POLITECNICO
MILANO 1863

Project Plan

Andrea Gussoni Federico Amedeo Izzo Niccolò Izzo

February 2, 2016

Contents

Chapter 1

Introduction

1.1 Revision History.

Revision 1.0

1.2 Purpose and Scope

1.2.1 Purpose

This document outlines the project plan for the development of the **myTaxiService** case. We will structure the document as follows:

- A brief introduction containing all the necessary informations for a correct understanding of the document, including the references to the previous documents.
- Two sections section where we apply the **Function Points** and **COCOMO** methodologies to estimate the project size and the effort and the risk related to it.
- A section dedicated to the identified schedule for the project (starting as requested from October 2015).
- A section that defines how we plan to allocate the available resources (the members of our group) on the development of the project.
- A section devoted to the evaluation of possible risks correlated to the development of the project.

1.2.2 Scope

The system to be developed is **myTaxiService**. This service aims to offer a simplified and reliable access to the preexisting taxi infrastructure of the city. Both taxi drivers and customers will benefit from this product. For example the customers will be able to:

- Have access to a taxi more quickly.
- Reserve in advance a taxi ride with fixed origin and destination.
- Share the taxi fee with others passengers going in the same direction.

And also the drivers will have a number of benefits with the adoption of our system, amongst them there are:

- Fair distribution of the customers.
- Virtual waiting queues instead of physical ones.
- Customer geographical localization.
- Automatic route planning.

1.3 List of Definitions and Abbreviations

1.3.1 Definitions

- Customers: those who will request the ride through the web application or the mobile application.
- Taxi drivers: registered users of the mobile application. They will upload their position and their availability to take rides to the system.
- Standard ride: action that begins with the customer's ride request and ends with the customer's payment at the end of the ride.
- Reserved ride: a ride that has been reserved at least two hours before the starting time. It begins from the reservations and ends with the customer's arrival at his destination.
- Shared ride: a different type of ride in which the customer give his availability to share the ride. A ride is considered shared when at least two customers are traveling in the same taxi cab.

- Smart-phone: a mobile device capable of connecting to the Internet and making and receiving calls and SMS.
- Geo-localization: the act of obtaining a user's geographic coordinates, eventually uploading them to an on-line service.
- Application: mobile or web app running on the user's device.
- System: refers to the part of the application logic that runs on the remote server.
- Taxi Zone: is an area of approximately $2km^2$ for which the taxi-queue is unique.
- Developers: all the people involved in the development of the Service.
- Stakeholders: all the people that may be affected by the Service activities.
- Scalable: used in describing the capability of adapting the resource usage in accordance to an increase/decrease of number of incoming requests.

1.3.2 Acronyms

- DD: Design Document.
- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface.
- UI: User Interface.
- OS: Operating System.
- UX: User eXperience.
- SOA: Service Oriented Architecture.
- IaaS: Infrastructure as a Service.
- SaaS: Software as a Service.
- PaaS: Platform as a Service
- MVC: Model-View- Controller.

- OO: Object Oriented.
- Java EE: Java Enterprise Edition.

1.3.3 Abbreviations

- Web app: Web-based application.

1.4 List of Reference Documents

Here a list of the documents we used as reference in the drafting of this document:

- **Project goal, schedule and rules** provided us by the professor of the course of Software Engineering 2.
- **Assignment 5: project Plan**
- Some examples of Function Points and COCOMO estimations from the previous years' project.
- **RASD**, [4]
- **Design Document**, [3]
- **Integration Test Plan**, [5]

Chapter 2

Functions Points Estimation

In this section we will report and use information and tables taken from [1].

2.1 Reference Tables

2.2 Tables for FP Counting Weights

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

Table 2.1: **For Internal Logical Files and External Interface Files**

	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

Table 2.2: **For External Output and External Inquiry**

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

Table 2.3: **For External Input**

2.3 Table for UFP Complexity Weights

	Complexity Weights		
Function Type	**Low**	**Average**	**High**
Internal Logical Files	7	10	15
External Interface Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Table 2.4: **For UFP Complexity Weights**

2.3.1 Relating UFPs to SLOC

For converting the obtained Unadjusted Function Points into Lines of Code we don't use the coefficient provided in the manual cited above as it doesn't seem to include one for J2EE, but instead we use the one found in [2], that is equal to **46**.

