Verification of Access Control Policies in Software Architectures
Proposal

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 giv 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 designend and implemented with the goal, to have a common ground on which differnet algorithms, scenarios and much can be tested. Currently CoCoMe abstracts the inventorymanagement 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 frontend for the services

2.1.2. Component based systems

Component based systems are the composition of differnet 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, serialisation, deserialisation. The variuos components communicate with each other via predefined interfaces. Also it is possible to add or swap out different components easily, because of the predefiendn 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 PickupShop

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 PickupShop. In the following I will give a short overview over this evolution of CoCoME. For futher reading on the differnet evolutions , please see [1].

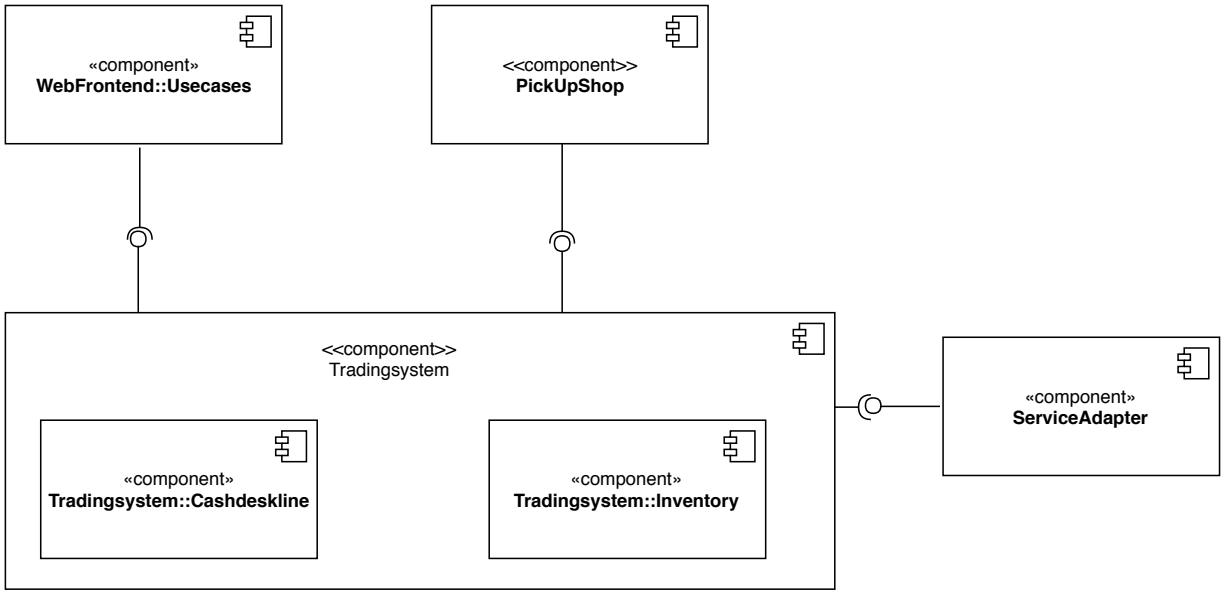

Figure 2.1.: Architectural overview over CoCoME

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 frontend, an layer for webservices and a final layer for the program logic. The frontend layer is used to have differnet frontends for customer access. The frontend forwards the request to the underlying structure and also shows the results.
The webservice layer is used to add different services, like an onlin market, to the system. This layer encapsulates the access to the program logic.
The programm logic layer holds the basic logic, the tradingsystem, of CoCoME. This tradingsystem divides into two components, the *Tradingsystem::cashdeskline* and the *Tradingsystem:inventory* component. The cashdeskline handles the various user inputs. This inludes 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 consitency 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 interagete 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 addtion 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 ar 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 differnet roles. Each role has differnet task and can perform differnet 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 stockmanager handles the differnet product information, receive a report over all the ordered products and can view customer reports
- **Storemanager**
  The storemanager orders Products, chane their price and view the stockreport
- **Cashier** The cashier sells procducts to the customer in the store via a cashdesk
- **EnterpriseManager**
  The enterprisemaanger can only view the deliveryreports
- **Admin**
  The systemadministrator 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.

