
# The CoCoME Platform for Collaborative Empirical Research on Information System Evolution

**Technical Report**

Robert Heinrich, Niko Benkler, Tobias Haßberg, Ralf Reussner

April 22, 2018

# Contents

# List of Figures

# 1 Introduction

# 2 Evolution Scenarios

We implemented distinct evolution scenarios covering the categories adaptive and perfective evolution. Corrective evolution is not considered in the scenarios as this merely refers to fixing design or implementation issues.

## 2.1 Evolution Scenarios of the Hybrid Cloud-based Variant

This section introduces the two evolution scenarios of the hybrid cloud-based variant of CoCoME.

### 2.1.1 Setting up a Docker environment

The CoCoME company must reduce IT administration costs but frequent updates to the enterprise and store software are necessary to continuously improve the entire system. As a consequence, IT staff need to update the system components as soon as a new software version is released. An Operations Team member has to get access to the actual server in order to undeploy the old version and replace it with the new one. This is time consuming and expensive as the updates have to be done manually.
Therefore, a Docker version is elaborated to simplify the administration process. As soon as a new software version of CoCoME is ready for delivery, the Development Team wrap it into a Docker Image. This Image can be automatically deployed to the destination server according to the principle of Continuous Deployment (CD) [2].

### 2.1.2 Adding a Mobile App

After successfully adding a Pick-up Shop, the CoCoME company stays competitive with other online shop vendors (such as Amazon). In times of smartphones, customer do not only want to buy exclusively goods from their home computers. Purchasing goods 'on the way' comes more and more into fashion. This raises the idea to create a second sales channel next to the existing Pick-up Shop in the CoCoME system. As a consequence, more customers can be attracted to gain a larger share of the market.
The customer can order and pay by using the app. The delivery process is similar to the Pick-up Shop: The goods are delivered to a pick-up place (i.e. a store) of her/his choice, for example in the neighbourhood or the way to work. By introducing the Mobile App as a multi OS

application, the CoCoME system has to face various quality issues such as privacy, security and reliability. Also the performance of the whole application can be affected if many customers order via the app.

## 2.2 Evolution Scenarios of the Microservice-based Variant

This section introduces the evolution scenario of the Microservice-based variant of CoCoME.

### 2.2.1 Defining different Microservices

After years of growth of the sales figures, the CoCoME company is thinking about steps to keep this trend. During the first meetings, they figured out, there should be a growth in income when they are establishing more branches. Later on, it became clear that the management system used so far would struggle to manage that situation. In consequence he CoCoME management decided to rebuild that system.
In their specification they mentioned at first, that the frontend should be similar to the old one and provide the same functionality. They decided to build an decentralized management system, which provides them a rapid registration of sales, and use different services which should be be able to duplicate them self, e.g. one project to manage the branches with an instance for each branch. In that moment a present computer scientist explained the concept of microservices to the managers.
Consequently they decided to re-engineer the given system and split it into microservices since it fulfills all their requirements.

# 3 Design Details for Evolution Scenarios

In this chapter we provide the detailed design documentation for each of the evolution scenarios introduced in the prior section. Sec. 3.1 sketches the design decision for the Mobile App that provides a second sales channel next to the existing Pick-up Shop. Sec. 3.2 describes the adaptive cahnges of setting up a Docker environment to simplify the update process. They are both based on, or at least use the Hybrid Cloud-based Variant of CoCoME [1]. In contrast, Sec. 3.3 provides a detailed design documentation of a new architectural version of CoCoME. This perfective evolution scenario is realized based on the Microservice idea.