2.4 Identification of Function Points for our system

2.4.1 Internal Logic Files

The Internal Logic Files (abbreviated as ILF) are *homogeneous set of data used and managed by the application.* MyTaxiService uses various set of logic files, each one with its own purpose. The first kind of set is represented by the user's account informations, they can be listed as following:

- Username -> String
- Email -> String

- Password Hash -> Boolean[512]

Also the taxi drivers have their own accounts with related fields:

- ID -> Integer
- Name -> String
- Password Hash -> Boolean[512]

Furthermore, each virtual taxi queue carries a couple of variables:

- ID -> Integer
- Area -> Float[4]
- Queue -> Driver[N]

While the ID interpretation is trivial, the floats are used to store a geographical rectangle (stored as its own boundaries). In addition to this each individual ride have its data fields:

- ID -> Integer
- Drive -> Integer
- Passengers -> Integer[4]
- IsShared -> Bool
- StartTime -> Float
- EndTime -> Float
- Departure -> Float[2]
- Arrival -> Float[2]

To sum up here are the internal data structures used by the service, each one with the corresponding Complexity and Function Pointers.

Logic File	**Complexity**	**Function Points**
User Account	Low	7
Driver Account	Low	7
Virtual Queue	Low	7
Ride	High	15
Total		36

Table 2.5: **Internal Logic Files.**

2.4.2 External Interface Files

The External Interface Files (abbreviated as EIF) are *homogeneous set of data used by the application but generated and maintained by other applications.* MyTaxiService uses only one external service to fulfill its purposes, this service is a geographical maps retrieve service, provided by the OpenStreetMaps APIs. The interaction with OpenStreetMaps can be summed up as a single operation: MapFetch(). This function takes as parameters a bounding box made of geographical coordinates, and will receive from the service the vector maps data regarding the selected area.

The External Interface Files, with the corresponding function points can be synthesized as following.

Interface File	Complexity	Function Points
MapFetch	High	10
Total		10

Table 2.6: **External Interface Files.**

2.4.3 External Input

An External Input (abbreviated as EI) is a *elementary operation to elaborate data coming form the external environment.* MyTaxiService handles various kinds of external input, they can be divided into several classes:

- Login/Logout: This input is composed by the user credentials and related password, these fields are a String and an array of 512 bits.
- Taxi Call: A taxi call carries some simple parameters which are the user ID, the user starting position, and some optional parameters depending on the call nature, such as the optional shared bool flag and the desired arrival position.
- Ride Reservation: A ride reservation carries the same parameters as a Taxi Call, of course in addition to this the request carries data about the time of departure that will be used to schedule the reserved ride.
- Driver position update: Our service receives the position refresh data from the taxi drivers mobile applications. Each application, whenever it detects a significant change in the user position uploads data about the new driver's position to myTaxiService.

- Insert/Update User: This particular request carries the personal data about the user, which include the user's username, email, hashed password, and personal informations.

We sum up the results of the External Input in the following table.

Input	Complexity	Function Points
Login/Logout	Low	3
Taxi Call	Avg.	4
Ride Reservation	Avg.	4
Driver Position Update	Avg.	4
Insert/Update User	High	6
Total		21

Table 2.7: **External Input.**

2.4.4 External Output

An External Output (abbreviated as EO) is a *elementary operation that generates data for the external environment.* To fulfill his purposes, myTaxiService has to send various kinds of data to the agent that interacts with it, they are:

- Ride Confirmed: This output carries an Integer representing the ID of the taxi who has accepted the ride.
- Taxi Arrival: This output is similar to the Ride Confirmed output, it carries the ID of the target taxi, plus it includes a string containing the serial code of the taxi, useful for the identification of the right cab.
- Shared Ride Notification: This notification carries informations about the person or people which are going to share our ride. Thus carrying the name of those people and some information about the route that the cab will take and the fee division coefficients.

The Function Points obtained by the External Outputs of the application can be summarized as following.

