Politecnico Di Milano

A.A. 2015/2016

Software Engineering 2:

”MyTaxiService”

Design Document

Paramithiotti Andrea (Matr. 788794) Rompani Andrea (Matr. 854052)

Zoia Lorenzo (Matr. 852392)

v1.0

4/12/2015

Contents

[1 Introduction 3](#_Toc435896242)

[1.1 Purpose 3](#_Toc435896243)

[1.2 Scope 3](#_Toc435896244)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc435896245)

[1.3.1 Acronyms 3](#_Toc435896246)

[1.4 Reference Documents 3](#_Toc435896247)

[1.5 Document Structure 4](#_Toc435896248)

[2 Architectural Design 5](#_Toc435896249)

[2.1 Overview 5](#_Toc435896250)

[2.2 High level Components and their Interaction 5](#_Toc435896251)

[2.3 Component View 7](#_Toc435896252)

[2.4 Deployment View 9](#_Toc435896253)

[2.5 Selected Architectural Styles and Patterns 11](#_Toc435896254)

[2.6 Other Design Decisions 11](#_Toc435896255)

# Introduction

## Purpose

This document represents the Design Document (DD). The main goal of this document is to describe all the components of the system and their interaction.

## Scope

The aim of the project is to create a brand new system for the management and organization of a city taxi service. This system offers a mobile application and a web interface in order to give the customer the possibility to benefit from the taxi service. Furthermore, the system provides an additional communication interface for the taxi driver.

The mobile application and the web interface accept requests and reservations for taxis from the users, with the possibility to organise a share ride among different users. The taxi driver is supposed to communicate his availability, acceptances and rejections of requests through the communication interface.

The system is created to simplify the access of passengers to the service, and to guarantee a fair management of taxi queues.

## Definitions, Acronyms, and Abbreviations

### Acronyms

* **DMZ:** Delimitarized Zone
* **DW:** Data Warehouse
* **DB:** Database
* **SOA:** Service-Oriented Architecture
* **RASD:** Requirements Analysis and Specification Document

## Reference Documents

This documents refers to the RASD document of the previous deliver.

This document is conformed to the “IEEE Standard for Information Technology—Systems Design— Software Design Descriptions”, (IEEE Std 1016™-2009 (Revision of IEEE Std 1016-1998)).

## Document Structure

* **Introduction:** it gives a description of the document and the information about the project scope
* **Architectural Design:** it contains all the information about the architectures designed for the system and the components that compose them. It provides the Component and Deployment diagram in order to clarify the exactly distribution of the components and their interaction. In this chapter are also included all the motivations that has determined the architectures’ choice
* **User Interface Design:** it contains all the mock-up useful for the description of the user’s experience
* **Requirements Traceability:** it explains how the requirements defined in the RASD map into the design elements that you have defined in this document
* **References:** it contains all the documentation

# Architectural Design

## Overview

The most important part of the System is structured using the SOA. For this reason, everything that finds place out of our physical system, so all the interface used by person, are defined as Service Requestors. All the operations that the system has to provide are split into atomic services.

In order to manage every request coming from the outside and message among all the services there is a Broker between the service providers and the requestors.

Regarding the data managing, there are two different approaches:

* Storing all the sensitive data in a database, contained into a DMZ, in order to guarantee a high level of security.
* Using a DW, connected to an analyzer software, to store all the other data in order to perform data mining operations.

The last component that defined is an http Web Server used for storing every file needed by the Website.

To perform the communication from the service providers to the requestors are used Push Notifications or directly the web server, depending on the user/Taxi driver interface involved.

## High level Components and their Interaction

The components of the system are:

* The service providers, which are the software components that must be installed on the servers and contain the logic in order to manage the services offered to the clients.

In this system these providers are:

* + The requests manager
  + The shared rides manager
  + The reservations manager
  + The registration manager
  + The authentication manager
  + The taxi manager, which perform either queue and Taxi drivers’ availability coordination
  + The emergency manager
* The service requestors, which are the interfaces that request the execution of a service. They all has an active internet connection in order to send requests to the providers.

