Cascianelli Silvia

Baldi Roberto

Project Plan

V 1.0

Abstract

This document aims at providing quantitative estimations about the resources, the time and the effort necessary to develop the MyTaxiService project, according with the outlines presented on the Requirement Analysis and Specification Document for the project itself. In particular, the Function Point approach has been used to evaluate a plausible value for the code size. Then we have exploited the algorithmic technique COCOMO II, to estimate the required effort, the duration and the team size that could be suitable to carry out the project. Furthermore, the document include a plan concerning the tasks to accomplish and a proposal for their schedule and allocation to the members of the project team.
Finally, we have defined a risk assessment, highlighting the risks that can occur during the development of the project and suggesting the associated recovery solutions.

Contents

1. Project Description

The project we will focus on is called MyTaxiService and it concerns the design of a system, which is able to manage conveniently and efficiently the taxi service of a large city. Hence, this project aims principally at simplifying the access of passengers to the service by exploiting the Net, but it also guarantees a fair management of taxi queues. To be more precise, MyTaxiService grants a potential passenger the possibility of requesting a taxy either through a web application or a mobile one and it assures the applicant to be informed about the code of the incoming taxi and the waiting time, provided that the system succeeds in allocating a taxi for the request. Moreover, MyTaxiService provides also the opportunity to demand for a reservation, as long as the booking occurs at least two hours before the expected ride; in this case the passenger must specify origin and destination of the ride as well as the meeting time during the booking process. In this way, the system will proceeds automatically to allocate a taxi for the reservation 10 minutes before the scheduled time; at that point, the applicant receives a confirmation with the code of the incoming taxi and any information concerning a possible delay. However, if there are not available taxis the allocation procedure fails also for an already booked reservation; in this eventuality the negative outcome associated with the reservation is reported to the applicant through an email or a notification on the app, to apologize for the disruption. Hence, in general, MyTaxiService works in order to handle efficiently the incoming taxi requests. As it has been already discussed on the RASD, essentially MyTaxiService arranges every available taxi in a specific queue associated with the zone of the city the taxi is located at that moment. Thus, the system can always forward an incoming request to the first taxy of the appropriate queue, according with the zone the origin of the ride belongs to and by using the special functionalities of the mobile app dedicated to drivers. Whether the chosen taxi driver does not accept the sent request MyTaxiService records a penalty for him and puts his taxi at the bottom of the local queue; at the same time the systems proceeds by forwarding the request to the following taxi, which has just became the first of the queue. If none among the first three chosen drivers accepts the request, the system forces the fourth one with a mandatory service call. In fact, MyTaxiService is designed to be efficient and so it cannot let a request missed unless there are not available taxi both in the local queue and in all the adjacent zones. In fact, if an incoming request refers to a zone in which the queue of available taxis is momentarily empty, MyTaxiService provides a taxi by forwarding the request to the longer queue among the ones of the adjacent zones. To conclude this argument, it has to be noted that each queue is scrolled cyclically and so a driver could receive the same request more than once and finally should be forced to accept it. Finally, as far as taxi driver are concerned, each of them must use the mobile app, logging in at the beginning of each shift; actually through the app a driver can notify the availability every time a ride has been concluded and the system can place it in the opportune queue depending on its actual GPS position. Besides, of this a taxi driver has to use the mobile app to accept an incoming call, which the system has conveniently sent to him, and also to signalize the occurrence of an urgent problem.

2. Function Points

2.1. Introduction

Function Point Analysis (FPA) is a sizing measure defined by Allan Albrecht of IBM in 1979. This technique quantifies the functions contained within software in terms that are meaningful to the software users, referring directly to the business requirements that the software is intended to address. In particular, the functionalities of the software are grouped into 5 different types (actually Internal Logic Files, External Interface Files, Inputs, Outputs and Inquiries) and each of these business functions is associated with a numeric index according to its type and complexity. The sum of all these function points is eventually equal to the Unadjusted functions point value (UFP), that can be used to estimate the Lines Of Code (LOC) of the application depending on the average number of lines for fuction point in a certain language. Moreover, other business measures, such as the Adjusted Function Point for representing an effort estimate, can be derived from the UFD. Anyway, we have used this technique simply to estimate the size of MyTaxiProject application according with the funcionalities presented in the Requirement Analysis and Specification Document. In this sense, the following table shows the number of Functional Point associated with each function type and related to the relative complexity. Furthermore, we have reported the conversion ratio needed to transform the value of the function points in the approximate number of Java code lines, necessary to write all those functions.