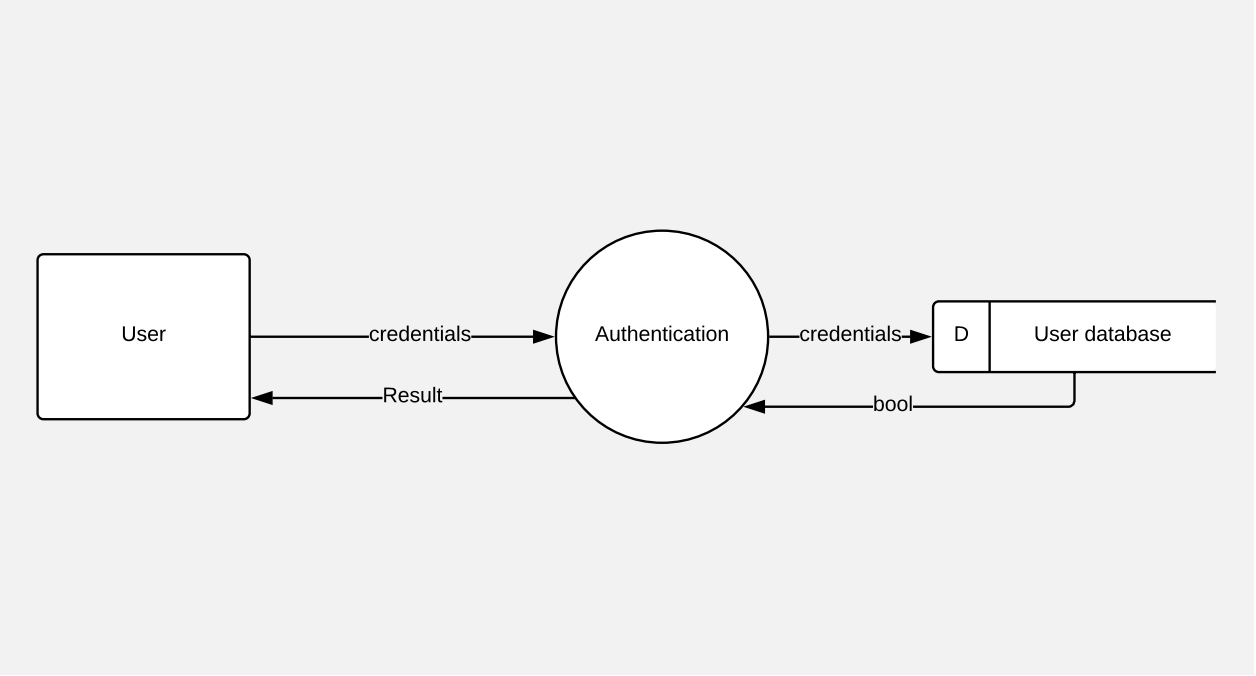

Figure 2.2.: A simple dataflow

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 ma ynot be violated. Any viable solution has to make sure the constriants aren't violated.

2.2.4. Dataflow

Dataflow 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 dataflow 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 modelling the systems architecture. It's also possible to compute predictions for the modeled system. These predictions ae predections for cost, reliability, maintainbility 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 componet developers to a fully functional system.

2.3.3. System deployer

The system deployer decides on how the system is ditributed between the avaible resources.

2.3.4. Domain expert

Domain experts creates usage model for the system. This describes user behaviour. 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. Workflow

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

2.5. State of the Art

Currently, in PCM the architectrue and the data processing (modeled via dataflows) arent't documented in the same model. Also analysis on the architectural level aren't added yet. Furthermore, dataflows 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] prposed 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 verification of role-based access control in software systems. For this goal, two main questions arise:

- How do I have to extend the system model, in this case the PCM component model, to allow analyses for role based access control on the architectural level ?
- How does a possible transformation for a constraint solver looks like?
- 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

The thesis will focus on models of the component developer (subsection 2.3.1) and the software architect (subsection 2.3.2). 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 differnet roles in the system. My main goal is to support a motivated software architect to identify errors in the architecture of a sytem in an early stage of the development process. Also all models will be in the PCM modelling language.

3.2. Approach

The apporach is divided in two different parts. The first part covers the modelling and transformation into a constraint system, which I can solve. The second part covers how to solve the constraint system. After that, I will show howto interpret the solution and the possible impact on the model.

