*Computer Science and Engineering*

*SOFTWARE ENGINEERING 2*

*A.A. 2015 / 2016*

***MyTaxiService***

RASD

**R**equirements **A**nalysis and **S**pecification **D**ocument

***November 6th, 2015***

**REFERENCE PROFESSOR**: **AUTHORS**:

*Mirandola Raffaela Polidori Lucia*

*Regondi Chiara*

**Table of contents**

1. **Introduction…………………………………………………………………….**
   1. RASD: definition and main features**………………………………………...**
   2. Requirement Engineering: introduction and definition **………………………**
   3. myTaxiService: description of the given problem**……………………………**
   4. Scope**……………………………………………………………………...**
   5. Glossary**…………………………………………………………………...**
   6. Stakeholders identification**………………………………………………….**
   7. Reference documents**……………………………………………………...**
   8. Overview**…………………………………………………………………..**
2. **Overall Description**
   1. Goals**……………………………………………………………………….**
   2. Domain properties**…………………………………………………………..**
   3. Assumptions**…………………...............................................................**
   4. The system we propose: current context and future system**………………….**
3. **Actors Identification**
4. **Requirements…………………………………………………………………..**
   1. Functional Requirements**…………………………………………………...**
   2. Non-functional Requirements**………………………………………………**  
       ***4.2.1*** User Interface**………………………………………………………..**  
       ***4.2.2***  Documentation**……………………………………………………….**  
       ***4.2.3*** Quality of Service attributes**…………………………………………..  
       4.2.4** Architecture**…………………………………………………………..**
   3. Pseudo - Requirements (constraints) **………………………………………**
5. **Scenarios Identification**
6. **UML Modeling………………………………………………………………….**
   1. Use Cases: identification and description**…………………………………...**  
       ***6.1.1*** … …  
       ***6.1.2*** … …  
       … …  
       … …
   2. Class Diagram**……………………………………………………………..**
   3. Sequence Diagrams**………………………………………………………..**

***6.3.1*** … …

***6.3.2*** … …

… …

* 1. State Chart Diagrams / Activity Diagrams**……………………………………**

***6.4.1*** … …

***6.4.2*** … …

… …

1. **Alloy Modeling………………………………………………………………….**
   1. Introduction to Alloy**………………………………………………………...**
   2. Alloy modeling**……………………………………………………………..**
   3. Generated worlds**…………………………………………………………..**
2. **Used Tools……………………………………………………………………..**
3. **Working Hours…………………………………………………………………**

***1* Introduction**

* 1. **RASD: definition and main features**

This RASD (Requirements Analysis and Specification Document) is a document where it’s analyzed the requirements analysis and specification phase of the development process of the web application “myTaxiService”.

The publication is produced in strict compliance with the standard IEEE 830-1998 (Recommended Practice for Software Requirements Specifications).

We realize this document in order to help you understand the requirements of the system, explaining both the application domain and the system that will be realized.

This document represents the basis for the project planning and it’s useful to estimate its duration and its cost.

It is also used to carry out test activities, verification and validation and to control requirements changes and, consequently, the evolution of the software. Therefore we must not think of it as a static document, but as a document that evolves and improves until it reaches its final form.

***1.2* Requirement Engineering: introduction and definition**

The Requirement Engineering is the process of discovering the degree to which the software meets the purpose for which it was intended, by identifying stakeholders and their needs and by documenting these needs in a form that is available for analysis, communication and implementation. So, in other words, the Requirement Engineering is a process whose goal is to identify and communicate the purpose and the requirements of a software system.

In this document we will follow the Jackson and Zave approach, “The World and The Machine”, in order to carry out our reasoning process and identify goals, domain assumption and requirements of our application.

According to this approach, the system that has to be developed (software-to-be) is the “machine”, while the portion of the real world which is affected by the machine is the “world”.

So, Jackson and Zave distinguish between “world phenomena”, which represent all those phenomena (events) occurring in the world and that the machine cannot observed, and “machine phenomena”, which represent all those phenomena entirely located inside the machine.

However the intersection between this two sets of phenomena is not empty: there are some “shared phenomena”, that the world share with the machine. The Requirement Engineering must be able to identify the shared phenomena, that represent the interface between the world and the machine, identifying all the features the system should have in order to produce benefits for the external world.

Analyzing this sets of phenomena, we can define the goals of our system, the properties of the application domain and the requirements.

***1.3* Description of the given problem**

We will project myTaxiService, an online application for requesting or reserving a taxi in a large city.