Function Type	**Simple**	**Medium**	**Complex**
Internal Logic Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Conversion Ratio for Java Language	53

2.2. FP Estimation

As we have already said, the functionalities taken into account in the Functional Point Analysis derive from the RASD document for MyTaxiService Project. These functionalities have been arranged and evaluated according with their type and complexity, as specified in the following list.

- Internal Logic File: The application includes ILFs storing information about Taxi Driver, Passenger, Queue, Request and Reservation
- External Interface File: MyTaxiService handles information about the position of the cabs that are originated from the external system of GPS trackers, placed on each taxi. Moreover, the system should include a programmatic interface to enable future evolution as well as the addition of new functionalities.
- External Imput: This type contains all the elementary operation that allows to elaborate data coming from the external final users: potential passengers and taxi drivers of the company. In particular, MyTaxiService allows the passenger clients to Make a Reservation, Make a Request, Cancel a Reservation and Cancel an already Allocated Ride. On the other hand the application permits the TaxiDriver to Login/Logout, Accept a Ride, Change the current State and Notify a Problem.
- External Output: This group comprises all the elementary operations that generates automatically a flow of data towards the outside of the application. For this reason, in the scope of MyTaxiService application, we have considered as part of this collection all the kind of notifications towards the users that are a mere consequence of the system normal work.
- External Inquiry: This last category includes the operation involving input and output; hence, in our case, they are the requests of visualization of the information about a ride (actually two variants respectively for the taxi driver client and for the passenger one).

The following table indicates the complexities that we have considered suitable to be associated with each function; furthermore, it summarizes the function points related to each functionality.

Internal Logic File		
Taxi Driver	Medium	10
Passenger	Simple	7
Queue	Simple	7
Request	Simple	7
Reservation	Simple	7
External Interface File		
Location	Complex	10
Programmatic Interface	Medium	7
External Input		
Make Reservation	Complex	6
Make Request	Complex	6

Cancel Reservation	Simple	3
Cancel Allocated Ride	Medium	4
Accept Allocation	Simple	3
Login	Simple	3
Logout		3
Change State	Simple	3
Notify Problem	Complex	6
External Inquires		
Visualize Ride Information (Taxi Driver)	Simple	3
Visualize Ride Information (Passenger)	Simple	3
External Output		
Show Ride Information to the Taxi Driver	Simple	4
Show Ride Allocation to the Passenger	Simple	4
Notify Cancellation to the Taxi Driver	Simple	4
Notify Cancellation and Reallocation to the Passenger	Simple	4
Confirm the Reservation	Simple	4
Notify Penalty	Simple	4
Notify the Delay to the Passenger	Simple	4
Sum		126

2.3. Estimate Source Line of Code

$\sum FP_i$	126

$SLOC = \sum FP_i \times ConversionRatio$	6678
$KSLOC = SLOC \div 1000$	6,678

We have found a value of one hundred and twenty-six for the function points and from this; we have derived a number of 6678 lines of code to complete the development. We have also calculated the number of thousands of lines of code because we need to use this parameter in the Cocomo Model.

3. COCOMO II

3.1. Introduction

The COnstructive COst MOdel algorithmic technique allows to achieve an estimation of effort, duration and team size for a project through a complex, non-linear model that takes in account the characteristics of the product but also of people and process.

