Politecnico di Milano, A.A. 2015/2016

Software Engineering 2: My Taxi Service
Project **P**lan **D**ocument

Belotti Nicola 793419
Chioso Emanuele 791621
Colombo Andrea 853381

February 2, 2016

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to define the plan for the my taxi service project, outlining the tasks to be done, the risks that might occur during the development and the costs of developing the project.

1.2 List of Reference Documents

- MyTaxiService Requirement Analysis and Specification document: `RASD.pdf`
- MyTaxiService Design document: `DD.pdf`
- Integration test plan document: `ITPD.pdf`

2 Function points

The Function Point estimation approach is based on the amount of functionalities in a software and their complexity. We will now provide a detailed description on the function points related to our application.

There are five estimators to take in account:

- Internal logic files
- External interface files
- External inputs
- External outputs
- External inquiries

This table defines the weights values that we've to use to perform the FP value.

Table 3. UFP Complexity Weights

Function Type	Complexity-Weight Low	Average	High
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Figure 1: Weights for all the different estimators

We will now proceed to identify all the elements and assign a value to them

2.1 Internal Logic files

An internal logic files is simply a data structure used in the application. The ILFs of our system are the following:

- User
- Passenger
- TaxiDriver

- Request
- Ride

User, passenger and taxi drivers can be assumed to be simple data structure. Request and ride are a little more complex so we'll treat them as average complexity.

2.2 External interface files

External Interface Files are a homogeneous set of data used by the application but generated by other applications.

The only EIF of our system is the interface of a map service we use for navigation. It can be considered as average complexity.

2.3 External inputs

External Inputs are operations to elaborate data coming from the external environment. The EIs of our system are:

- Registration
- Login
- Simple request
- Taxi reservation
- Driver availability

Login, request, reservation and driver availability are inputs with low complexity. Registration is more complex (average complexity)since it involves a bigger amount of data and components.

2.4 External outputs

External Output is an elementary operation that generates data for the external environment. The EOs of our system are:

- Notifications to users
- Notifications to taxi drivers

Both of this outputs can be considered of low complexity.

2.5 External Inquiry

External Inquiry is as an operation that involves input and output, without elaboration of data. The EQs of the system are:

- Taxi driver ride request
- Ride sharing

Ride request to taxi driver is simple (low complexity). On the other hand, ride sharing requires a lot of operations since it can be considered of high complexity.

2.6 Summary

Internal logic files score: 3 low complexity + 2 average complexity: $3 * 7 + 2 * 10 = 41$

External onterface files score: 1 medium complexity: 7

External inputs score: 4 low complexity + 1 average complexity: $4 * 3 + 1 * 4 = 16$

External outputs score: 2 low complexity: 4

External inquiries: 1 low complexity + 1 high complexity: $1 * 3 + 1 * 6 = 9$

The total is of $41 + 7 + 16 + 4 + 9 = 77$

2.7 Source lines of code

The conversion multiplicator from function points to SLOC is 53 for the Java language: hence we have $77 * 53 = 4081$ lines of code.

3 Cocomo

We use cocomo model to provide an approximate cost estimation on the project and the time needed to develop the software.

3.1 Effort estimation

COCOMO formula for calculating the effort is $ffort = 2.94 * EAF * KSLOC^E$ Where:

- KSLOC is kilo source lines of code calculated in the previous section.
- EAF: effort adjustment factor derived from cost drivers
- E: exponent derived from scale drivers
- 2.94 a parameter

We consider our project with all nominal Cost Drivers and Scale Drivers, thus EAF will be equal to 1 and E will be equal to 1.0997. So ffort = $2.94 * 1 * 4.081^{1.0997} = 15.3709809$ person-months

3.2 Time estimation

The duration of the project is calculated with the formula: Duration = $3.67 * ffort^E$ Where:

- ffort is the effort calculated in the previous point.
- E: exponent derived from scale drivers
- 3.67 a parameter

So Duration = $3.67 * 15.3709809^{0.3179} = 3.60379248$ months

3.3 Conclusion

Now the we have effort (expressed in person-months) and duration (expressed in months) we can calculate the number of people necessary to develop the software $N = \frac{ffort}{duration} = \frac{15.3709809}{3.60379248} = 4.26522365 \approx 5 people$. I decided to roundup to 5 to maintain a relatively large margin in case of problems.