The system that will be developed has to allow to the non-registered users to request a taxi. The system will answer to the request by informing the passenger about the code of the incoming taxi and the waiting time.

The system has also to allow the registration of new users and allow them to reserve a taxi by specifying the origin and the destination of the ride.

The system has to guarantee a fair management of taxi queues: it automatically computes the distribution of taxis in the various zone based on the GPS information it receives from each taxi. In this way, when a request arrives from a certain zone, the system has to forward it to the first taxi queuing in that zone. If the taxi confirms the system will send a notification to the passenger, otherwise the system has to forward the request to the second in the queue. The system will receive the information about the availability through a mobile application used by the taxi drivers to communicate with the system itself.

***1.4* Scope**

The software-to-be will be developed as a web base application that will let the user request of reserve a taxi for a specific ride, with notifications about the availability and the code of the incoming taxi, in addition to the waiting time.

***1.5* Glossary**

***1.6* Stakeholders identification**

Our “financial” stakeholder is the professor who assigned us this project.

We have to deliver this project to the professor within the end of January 2016, showing that we have understood the requirements analysis and design phases of the software development process.

The government of city X “virtually” gave us this project in order to simplify the access of passengers to the taxi service and guarantee an efficient management of taxi queues.

The government needs to have a product that works fine and respects all the specification given to us.

We think that our typical users can be:

* occasional users that sometimes can use this application to request a taxi when they need it
* habitual users that need regularly a taxi and that can exploit the registration in order to reserve a taxi for a specific ride

***1.7* Reference documents**

We wrote this document following the structure and the subdivision into paragraphs of the standard IEEE-STD-830-1993 and arising from old documents of previous projects.

***1.8* Overview**

This document is structured into five main parts:

* *Section 1*: Introduction  
  this section contains an explanation of the document, a description of the problem that will be faced and some other general information about the software.
* *Section 2*: Overall Description  
  this section gives information about the goals and the domain properties of the application, in addition to a brief description of the actual system and the software-to-be.
* *Section 3*: Requirements  
  this section gives information about all the actors involved and contains all the specific requirements of the software (functional, non-functional, constraints).
* *Section 4*: UML Modeling  
  in this section all the typical scenarios are identified, together with the main use cases;  
  in order to better understand all the functionalities of the software, UML diagrams are here provided, modeling all the specific requirements listed before.
* *Section 5*: Alloy Modeling  
  this sections contains a simple introduction to the Alloy notation and the alloy modeling of the previous requirements.

***2* Overall Description**

***2.1* Goals**

For our system it’s possible to identify the following minimum goals:

* [G1]: for every request, a taxi should arrive in the correct zone of the city;
* [G2]: for every reservation, a taxi should arrive in the correct zone at the time established;
* [G3]: all registered users must be able to call a taxi, since the time of registration;
* [G4]: all registered users must be able to book a taxi from point A to point B, at the time X;
* [G5]: all registered users must be able to receive a notification containing the code and waiting time when they require a taxi;
* [G6]: all registered users must be able to receive a notification including the confirmation of the reservation when they book a taxi;
* [G7]: all unregistered users have to able to register to the application;
* [G8]: all taxi drivers must be able to notify the system with their availability;
* [G9]: all taxi drivers must be able to notify the system saying that he’s busy because he’s taking care of another request;

***2.2* Domain properties**

We suppose that the following conditions hold in the world we analyze:

* for each reservation the origin and the destination of the ride, the time in which the reserved taxi should arrive, in addition to the name of the person who does the request are given;
* for each call the location and name of the person who does the request are given;
* when a taxi is mobilized, it will reach the site of the request as quickly as possible;
* when a reservation is done, the system allocated a taxi 10 minutes before the meeting time with passenger;
* accurate taxi location are known by the GPS signal of each taxi;
* the city is divided into zones and each zone is associated with a taxi queue.  
  Each taxi is queued in his area according to the position obtained by the GPS;
* when a taxi is busy, it is placed at the bottom of his queue;
* availability of taxis are known thanks to a mobile app owned by each driver;
* every reservation has to be done at least 2 hours before the ride;

***2.3* Assumptions**

There are few points that could result not very clear in the speciﬁcation document, so we will have to assume some facts:

* only users who have already registered to the application can exploit the request and reservation functionalities;
* all registered users can call a taxi: when a taxi is coming , the user is "waiting for taxi" and cannot call others;
* a registered user cannot cancel the immediate request for a taxi;
* a registered user can cancel a reservation up to 30 minutes before the reservation time;
* a registered user who has reserved a taxi is "waiting to taxi" from 30 minutes before the reservation;
* when the taxi driver receives a call, his state changes from "available" to "occupied";
* when the taxi driver finishes a race, his status changes from "occupied" to "available";
* when the taxi driver takes care of a reservation, its status remains "occupied" 10 minutes before the meeting until the end of the race;
* when a request from a certain area is received, the system checks taxi queue of that area, and sends a request to the first taxi in the list. If the taxi confirm its availability, then the system sends the confirmation to the passenger, otherwise the taxi is moved at the end of the queue;
* the app used by the drivers is installed on board, on a mobile data terminal placed in each taxi

***2.4* The system we propose: current context and future system**

MyTaxiService is a completely new software. Currently, there are no applications of this kind, so our project is not an evolution of something already existing, but it’s something new, built of the specific requests of our customer.

Currently, the taxi service of this city can be accessed only by the telephone and the management of the taxi queues in the different zones of the city turns out to be pretty complex with the growth of the requests.

We propose a web based platform that allows a user to register, and after the registration, to call a taxi which will start immediately or reserve one for a specific race at a particular time. The system will notify the taxi code and the wait time for the immediate demand for taxi, and will send a confirmation notification for the reservation of the taxi. The system allow you to only delete the reservation and not the immediate demands of taxi. The server will generate different user experience and different pages depending on the actor who is at that moment using the platform.

Together with the web application, our system will provide a mobile app for the taxi drivers, so that they can communicate their availability or their tasks to the system, in order to allow a better management of the taxi queues and of all the requests or reservations received.

***3* Actors Identification**

GUEST: he can visit the homepage and register into the system, creating a personal account, in order to become a registered user.

REGISTERED USER: after successful login, he can access to his personal profile, modify his personal information and immediately request or reserve a taxi filling out the dedicated forms.

He can receive notifications about the incoming taxi, visualize his past and current reservations and eventually decide to delete one of the current ones.

TAXI DRIVER: he is already pre-registered into the mobile application installed on his taxi and can use it to answer to specific requests forwarded by the system, communicate his availability and eventual calls he’s taken care of.

**4 Requirements**

According to the model of The World and The Machine presented by Jackson and Zave, assuming that all the domain properties and the assumptions we have defined in the previous chapter hold in the world, we have identified the requirements that satisfy the goals described earlier in the document.

These requirements are the main features characterizing our system:

* Registration  
   - the system shall provide a “sign up” functionality;  
   - when a new registration is done, the system shall create a new profile in the   
   passengers database and allow the new user to log in;  
   - the system shall store user information including name, surname, email, mobile   
   phone number and credit card.
* Log In  
   - the system shall provide a “log in” functionality to access the application;  
   - the system shall grant the access to the application only after a user types a   
   correct username and password;  
   - when the log in is corrected, the system shall show the user his personal page;  
   - the system shall show an error message whenever the authentication is wrong.
* Insert, edit, delete personal information   
   - the system has to allow the user to insert new personal data inside his profile  
   information;  
   - the system has to allow the user to modify existing personal information;  
   - the system has to allow the user to delete existing personal information.
* Request a taxi  
   - the system shall provide a “request” functionality, allowing a passenger to request   
   a taxi when he wants and where he wants;   
   - the system has to forward each request to the first taxi situated in the requested   
   zone, in order to check its availability, and then mobilize the first available taxi in   
   the queue of that zone;  
   - the system shall move a taxi to the bottom of the queue if not available for one of   
   the current requests;
* Reserve a taxi  
   - the system shall provide a “reserve” functionality in order to allow a passenger to   
   reserve a taxi for a specific ride;  
   - when a reservation is received at least two hours before the ride, the system   
   confirms it to the passenger;  
   - when a reservation is not done at least 2 hours in advanced, the system shall show   
   an error message;  
   - the system shall allocate for every reservation one of the available taxis 10 minutes   
   before the established meeting time;  
   - the system shall provide a function that allows a user to visualize the list of   
   reservations he has done.
* Drivers communication  
   - the system shall be able to receive communication and information about   
   availability from each taxi driver;  
   - the system shall store all the information about each taxi’s availability;  
   - when a request from the system is received, a taxi driver should communicate its   
   availability to the system itself;  
   - when a taxi drivers decides to take care of certain calls, he should inform the   
   system.
* Localization  
   - the system has to be able to accurately localize the distribution of the taxis in each   
   zone using a GPS locator.
* Notifications  
   - the system has to notify a user, confirming his request, when a taxi is available  
   - after confirming a request, the system shall communicate to the passenger the   
   code of the incoming taxi and the estimated waiting time;  
   - the system shall confirm a successful reservation sending a notification to the   
   passenger.
* Other services offering  
   - the system should offer the possibility of exploiting all the other different services   
   among the offered ones.

