Project Plan

Filippo Agalbato
850481

Andrea Cannizzo
790469

February 1, 2016

Contents

Chapter 1

Introduction

1.1 Purpose and scope

This is the Project Plan Document for the software project My Taxi Service. The goal of this document is to perform a preliminary analysis on the project feasibility and to indicate its required effort and time, to designate and allocate tasks, and to perform risk assessment and recovery protocols.

This document is targeted to the project managers, designers and developers, as the first step in assessing what needs to be done.

General information on the My Taxi Service project itself may be found in the reference documents, as specified below.

1.2 References

- Specification Document: My Taxi Service project specification for the Academic Year 2015-16 – *Assignments 1 and 2 (RASD and DD).pdf*;
- Specification Document: My Taxi Service project specification for the Academic Year 2015-16 – *Assignment 5 – Project Plan.pdf*;

Chapter 2

Effort metrics

2.1 Function points analysis

A first estimation on the project's size and required effort can be obtained by employing a Function Point analysis.

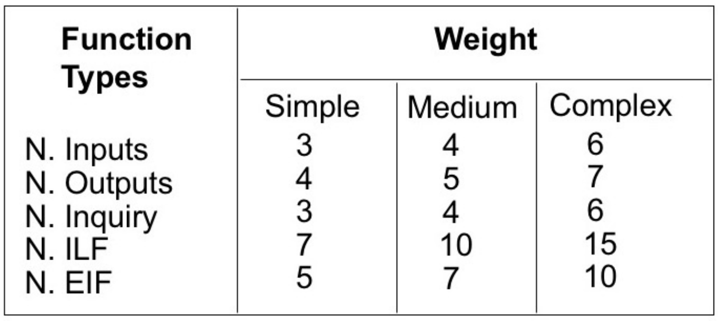
2.1.1 Counting Function Points

The bulk of the Function Points count comes from inputs and outputs, as should be expected from a system oriented to massively sending and receiving messages and orders and reservations and whatnot. The tables are self-explanatory, but to elaborate a bit further, some of the choices are explained here. Driver signup requires more steps for identity verification than the simpler passenger signup, so the former weighs more than the latter. Output functions weigh more if they require complex elaboration before sending the result away, as is the case for the *Driver move elsewhere*, *Passenger request confirmation* and *Passenger reservation confirmation*, which must run complex routines on the data present in the system at the time to ensure city-wide taxi coverage (for the first) or to ensure that the passenger can receive what asked (for the other two). Notice that simply checking if a reservation can be satisfied in the future is less complex than checking *now* – because in the first case the analysis is simplified by lack of complete data. Database interaction has been modeled as Internal Logic Files functions, since these are properly part of the system and not external, standalone ones – thus their interface is well-known and can be adapted to better serve the needs of the system. As a result, there are no External Logic Files functions (which greatly reduces Function Points count).

2.1.2 Conclusions

By counting functional operations in the system and assigning them appropriate values, we obtain an Unadjusted Function Points (UFP) count. If we multiply this by the appropriate programming language scale factor, which equals to 46 for Java EE (according to this resource), we obtain the Lines of Code (LOC) count for the project, thereby obtaining a size estimate.

$$LOC = UFP \times 46$$



Function Types	Weight		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. ILF	7	10	15
N. EIF	5	7	10

Figure 2.1: Function Points values

Function	Complexity	FP Cost
Login	LOW	3
Logout	LOW	3
Passenger signup	LOW	3
Driver signup	MEDIUM	4
Require Taxi	MEDIUM	4
Reserve Taxi	MEDIUM	4
GPS reading	LOW	3
Modify or delete reservation	LOW	3
Driver availability	LOW	3
Driver accepting or declining	LOW	3
Subtotal		33

Table 2.1: Input functions

Function	Complexity	FP Cost
Driver assignment notification	LOW	4
Driver move elsewhere	HIGH	7
Passenger request confirmation	HIGH	7
Passenger reservation confirmation	MEDIUM	5
Passenger res unavailable	MEDIUM	5
Subtotal		28

Table 2.2: Output functions

Function	Complexity	FP Cost
See reservation history	LOW	3
Subtotal		3

Table 2.3: Inquiry functions

Function	Complexity	FP Cost
Passengers database	LOW	7
Taxi and taxi driver database	LOW	7
Requests database	LOW	7
City zones database	LOW	7
Subtotal		28

Table 2.4: Internal Logic Files functions

Driver	Magnitude	Value
Precedentness	VERY LOW	6.20
Flexibility	NOMINAL	3.04
Risk resolution	NOMINAL	4.24
Team cohesion	HIGH	2.19
Process maturity	NOMINAL	4.68
Total sum		20.35

Table 2.5: COCOMO Scale drivers

Function Point count, as it emerges from the analysis depicted in this section's tables, is:

$$UFP = 33 + 28 + 3 + 28 = 125$$

And this finally gives us:

$$LOC = 125 \times 46 = 5750$$

This estimation on the source lines of code can then be used as input for other analysis methods, such as COCOMO II.