4 Tasks

4.1 Overview

In this chapter we will provide an overview of the tasks and how we will divide them within the group.
We will do a simply overview of the Macro-tasks and we'll create a Gantt chart to make this section simple and clear because the graph will be self explanatory.
The COCOMO II overview gave the resoult of about 3,6 months, so our schedule will start the 1 of october and will finish before the end of January.

4.2 MacroTasks

4.2.1 RASD

The initial phase will be the creation of the RASD to describe the system in terms of functional and nonfunctional requirements, it will be a contractual base between the customer and the developing.

4.2.2 Design

We will write the Design Document, working on the Components of the application, the database projecting and the deployment and run-time definition.

4.2.3 Test Plan

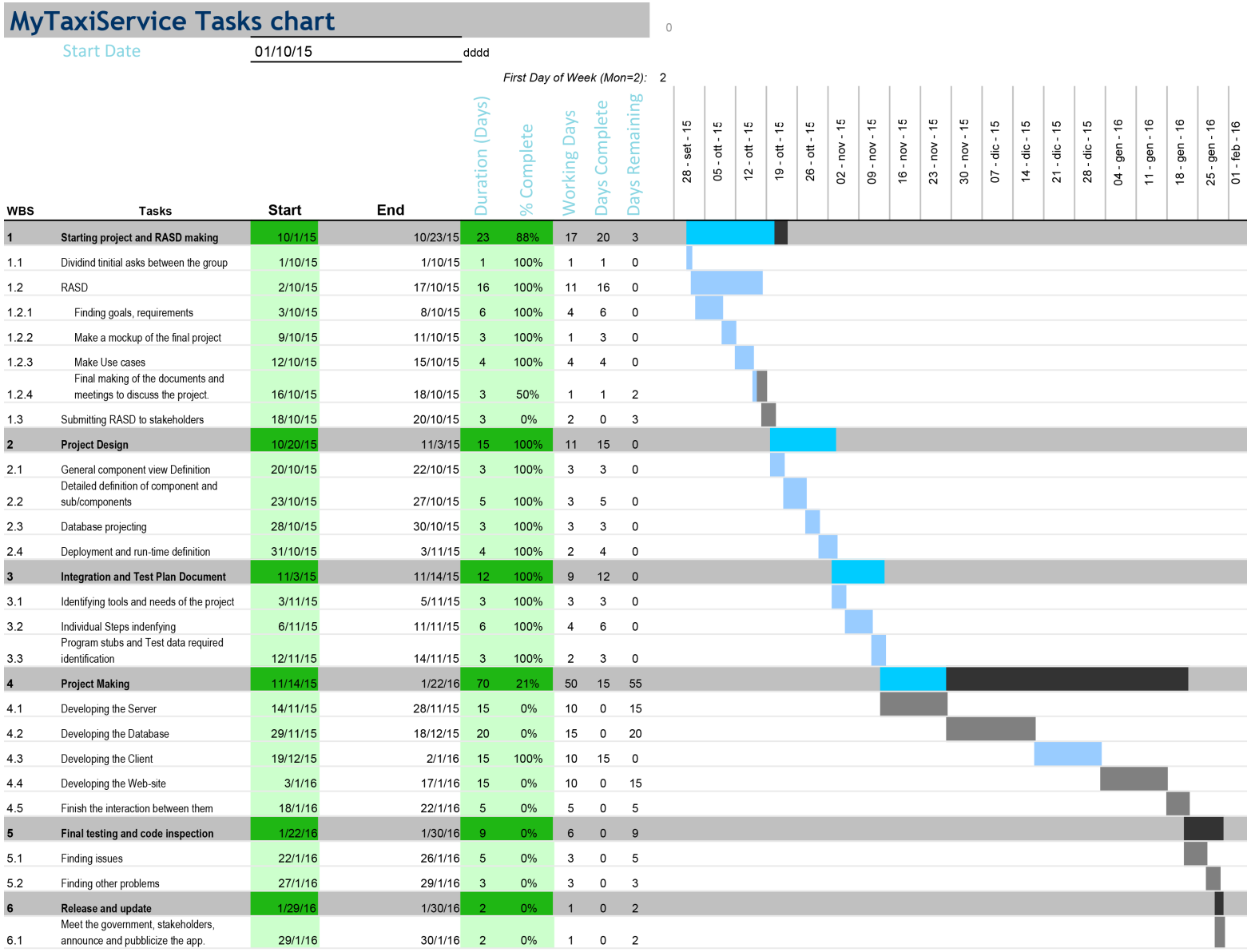
We will then study our project and start thinking about objectives, target market, processes for a specific beta test for the software.

