Politecnico di Milano

A.Y. 2015/2016

myTaxiService

Project Plan Document

Bernardis Cesare matr. 852509 Dagrada Mattia matr.852975

January 21, 2016

Contents

1 Functional Point Approach

The Functional Point approach is a technique that is used to evaluate the effort needed for the design and implementation of an application. This technique evaluates an application analysing the functionalities of the application itself. For what concerns our application all the functionalities can be derived from the RASD document and from the Design Document. Each functionality is evaluated in its totality. This technique groups the functionalities of an application in:

- Internal Logic File: it represents a set of homogeneous data handled by the system. In our application, MyTaxiDriver, in this category are included all the data structure saved into the database.
- External Interface File: it represents a set of homogeneous data used by the application but generated and handled by external application. In our application, MyTaxiDriver, in this category is included the map service offered by GoogleMaps.
- External Input: elementary operation to elaborate data coming from the external environment.
- External Output: elementary operation that generates data for the external environment.
- External Inquiry: elementary operation that involves input and output. operations.

The following table outline the number of Functional Point based on functionality and relative complexity:

Function Point	Complexity		
	Simple	Average	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

1.1 Internal Logic File

The system stores only little information about users. Guests are stored only if they make a request and only by adding their IP address, while for registered users and taxi drivers are stored personal data. Request and reservations are stored in the system in order to have an history always accessible. Requests are stored saving the position of the request and the IP of the user, while reservations are stored saving the origin and the destination and the informations of the user who made it. Areas are considered of high complexity due to the fact that they are stored along with the associated queue.

Internal Logic File	Complexity	FP
User	Simple	7
Request	Simple	7
Reservation	Simple	7
Area	Complex	15
Total		36

1.2 External Logic File

The application has to manage the maps coming from the external interfaces provided by Google Maps in order to calculate taxi paths and show the position of the users. A map is a complex set of data that probably must be elaborated before to be used. So it is necessary that the developer in charge of building this functionality studies Google Maps APIs as well as possible, a task that could require an important amount of time.

External Interface File	Complexity	FP
Map	Complex	10
Total		10

1.3 External Input

This section involves all the possible way of interaction between the user and the application.

- Login/Logout: these are simple operations related only to the client and the user on the frontend server.
- Registration: this is a simple operation which again, as the login/logout is related only to the client and the user on the frontend server, with the addition of the storage of the newly created RegisteredPassenger into the database.
- Make Request: this operation is considered complex because not only it requires to insert personal data, to retrieve the GPS position and to store the IP, but also starts the process of searching an available taxi to whom submit the request, therefore involving Area, Queue and TaxiDriver.
- Make Reservation: this operation is considered complex since it requires to store some information into the database but also once the

timer expires it will start a request process, which is considered complex as said above.

- Delete Reservation: this operation is considered average because it requires the retrieval of the list of reservations made by a Registered-Passenger from the database at first, and then delete itself.
- Give/Revoke Availability: these operations are considered average since they involve a TaxiDriver and the QueueManager which will add or remove the TaxiDriver from the Queue he currently is in.
- Accept/Refuse Request: these operations are considered average since they involve a TaxiDriver and a Request but also the QueueManager because the TaxiDriver is preventively removed from the Queue when he receive a Request notification. In case the TaxiDriver refuses the request, the QueueManager has to place him again in Queue.

External Input	Complexity	FP
Login/Logout	Simple	2×3
Registration	Simple	3
Make Request	Complex	6
Make Reservation	Complex	6
Delete Reservation	Average	4
Give/Revoke Availability	Average	2×4
Accept/Refuse Request	Average	2×4
Total		43

1.4 External Output

The application does not create important objects for the output. The only external output can be considered the generation of the notifications, which are simple messages shown on the device screen.

External Interface File	Complexity	FP
Notificaiton	Simple	4
Total		4

1.5 External Inquiries

Our application is mainly concentrated on the request functionality, so there are many inputs, but only few inquiries. The only inquiries we have, are:

- the list of reservation, simple objects, to let the registered passenger select the reservation to delete
- the taxi profile, another simple set of data, where he can set his availability.

External Interface File	Complexity	FP
Reservation	Simple	3
Taxi driver profile	Simple	3
Total		6

- Total number of Function Points: we have computed the following value of FP: 99. We will now estimate the size of the application in KLOC and use COCOMO in order to estimate the effort.

2 COCOMO

To estimate the SLOC starting from the FP we use an average conversion factor of 46 as described at http://www.qsm.com/resources/function-point-languages-table, an updated version that adds J2EE of the table included in official manual (http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf).

$$LOC = 46 \times 99 = 4554$$

We will consider our application as a project with all "nominal" Cost Drivers and Scale Drivers. Therefore we will use an EAF equal to 1.00 and exponent E equal to 1.0997 We Consider a project with all Nominal Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent E of 1.0997. With these values we can proceed and calculate the effort estimation:

$$effort = 2.94 \times EAF \times (KSLOC)^E$$
$$effort = 2.94 \times EAF \times (4.554)^1.0997 = 15.57 \ ^{Person}/_{Month}$$

Where

- EAF: Effort Adjustment Factor which is derived from Cost Drivers.
- E: Exponent derived from Scale Drivers.