## 3.1 Design Decisions for the Mobile App

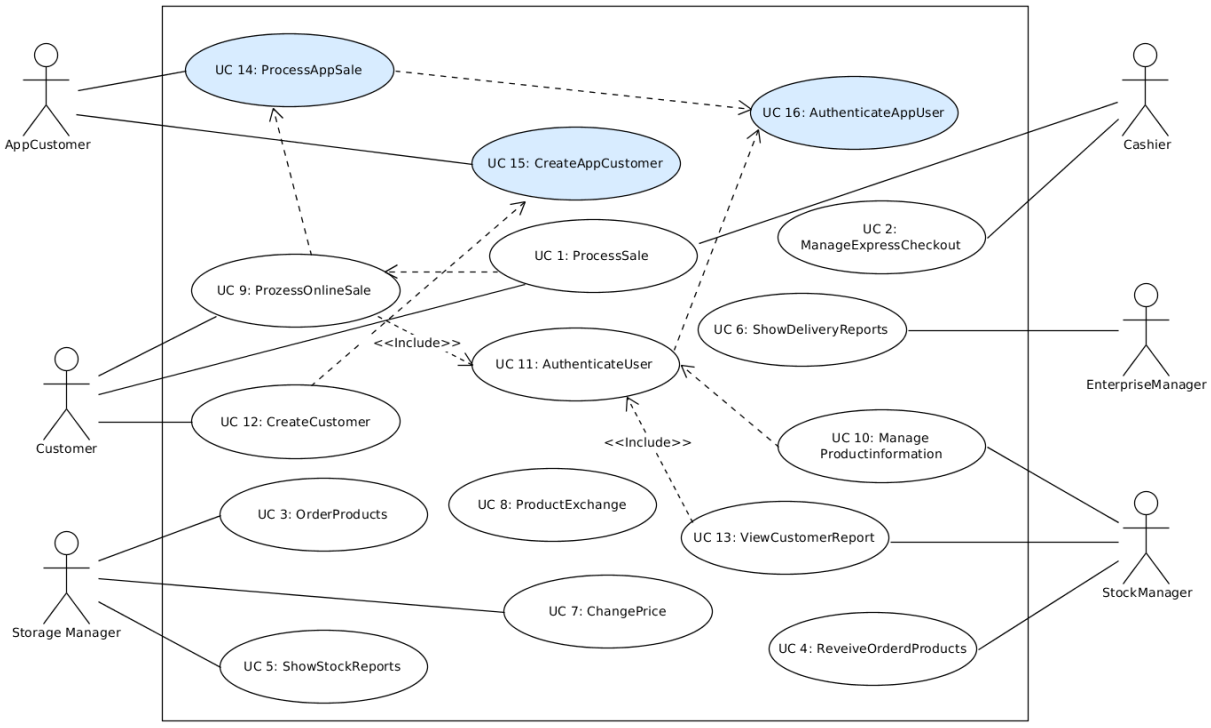


Figure 3.1: Use Case Diagram CoCoME Mobile App

## 3.2 Setting up a Docker environment

Looking to the changes for the Docker project, you can see in figure 3.2 the changes are affecting the technology stack in the form of adding additional layers. More detailed, the given CoCoME project is moved into the Docker Deamon, which runs a Linux distribution.The original parts of the stack, like Glassfish and the Java Virtual Machine, are still a part of the stack.

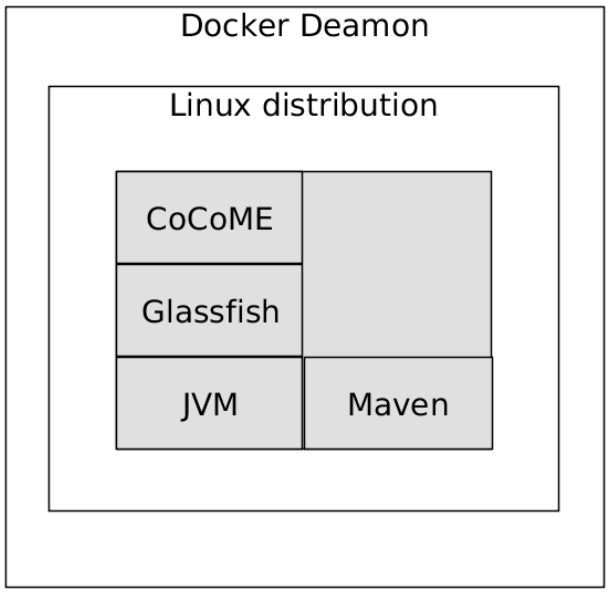


Figure 3.2: Extended technology stack CoCoME

The Dockerfile defines an environment based on the latest version of Ubuntu 16:04. Onto it there is installed Maven, Git and Java by using the Ubuntu package manager.
Git has two purposes: On the one hand it is used to download the most recent version of CoCoME. On the other hand, it is used to download a prefabricated version of Glassfish that already includes domains and other adjustments required for CoCoME. Java is required by Glassfish and CoCoME as they need the Java Virtual Machine. Maven is needed to deploy the latest version of CoCoME onto the provided Glassfish servers.

