myTaxiService

Project Plan

Monica Magoni 854091

Alberto Cibari 852689

February 1, 2016

Contents

1 Introduction

In this document our attention is focused on the estimation of cost and effort in the development of our project. This evaluation will be performed following the Function Points Analysis and the COCOMO II model.
Then, we will identify the tasks that are part of our project and their schedule, the allocation of resources for each specific task and the risks that our project has to deal with.

2 Function Points

In this section we perform the Function Points Analysis; this analysis has the aim to give an estimation of the project size, according to its functionalities. Indeed, we can estimate the LOC (Lines of Code) starting from the Function Points count. This approach identifies different function types:

- Internal Logic File;
- External Interface File;
- External Input;
- External Output;
- External Inquiry.

The meaning of each function type will be clarified later on.
Before computing the effort, we have to count the number of operations/data that correspond to a specific Function Points type; a weight is associated to each of the Function Points type, depending on its complexity; the total effort is computed by multiplying each count by the weight and summing all values.

The weights we will use to calculate the effort are listed in the following table:

Function Types	Complexity		
	Simple	Medium	Complex
Internal Logic Files	7	10	15
External Interface Files	5	7	10
External Input	3	4	6
External Output	4	4	7
External Inquiry	3	4	6

2.1 Internal Logical Files

This function type refers to data that are managed and used by the applications.
In our system, these data are related to:

- User personal information.
- Requests/reservations.
- History of rides.
- Queues and Zones.

All the entities have a simple complexity, except for the Queue and Zones entity that has a medium complexity. 3 x 7 + 1 x 10 = 31 FPs

2.2 External Interface Files

This function type refers to data that are used by the applications but managed and created by external applications.
In our system, these data are the ones concerning the maps; indeed, these data are generated and maintained by Google Maps (as described in the

RASD).
In this case, the rating is based upon the number of data elements and the record types and so, from our system's point of view, they both have a simple complexity: 5 FPs.

2.3 External Input

This function type refers to elementary operations that elaborate data coming from the external environment.
In our system, the External Input operations that concern the general user (either taxi driver or passenger) are:

- Sign up into the system.
- Login/Logout.
- Change password/information of the account.
- Delete account.

The Login and Logout operations are simple operations, so we can use a simple weight for them: 2 x 3 = 6 FPs. The change account information and the Sign up operation are more difficult, and we gave them a medium complexity: 2 x 4 = 8 FPs. The Delete account is a complex operation: 6 FPs.

There are others External Input operations that refer only to taxi drivers:

- Notify the system of their availability.
- Accept requests.
- Decline requests.
- Activate/Deactivate taxi service.
- Track their position via GPS.

The tracking operation and the management of the taxi service are simple operations, so they have a simple complexity: 2 x 3 = 6 FPs. The other 3 operations have a medium complexity : 3 x 4 = 12 FPs.

There are others External Input operations that refer only to passengers:

- Request a taxi.
- Reserve a taxi.
- Delete requests.
- Delete reservations.

The Request and Reservation have a medium complexity: 2 x 4 = 8 FPs. Instead the delete operation are complex operations: 2 x 6 = 12 FPs.
All these operations have been identified referring to the RASD.[1]

2.4 External Output

External Output refers to elementary operations that generate data for the external environment (usually including the elaboration of data from logic files).
In our system, we have identified the following External Output operations that are performed by the server:

- Notification to taxi drivers of a new request: when a passenger makes a request for service, the system will send a notification to taxi drivers, according to the queue algorithm.
- Notification to taxi drivers of a new reservation: when a passenger asks for a reservation, the system will send a notification to taxi drivers, according to the queue algorithm.

[1]Section "Overall Description"

- Confirmation to passenger who asked for a request: once the system found a taxi that is able to satisfy the passenger's request, it sends a confirmation to the passenger with the taxi identifier and the possible arrival time.
- Confirmation to passenger who asked for a reservation: the system sends a confirmation to the passenger with the taxi identifier ten minutes before the ride.
- E-mail to users that successfully registered in the system.

All the operations described in the section are simple operation: 5 x 4 = 20 FPs.
All these operations have been identified referring to the RASD.[2]

2.5 External Inquiry

External Inquiry refers to elementary operation that involves input and output, without significant elaboration of data from logic files.
In our system, we have identified these External Inquiries concerning the general user (either taxi driver or passenger):

- Visualize all the requests/reservations done.
- View the history of all the rides done.

Other External Inquiries specific for taxi drivers are:

- View the pending requests.