We can then calculate the expected duration of the project in month

$$Duration = 3.67 \times (effort)^E$$
$$Duration = 3.67 \times (15.57)^{0.3179} = 8.78 months$$

And finally we can estimate the number of people needed to complete the project

$$N_{people} = \frac{effort}{Duration}$$
$$N_{people} = \frac{15.57}{8.78} = 1.77 \rightarrow 2 people$$

3 Tasks

3.1 Identification

We present a brief description of the tasks that our project requires to be executed.

T1: Requirements Analysis and Specification
This is the first step of the project, which implies a deep analysis of the requirements submitted by the customer and the drafting of a proper document. It is very important that the requirement analysis and transformation is very clear and close to the customer necessities.

T2: Design definition
In this step, we choice the architecture of our whole system that will be the base of the project development. As T1, this task requires the drafting of a proper document.

T3: Test Plan
In this step we have to think about the plan to test our system. We define what has to be tested, when and with which other components. This step too requires a proper document.

T4: Project Plan
We define how much time we will have to spend on the project, the risks that we could face during and after the whole development process and allocate our available time to perform the defined tasks.

T5: Development of system
The coding part of the project, we have to write the real application based on the previous documents. It can be divided in the development of the various subsystems:

- Database
- Backend system
- Frontend system
- Clients

T6: Unit testing
In this task we have to test each component independently from the others

T7: Integration testing
In this step we have to start integrating the various components each other in order to test the different functionalities across the single entities and subsystems.

T8: System testing

This is the final test of the system. Global functionalities are tested and some performance tests are performed to improve the system usability.

There are some dependencies that must be respected and that limit the number of parallel tasks that can be executed. Generally, given the order of the tasks above, we can say that every task requires that all the previous tasks must be consolidated, before to be considered. We will see particular cases in the allocation.

3.2 Allocation

As premise we have to consider that we are still students, so our knowledge, experience and time to dedicate at this project are quite limited.

Step	Cesare Bernardis	Mattia Dagrada
1	T1	T1
2	T2	T2
3	T3	T3
4	T4	T4
5	T5, T6	T5, T6
6	T7	T7
7	T8	T8

The first thing to consider is that our experience is very limited, so in most of the cases two thinking heads are better than only one. This is why we decided to analyse the project and write the relative documents together, in order to share our individual knowledge and to be both aware of the project construction. The individualities come out during the development process, where the system can be split up in different parts to save time. The last tests require both the parts of the project, so we will be both involved. Let's analyse the steps a bit deeper:

Steps 1,2,3,4

These steps are preliminary analyses and documents, so, as said before, we think could be a good choice to spend some time thinking together about the requirements, design, architecture and development process of our project. Maybe the various parts of each document can be divided between us, but we are both focused on the same document (and task).

Step 5

The development process can be divided in two branches, trying to keep the two parts with the same size. So we can think that one of us writes the code of the Clients and the Frontend system, while the other writes the code of the backend and the database interface. Subsequently every component of the team can "unit test" the modules that he wrote. If one of the developers ends before the others can help with the development of the other parts or maybe prepare the tests for the Unit testing of those parts.

Steps 6,7

The project ends with the integration and test of the complete functionalities of the system. To accomplish these tasks the code must be complete, so all the team must have completed the implementation of each part. Eventual errors may be found and corrected by the same person that wrote the wrong code.

4 Risk and Recoveries

In the development of a project, handling the risks is a very important task. The first thing to do is to identify all the most relevant risks that can occur. Then it is important to determine recovery actions in order to solve the risks. In our project the main risks that can be found are:

- **Schedule.** It may happen that there are difficulties following the project schedule, which means that the project will be finished later than the estimated time. This can lead to an increase in effort and cost. In order to avoid this kind of problem, that is one of the main and more frequent problem that happens in doing a project, it is important to monitor all the phases of the project, in order to understand if something is slowing down and why.
- **Technology.** Another risk that can be critical. At the moment of deploying the project on an infrastructure the requirements may be higher than expected and lead to an increase in the overall cost of the project. A good strategy can be to overestimate the requirements in the beginning.
- **Lack of experience.** If the team developing the project doesn't have a good amount of experience there can be problems while developing some parts of the project since they may result too difficult for the developer and this may lower the overall quality of the project. Therefore it is important to specify at best all the requirements for the project, in this way the team of developers can be properly selected without facing this kind of problem.
- **Budget.** It may happen that the initial budget assigned to the project may not be enough to cover the complete development of the project and if there are problems in recovering the needed money the entire project may be slowed down or stopped. It is also difficult to prevent this kinda of problem because if the starting budget is limited it can be hard to find more money. A good strategy can eventually be to find more investors and to wait before starting the project until the starting budget seems fine.

We will adopt a proactive strategy, that tries to avoid risk in advance instead of trying to solve them after they are already become a problem. All the potential threat will be identified and analysed, also taking in consideration the probability of happening and the potential damage. This analysis produces a ranking of risks, allowing to create a contingency plan that will give priority to the risks higher in the ranking.

5 Other Info

5.1 Hours of work

For each component follows an approximate indication of how much time was spent on the realization of this document

Component	Hours
Bernardis Cesare	18
Dagrada Mattia	16

5.2 Tools

We used various tools to develop this document:

- $\LaTeX\,2_\varepsilon$ and TeXMaker editor 2.10.4
- Draw.IO