4.2.4 Project Making

This is the main phase of the project, we will build the application and whatever concerns it's work, Following the steps of the ProjectPlan and the other phases.

4.2.5 Final Tests

As descripted in the Test plan documents, there will be a phase of testing and even implementations to be sure that everything works fine in every situation.



WBS	Tasks	Start	End	Duration (Days)	% Complete	Working Days	Days Complete	Days Remaining
1	**Starting project and RASD making**	10/1/15	10/23/15	23	88%	17	20	3
1.1	Dividind tinitial asks between the group	1/10/15	1/10/15	1	100%	1	1	0
1.2	RASD	2/10/15	17/10/15	16	100%	11	16	0
1.2.1	Finding goals, requirements	3/10/15	8/10/15	6	100%	4	6	0
1.2.2	Make a mockup of the final project	9/10/15	11/10/15	3	100%	1	3	0
1.2.3	Make Use cases	12/10/15	15/10/15	4	100%	4	4	0
1.2.4	Final making of the documents and meetings to discuss the project.	16/10/15	18/10/15	3	50%	1	1	2
1.3	Submitting RASD to stakeholders	18/10/15	20/10/15	3	0%	2	0	3
2	**Project Design**	10/20/15	11/3/15	15	100%	11	15	0
2.1	General component view Definition	20/10/15	22/10/15	3	100%	3	3	0
2.2	Detailed definition of component and sub/components	23/10/15	27/10/15	5	100%	3	5	0
2.3	Database projecting	28/10/15	30/10/15	3	100%	3	3	0
2.4	Deployment and run-time definition	31/10/15	3/11/15	4	100%	2	4	0
3	**Integration and Test Plan Document**	11/3/15	11/14/15	12	100%	9	12	0
3.1	Identifying tools and needs of the project	3/11/15	5/11/15	3	100%	3	3	0
3.2	Individual Steps indenfying	6/11/15	11/11/15	6	100%	4	6	0
3.3	Program stubs and Test data required identification	12/11/15	14/11/15	3	100%	2	3	0
4	**Project Making**	11/14/15	1/22/16	70	21%	50	15	55
4.1	Developing the Server	14/11/15	28/11/15	15	0%	10	0	15
4.2	Developing the Database	29/11/15	18/12/15	20	0%	15	0	20
4.3	Developing the Client	19/12/15	2/1/16	15	100%	10	15	0
4.4	Developing the Web-site	3/1/16	17/1/16	15	0%	10	0	15
4.5	Finish the interaction between them	18/1/16	22/1/16	5	0%	5	0	5
5	**Final testing and code inspection**	1/22/16	1/30/16	9	0%	6	0	9
5.1	Finding issues	22/1/16	26/1/16	5	0%	3	0	5
5.2	Finding other problems	27/1/16	29/1/16	3	0%	3	0	3
6	**Release and update**	1/29/16	1/30/16	2	0%	1	0	2
6.1	Meet the government, stakeholders, announce and pubblicize the app.	29/1/16	30/1/16	2	0%	1	0	2

Figure 2: Gantt Chart

5 Risks

In this section we will analyze risks that may occur during and after the development.

5.1 Project Risks

Requirements Change: Requirements may change during the project lifecycle. For example new features may be requested by the customer. (Marginal)

Lack of Skill: Cruise skill may be not high enough and could delay the delivery. (Negligible)

5.2 Technical Risks

Bugs in the code: There may be some bugs in the code that haven't been identified. (Marginal)

Integration Test Failure: After the implementation of the components, they may not pass the integration phase. (Critical)

Downtime: System might go down for any reason, for example excessive load and hardware failure. (Critical)

Data Loss: Data might be lost due to hardware failure and bugs. (Critical)

5.3 Business Risks

Market risk: People could not be interested in using our app, preferring traditional method.

Competitors: Idea could not be innovative, for example during the development another company might release a similar software.

Lack of fundings: Fundings might not be enough to develop the project.

6 Appendix