2.2 COCOMO II analysis

COCOMO II analysis uses two different sets of parameters: Scale and Cost Drivers. Definitions, tables and values are taken from the COCOMO II Model Definition Manual.

2.2.1 Scale Drivers

Precedentness relates to how familiar the type of project is to its developers, and this is being a first for us, it can only take the lowest value. Flexibility and Risk resolution have been rated to a medium value because of the didactic nature of this project – as a toy example and not an actual real-life business application, we had much more freedom that what would have otherwise been. Team cohesion has been rated high thanks to our mutual understanding and a bit of history of working together. As for the Process maturity, it has been estimated through a rough application of a CMM-like classification, as suggested.

Driver	Magnitude	Value
Required software reliability	VERY LOW	0.82
Database size	NOMINAL	1.00
Product complexity	NOMINAL	1.00
Required reusability	NOMINAL	1.00
Documentation match	NOMINAL	1.00
Execution time constrain	NOMINAL	1.00
Main storage constraint	n/a	n/a
Platform volatility	LOW	0.87
Analyst capability	HIGH	0.85
Programmer capability	NOMINAL	1.00
Personnel continuity	VERY HIGH	0.81
Application experience	VERY LOW	1.22
Platform experience	LOW	1.09
Language and tool experience	LOW	1.09
Usage of software tools	NOMINAL	1.00
Multisite development	EXTRA HIGH	0.80
Required development schedule	NOMINAL	1.00
Total product		0.57

Table 2.6: COCOMO Cost drivers

2.2.2 Cost Drivers

Required software reliability The application does not manage important data at all, and even if a failure leads to a loss of service for some users, they lose a taxi ride at most;

Database size The real size of the databases is unknown, but they would be at most of medium size, and the estimated SLOC are in the thousands;

Product complexity The value was estimated as medium following the CPLEX rating scale;

Required reusability A modular structure is always good, but there is no need to overachiev;

Documentation match to life-cycle needs The documentation provided in other documents, and the requirement and design detailed, are neither too much nor too little work;

Execution time constraint The system does not need to satisfy narrow, real-time constraints, but it is still required to work with dates, hours and minutes and send messages appropriately, working within seconds (or tens of seconds) of the timestamp triggers, and moreover it must constantly monitor city-wide taxi coverage, reacting to changes and movements;

Main storage constraint In 2016 there is no such thing as a *space constraint* for a project of this size – the databases emplyed here are far smaller than what a typical business would have;

Platform volatility We can assume a very stable work situation without frequent innovations;

Analyst capability We put a lot of effort in defining requirements and design, and even if these have not been tested on the field (implementing properly), we believe them to be sound enough;

Personnel continuity In a real case in which we were developing this, we would surely be there from the beginning to the end of the process;

Application experience We never did anything of the sort (developing in Java EE, building a web app, and so on);

Platform experience While we have been studying the theoretical workings of the components (databases, user interfaces, etc) we never actually used them anywhere in practice;

Language and tool experience While we have been studying the theoretical workings of language and tools we never actually used them anywhere in practice;

Usage of software tools While not extensive, we do have knowledge of use of Integrated Development Enviroments, dependencies managers, collaborative effort trackers and version controllers;

Multisite development We worked together most of the time, in person, and communicated with today's means when needed;

Required development schedule We were always on time and the workload was always balanced, leading to no need to stretch or compress the schedule.

2.2.3 Conclusions

The Effort equation is as follow:

$$E_f = A \times EAF \times KSLOC^e$$

where A is a standardized parameter equal to 2.94, EAF is the product of the Cost Drivers, equal to 0.57, KSLOC is thousands of lines of code (estimated at 5.75 through Function Point analysis) and e is another parameter calculated as:

$$e = B + 0.01 \times \sum_{SF_i \in SF(I)}$$

where B is a standardized parameter equal to 0.91 and SF is the set of the Scale Factors, whose sum is 20.35. This leads to:

$$e = 0.91 + 0.01 \times 20.35 = 1.1135$$

which in turn entails:

$$E_f = 2.94 \times 0.57 \times 5.75^{1.1135} = 11.75$$

which means a little less than a full person-year, or twelve person-months.

With Effort in hand, we may estimate Duration with the following:

$$D_u = 3.67 \times {E_f}^d$$

where d is the schedule equation exponent, whose value is:

$$d = D + 0.2 \times (e - B)$$

where D is a standardized parameter equal to 0.28 and e and B have been previusly defined. This leads to:

$$d = 0.28 + 0.2 \times (1.1135 - 0.91) = 0.3207$$

so that we may obtain the final result for the Duration:

$$D_u = 3.67 \times 11.75^{0.3207} = 8.09$$

From this, we can finally extract the required number of people with the formula:

$$N_p = \frac{E_f}{D_u} = \frac{11.75}{8.09} = 1.45 \approx 2$$

Chapter 3

Project tasks and resource allocation

3.1 Project Plan

The elaboration of Project Plan document is the first step of Software Development.

