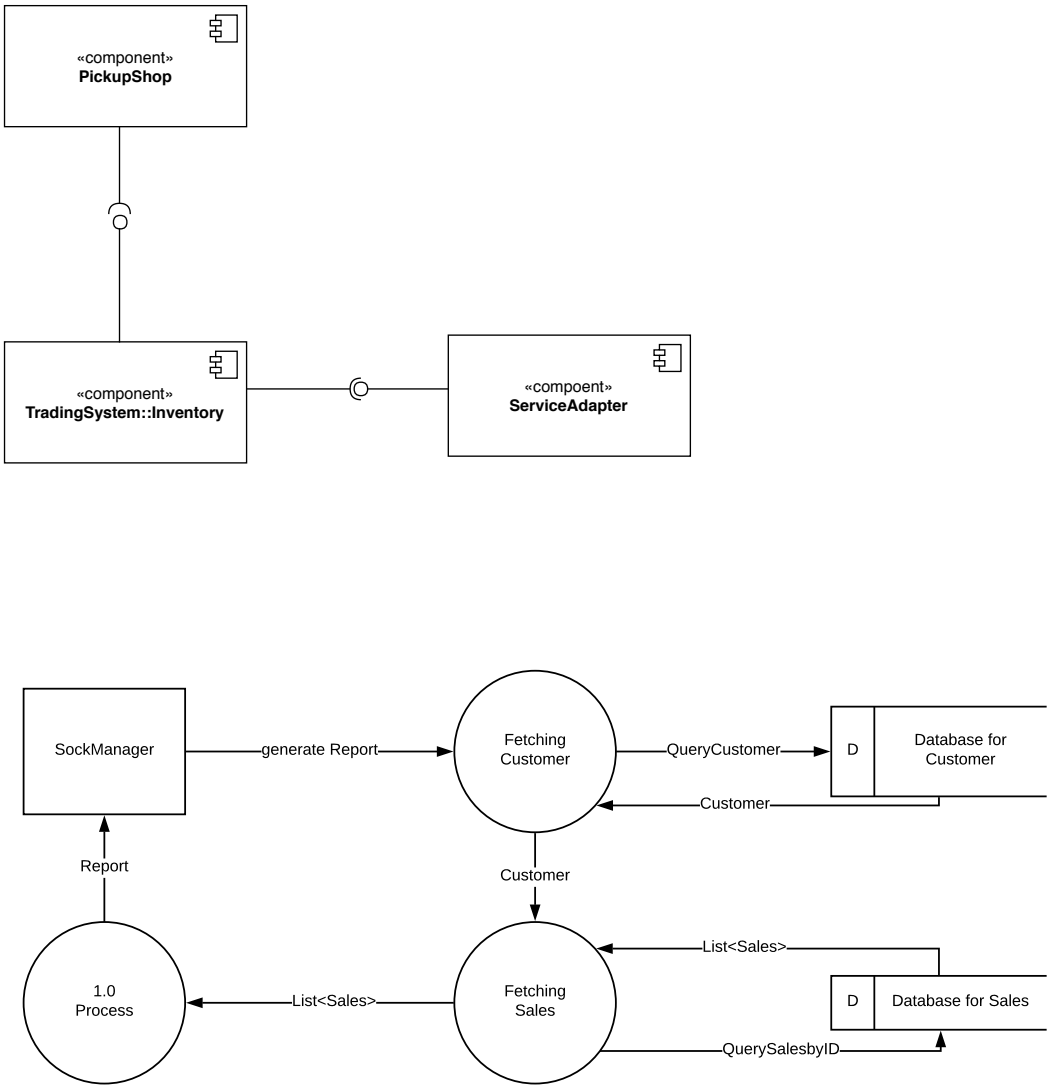

Figure 3.1: Added a data flow to a simplified architecture model

Figure 3.2: The resulting model after the constraint system has been calculated

constraint system is solved. For this task, there are various approaches. The approach I am going to use is the on from Seifermann, he uses the programming language Prolog to solve. this the approach that I am going to use.
After the result is calculated, the result is mapped back to the model to show the software architect if problems arose or if the system in this specification is flawless.How such the result for the scenario should look like can be seen in

3.3.3 CoCoME: Evaluation

The characteristic I am going to check is the correctness. The following question that I am going to answer is, works the modeled system in the current specification as expected. These is verified via the constraint system. Currently, UC13 isn't implemented in CoCoME, because of that I have to make a prediction how this case may be implemented. Another characteristic we may address could be performance. The question I am going to answer for this is, does CoCoME after the model is fixed performs worse than in the previous state? To measure this, I am going to use the PCM predictions that can be made in the modeling language.

3.4 Additions

In this section, I will cover my additions to PCM and the CoCoME system, to make the approach described in section 3.2 work.

- **Extend PCM Model on a meta level to include data flows**
  I will add to the component model, designed by the component developer, data flows as an first level entity.
- **Update the concrete CoCoME models**
  Currently, CoCoME is modeled completely in PCM. I am going to extend these models with the addition of data flows.
- **Extend CoCoME**
  Currently, CoCoME is for the goal it is aiming for a relative basic system. It currently only implements the specified use cases (see for further information the technical report [1]). I am going to create new use cases and add them to the CoCoME system.
- **Build a constraint solver**
  Currently, a basic constraint solver is implemented by Seifermann. This solver I am going to extend.
- **Transformation from model to constraint system and the other way around**
  In the current state, CoCoME is completely modeled in PCM. I am going to specify how the transformation is made in both directions.

4 Organization

4.1 Schedule

4.2 Risk management

In this section I will focus on risk management of my thesis. Therefore, I will point out some aspects, that may cause delay of my work.

- **Roles in CoCoME**
  Currently a role management is not implemented in CoCoME. I will add this. For this I may use more time than I expected.
- **Implementation of Scenarios**
  Some of the scenarios I am going to use, aren't part of CoCoME yet. It may take more time than expected to implement these.
- **Extension of PCM**
  Currently PCM doesn't support data flows in the architecture model. It may cost more time than anticipated to extend the PCM.

Bibliography

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

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

[3] 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.