3.2.1. ~~First part:~~ Defining goals, adding dataflows and transforming into a constraint system

First of all, goals are defined for the system. These goals make sure that the data processing in the system works the way it is intendend to. It exists two type of goals, predefined goals and self

defined goals. Predefined goals are mostly for compliance with law regulations. Self-defined goals are very specific to the system and cannot be generalized.
Then the component models are extended. Dataflows are added to the component models to portray the processing of data in the system. Note, that only data mentioned in the goals will be modeled in a dataflow diagram. After the goals are modeled and the dataflows added the architecture, the model is transformed into a constraint system.

3.2.2. Second part: solving the constraint system and interpretation

To solve the emerging constraint system I will use logic programming, most likely the language prolog. After the result is avaible, it will be mapped back to achitecture to show errors, if there some. The the software architect can react to the results.
The whole process allows to identify flaws in the architecture on an early stage in the PCM Process. Therefore it is relatively cheap to fix the errors.

3.3. Case study: CoCoME

In this section, I will apply the approach on the CoCoME system (section 2.1). I will do this on an excerpt of the system. In this excpert I will discuss a scenario, to clarify all aspects of the thesis.

3.3.1. CoCoME: Approach

As my scenario to work on, I choose the Use case 13 (UC13) defined in the technical report ~~(see~~ [1]~~)~~. In this scneario, the stockmanager request a full report of all orders a specific customer has made. For this the ID of the customer is known to the stockmanager.
For this scenario the access control matrix is the one shown below. The cashier and the storemanager are able to access the personal data to verify the right person stands in front of him, the stockmanager is not, because he oesn'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.

Role	Access to customer personal data
stockmanager	denied
cashier	granted
storemanager	granted

Then the dataflow is added to the architecture model. Currently the architectural modle is simplified and only the relevant components are shown. In this model, the components *PickupShop*, *Tradinsystem::inventory* and *SeviceAdapter* are involved. ~~The~~ I keep them seperate for comprehensibility. The Figure 3.1 shows the resulting model with the addition of the dataflow.

The stockmanager inputs the ID of a customer, then CoCoME excutes two queries. The first query to retrieve the customer, then the second query to get all orders for a specific customer. As the last step, the report is assembled and returned to the stockmanager.