Output	Complexity	Function Points
Ride Confirmed	Low	4
Taxi Arrival	Low	4
Shared Ride Notification	Avg.	5
Total		13

Table 2.8: **External Output.**

2.4.5 External Inquiry

An External Inquiry (abbreviated as EQ) is a *elementary operation that involves input and output, without significant elaboration of data from logic files.* MyTaxiService uses various kind of interactive communication with its own users, amongst them there are:

- User Profile: This response yields informations about the current user profile, including the current username, email and personal informations, including an optional profile picture.
- Taxi ETA: This response yields a value expressing the estimated time of arrival of the current cab.
- Estimated Cost: The estimated cost response will return, an estimation of the trip cost considering the fixed fees and a rough estimation of the variables fees evaluated in the planned route.
- Shared Ride Route: This response will return a list of geographical points, representing the estimated route that will be followed during a shared ride.

The Function Points obtained by the External Outputs of the application can be summarized as following.

Inquiry	**Complexity**	**Function Points**
User Profile	High	6
Taxi ETA	Low	3
Estimated Cost	Low	3
Shared Ride Route	High	6
Total		18

Table 2.9: **External Inquiry.**

2.4.6 Function Points Results

Inquiry	Function Points
Internal Logic Files	36
External Interface Files	10
External Input	21
External Output	13
External Inquiry	18
Total	98

Table 2.10: **Function Points Results.**

2.5 Summary

As we sum up all the function points of our system we obtain a value of **98**. We multiply this value with the coefficient of our programming language. This coefficient, as stated before is **46**. The result of the computation are **4508** Source Lines Of Code for the realization of our system.

Chapter 3

COCOMO II Estimation

The COCOMO estimation is a methodology that use a complex *non-linear* model that takes in account a lot of different factors, not only related to the characteristics of the final product, in order to provide an accurate estimation of the effort.

As for the function points chapter we refers to [1] for tables and procedures used in the estimation process.

3.1 Scale Drivers

Here a first list of **Scale Drivers** that have to be evaluated:

Precedentedness : reflects the previous experience of the organization with this type of project. In our case this value will be **low**, due to the fact that for us this was the first work in which we had to use the advanced methodologies of software engineering and the Java EE framework.

Development flexibility : reflects the degree of flexibility in the development process. We will set this value to **nominal** because the specifications that we had were not too strict, and in the development project there had been some occasions were we could influence in a consistent way the process.

Risk resolution : reflects the extent of risk analysis carried out. In our cause this value will be **high**. Indeed we tried to do an extensive risk analysis, but due to the fact that as mentioned before this was our first experience in this field we will not give an higher value to this field.

Team cohesion : reflects how well the development team know each other and work together. We will assign to this field the value **high**. At first

we had some problems in finding a efficient methodology to synchronize the work, often resulting in duplication of the work, loss of time and similar things, but then we found a more productive approach to the work and the coordination overheads and problems decreased.

Process maturity : reflects the process maturity of the organization. For evaluating this field we use the 18 Key Process Area (KPAs) in the SEI Capability Model. The value obtained is 2.5, that we will round to 2 that is **nominal**.

Scale Driver	**Factor**	**Value**
Precedentedness	Low	4.96
Development flexibility	Nominal	3.04
Risk resolution	High	2.83
Team cohesion	High	2.19
Process maturity	Nominal	4.68
Total		17.7

Table 3.1: Scale Drivers estimations for the system.

3.2 Cost Drivers

- Required Software Reliability: This is the measure of the extent to which the software must perform its intended function over a period of time. In our case we assign to the field the value **high** due to the fact that a failure of the system can potentially cause a huge financial loss for the taxi drivers.

- Data base size: This cost driver attempts to capture the effect large test data requirements have on product development. For this particular driver we don't have enough information to do a reliable estimation, so we set this value to **nominal**

- Product Complexity: Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. In our case we decided to assign a value **very high** on the basis of the complex operations that our systems has to perform with the queue management, with the distributed database scenario and with a moderate complex user interface.

- Required Re-usability: This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. In this case we decide to set the field to a value equal to **nominal** considering that no particular requirements on this aspect have been specified.

- Documentation match to life-cycle needs: The rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs. Also in this case we set the value to **nominal**, a value that should permit an adequate documentation also in anticipation of some future extensions.

- Execution Time Constraint: This is a measure of the execution time constraint imposed upon a software system. In our case we decided to set this value to **nominal**, considering the fact that the techniques mentioned in the DD [3] for the dynamic allocation of resources should maintain the load of the system in an acceptable zone.