All these calculations are based on estimates of the project' size in Source Lines of Code, on a great number of meaningful factors (grouped in Scale Drivers and Cost Drivers) and eventually in some assumptions about the realization of the application itself. As mentioned above, the COCOMO II uses a non-linear model that leads to estimate the required effort and time for a software to be developed respectively through the effort equation and the schedule equation. All the values and formulas used in this analysis have been taken from COCOMO II, Model Definition Manual at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

Scale Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	6,20	4,96	3,72	2,48	1,24	0,00
Development Flexibility	5,07	4,05	3,04	2,03	1,01	0,00
Architecture and Risk Resolution	7,07	5,65	4,24	2,83	1,41	0,00
Team Cohesion	5,48	4,38	3,29	2,19	1,10	0,00
Process Maturity	7,80	6,24	4,68	3,12	1,56	0,00

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Required Software Reliability	0,82	0,92	1,00	1,10	1,26	
Database Size		0,90	1,00	1,14	1,28	
Product Complexity	0,73	0,87	1,00	1,17	1,34	1,74
Development for Reusability		0,95	1,00	1,07	1,15	1,24
Documentation Match to Lifecycle Needs	0,81	0,91	1,00	1,11	1,23	
Execution Time Constraint			1,00	1,11	1,29	1,63
Main Storage Constraint			1,00	1,05	1,17	1,46
Platform Volatility		0,87	1,00	1,15	1,30	
Analyst Capability	1,42	1,19	1,00	0,85	0,71	
Programmer Capability	1,34	1,15	1,00	0,88	0,76	
Personnel Continuity	1,29	1,12	1,00	0,90	0,81	
Application Experience	1,22	1,10	1,00	0,88	0,81	
Platform Experience	1,19	1,09	1,00	0,91	0,85	
Language and Tool Experience	1,20	1,09	1,00	0,91	0,84	
Use of Software Tools	1,17	1,09	1,00	0,90	0,78	
Multi-site Development	1,22	1,09	1,00	0,93	0,86	0,80
Required Development Schedule	1,43	1,14	1,00	1,00	1,00	

Equation for the Model	Meaning
$PM = A \times KSLOC^{E} \times \prod_{i=1}^{17} CD_i$	The Effort to Complete the Project
$E = B + 0.01 \times \sum_{i=1}^{5} SD_i$	
$TDEV = C \times PM^{F}$	The Duration of the Project
$F = D + 0.2 \times (E - B)$	
$NS = PM \div TDEV$	The Number of Needed Staff Members

Constants to Calibrate the Model

A	2,94	B	0,91	C	3,67	D	0,28

As far as MyTaxiService project is concerned, we have assumed to use also already existing components for the implementation and therefore we have considered the suitable constants to be used in the equations. In the following paragraphs we have summarized the values obtained by applying the COCOMO II model to MyTaxiService project.

3.2. Scale Drivers

Scale Drivers		
Precedence	Low	4,96
Development Flexibility	High	2,03
Risk Resolution	Very High	1,41
Team Cohesion	Extra High	0
Process Maturity	High	3,12
Sum		11,52

3.3. Cost Drivers

Cost Drivers		
Program		
Required Software Reliability	Nominal	1
Database Size	High	1,14
Product Complexity	High	1,17
Developed for Reusability	Nominal	1
Documentation Match to Lifecycle Needs	Very High	1,23

Platform		
Time Constraint	High	1,11
Storage Constraint	Nominal	1
Platform Volatility	Low	0,87
Personnel		
Analyst Capability	Nominal	1
Programmer Capability	High	0,88
Personnel Continuity	Very High	0,81
Application Experience	Nominal	1
Platform Experience	Nominal	1
Language and Toolset Experience	High	0,91
Project		
Use of Software Tools	Nominal	1
Multisite Development	High	0,93
Required Development Schedule	High	1
Product		0,955718709

3.4. Effort Estimation

$KSLOC$	6,678
$\sum_{i=1}^{5} SD_i$	11,52
$\prod_{i=1}^{17} CD_i$	0,955718709

$E = 0{,}91 + 0{,}01 \times 11{,}52$	1,0252
$PM = 2{,}94 \times 6{,}678^{1{,}025} \times 0{,}956$	19,68361775

Proceeding with the COCOMO model and inserting all the data, we have learnt that the project needs an effort of twenty person-months.