These operations involve only one entity each, so they have a simple complexity: 3 x 3 = 9 FPs.
All these operations have been identified referring to the RASD.[2]

[2] Section "Overall Description"

2.6 Total FP and Summary

In summary, we have computed that the value of the unadjusted FPs is 123, as described in the following table. We can use this result as a basis to estimate the size of the project in KLOC and then use another approach (COCOMO II) to evaluate the effort.

Function Types	Function Points
Internal Logic Files	31
External Interface Files	5
External Input	58
External Output	20
External Inquiry	9
Total	123

3 COCOMO

In order to estimate the effort of the project, we refer to the COCOMO II model; During the estimation process, in order to give a weight to Scale Drivers and Cost Drivers, we have referred to the tables presented in this document: COCOMO II - Model Definition Manual (link)

The effort is expressed as Person-Months (PM) and it is:

$$\text{Effort} = A * EAF * (KSLOC)^E = 2.94 * 1.32 * (6.519)^{1.066} = 28.631 PM \quad (1)$$

Parameter	**Value**	**Section**
A	2.94	defined by COCOMO
EAF	1.32	3.2
KSLOC	6.519	3
E	1.066	3.1

How we found the values of the equation is described in the upcoming sections 3.1 and 3.2.

The value of KSLOC can be evaluated from the FP count:

$$SLOC = FPs * 53 = 123 * 53 = 6519 \quad (2)$$

The value 53 refers to the use of Java as the implementation language of our software.
Since we have not implemented the software, we are not able to make any comment about the estimation of the SLOC.

From the Effort value we derived the project duration:

$$\text{Duration} = 3.67 * (\text{Effort})^F = 3.67 * (28.631)^{0.3112} = 10.42 \quad (3)$$

where

$$F = 0.28 + 0.2 * (E - 0.91) = 0.28 + 0.2 * (1.066 - 0.91) = 0.3112 \quad (4)$$

We have worked on our project for five months, while in the evaluation the duration is approximately ten months. The difference is due to the fact that during the five months we have not worked on the development and the testing of the software, while the estimation refers to the total duration of the project.
So, even if we do not have a real value to be compared with the one estimated by COCOMO II, we think that the evaluation is right.
With the Effort and the Duration we can find the number of people working on the project

$$N_{people} = \frac{\text{Effort}}{\text{Duration}} = \frac{28.631}{10.42} = 2.74 \tag{5}$$

This value is almost the real one: indeed our team was formed of two people.

3.1 Scale Drivers

The COCOMO II model identifies five Scale Drivers:

- Precedentedness: reflects the previous experience of the organization with this type of project.
  Since we do not have a lot of experience concerning this kind of project, the value of precedentedness of our project is low.

- Development flexibility: reflects the degree of flexibility in the development process; for example, very high means that the client set only general goals.
  In our project, goals were well defined by the client but not in a very specific way; indeed, they were general. So the development flexibility can be very high.

- Architecture/risk resolution: reflects the the extent of risk analysis carried out.

Thanks to session control, security access, and GPS tracking mostly of the risks were eliminated then this value will be high.

- Team cohesion: reflects how well the development team know each other and work together.
  During the project, each member of the team was assigned to a specific task; we scheduled some meetings in order to overcome the difficulties we could have and to discuss about the main topics of each task. Our strategy was effective since we have been able to accomplish every task in time and accurately.

- Process maturity: reflects the process maturity of the organization.
  This factor has been evaluated according to the 18 Key Process Area (KPAs) in the SEI Capability Model. The value obtained is nominal.

The values associated to each Scale Driver are listed in the following table:

Scale Drivers	**Value**	**Weight**
Precedentedness	Low	4.96
Development flexibility	Very High	1.01
Architecture/risk resolution	High	2.83
Team cohesion	High	2.19
Process maturity	Nominal	4.68
Total		15.67

To calculate the exponent E in the effort formula, we use the value we found of the scale drivers.

$$E = B + 0.01 * 15.67 = 0.91 + 0.01 * 15.67 = 1.066 \tag{6}$$

Where B is the scaling base-exponent, and it is given by COCOMO. It's value is 0.91.

3.2 Cost Drivers

In order to estimate the effort, we need to evaluate the EAF (Effort Adjustment Factor), that is the product of Cost Drivers.
So, we now evaluate each Cost Driver:

- Required Software Reliability (RELY): this factor measures the extent to which the software must perform its intended function over a period of time.
  In our project, we can consider this factor moderate and so easily recoverable; this is due to the fact that a system failure would not put on risk human life, but it can create problems or difficulties to people.
- Data Base Size (DATA): this cost driver attempts to capture the effect large test data requirements have on product development.