During the development, it was decided to implement and provide two different versions. The first version always pulls the most recent CoCoME source code from GitHub, downloads the entire dependencies with maven, compiles and builds the project and finally, deploys CoCoME on the Glassfish servers. . As a consequence. creating and starting a Docker Container takes about one hour.
In contrast, the second version only pulls a prefabricated version of CoCoME from GitHub. Therefore, pulling the source code up to building the project is skipped. As a consequence, Maven does not have to be included in the technology stack. Solely, deploying CoCoME on the glassfish server is necessary.
This reduces the deployment time to a few minutes but has a disadvantage: The prefabricated

version is updated manually. Therefore, it is sometime not the most recent version.
By providing both, a fast deploying version and a current version, the user can choose whats the best for its situation.

## 3.3 Using Microservices Technology

- je microservice absatz mit entsprechenden Sequenzendiagram
- frontend? muss dazu auch das gemacht werden?
- ein blocktext zu mehereren diagrammen oder diagramme zwischen text?
- je die einzelnen module und szenarien erlaeutern in dem diese sinnvoll sind

### 3.3.1 Products

abstrahieren der Produktinformationen

### 3.3.2 Stores

einzelne Laeden alleinstehend abbilden um nach bedarf neue microservices alias laeden starten zu können

### 3.3.3 Enterprise

aehnlich zu stroes

### 3.3.4 Reports

stellt alleinigen aufgaben bereich dar, entsprechen undabhaengig darzustellen von anderem.

# 4 Implementation of Evolution Scenarios

## 4.1 Docker

As shown in figure 4.1 the docker Container contains five different Glassfish servers. In particular they are called *WEB*, *ENTERPRISE*, *STORE*, *REGISTRY* and *ADAPTER* and correspond to the given by the CoCoME deplyoment setup. By default, Glassfish provides a Derby DB that is connected to the server Adapter using Java Database Conectivity (JDBS) interface.

