Politecnico di Milano

Software Engineering 2 Project
a.y. 2015-16

MyTaxiService
Project Plan Document
Version 1.0

Casati Fabrizio, 853195
Castelli Valerio, 853992

Referent professor: Di Nitto Elisabetta

January 26, 2016

Contents

Chapter 1

Introduction

1.1 Revision History

Version	Date	Author(s)	Summary
1.0	26/01/16	Valerio Castelli & Fabrizio Casati	Initial release

1.2 Purpose and scope

This document represents the Project Plan Document for myTaxiService.

Its main purpose is to analyze the expected complexity of myTaxiService and assist the project leader in the delicate phase of cost and effort estimation. This information can be subsequently used as a guidance to define the required budget, the resources allocation and the schedule of the activities.

In the first section, we're going to use the Function Points and COCOMO approaches together to provide an estimate of the expected size of myTaxiService in terms of lines of code and of the cost/effort required to actually develop it.

In the second section, we'll reuse these figures to propose a possible schedule for the project that covers all activities from the requirements identification to the implementation and testing activities.

In the third section we're going to assign the different members of our development group to the various tasks.

Finally, we're going to elaborate on the possible risks that myTaxiService could face during the various phase of the project and provide some general conclusions.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1.3.2 Acronyms

- FP: Function Points.
- ILF: Internal logic file
- ELF: External logic file.
- EI: External Input.
- EO: External Output.
- EQ: External Inquiries.
- DBMS: Database Management System.
- API: Application Programming Interface.
- ETA: Estimated Time of Arrival.
- UI: User Interface.
- GPS: Global Positioning System.

1.3.3 Abbreviations

1.4 Reference Documents

- Assignment document: Assignment 5 - Project Plan.pdf
- myTaxiService Requirement Analysis and Specification Document: RASD.pdf
- The Project Plan Example documents: Example of usage of FP and COCOMO for Assignment 5.pdf and Second example of usage of FP and COCOMO for Assignment 5.pdf
- The Function Points complexity evaluation tables.
- The COCOMO II Model Definition Manual (version 2.1, 1995 – 2000 Center for Software Engineering, USC).

Chapter 2

Project size, cost and effort estimation

This section is specifically focused on providing some estimations of the expected size, cost and required effort of myTaxiService.

For the size estimation part we will essentially use the Function Points approach, taking into account all the main functionalities of myTaxiService and estimating the correspondent amount of lines of code to be written in Java. This estimation will only take into account the parts of the project that concur to the implementation of the business logic and will disregard the aspects concerning the user interface.

For the cost and effort estimation we will instead rely on the COCOMO approach, using as in initial guidance the amount of lines of code computed with the FP approach.

2.1 Size estimation: function points

The Function Points approach provides an estimation of the size of a project taking as inputs the amount of functionalities to be developed and their complexity.

The estimation is based on the usage of figures obtained through statistical analysis of real projects, which have been properly normalized and condensed in the following tables:

For Internal Logic Files and External Logic Files

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

For External Output and External Inquiry

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

For External Input

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

UFP Complexity Weights

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

2.1.1 Internal Logic Files (ILFs)

myTaxiService relies on a number of ILFs to store the information it needs to offer the required functionalities. In the next few paragraphs, we'll analyze in detail the various ILFs we have identified.

First of all, the system has to store information about taxis and taxi drivers. These data are condensed in a single table that holds the first name, last name and birthdate of a taxi driver as Strings, together with his home address, SSN, email address and mobile phone number as contact information. It also stores the taxi driver's driving license, his taxi license, the taxi plate number and the taxi status (available / unavailable / on a ride / outside city) for convenience.

As for the zones, they are stores using a two-level structure. The first level of the structure holds the identifiers of all the zones, while a secondary table contains all the location coordinates (as ¡latitude, longitude¿ pairs) necessary to identify the vertices of the zone polygon.

The system furthermore needs a queue for each zone in which it can store the identifiers of the taxi drivers waiting in that zone and their relative position in the queue, and similar structures to store the lists of taxi drivers who are unavailable, outside the city or currently on a ride. These data are stored on disk primarily to facilitate a fast recovery of the system in case of failure.

Reservations are stored in a dedicated table that holds all the information about the identifier of the passenger who booked them, the timestamp of the moment when the tuple is created, the date and time of the pickup, the origin and destination locations as addresses and the status (pending / being served / completed / cancelled).

Requests are also stored in a dedicated table with a similar structure, the only difference being the absence of the destination field and the presence of an extra field for the secret code.

The system has a dedicated table to store the passenger data. Each passenger is associated with an identifier, a name, surname and birthdate, the username, an email address and a mobile phone number. The passenger identifier is used as foreign key in a secondary table that stores his the personal settings.

Passengers and taxi drivers login information, that is username, password, identifier and user type are stored in a dedicated user table. Admin accounts are not stored in this table to achieve a greater level of security.

To keep track of who can access the administration services, information on the admin accounts is stored in a dedicated table. The main fields are name, surname, username, password and the id of the privilege level of the account, which is referencing a correspondent entry in a dedicated privileges table.

Finally, the system keeps a list of application and plugin identifiers that have access to the privileged API. Each identifier is associated with the contact information of the developer, a description of what the application or plugin does and the exhaustive list of methods that can be called.

(should we add the statistical data here? or keep it as a computed set of things?)

Using the previously defined tables, this is the count we obtain:

ILF	Complexity	FPs
Login data	Low	7
Passenger data	Low	7
Taxi drivers	Low	7
Zones	Low	7
Queues	Average	10
Reservations and requests	Low	7
API permissions	Average	10
Total		55

2.2 Cost and effort estimation: COCOMO

- **Internal Logical File (ILF)**: homogeneous set of data used and managed by the application –¿
  - taxi drivers: firstname, lastname, birthdate, address, email, mobile phone number, taxi license, driver license, taxi plate number.
  - zones: coordinates...
  - queues: taxi number, position
  - reservation list: passengerId, insertion timestamp, data, time, origin, destination, status
  - request list: passengerId, insertion timestamp, origin, status, secret code
  - tabelle che memorizziamo (utenti per loggarsi/admin?, dati tassisti, dati zone, code x zona (e non solo), reservation, request (pending/soddisfatte), stato taxi, elenco app registrate x API con permissions, elenco permissions (in generale), settings x utenti?, dati statistici?)
- **External Interface File (EIF)**: homogeneous set of data used by the application but generated and maintained by other applications –¿ dati del mapping service
- **External Input (EI)**: Elementary operation to elaborate data coming form the external environment –¿ login/logout, password retrieval, change settings for user, insert/update/delete zones, insert/update/delete taxi (drivers), register new passenger, delete passenger, new request, new reservation, delete reservation, accept ride, refuse ride, set availability, end ride, request stats, insert new app API key, refuse App API key, view reservation history
- **External Output (EO)**: Elementary operation that generates data for the external environment. It usually includes the elaboration of data from logic files –¿ notification to taxi driver that he has been assigned to a request/drive, notification to passenger that his request has been accepted / refused (no one is able to serve him), notify driver his zone has changed, notify driver his position in zone queue has changed...
- **External Inquiry (EQ)**: Elementary operation that involves input and output Without significant elaboration of data from logic files –¿ taxi driver request his position in zone queue, passenger request his

reservation history, show stats (only if precomputed), get taxi drivers in the system, get zones (admin.), get list of approved app APIs, get list of passengers

Chapter 3

Schedule

Chapter 4

Resource allocation

Chapter 5

Risk management

Chapter 6

Conclusions

Appendix A

Hours of work

To redact this document, we spent ?? hours per person.