Project Plan Document

myTaxiService

Fabio Catania, Matteo Di Napoli, Nicola Di Nardo

February 1, 2016

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a project plan for *myTaxiService* application, evaluating the size of the project by means of the Function Points and then the efforts and the costs required to develop it using COCOMO II tool.

Moreover, it presents overviews regarding tasks scheduling and organization and risk management.

1.2 Scope

The system aims at optimizing the taxi service in London. The software is intended to:

- integrate the current telephonic reservation system by offering the possibility to request a taxi through a web application and a mobile app;
- guarantee a fair management of the taxi queues;
- offer the possibility to share the taxi with other people. The goals of the application are illustrated more in detail in the section 1.5 Goal of the Requirements Analysis and Specifications Document.

1.3 References

- RASD_definitive.pdf
- Design Document.pdf
- Integration Testing Plan Document.pdf
- Assignment 5 - Project Plan
- Example of usage of FP and COCOMO for Assignment 5.pdf
- Second example of usage of FP and COCOMO for Assignment 5.pdf

1.4 Document Overview

This document is structured following the specification by the *Assignment 5 - Project Plan.pdf* document.

It is essentially organized in five parts:

- Section 1: Introduction, it gives a description of the document and it is thought as supporting material for its lecture;
- Section 2: Project Estimation, it provides an estimation regarding size, efforts and cost of the project by means of Function Points and COCOMO II tool.

- Section 3: Task Identification, it provides the definition and the scheduling of the tasks subdivision of the project.
- Section 4: Resource Allocation, it provides an overview over the assignment of existent resources to the identified tasks.
- Section 5: Risk Management, it identifies and treats the possible risks related with the project.

2 Project Estimation

2.1 Function Points Computation

2.1.1 Overview

The Functional Point approach is a technique that allows to evaluate the eort needed for the design and implementation of a project, identifying the various functionalities it has to provide and assigning to them a relative computable weight. The fundamental assumption is that the dimension of software can be characterized by abstraction, since the effort to develop a software product strictly depends on its functionalities. Functionalities are divided in the following categories:

- **Internal Logical File (ILF)**: homogeneous set of data used and managed by the application.
- **External Interface File (EIF)**: homogeneous set of data used by the application but generated and maintained by other applications.
- **External Input**: elementary operation to elaborate data coming from the external environment.
- **External Output**: elementary operation that generates data for the external environment. It usually includes the elaboration of data from logic files.
- **External Inquiry**: elementary operation that involves input and output Without significant elaboration of data from logic files.

In the following table is defined the weight in Function Points associated to each of the categories depending on its complexity:

Function Type	Simple	Medium	Complex
Internal Logic File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

2.1.2 myTaxiService Application Analysis

Analyzing myTaxiService functionalities referring to what has been stated in RASD Document, the following functionalities profile has been identified:

Internal Logic Files

Functionality	Complexity	Weight
Customer file with telephonic number	Simple	7
Taxi Driver file with info about Vehicle ID and personal data	Simple	7
Zone of the City simple data file with established starting points for the sharing calls	Simple	7
Queue file with dynamic control over its organization	Complex	15
Regular Call Request data file (it has fields for request position and required destination)	Simple	7
Reservation data file (it has fields for request position, time of departure and required destination)	Simple	7
Sharing Request data file (it has fields for request position, maximum allowed time to be coupled, required destination)	Simple	7
	Total	**57**

External Interface File

Functionality	Complexity	Weight
GPS position file incoming from the client-side GPS service	Simple	5
	Total	**5**

External Input

Functionality	Complexity	Weight
Sign In operation regarding Taxi Drivers	Medium	4
Log In/Log Out operations of identified Taxi Drivers	Simple	3
Acceptation/Rejection by Taxi Driver of an incoming call	Simple	3
Declaration of Availability/Unavailability by the Taxi Driver	Simple	3
Regular Call request by customer	Simple	3
Reservation Call request by customer	Simple	3
Reservation Cancellation by customer	Simple	3
Sharing Call request by customer with data about his maximum waiting time and his position	Medium	4
	Total	**26**

External Output

Functionality	Complexity	Weight
Notification to the user about his request, with expected time of arrival calculated via proprietary algorithm if accepted, and position of the starting point and weighted fee in case of Sharing request	Complex	7
Notification to the taxi driver of incoming call	Simple	4
	Total	**11**

External Inquiry

Functionality	Complexity	Weight
Customer can see and access the data of a Reservation Call he previously made	Simple	3
	Total	**3**

Total Unweighted Function Points **126**

To pass from UFP to a SLOC estimation we use an average conversion factor of 53 as described at *http://www.qsm.com/resources/function-point-languages-table* regarding Java language, the one that will be most probably (but not necessarily) used to develop the application once has been chosen to rely on Google Cloud services. We obtain: 102*53 = **5406 SLOC**.