In this system, according to the RASD device description, these requestors are:

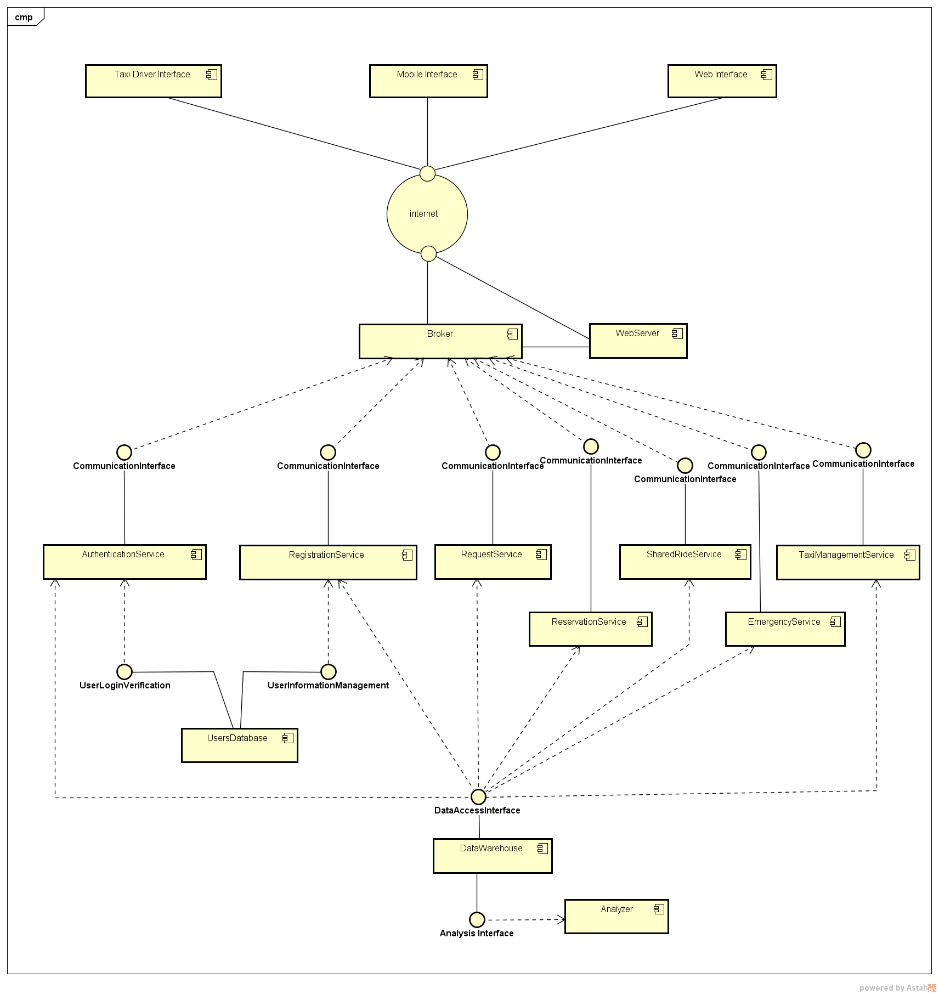
* + The Taxi driver interface
  + The web interface
  + The mobile interface
* The Broker, which is the central core of the system. It manages all the communication between either the requestors and the providers or the providers themselves.

