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

2.1.2 External Logic Files (ELFs)

The only external data source myTaxiService relies on is represented by the Mapping Service.

The interaction between the core system and the remote service provider happens through a RESTful API and data can be returned in JSON or XML format. The results have then to be processed before they can be used as part of our computation.

There are two main kind of interactions:

- Given the coordinates of two locations, get an estimate of the time that is necessary to drive from one to the other
- Given an address, get the correspondent pair of coordinates (reverse geocoding)

On the client side, the mapping service is also used to retrieve the graphical representation of the city map to be displayed on the smartphone of the taxi driver.

Given the complexity of the interaction and the amount of data that is retrieved, it is reasonable to classify this logic file as a complex one.

ELF	Complexity	FPs
ETA computation	Low	10
Reverse geocoding	Low	10
Map data retrieval	Low	10
Total		30

2.1.3 External Inputs (EIs)

myTaxiService supports many kind of interactions with different categories of users.

We are now going to summarize the impact of the offered features, grouping them by user category. All users:

- Login/Logout: these are simple operations that involve only the account manager. They contribute 3 FPs each.

Passengers:

- Password retrieval: this operation has an average complexity, as it involves a number of steps in order to be sure the user is really entitled to retrieve his password. For this reason, it contributes 4 FPs.

- Change settings: this operation also has an average complexity, as the number of settings to be managed can be quite high. As such, it contributes 4 FPs.
- Request or reserve a taxi: these are both very complex operations that involve a large number of components. For this reason they account 6 FPs each.
- Delete a reservation: since this is straightforward operation, it yields 3 FPs.
- Register a new account: this operation also has an average complexity, as it involves a number of checks on the validity of the fields. As such, it contributes 4 FPs.
- View reservation history: since this is straightforward operation, it yields 3 FPs.

Administrators:

- Insert, delete and update zones: these are very complex operations that involve a great number of components. For this reason they account 6 FPs each.
- Insert, delete and update taxi drivers: as for the zones, the complexity of these operations is high, so they account another 6 FPs each.
- Request service statistics: as this operation involves some fairly complicated aggregate queries on the database, it can be considered complex and thus contributes 6FPs to the total amount.
- Grant and revoke privileges to an application or plugin: while these two specular operations involve a large number of fields that can be set, the impact on the database is quite limited. For this reason we think they have an average complexity and they should contribute 4 FPs each.

Taxi drivers:

- Accept, refuse and end ride: even though from a client point of view these operations seem trivial, the steps required to properly rearrange the taxi queues are quite complex. For this reason they account 6 FPs each.
- Set availability: this operation also involves a few rearrangements of the taxi queues, but has a smaller impact on the overall behavior of the system. For this reason it can be though as having an average complexity and it contributes 4 FPs.

The final results are shown in the following table:

EI	Complexity	FPs
Login/Logout	Low	2x3
Password retrieval	Average	4
Change settings	Average	4
Request or reserve a taxi	High	2x4
Delete a reservation	Low	3
Register a new passenger account	Average	4
View reservation history	Low	3
Insert, delete and update zones	High	3x6
Insert, delete and update taxi drivers	High	3x6
Request service statistics	High	6
Grant and revoke app privileges	Average	2x4
Grant and revoke plugin privileges	Average	2x4
Accept, refuse and end ride	High	3x6
Set taxi availability	Average	4
Total		112

2.1.4 External Inquiries (EQs)

As specified by the FP guidelines, an inquiry is essentially a data retrieval request performed by an user.

myTaxiService supports a few interactions of this type that don't require complex computations:

- A taxi driver can retrieve his position in the queue of his current zone.
- A passenger can retrieve his reservation history.
- An administrator can retrieve the full list of taxi drivers, zones, passengers or approved applications and plugins that are making usage of the APIs.

All these operations can be though as fairly simple. The resulting table is the following:

EQ	Complexity	FPs
Retrieve taxi position in queue	Low	3
Retrieve passenger reservation history	Low	3
Retrieve list of taxi drivers	Low	3
Retrieve list of zones	Low	3
Retrieve list of passengers	Low	3
Retrieve list of approved applications	Low	3
Retrieve list of approved plugins	Low	3
Total		21

2.1.5 External Outputs (EOs)

As part of its normal behavior, myTaxiService occasionally needs to communicate with the user outside the context of an inquiry. These occasions are:

- Notify a taxi driver that he has been assigned to a request.
- Notify a passenger that his request has been accepted.
- Notify a passenger that his request has been dropped.
- Notify a taxi driver that his zone has changed.
- Notify a taxi driver that his position in the zone queue has changed.

All these operations can be though as fairly simple. The resulting table is the following:

EQ	Complexity	FPs
Taxi request assignment notification	Low	4
Request accepted notification	Low	4
Request dropped notification	Low	4
Zone changed notification	Low	4
Position in the queue changed notification	Low	4
Total		20

2.1.6 Overall estimation

The following table summarizes the results of our estimation activity:

Function Type	Value
Internal Logic Files	55
External Logic Files	30
External Inputs	112
External Inquiries	21
External Outputs	20
Total	238

Considering Java Enterprise Edition as a development platform and disregarding the aspects concerning the implementation of the mobile applications (which can be thought as pure presentation with no business logic), we can estimate the total number of lines of code.

Depending on the conversion rate, we have a lower bound of:

```
SLOC = 238 * 46 = 10948
```

and an upper bound of

```
SLOC = 238 * 67 = 15946
```

2.2 Cost and effort estimation: COCOMO

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

- **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 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.