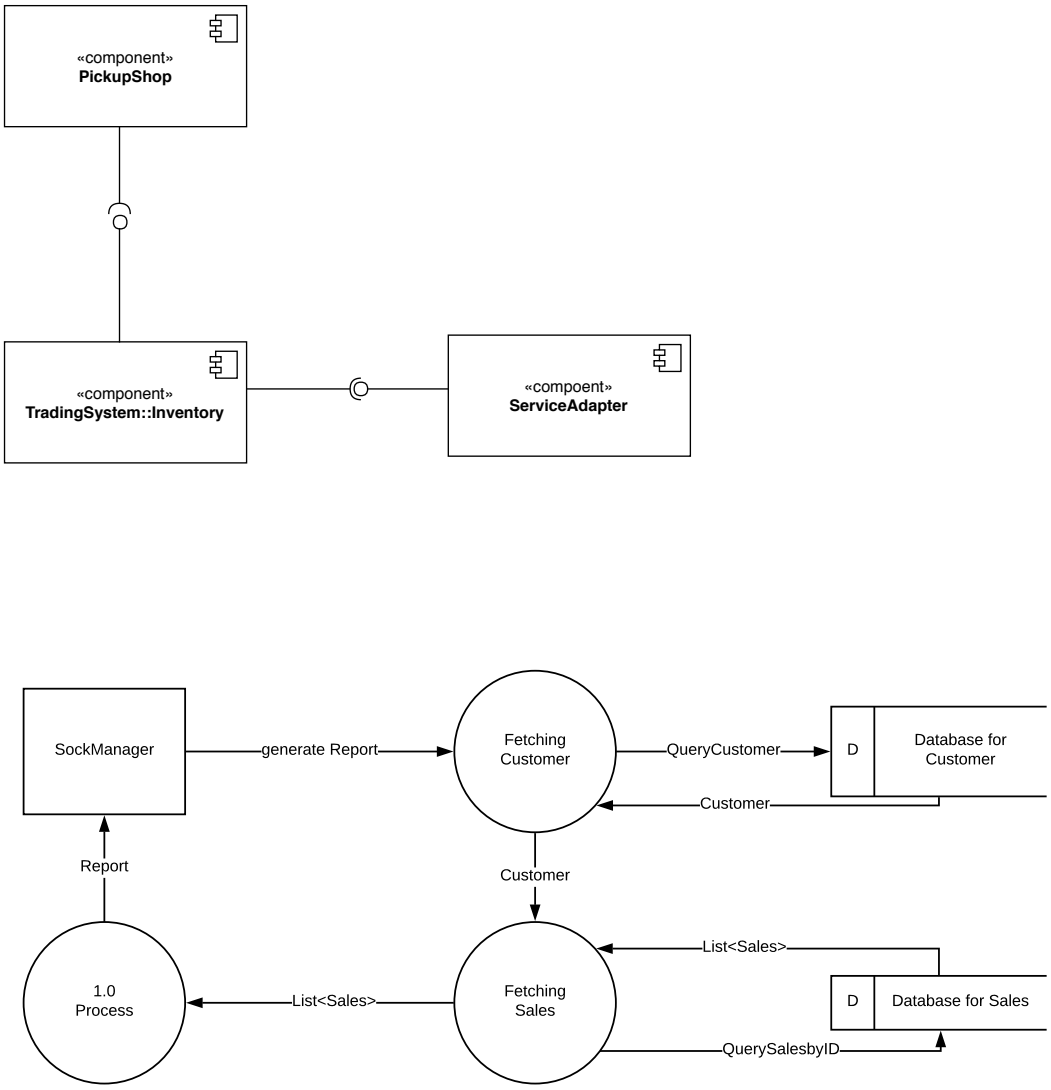

Figure 3.1.: Added a dataflow to a simplified architecture model

3.3.2. CoCoME: Evaluation

After the addition of the dataflow, the model is transformed to generate constraints. In the next step, a constraint solver determines, if the constraints can be met with the defiend goals in the access control matrix (subsection 3.3.1).
The result of the constraint solver shows, that the dataflow from the *Databse for Sales* to the *stockmanager* is invalid (see Figure 3.1). For the last step, this result has to be mapped to the architectural model. Now the software architext can review the model and change it if necessary.
The desired state would be if the dataflow from the *Databse for Sales* to the *stockmanager* couldn't exist.

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 to include dataflows**
  I will add to the component model data flows as an firs tlevel entity.
- **Extend CoCoME**
  Currently, CoCoME is for the goal it is aiming for a relativ basic system. It currently only implements the specified use cases (see for further information the techncal report [1]). I am going to extend the code base to create more dataflows than the basic ones. The details will be specified in my thesis.
- **Create a constraint solver**
  in the current state, Prolog does the trick. In the future with more sophisticated dataflows the constraint may become more complex. I am going to find a solver for this case. Either I am implementing it myself or use an already existing one.
- **Update existing PCM models**
  In the current state, CoCoME is completly modeled in PCM. I am going to update these models, with the dataflows I am going to analyse in my thesis.

4. Organisation

4.1. Shedule

The next figure (Figure 4.1) shows a rough overview over my schedule in the thesis.

4.2. Riskmanagement

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 rolemanagement 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 par tof CoCoME yet. It may take more time than expected to implement these.
- **Extension of PCM**
  Currently PCM doesn't support dataflows in the architecture model. It may cost more time than anticipated to extend the PCM.

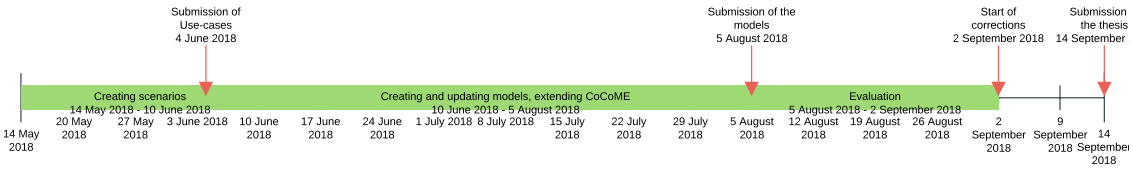

Figure 4.1.: Overview of the schedule for the thesis

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.

A. Appendix

A.1. First Appendix Section

Figure A.1.: A figure

…