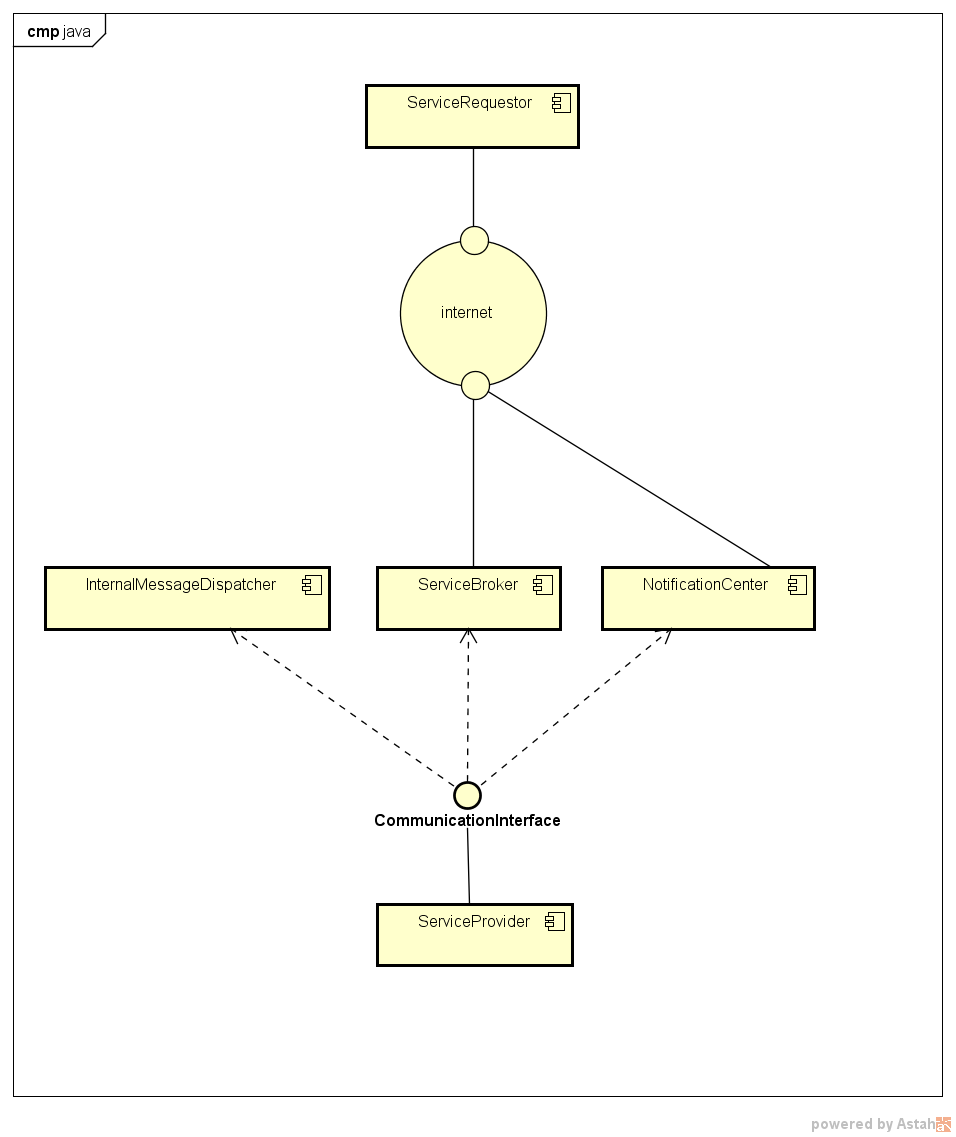
It is composed by three elements:

* Service Broker: this component receive the requests from the requestors, through an internet connection, and is able to forward them to the designated provider. Consequently, it has a direct connection with internet and all the Service Providers.
* Notification Center: this component allows the service provider to communicate to the requestors using the messaging protocol Push Notification. In order to perform his job, as the Service Broker, it is connected with internet and all the Service Providers
* Internal Message Dispatcher: the aim of this component is to allow and manage the communication among the service providers, so it has a direct connection with each of them.
* The Web Server, which manages the all the https request and where are stored all the files useful for the website. It is also connected with the Service Broker and with it and the Notification Center, are the only component that are connected to internet.
* A Data Warehouse, which is used directly by the providers in order to store or read any useful or sensitive data. It works with an OLAP technology for the storage of the data.
* A database, storing the users information, contained into a DMZ in order to increase the security and prevent unauthorized access to sensitive information
* The analyzer, a software application that have access to the data in the Data Warehouse and performs the statistical analysis