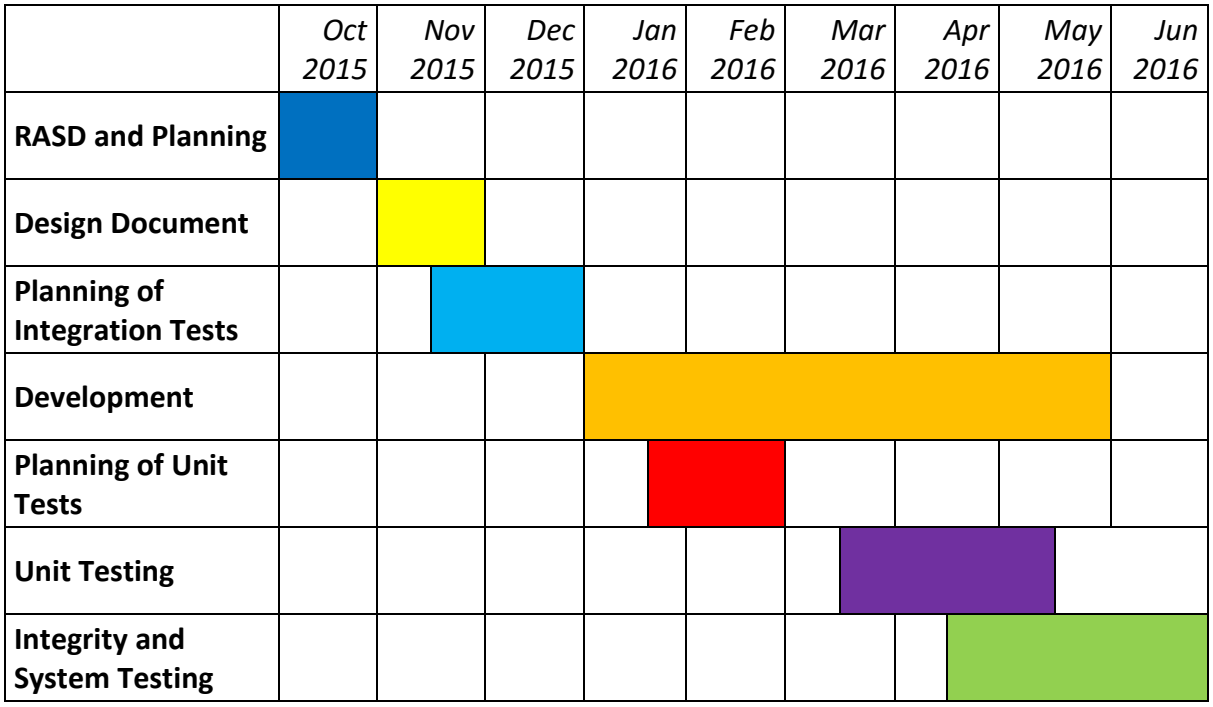
3.5. Schedule Estimation

$F = 0{,}28 + 0{,}02 \times (1{,}025 - 0{,}91)$	0,30304
$TDEV = 3{,}67 \times 19{,}684^{0{,}303}$	9,05381508
$NS = 19{,}684 \div 9{,}054$	2,174068895

The COCOMO model has granted the opportunity to evaluate the time needed to complete the project and to derive from it the number of staff members that should work on the project to carry it out in time. Eventually, the obtained value for the development time is nine months and therefore, considering the estimated effort of twenty person-months, we have found the team size suitable to accomplish, theorically, all the tasks in time (i.e. two members for the project team).

4. Schedule for the Project

4.1. Schedule of the Tasks



According with the result obtained from the application of the COCOMO model, we have planned the tasks of our project through a period of nine months. We have decided to reserve the biggest slot of time to the development, but we have also taken into particular account others important tasks like the planning and the execution of the unit and integration tests. In effect, all the testing parts, concerning both planning and execution of tests, have a meaningful role in the project schedule and this is very important to be sure to release a well-working application at the end of the development.
The schedule includes also a slot related to the writing of the RASD and of this project plan

document, as to offer an authentic and complete vision on the entire project with all its phases.

The results of the COCOMO model concerning the required time and effort to fulfil the schedule are actually a little conservative since this model is generally thought for evaluating bigger projects than MyTaxiService. For this reason, we think that the entire project could be completed in less than nine months; anyway, using this estimation allows us to manage any problems that could arise during the development.