***4.1* Functional requirements**

After defining in the previous paragraph the main features of MyTaxiService, we can now identify some functional requirements for each actor involved:

* Guest: he can
* sign up
* Registered user: he can
* log in
* modify his profile information
* request a taxi compiling the related form
* reserve a taxi compiling the related form
* receive a taxi request confirmation
* receive information about the code of the incoming taxi and the waiting time
* receive a notification confirming a reservation
* visualize all his previous and current reservations
* exploit all the other different services offered by the application
* Taxi driver: he can
* send information about his availability
* send information about the calls he takes care of
* receive requests about his availability
* receive information about the location of the requests he has to satisfy

***4.2* Non-functional requirements**

The non-functional requirements describe all those aspects of the system not directly related to its behavior and its functionalities.

***4.2.1 User interface***

The interface of our passenger application is thought to be used via web, while the interface for the taxi drivers is offered through a mobile application installed on board on a data terminal.

We want the web application to be a very user friendly interface, minimal and easy to use for all the passengers: buttons to come back to the homepage or to the personal profile page and to logout are always visible.

Here are now presented some mockup that represent our idea of the structure of the application pages:

***Homepage***

The mockup below show the homepage of MyTaxiService: here users can decide to log in, if he’s already registered, or sign up, in order to create a new account.