## Component View

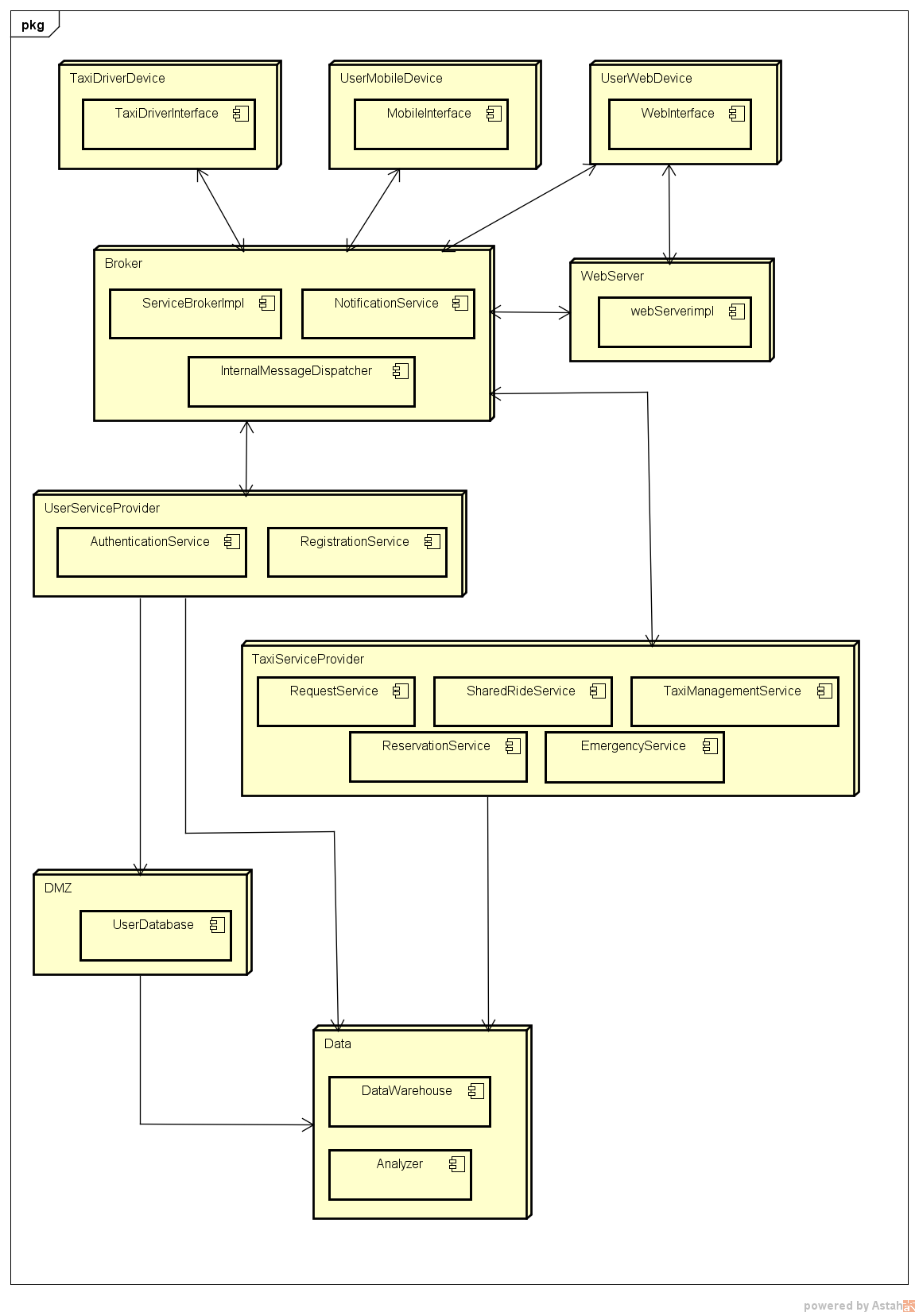
In this component diagram, the internal division of the Broker is implied in order to increase the overall comprehensibility.

 This diagram, instead, shows in more detail the division aforementioned. In this diagram, all the service providers and requestors are compressed in a single component in order to focus the attention on the interaction of the Broker’s parts.



## Deployment View

The division in node of the deployment diagram is just a logic one and it does not represents a constrain for a future hardware implementation. For this reason, the components are grouped only by functionalities, as they has been presented in the High level Components and their Interaction paragraph (2.2).



## Selected Architectural Styles and Patterns

The main architectural style designed for this system is the SOA one.

In fact, as described in the High Level Components and Their Interaction paragraph (2.2), is possible to recognize the SOA typical three-ways division. In fact, there are both the Service Requestors and the Service Providers, then the Service Broker that allows communication between each element.

We decided to design our project with this architecture for these reasons:

* Good management of a big number of users

Thanks to the division of the services that the system provides into atomic ones, the application is able to manage a large amount of different request in parallel

* Future developments

The division in services ensures a simple way for the addition of the new implementations and service upgrades made with the APIs considered in the RASD

* Scalability

The division in services allows a better management of the resources also in the case of an unexpected increase of the requests

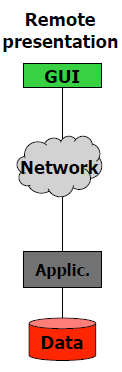
* Reliability

Having the possibility to add new servers to our cluster, we can guarantee a high level of availability. For example, we can have one or more auxiliary server that in case of failure can be easily used in order to keep on manage the external requests in a transparent way towards the external users.

* Maintainability

The quite small range of our application makes easy to handle the maintenance of the individual services. If the application becomes bigger, this point will forcedly be an issue

The other architectural style designed for this system is a Client-Server, regarding all about the management of the website. In order to give the possibility to use our website from every html 5 browser, we have chosen, as the image below shows, a Two-tier architecture, where both data and application find place in the system server and the GUI is handle directly by the device’s browser.



## Other Design Decisions

We decided to use a Data Warehouse in order to exploit its peculiarity of extraction of knowledge from data, for the prevision of a possible future increase of the requests and the creation of statistics useful for the economy of the company.

Usage data can also be used to better manage the work division on the various components.

For every event notification from system to user interfaces and also from system to taxi interfaces we will use a push notification service provided by the operating system of the interfaces