- Main Storage Constraint: This rating represents the degree of main storage constraint imposed on a software system or subsystem. Also this value should be put to the **nominal** value for the same reasons of the previous case.

- Platform Volatility: "Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. We decided to set this value to **high** because especially for the client side is imperative to stay updated with the changes of the operative systems of smartphones and to correct potential bugs in a very short time to stay on the market and avoid competition.

- Analyst Capability: Analysts are personnel who work on requirements, high-level design and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. We decided to set this value to **nominal** because as stated before this is our first experience with this particular methodology of development.

- Programmer Capability: Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. In our case we think that an appropriate value for this factor is **high** considering that this is our particular field of specialization.

- Application Experience: The rating for this cost driver (formerly labeled AEXP) is dependent on the level of applications experience of the project team developing the software system or subsystem. In our case for obvious reason we set this factor to **very low**.

- Platform Experience: The Post-Architecture model broadens the productivity influence of platform experience, PLEX (formerly labeled PEXP), by recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities. We decide to assign a value **nominal** considering that at this point of our studies we have some experience with the platform used in this project (excluding Java EE).

- Language and Tool Experience: This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. We assign to this factor a value equal to **low** considering that this is our first experience with most of the tools used in the development of this project.

- Personnel Continuity: The rating scale for PCON is in terms of the project's annual personnel turnover. In our case is obvious that this value should be set at **very high**.

- Usage of Software Tools: The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high. We think that in our case we can assign a value **high** at this factor.

- Multi-site Development: Determining this cost driver rating involves the assessment and judgement-based averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia). In our case a value of **high** seems adequate.

- Required Development Schedule: This rating measures the schedule constraint imposed on the project team developing the software. In our case no particular constraints have been specified, so a value of **nominal** should be fine.

Cost Driver	Factor	Value
Required Software Reliability	High	1.10
Database size	Nominal	1.00
Product Complexity	Very High	1.34
Required Re-usability	Nominal	1.00
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	Nominal	1.00
Main Storage Constraint	Nominal	1.00
Platform Volatility	High	1.15
Analyst Capability	Nominal	1.00
Programmer Capability	High	0.88
Application Experience	Very Low	1.22
Platform Experience	Nominal	1.00
Language and Tool Experience	Low	1.09
Personnel Continuity	very High	0.81
Usage of Software Tools	High	0.90
Multi-site Development	High	0.93
Required Development Schedule	Nominal	1.00
Total		1.34

Table 3.2: Cost Drivers estimations for the system.

3.3 Effort Equation

The general form for calculating the effort using the COCOMO II methodology is like this:

$$\text{Effort} = A \times EAF \times KSLOC^{E} \tag{3.1}$$

Here an explanation of all the parameters of the equation:

- $A = 2.94$
- $EAF = \prod_i C_i$: product of all cost drivers, equal to **1.34**
- $KSLOC$: Kilo-Source Lines of Code, estimated in the Function Points chapter, equal to **4,5**.
- $E = 0.91 + 0.01 \times \sum_i SF_i$: calculated in the scale drivers section, equal to **1.09**

The total effort results in **20.3** persons months.

3.4 Duration

$$\text{Duration} = 3.67 \times (PM)^{0.28+0.2\times(E-0.91)} \tag{3.2}$$

Here an explanation of all the parameters of the equation:

- PM is the effort calculated previously
- E is the exponent calculated in the scale drivers section.

This equation brings us to a estimated duration of **9.5 months**.

Using the following formula to calculate the number if persons to allocate to the project:

$$\text{Number of developers} = \frac{\text{Effort}}{\text{Duration}} \tag{3.3}$$

This brings to an estimation of **2.2** developers to allocate to this project. This seems an excessive underestimation, and also since our team is composed by 3 people, on the basis of the estimations obtained until now we can divide the effort obtained by the real number of people and obtain an estimation of **6.7** months for completing the project.

This really seems an underestimation, maybe caused by a non perfect and complete analysis of all the COCOMO II parameters(considering that this is only a simulation and we don't have all the necessary information that a real company have to use the model).

So we think that allocation a period of at least **9/10** months and a team of **3** people for the development of this project should be a good idea.

