Karlsruhe Institute of Technology

Software Design and Quality

Vertraulichkeitsanalyse mit Data-Centric Palladio
Proposal

Proposal 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. Mai 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. Introduction

1.2. Scope

As systems grows in complexity and connection, new challenges arise for software developers. Non-functional requirements becoming more important. Some of them are legal regulations, performance, confidentiality, data protection and so on. Software developers needs to pay attention to these in the design process so the upcoming systems are well designed regarding these new challenges. Also the transparency of data and the flow of data in a newly developed system is an important point as confidentiality become more important in the future. Users get more aware about data protection and want to know, what happens to confidental information.

1.3. Problem

The current situation is, that dataflows and and security are seperatd from the design process. There are no archtitectural entity for either of them and often security and dataflow examined at a later stage of the development process. Commonly, security is examined directly in the code. In the design timeline, it is one of the latest stages. Therefore if major flaws are revealed the fixing of this/these problems takes up a lot of resources in time and manpower. In the worst case the system has to be redesigned.

1.4. Gap

In design process dataflows aren't modeled on an architectural level and added on the later stages of the development. In result the earlier stages aren't covered sufficiently. It exists no architetural level entity for dataflow or access control. Therefore dataflow in the system or access control don't influence some design decison where they should.

1.5. Solution

As solution for the problem, the author proposed to model dataflow on an architectural level. Therefore it was proposed to extend the existing architectural description languages (ADL) to include dataflow diagrams and access control models. After including the new models and diagrams in the architectural artifact, there are new possibilities for analyses.

So the verification if the non-functional requirements are met becomes simpler. In other words, if flaws are found it is not that expensive to correct them. Expensive in this case means that it is not neccessary to remodel the system after deployment.

1.6. Validation

To verify that the proposed models are more of an asset than a liabilty, I will look closely in the CoCoME system. In CoCoME I will identify dataflows between components that pose a threat to data protection and the confidentiality of the user. After that constraints are defined, considering which information may flow through which components and which not.Then logic programming is used to solve the created constraints system. Additional to dataflows, access control will be modeled. I am using this model to ensure only authorized roles are capable of accessing confidental data. Also data leaks to unauthorized users are nullified.

2. Basics

2.1. Componentbased systems

Componentbased systems consists of, as the name states, components. A component is a unit of code in which a functionality is encapsulated. For clarification, in big systems you have components handling the access to a database, components for the UI and so on. The different components communicate with each other via predefined Interfaces. Note that a componentbased system is modular, which means you can easily swap components in and out. Often it is possible to reuse design know-how from other components.
In the next section, I will give a brief overview over CoCoME, the system I am going to work on and a big part of my upcoming thesis.

2.2. CoCoME

CoCoMe is a componentbased System, which are abstracts the inventory-management and selling process of a big vendor like Lidl, Aldi or some others. This system is componentbased (2.1). It consists out of many components, which handle the differnet Use-Cases for CoCoME. THe following list is a quick overview of what CoCoMe is (currently) able to achieve.

- Selling goods with express checkout
- Order products for the inventory and review the ordered products.
- adding different services through an API.
- Connectiong to a database

CoCoME exist in various variants, which were developed over the time.The codebase, used technologies and the deployment may vary on each variant. Likewise for some variant there are also evolutionary scenarios that may change the deployment of CoCoME or adding additional technologies to the system.For further reading on CoCoME please see
For my thesis, I will look closely in Hybrid Cloud-based Variant with the addition of a Pickup-shop. The deployment and some mentionable aspects will be covered in the following.

2.2.1. Archtitectural overview

First of all, I will take a look on the architectural view on The fundamental layout of the CoCoMe, 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 webservice layer is used to add different services, like analytics, to the system.
The programm logic layer holds the basic logic, the tradingsystem, of CoCoME. This tradingsystem divides into two parts, the cashdeskline and the 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 inventory component handles the connection to the underlying databases and the consitency of the stock items.

2.2.1.1. Adding a PickUp-shop

The concept of a PickUp-shop is, that a customer orders goods online and pay them directly or pay at the PickUp-shop . The purchased goods are provided by the chosen PickUp-shop, where the customer collects them. In this scenario CoCoME changes from a closed system (finite employees access from finite locations) to an open system (user may connect from all over the world). This system may be deployed on one or various servers. The specification states at least three different servers.

2.2.1.2. Database Migration

In this case the CoCoME system deploys the database in a cloud-environment. The purpose of the decison is, if CoCoME grows in user numbers that the scalability of the current cloud isn't sufficient enough anymore and more CPU power is needed. Note that, this change of the cloud environment can be done during the runtime.

2.2.1.3. Adding a Service Adapter

The service adapter is a technique to abstract the database access. Its mainly used the not be raliant to the layout of the underlying database by adding a layer of abstration

2.3. Definitions

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

2.3.1. Confidentiality

Confidentiality is prsent, if there is no unauthorised informatio retrieval possible.

2.3.2. Data protection

Ability of a real person to control the flow of his/her personal data in a given system

2.3.3. Dataflow

One can imagine a dataflow as the journey of information through a system. More precisely, a dataflow is how information are altered by process until the reach their final station. The two figures below will clarify this definition:

2.3.4. Constraints

Constraints define limits for a system or in this case, a dataflow.

3. Concept

I will present a scenario (from CoCoME) to show the basic idea of my thesis. This includes identifying a dataflow, show violations to confendentiality, building constraints and using Prolog to solving them.

3.1. Scenario

3.1.1. Scenario: UC13

The UC13 is a violation to data protection. The stockmanager is able to get specific information of a customer. This open the system to various threats, like analyzing the current situation of customer.
I assume the id's in the Customer class are distinguishable. Currently a connection between the custome r and the ordered products are not yet made. I will looking at two different possibilities, how the connection may be implemented.

- The Store queries past sales to the stores database and saves the customer in an attribute.
- A container class is added, which stores the customer-order-pair.

The resulting dataflow I assume an alternative and more seure implementation is to get the purchased items from a specific shop instead of a specific customer. Currently this case isn't implemented yet.

3.2. Evaluation

4. Organisation

A. Appendix

A.1. First Appendix Section

Figure A.1.: A figure

...