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

As systems gets more connected and complex in the past few years, some new requirements emerged. Such requirements may be compliance with law, performance and many more. So it seems to be a good solution to model dataflows on an architectural level to get an overview over how data is processed before any line of code is written. This also implies that non-functional requirements as mentioned above may be considered during the modelling of the system. Currently dataflows aren't an entity on the architectural level, and therefore aren't considered in the modelling process.
In my upcoming thesis, I will address this issue and modelling dataflows on an archtitectural level. For this I will look into an existing system and extracting existing dataflows.

1.2. Main Question

The main questions I will look into are :

- How to identify dataflows
- Identifying flaws to non-functional requirements
- Building constraints for the system
- Propose solutions that the constraints are met

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. 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

Currently, there are various variants of CoCoME, which were developed over 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.
The following sections are used to introduce all of the mentioned variants.

2.2.1. Plain Java Variant

This variant is one the first implementations of the CoCoMe system. It was implemted in Java and does not use any JavaEE technologies. Also it contains the core components for the inventory-management and the cashdeskline, where the selling of itmes is handled. Note that an User Interface is not implemted yet.

2.2.2. Service Based Variant

This variant puts a web-service-layer(WSL) on top of the components implemnted in the Plain Java Variant (2.2.1). The additional WSL allows to add different services fairly easy to CoCoME. Such services could be, for example, an User Interface or some analytics tool.

2.2.3. Hybrid Cloud-based Variant

This variant adds a frontend to the already existing backend (2.2.1 and also keeps the additional layer (2.2.2). This variant evolved from the Plain Java Variant (2.2.1) by relocating some resources to a cloud environmentand implementing four evolutionary scenarios, which are described in the following sections.

2.2.3.1. Platform Migration

This evolution transfers the attached database to the cloud.

2.2.3.2. 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 collect 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.

2.2.3.3. 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.

2.2.3.4. 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

…