Chapter 4

Task Identification and Scheduling

The tasks involved in our project are:

1. *Requirement Analysis and Specification Document* [4] delivery.
2. *Design Document* [3] delivery.
3. *Testing Plan Document* [5] delivery.
4. *Project Plan* delivery.
5. Delivered documents presentation to the project committee.
6. Software development and unit test verification.
7. Integration testing on the system.

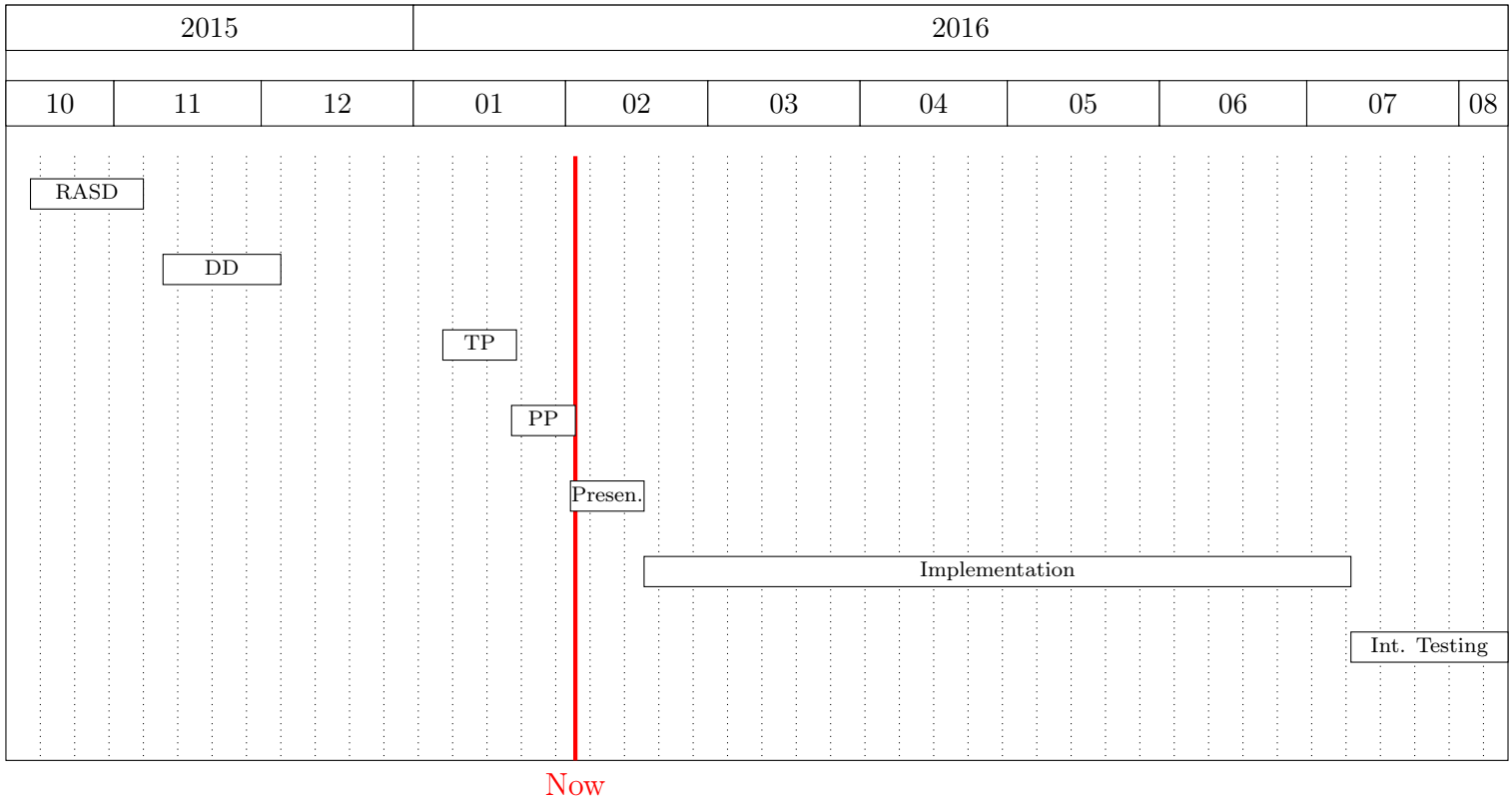

Figure 4.1: Gantt chart of the project.