This parameter has a value of high, because the test database must have enough data to cover all possible test cases and all possible execution faults. So the ratio between the size of the test DB and the SLOC is between 100 and 1000.

- Product Complexity (CPLX): the complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.
  The control operations, computational operations, device-dependent operations and the user interface management operations have a nominal value, while the data management operations has a high value. So the average weigh of the areas is nominal.

- Developed for Reusability (RUSE): This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects.
  We prepared the project to maintain a more generic design as possible so the components can be used across the project. So the value given to this field is nominal.

- Documentation Match to Life-Cycle Needs (DOCU): this cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs.
  The documentation presented is right-sized to the life-cycle needs of the project. So this parameter is set to nominal.

- Execution Time Constraint (TIME): this is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system consuming the execution time resource.
  As said in the RASD, the system must have a minimum availability of 97%, so the value of this parameter is very high.

- Main Storage Constraint (STOR): this rating represents the degree of main storage constraint imposed on a software system or subsystem.
  This parameter in our application is not relevant. We do not expect the data to increment significantly, so we set it to a nominal value.

- Platform Volatility (PVOL): "Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.
  Changes to platforms will happen rarely (between 12 months and 1 month) so this value is set to low.

- Analyst Capability (ACAP): the major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate.
  We can consider this factor high due to the fact that we spent a lot of time in analyzing the requirements and designing our system-to-be; indeed, we gave high priority to those parts of the project.

- Programmer Capability (PCAP): this factor refers to the programmers capability as teams rather than individuals.
  This factor in our system can be evaluated as high since our team had already worked together on another project and so we know each other's ability to program.

- Personnel Continuity (PCON): this parameter measures the project's personnel turnover.
  We give a high value to this factor since we have worked with continuity (almost everyday) on this project for the last five months, without personnel turnover.

- Applications Experience (APEX): this rating is defined in terms of the project team's equivalent level of experience with this type of application.
  Since this is our first project of this type, this parameter is set to low.

- Platform Experience (PLEX): this factor measures the knowledge of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.
  Since we have only a bit of knowledge about those platforms, we can consider this parameter low.

- Language and Tool Experience (LTEX): This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem.
  Our level of experience concerning the language is high since we did a quite complex project in Java last year; on the contrary, our knowledge regarding the tools is low. So, we can set the value to nominal.

- Use of Software Tools (TOOL): the tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.
  Our level of experience in this field is restricted, as said in the LTEX parameter, so we set this value to low.

- Multisite Development (SITE): this cost driver rating involves the assessment and judgement-based averaging of two factors: site collocation and communication support.
  Our approach concerning the development phase would be multi-city and interactive multimedia; so we give a high value to this parameter.

- Required Development Schedule (SCED): this rating measures the schedule constraint imposed on the project team developing the software.
  For sure, there was a time constraint that has forced us to get faster in the development of the whole project; indeed, we had a lot of work to do and the schedule was stringent. This parameter is set to high.

The values associated to each Cost Driver are listed in the following table:

Cost Drivers	**Value**	**Weight**
RELY	Moderate	1.00
DATA	High	1.14
CPLX	Nominal	1.00
RUSE	Nominal	1.00
DOCU	Nominal	1.00
TIME	Very High	1.63
STOR	Nominal	1.00
PVOL	Low	0.87
ACAP	High	0.85
PCAP	High	0.88
PCON	High	0.90
APEX	Low	1.10
PLEX	Low	1.09
LTEX	Nominal	1.00
TOOL	Low	1.09
SITE	High	0.93
SCED	High	1.00
EAF		1.32

4 Tasks Identification

We have identified the following tasks:

- Draw up of the Requirement Analysis and Specification Document (RASD).
- Draw up of the Design Document (DD).
- Draw up of the Integration Test Plan Document (ITPD).
- Draw up of the Project Plan Document.
- Prepare the final presentation.
- Develop the software.
- Test the system.