3.2 Requirement Analysis and Specification Document

Elaboration of the Requirement Analysis and Specification Document. The goal of this document is to describe the system in terms of goals, requirements, assumptions and design, to define the different classes of users that will interact with it, and to analyse its most typical and most critical use cases. This document is targeted to the customer's project managers, to evaluate the project specification from a high-level point of view, and to the intended designers, developers, programmers, analysts and testers, to build the actual software and to maintain, integrate and expand it in the future.

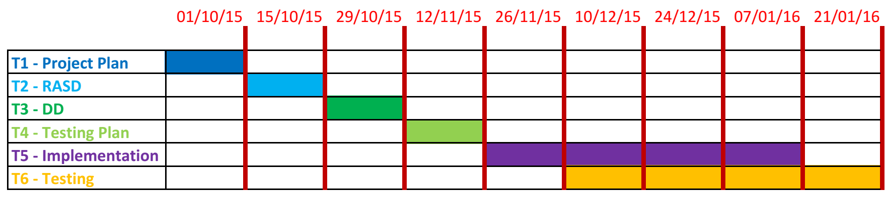
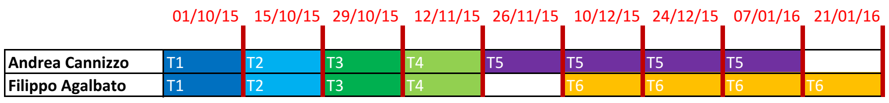


Figure 3.1: Task scheduling



	01/10/15	15/10/15	29/10/15	12/11/15	26/11/15	10/12/15	24/12/15	07/01/16	21/01/16
Andrea Cannizzo	T1	T2	T3	T4	T5	T5	T5	T5	
Filippo Agalbato	T1	T2	T3	T4		T6	T6	T6	T6

Figure 3.2: Resource allocation

3.3 Design Document

Elaboration of Design Document. The goal of this document is to describe the system in term of architectural choices and design decisions, to provide an overview of the interactions between its components and the users, and to present samples of algorithmic pseudocode for the core systems. This document is targeted to the customer's project managers, to evaluate the project architecture from a high-level point of view, and to the intended designers, developers, programmers, analysts and testers, to build the actual software and to maintain, integrate and expand it in the future.

3.4 Testing Plan

Elaboration of Testing Plan Document. The goal of this document is to describe the system's required integration testing framework and infrastructure that ought to prove its compliance with specifications, starting from these and proceeding up to the design phase of the different testing procedures. This document is targeted to the project testers, to correctly build their infrastructure so that the integration testing phase can proceed according to plan.

3.5 Implementation

This task consists in writing source lines of code of the software using the chosen development language.

3.6 Testing

This task consists in testing the software following Testing Plan Document (Code Inspection, Unit Test, Integration Test, System Test, and so on).

Chapter 4

Risk analysis

4.1 Risk identification

We identified these main risks in our project planning:

- R1 – Unrealistic schedule;
- R2 – Unavailability of staff caused by sudden problems (illness, other employments);
- R3 – The discovery of a badly designed project;
- R4 – Inability to find skilled developers for the code writing;
- R5 – Wrong functionality;
- R6 – Inability to create some critical algorithms;
- R7 – New members in staff.

Ranks go as follow: Negligible, Marginal, Serious, Catastrophic.

Risk	Probability	Effects
R1	High	Marginal
R2	Moderate	Serious
R3	Moderate	Serious
R4	Hight	Catastrophic
R5	Low	Catastrophic
R6	Low	Serious
R7	Low	Marginal

4.2 Contingency strategies plan

4.2.1 R1 – Unrealistic schedule

Alert costumers to the possibility of delays and reorganize the project plan with new deadlines. Investigate buying-in components. Investigate use of a program generator.

4.2.2 R2 – Unavailability of staff caused by sudden problems (illness, other employments)

Reorganize resources allocation in order to give more work to someone that in future will have less work.

4.2.3 R3 – The discovery of a badly designed project

Alert customers of delays, push back release date, and change the design as little as possible to fix it while salvaging as much as possible from what has already been done. Investigate buying-in components to replace badly designed ones.

4.2.4 R4 – Inability to find skilled developers for the code writing

Investigate buying-in components. Analyse the possibility to train actual staff.

4.2.5 R5 – Wrong functionality

Find out if its a requirement analysis problem or a implementation problem. In the first case derive traceability information in order to redesign that function. In the second case just recode the function. Maximize information hiding in the design.

4.2.6 R6 – Inability to create some critical algorithms

Investigate buying-in components.

4.2.7 R7 – New members in staff

Reorganize team and schedule. Reallocate staff to different tasks.

Appendix A

Document and work information

A.1 Revisions

This is the first version of this document. There are currently no revisions.

A.2 Tools used

TeXworks editor With PDFLaTeX, for composing and editing this document.

A.3 Overall time spent

The authors spent about 8 hours of their time, equally divided among them, working on this document.