Chapter 5

Resources Allocation Plan

In this chapter we will show an overview of how the various tasks necessary to the completion of the project will be distributed among the team members.

As stated before our team is composed by 3 persons, **Andrea Gussoni**, **Federico Amedeo Izzo** and **Niccolò Izzo**. From now on we will refer to the team members only with their first names.

Obviously we tried to allocate the work in a way were each of us could contribute in an equal way, and as stated before for the main tasks we tried to do as much teamwork as possible.

In general for the tasks that we completed until now we tried to use this schema for organizing the division of the work:

First Brainstorm In this phase we usually met in person in university after the lesson in which the assignment was presented in order to do a first exchange of ideas, identify the a possible division into sub-problems of the whole assignment and estimating the effort of each one.

Task Allocation Then in the first weekend after the explanation of the assignment we organized a conference call (in the weekend Andrea usually returned to his hometown near Varese so it was impossible to meet in person) in which we tried to find a first skeleton for the assignment, and decide the task allocation. Usually each one of us chose the set of tasks that he preferred, and another team member proposed himself to do the revision of that parts after their completions. In this phase we also tried to define a first deadline to respect for the completion of the work. Also we tried to identify tasks that needed to be completed by at least two members, and tried to find a possible time slot during the week were to do the work.

Work When the tasks were allocated to the team members each one of

us allocated the work in his available time slots, which very different among the members. For example Andrea usually postponed the work during the week to do it in the weekend maybe allocating an entire day only for that. Niccolò and Federico usually preferred to spread the work in lunch-breaks or in the evenings after the lessons.

Final Cross-Overview and Delivery After the decided deadlines each one of use reviewed the parts done by the others (as stated before each part has been reviewed at least by another member), suggesting corrections or raising observation and doubts. At this point usually we compiled the final version of the delivery and upload it to the official repository.

For what concerns instead the implementation and part we came up with this division of work similar to this:

- Basic structure of the business logic: In this task all three of us will work simultaneously . This because in this part we will define and develop the common structure of all the system and is essential that each one of us understand correctly the basic functionalities.

- Advanced functions (reserved ride and shared ride): We decide to allocate Andrea to the reserved ride functionality development, and Federico and Niccolò the the shared ride functionality development that is more complex.

- Front End design: This task will be assigned to Niccolò and Federico, that are more familiar with design techniques and user interface development.

- Web Tier development: This task will be allocated to Andrea, that is more familiar with web servers in general and the deployment of web servers.

- Database and Business Logic integration: For this task we will allocate Andrea that will work full time on this and Federico that will allocate half of his work in this phase for this task. Andrea has some experience with deployment of databases and Federico will help also coordinating the work with the next task.

- Front End and Business Logic integration: For this task we will allocate Niccolò full time and Federico half time. Niccolò already has some experience in this kind of work.

- The role of Federico will be to support both Niccolò and Andrea in the the two task and mainly to provide coordination between the two phases. Due to the fact that he is involved in both the tasks he will reports problems between the two tasks, and help resolving them given the fact that he is involved in both the tasks.

Chapter 6

Risk Definition and Recovery Actions

In this section we will discuss about an important aspect that is the risk analysis for our project. It is important not to underestimate the weight of this task because otherwise we can incur in major and potentially critical failures, errors or problems that we haven't foregone.

There various type of potential risk in which we can incur:

- Risks related basically to the project development procedure.
- Risks related to business factors (like the financial situation of the company).
- Risk related to technical factors (for example the software or hardware failures that can occur during the normal operations of the final system).

6.1 Project Risks

We can analyze in detail the risks concerning the project development procedure:

Delays on the expected schedule Due to the fact that our team is at his first experience with the methodologies of development that have been taught us in the software engineering 2 course and also with the Java EE framework there is an high chance that the development will suffer of delays. This can bring to a situation where we won't have a complete functional version of the system. To mitigate this potential problem we can approach the development focusing on the main functionalities

that the final product should have. For example in the case we detect some delays we can postpone the development of advanced features (such as the shared ride functionality) and we can focus on the basic ones, in order to release a first version of the software in time with the expected schedule.