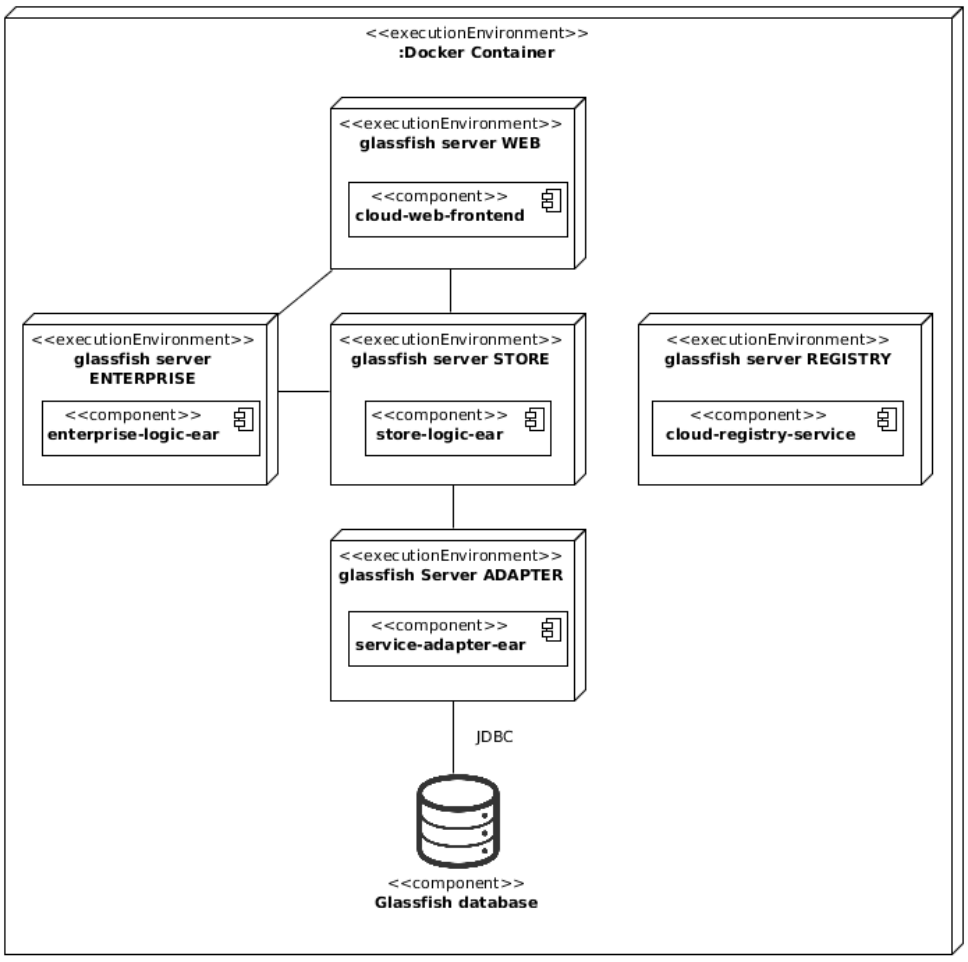


Figure 4.1: Deployment diagram CoCoME

CoCoME is deployed inside the docker container on the same way it is usually deployed. This means the maven generated archive files *cloud-web-frontend*,*enterprise-logic-ear*,*store-logic-earcloud-registry-sevice* and *service-adapter-ear* are deployed to the servers with the following assignment:

| Server | Deployment file |
|---|---|
| WEB | cloud-web-frontend |
| ENTERPRISE | enterprise-logic-ear |
| STORE | store-logic-ear |
| REGISTRY | cloud-registry-service |
| ADAPTER | service-adapter-ear |

Figure 4.2: Assignment of archive files to Servers

4.2 demonstrates the assignment between the archive files and the servers as it is implemented and also recommended by the CoCoME deployment guide. This information is also represented in Fig. 2. As mentioned earlier, there are two versions of this Docker project. Both deploy the CoCoME main program with this assignment.

In addition, the fast version can be extends by the pickup shop[1]. This pickup-shop runs inside a separate container which is shown in figure 4.3. As shown in figure 4.3, this container

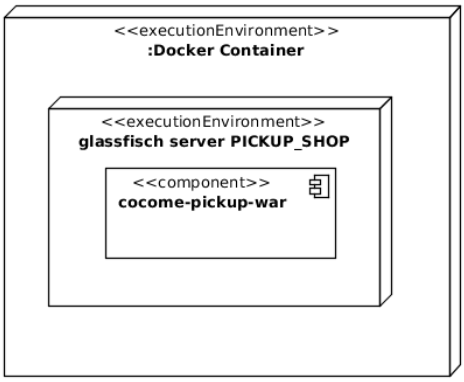


Figure 4.3: Deployment diagram CoCoME Pickup Shop

provides only one Glassfish server.

| Server | Deployment file |
|---|---|
| PICKUP_SHOP | cocome-pickup-war |

Figure 4.4: Assignment archive files to Servers

[1] https://github.com/cocome-community-case-study/cocome-cloud-jee-web-shop

To control the start of both containers, precisely the CoCoME and the Pick Up Shop, another specific file is needed: the Docker Compose file. It ensures that the CoCoME Container is active, before the pickup-shop container is starting. This is necessary as the Pickup Shop requires a running instance of CoCoME to register itself.
Also CoCoME runs without the pickup-shop, the pickup-shop does not work without an running instance of CoCoME.
Both containers need to communicate with each other. By default, docker prohibits any outgoing and ingoing communication from an in a container. This is solved by opening specific ports through which the communication is possible. Which ports the containers can use is specified in the Docker Compose file as well.

# 5 Conclusion

# Bibliography

[1] R. V. Heinrich. The cocome platform for collaborative empirical research on information system evolution, [2016].

[2] H. H. Olsson, H. Alahyari, and J. Bosch. Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 392–399. IEEE, 2012.