2.2 Effort Estimation with *COCOMO II*

We will use the calculated Total Unweighted Function Points as a size estimation of the project, to allow use to use COCOMO II model to obtain an accurate effort estimation. Following is an explanation of the model's Scale and Cost drivers adjustment.

2.2.1 *COCOMO II* Scale Drivers

- **Precedentedness**: LOW. All the developers have previous experience with the languages used in the development process, but they lack of any experience in this kind of architecture and it is the first time they face a cloud-based approach.
- **Architecture/risk resolution**: NOMINAL. A high-level analysis has been carried out.
- **Development flexibility**: VERY HIGH. The provided specification only set general goals to be fulfilled, so theres a quite high flexibility allowed on how to achieve them in the development process.
- **Team cohesion**: EXTRA HIGH. Al team members know each other very well and work together since a decent amount of time, and they have no communication problems at all as they already has cooperated in previous projects without any issue in this specific matter.
- **Process maturity**: NOMINAL. The development process passed through a detailed and specific waterfall procedure so the standardized approach grants an at least nominal process maturity.

2.2.2 *COCOMO II* Cost Drivers

Product Cost Drivers

- **Required software reliability**: NOMINAL. Software failures do not have critical consequences
- **Data Base Size**: NOMINAL. The Data Base expands itself dynamically thanks to the cloud services chosen for the architecture. However, according to the operational domain of the application, it has a medium size.

- **Product complexity**: HIGH. The complexity of the whole system is set to high because of its server-client style architecture relying on a cloud structure.
- **Required reusability**: HIGH. The project is designed to be as modular as possible. Therefore its reusability is high.
- **Documentation match to life-cycle needs**: NOMINAL. This parameter describes the relation between the provided documentation and the application requirements. It is set to nominal since each aspect of the system to be described is expressed in the RASD or in the DD.

Personnel Cost Drivers

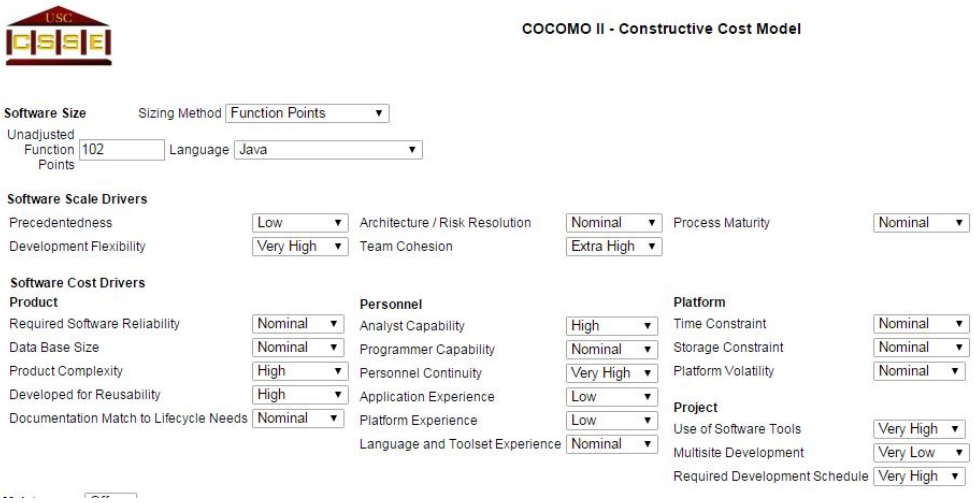
- **Analyst capability**: HIGH. Design and analysis abilities are set to high, since we intentionally dedicated a lot of effort in analyzing the problem requirements and its potential integration in a real word scenario.
- **Programmer capability**: NOMINAL. The development team has a relatively low experience in programming.
- **Personal continuity**: VERY HIGH. The team dedicates all his time to this work and is actively present during all the time of the project.
- **Application Experience**: LOW. The previous experience of the development team in web projects is low.
- **Platform Experience**: LOW. The previous experience of the team is medium according to the SQL DB, but low in cloud programming.
- **Language and tool experience**: NOMINAL. All the developers can program with the languages that most probably are going to be used to accomplish the requirements, but some of the tools has not been used before.