******

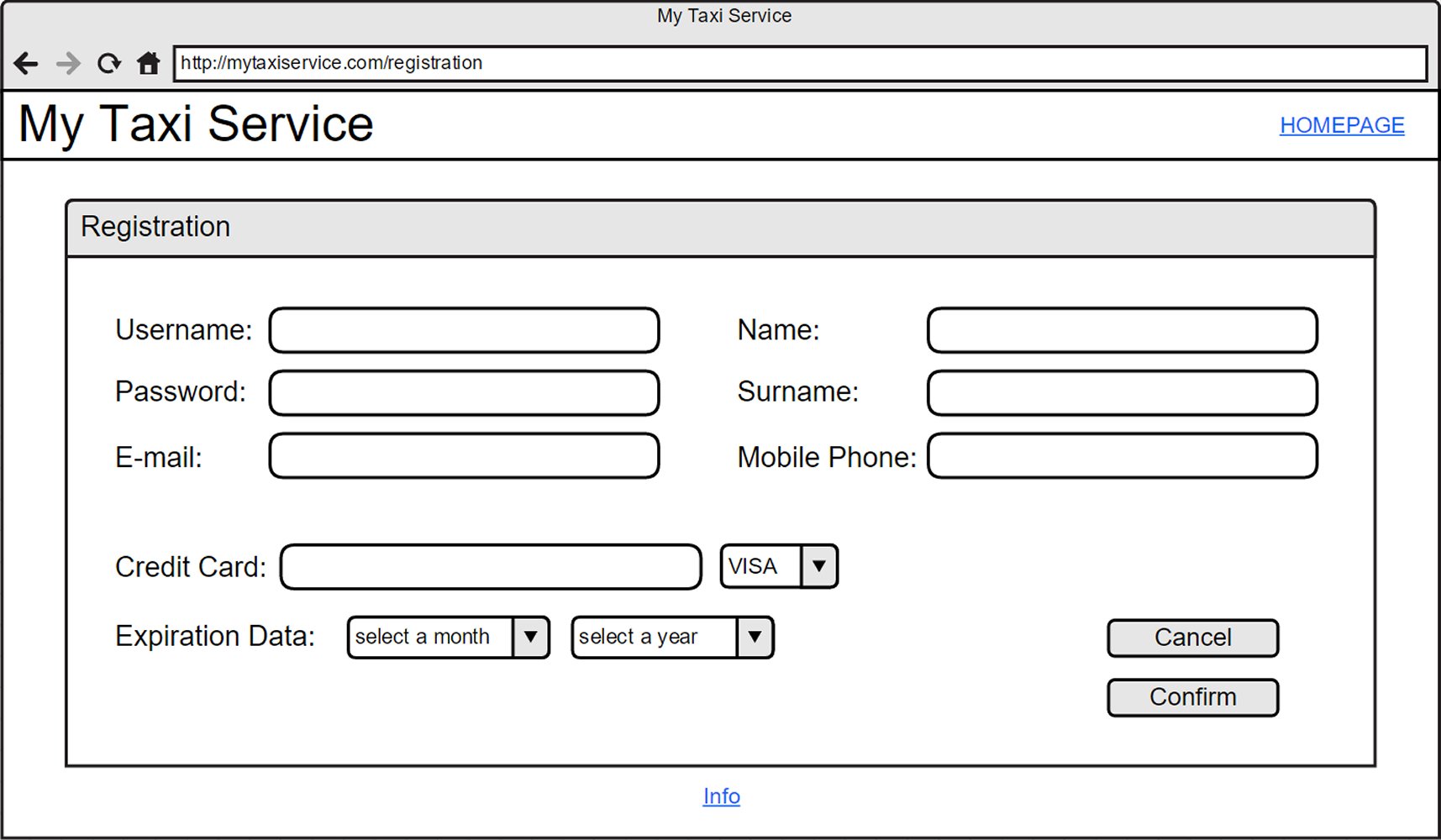
***Log In***

The mockup below show the log in page: here users can access the system filling out the form with their credentials.



***Registration***

The mockup below show the registration form page: a guest has to fill out the form in order to became a registered user and create his personal account.



***4.2.2 Documentation***

We decide to produce and release the following documents in order to better explain our project and all the development phases:

* *RASD* (*Requirements Analysis and Specification Document*), in order to understand better and explain efficiently our project, defining all the goals, domain properties and assumptions, describing all the main features of the systems and modeling all the requirements and specifications through specific notations.
* *DD* (*Design Document*), in order to define the real structure, the architecture and eventual design constraints of our application.
* *Inspection Document*, in order to prove our code inspections and bug identifications abilities, analyzing an existing well-known open source project
* *Testing Document*, containing the results of a test analysis on the same project considered for the inspection
* *Function Points Document*, in order to assess the effort and the cost required for the development of our own application

***4.2.3 Quality of Service attributes***

* ***Usability***: the web application is addressed to different varieties of people, with different ages and different skills and level of knowledge concerning informatics systems.  
  The web application has to be user-friendly, minimal and quite intuitive, in order to be used without particular problems by every kind of user. In order to achieve this goal, not too much functionalities has to be developed and the user interface shall be simple and intuitive.
* ***Performance***: the application has to be able to manage a high number of users connected at the same time and a more than one request or reservation done simultaneously.  
  The response time (confirmation and communication about the details of the incoming taxi) to a specific request has to be reasonable, within at most 3 minutes since the request.
* ***Accessibility***: the application shall be available every day, from 7:00 am to 7:00 pm;
* ***Reliability***: the system shall be reliable and it has to be able not to lose passengers data contained in the database even in case of breakdowns or damages. The system shall do a daily backup of all the available data in order to grant reliability.
* ***Security***: every user shall access only to the functionalities he is allowed to. For this reason an authentication mechanism exploiting access credentials is needed.

***4.2.4 Architecture***

According to the requirements specified below, as for the architecture, the system will need one or more dedicated servers and a database to store all the user’s information.

Eventually another persistent and stable memory system will be added in order to allow the backup of all the data and increase the system’s reliability.

An internet connection is required to access the application.

***4.3* Pseudo-requirements (constraints)**

The so called “Pseudo-requirements” simply represents constraints on the system imposed by the client or the environment in which the system operates.

MyTaxiService doesn’t have particular constraints concerning interfaces or specific operations for the management of the system.

However it is subjected to constraints concerning the delivery and the tools needed in order to use the software.

As for the tools, it’s necessary for the user to have an internet connection and to use a web browser that allows the navigation. Also the taxi drivers has to have installed on board a mobile data terminal which allows the installation of the mobile app they should use to communicate with the system.

As for the delivery, the project and all the related documents have to be completed within an fixed period of time.

Here the deadlines to respect:

* October 14th, 2015: *Group Registration*
* November 6th, 2012: *Requirements Analysis and Specification Document (RASD)*
* December 4th, 2015: *Design Document*
* January 5th, 2016: *Inspection Document*
* January 21st, 2016: *Testing Document*
* January 30th, 2016: *Function Points*