There are no explicit deadlines concerning the development phase and the testing phase. However, in the following table we will evaluate the deadlines for the development phase and the testing one referring to the estimation of the duration made by COCOMO II.[3]

Task	**Start date**	**End date**
RASD	2015/10/15	2015/11/06
DD	2015/11/06	2015/12/04
ITPD	2016/01/07	2016/01/21
Project Plan	2016/01/21	2016/02/02
Presentation	2016/02/02	2016/02/10
Implementation	2016/02/10	2016/06/15
Integration Testing	2016/06/15	2016/08/15

[3]See Section "COCOMO"

5 Resource Allocation

In this section we show how the tasks of the project were assigned to each team member; we want to specify that each member of the team worked on every single part of each deliverable.
Subdivision of tasks between the team members:

- RASD:

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Section "Overall Description" • Parts of section "Specific Requirements" • All the diagrams (Sequence Diagrams, Class Diagrams)	39
Monica Magoni	• Section "Introduction" • Parts of section "Specific Requirements" • Alloy • Mockups • All the images	39

- DD:

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Section "Algorithm Design" • Section "Requirements traceability" • Runtime view in section "Architectural Design" • Section "User Interfaces"	27
Monica Magoni	• Section "Introduction" • Section "Architectural Design" • All the images	27

- Integration Testing Plan:

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Section "Individual Steps" • Section "Program Stubs"	10
Monica Magoni	• Section "Introduction" • Section "Integration" • Section "Tools"	10

- Project Plan:

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Parts of Section "Function Points" • Parts of Section "Cocomo" • Section "Risks"	11
Monica Magoni	• Parts of Section "Function Points" • Parts of Section "Cocomo" • Section "Task" • Section "Resource Allocation"	13

- Final Presentation:

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Slides	3
Monica Magoni	• Slides	3

- Implementation:
  The following table shows a possible resource allocation for the implementation phase; indeed, we have not performed the implementation. In the estimation of the total hours, we have supposed that we will not work on the project for 8 hours/day (as an usual worker) but more or less 10 hours/week. In this case it is quite difficult to predict the number of hours needed for this phase especially because these hours also refer to the time used to learn Java EE framework.

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Business Logic • Mobile Application	150
Monica Magoni	• Business Logic • Web Application	150

- Integration Testing:
  The following table shows a possible resource allocation for the Integration Testing phase; indeed, we have not performed the Integration Testing.
  In the estimation of the total hours, we have supposed that we will not work on the project for 8 hours/day (as an usual worker) but more or less 10 hours/week.

Team member	**Tasks**	**Total hours**
Alberto Cibari	• Test the integration between the sub-components on the client tier • Test the integration of the sub-systems client tier and server tier	60
Monica Magoni	• Test the integration between the sub-components on the server tier • Test the integration of the sub-system data tier and server tier	60

6 Risks

In this section we talk about the risk this project can face during its lifetime, from the project planning to the release and maintenance of the system.

The risk are divided into 3 groups:

- Project risks.
- Technical risks.
- Business risks.

6.1 Project risks

This kind of risks threaten the project plan and can road to a slip of the project schedule and an increase of costs.

- Delays. The project could require more time than expected, and go beyond the deadline. If this happens, it is possible to release an incomplete version of the product where the most important features are developed, leaving the less important for a second release
- Lack of experience. In fact the team members have no experience in programming in the Java EE framework. This problem can have moderate effects at the beginning of the developing, but once become confident with the framework, its effect are less felt.
- Change of requirements. The requirements of the project may change and some sections of the project should be redefined.
- Team misunderstandings. The project requirements can misinterpreted due to a lack of communication.

Risk	Probability	Effects
Delays	High	Moderate
Lack of experience	Very High	Moderate
Change of requirements	High	Moderate
Team misunderstandings	Low	Moderate

6.2 Technical risks

This kind of risks threaten the quality and timeliness of the software to be produced.

- Architecture lacks flexibility. The architecture is incapable of supporting change requests and needs to be reworked.
- Scalability. Components can't be scaled to meet performance demands.
- Vulnerability. Technology components have security vulnerabilities due to software bugs. This can expose personal data of the users.
- Stability. Some components of the system can crash and lead to data losses.
- Project management tool problems & issues. For example the maintenance of a clean code can lead to bugs in the system. To resolve this problem it is possible to use tools for the code inspection.

Risk	Probability	Effects
Flexibility	Moderate	Serious
Scalability	Low	Serious
Vulnerability	Moderate	Catastrophic
Stability	Moderate	Serious
Code issues	Low	Moderate

6.3 Business risks

This kind of risks threaten the viability of the software to be built.

- Financial problems. During the development of the project or after its deployment the funds may be insufficient to maintain the project. This problem can be solved doing an accurate analysis before the beginning of the development.
- Product incurs legal liability. The product has quality issues that harm the customers.

Risk	Probability	Effects
Financial problems	Low	Catastrophic
Legal problems	Low	Catastrophics

Appendix

Tools used

- ShareLaTeX: to write this document. [4]
- GitHub: to share our work. [5]

Hours of work

- Alberto Cibari: 11 hours
- Monica Magoni: 13 hours

[4] https://it.sharelatex.com
[5] https://github.com