Team Organization Also relating to the problem explained above we can expect that the coordination between the team members can suffer. We already noticed this in the work done: for example each one of us has its own personal schedule, it prefer to work in a certain setting (at home, at university), in different time slots (during the day, during the night). Also one team member can be more productive in performing alone a single task, avoiding the communication overheads. Someone else instead can benefit from team-work and perform better in this situation. In general the job of diving the work between the team members can be a difficult aspect of the work, and a bad division can cause excessive communication overheads and problems due to the synchronization of various task, maybe due to a chain of work-dependencies not satisfied.

Change of requirements We can have also risks due to a change of requirements after the analysis phase. We can't take particular actions against this issue, we can only react in the fastest way possible and inform of this problem the stakeholders in order to get an immediate notice of eventual changes in the requirements.

6.2 Business Risks

For what concerns the business factors:

Financial problems The expected income from the sales of the product can be insufficient to cover the development expenses and to guarantee the necessary profit. This type of risk can have severe effects and must be avoided by putting the a lot of attention in the feasibility study phase.

Competition We may incur in a situation in which a third entity came up with a product that places itself in the same market-segment of myTaxiService. The only way to avoid this is to carry out policies to ensure that the employers can't work for a society that is in competition with us. We can think at NDA agreements or similar policies.

Laws Last but not least we have the risk of changes in the regulation policies of the taxi market, the mobile application market and correlated aspect. We can't predict this kind of issue, the only thing we can do is to stay updated on this field and maybe find a professional figure that can help us with this kind of work.

6.3 Technical Risks

Then we also have to consider potential technical risks that we can face:

Hardware Failures During the normal operations of the systems the eventuality of a hardware failure must be always taken into consideration. Thanks to the use of dynamic resource allocation and the virtualization of the computational resources (as explained in the DD document [3]) we expect to be able to control this kind of issue and to perform very fast recovery actions. Anyway we have to put particular attention in the management of the data on which our application relies. We need to introduce very substantial policies regarding the backup plan adopted, considering various types of solution, always having physically separated copies of backups.

Software Failures We also have to take into consideration possible failures of the software side of the system. For example we can have major failures during the integration phase of the various components and subsystems. This kind of behavior should be averted with an adequate quality of the software, but we cannot exclude a priori this kind of possibility. The only way to avoid this is to begin early in the development phase to integrate as much components and subsystems as possible, an to pay particular attention in the design of interfaces.

Bad code quality Another risk that we can face is having a system that works very well and as expected but that has a very low quality source code and very poor documentation. A way to avoid this is to perform regular **code inspections**.

Chapter 7

Appendix

7.1 Used Software

For the redaction of this document we used various software, here a little list:

- LaTeX framework (MacTeX on OS X and TeX Live on GNU/Linux) to generate the document.
- Various editor to edit the source file:
  - Sublime Text 3 (Beta)
  - Atom.io
  - Vim
- Self-Hosted ShareLaTeX to collaboratively edit the document.
- Git to version the source, GitHub to host the repository.
- Gimp, Inkscape and ImageMagick to edit some images (.svg to .png).
- Draw.io for drawing some diagrams.
- Teamspeak and Hangouts used to organize conference-calls in order to work together.

7.2 Hours of Work

We tried to distribute in an equal way the workload for each team-member. In particular we tried to arrange physical meetings or conference calls for the parts where important choices had to be made. We estimated that each of

us spent on average 12 hours on the drafting of this document. Therefore this task took a total of, more or less, 36 hours of work.

Bibliography

[1] COCOMO II - Model Definition Manual, Version 2.1, 1995 – 2000, Center for Software Engineering, USC. http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

[2] Function Point Languages Table. http://www.qsm.com/resources/function-point-languages-table

[3] Design Document of myTaxiService, first revision. https://github.com/andrealinux1/se2project/blob/master/Deliveries/DD_01.pdf

[4] Requirements Analysis and Specification Document of myTaxiService, first revision. https://github.com/andrealinux1/se2project/blob/master/Deliveries/RASD_01.pdf

[5] Integration test Plan of myTaxiService, first revision. https://github.com/andrealinux1/se2project/blob/master/Deliveries/TP_01.pdf