Platform Cost Drivers

- **Execution time constraint**: NOMINAL. Our application doesnt require particularly rapid times of execution.
- **Main storage constraint**: NOMINAL. Our application doesnt require particularly high degrees of Main Storage resources consume.
- **Platform volatility**: NOMINAL. Our main platform is provided by Google Cloud services, and its eventual changes would be with any reason compatible with the existing structure, so platform volatility shouldnt be a constraint to be especially accounted for.

Project Cost Drivers

- **Use of Software Tools**: VERY HIGH. Every phase of the development process of our product is highly supported by the most appropriate tools for every case.
- **Multisite Development**: VERY LOW. Since the whole development team resides in the same city the application will be entirely developed in the same site and no assessment for remote communication has to be taken into account.
- **Required Development Schedule**: VERY HIGH. The schedule imposed on the application development project is particularly strict.



With these settings, COCOMO II gives the following effort estimation:

Results

Software Development (Elaboration and Construction)

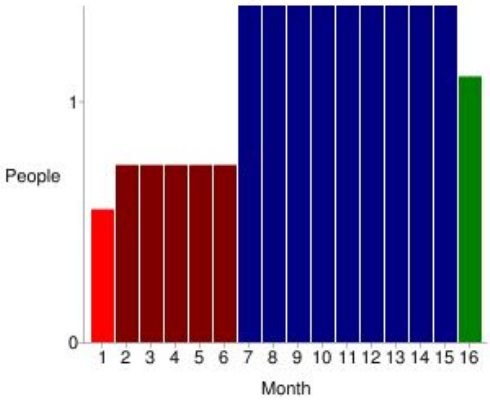
Effort = 17.3 Person-months
Schedule = 15.0 Months
Cost = $34527

Total Equivalent Size = 5406 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.0	1.9	0.6	$2072
Elaboration	4.1	5.6	0.7	$8287
Construction	13.1	9.4	1.4	$26241
Transition	2.1	1.9	1.1	$4143

Staffing Profile



Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.5	1.3	0.3
Environment/CM	0.1	0.3	0.7	0.1
Requirements	0.4	0.7	1.0	0.1
Design	0.2	1.5	2.1	0.1
Implementation	0.1	0.5	4.5	0.4
Assessment	0.1	0.4	3.1	0.5
Deployment	0.0	0.1	0.4	0.6

3 Task Identification

3.1 A Waterfall Model

In *myTaxiService* project, requirements and scope are fixed and the technology of the operational domain is clearly understood. As a consequence, a waterfall model can be suited to this software system.

The waterfall model is a sequential, structured design process, in which progress is seen as the linear sequence of discrete phases. It can include variations on the process, like returning to the previous phases after flaws were found downstream. This developing style places emphasis on documentation and after each phase a structured document has to be released.

As Royces original waterfall approach, the model in case is organized as follows:

- **Requirements and analysis specification**: in this phase is analyzed the domain in which the application takes place and are identified the re-

quirements and the specifications of the S2B (System To Be). It ends with the production of the Requirements Analysis and Specification Document (RASD).

- **Design**: it defines the software architecture and as a result is produced the Design Document (DD).
- **Coding**: in this phase each module is implemented using the chosen programming language and each module is tested in isolation by the module developer. Programs include their documentation.
- **Verification**: inspection is used as quality assurance approach. Then, modules are integrate into subsystems and integrated subsystems are tested.

3.2 Task List

Those are the tasks identified for each phase of the project development:

Requirements and analysis specification

T1: Requirements Identifying

T2: Mock Up Design

T3: UML Modeling

T4: Alloy Modeling

Design

T5: Architectural Design

T6: Algorithm Design

T7: User Experience Design

Coding

T8: Client-Side Implementation and Unit-Testing

T9: Server-Side Implementation and Unit-Testing

T10: Cloud Interfaces Specification and Unit-Testing

Verification

T11: Code Inspection

T12: Integration Testing Plan

3.3 Task Scheduling

To create a valid schedule for the various phases of the application development process we refer primarily to the studies within HP that distributed the effort required in the following approximate percentages:

18% requirements and specification

19% design

34% coding

29% testing

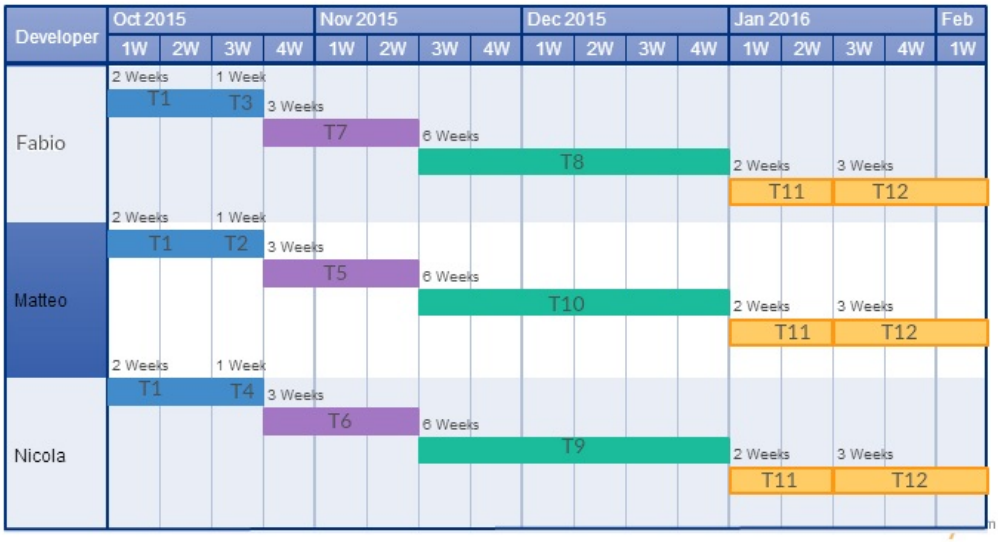
Since the total scheduled time defined for the project development goes from the beginning of October 2015 to the beginning of February 2016, we can consider 80 working days to be split among the various stages of the process. These considerations bring us to the following task scheduling:

Task	**Duration**	**People**	**Dependencies**
T1	2 weeks	3	-
T2	1 week	1	T1
T3	1 week	1	T1
T4	1 week	1	T1
T5	3 weeks	1	T2-T3-T4
T6	3 weeks	1	T2-T3-T4
T7	3 weeks	1	T2-T3-T4
T8	6 weeks	1	T5-T6-T7
T9	6 weeks	1	T5-T6-T7
T10	6 weeks	1	T5-T6-T7
T11	2 weeks	3	T8-T9-T10
T12	3 weeks	3	T11

The plan also includes an activity of synchronization at the end of each phase in order to provide the corresponding formal document.

4 Resource Allocation

The previous task identification analysis leads to the following allocation of the developers available on the different tasks:



5 Risk Management

The last but not least section of this PP (Project Plan) consists in the analysis of the risks that could damage the developing process and its final product. Risks are potential problems. Each one is associated to a risk factor, that represents a valuation of his probability w.r.t. the unwanted consequences when it occurs. For each risk is explained a strategy, that has to be thought as a countermeasure to it.

Risks of this project have been categorized into three main groups and are presented below.

5.1 Process Risks

Risk Identifier	R1
Description	Key staff are ill or generally missing at critical times in the project
Probability	Moderate
Effects	Serious
Strategy	The work is organized in phases and at the end of each one there is a discussion among the team developers in order to create more awareness of the each others jobs. As a consequence, after a period of study and analysis, each team member could be temporary substituted during his absence.

Risk Identifier	R2
Description	Changes to requirements that require major design rework are proposed
Probability	Moderate
Effects	Serious
Strategy	As a proactive risk strategy, information hiding is maximized in the design phase. As a reactive strategy, when a requirements change is proposed must be derived traceability information about his impact in the process and in the final software product.

Risk Identifier	R3
Description	A wrong estimate is made during the initial phase of the project about its cost, duration or requirements satisfiability
Probability	Moderate
Effects	Serious
Strategy	When a misestimate occurs, it is necessary to evaluate the possibility to change the software specifications, to modify or to cancel a functionality and to produce a new version of the project schedule

5.2 Technical Risks

Risk Identifier	R4
Description	Failure of the infrastructure provided by the cloud platform (Google)
Probability	Low
Effects	Catastrophic
Strategy	Define a high level procedure to migrate the server on another similar cloud platform

Risk Identifier	R5
Description	The external GPS integrated to the client does not work as expected
Probability	Low
Effects	Serious
Strategy	When the problem regards a unique taxi driver (or few taxi drivers), it is denied the access to the server resources to that particular client until the problem has been solved. On the other hand, when it is a common bug, it is considered the possibility to change the positioning external service technology

Risk Identifier	R6
Description	Malfunctioning of the client mobile application as a consequence of release of a new version of the mobile OS
Probability	Low
Effects	Serious
Strategy	The cause of the problem must be detected and solved with an update of the used technology. That risky situation can be avoided with a careful maintenance activity.

5.3 Business Risks

Risk Identifier	R7
Description	Customer financial problems force reductions in the project budget
Probability	Low
Effects	Catastrophic
Strategy	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost effective