4.2. Allocation of the Resources

<table>
<tr><th></th><th>Dec 2015</th><th>Jan 2016</th><th>Feb 2016</th><th>Mar 2016</th><th>Apr 2016</th><th>May 2016</th><th>Jun 2016</th></tr>
<tr><td>Roberto Baldi</td><td rowspan="2">Planning of Integrity Tests</td><td>Developing of MyTaxiService Manager</td><td colspan="2">Developing of Queue Manger</td><td>Unit Testing</td><td></td><td>Integrity Testing</td></tr>
<tr><td>Silvia Cascianelli</td><td>Developing of Client</td><td>Planning of Unit Tests</td><td>Developing of Data Manger</td><td></td><td colspan="2">Integrity Testing</td></tr>
</table>

We have scheduled the resources (the components of the team) for the tasks that we must face after the design of the project, since all these phase come after the writing of the design document. The development phase has been properly divided, considering a specific slot of time for the realization of each component; these components are the high-level components defined in the design document. We have supposed to arrange the implementation in this way but this order can change during the actual realization in order to manage situations or problems that could arise.
Furthermore, we have decided to distribute the slots for the testing among the phases of the development; in this way, we could plan and write some more efficient tests for the software, exploiting this strong connection between these two stages (development and testing).

5. Risk Analysis

In this last paragraph, we have described the risks related to the development of MyTaxiService project, according with the risk assessment we have performed. For each of these risks we have estimated a plausible probability of occurrence as well as the seriousness of the consequences and a suitable contingency plan.

Risk	Probability	Effect
The estimation concerning the required budget could be unrealistic	MEDIUM	SERIOUS
The estimations concerning the schedule and required resources could be unrealistic	MEDIUM	REMARKABLE
The government of the city that has commissioned the project could cut the pre-established budget for financial problem	MODERATE	SERIOUS
The commissioners of the project could require some changes in the functionalities or some additions	MEDIUM	REMARKABLE
The overall technical and project experience of the team members could be not sufficient to guarantee a good work within the estimated time	HIGH	SERIOUS
A team member, assigned to a task, becomes unavailable for a serious cause during the exercise of that task	LOW	REMARKABLE
The external system of GPS trackers used for localizing the cabs and handling consequently the queues could not working well	MODERATE	CATASTROPHIC
The database used for storing all the meaningful data could not assure suitable performances	MODERATE	SERIOUS

Risk	Strategy
The estimation concerning the overall costs could be unrealistic	It is important to report and supervise continuously all the expenses for the equipment, for the working hours of the team members as well as every accessory cost. If the overall cost increases, exceeding the estimated one, it is important to try to guarantee the respect of the budget approved by the commissioners. Anyway, in some cases, it could be necessary a revision to cut the superfluous costs and/or a properly motivated request to the commissioners, trying to obtain some extra funds.
The estimations concerning schedule, resources and required budget should be unrealistic	It is sufficient to verify the progresses through a periodical reviewing approach; we can fix some milestones in the schedule as to define when some deliverables or some parts of the project should be ready. In this way, we can monitor the accomplishment of the various tasks and whether they are late, we can intervene, also adopting changes in the allocation of the assignments if necessary.

The government of the city that has commissioned the project could cut the pre-established budget for financial problem	We can adopt a proactive strategy, trying to avoid this kind of situation. In particular, we could prepare a document that illustrate how possible cuts to the project budget would not be cost effective, since they could oblige to abandon the realization of the project itself. Furthermore, we could periodically provide the commissioners with the updated list of real costs. In this way, we could discourage the commissioners from cut our project budget, even in case of some financial problems.
The commissioners of the project could require some changes in the functionalities or some additions	In this case, we can limit ourselves to adopt a reactive strategy whether some additions or changes are required. This approach is made possible and suitable to handle this risk thanks to the development of a programmatic interface for additional features or future evolution that was one of the goal presented in the RASD. Anyway, if this situation occurs, it is important to modify coherently all the written documentation as well as all the parts of the project involved in such a variation.
The overall technical and project experience of the team members could be not sufficient to guarantee a good work within the estimated time	To limit the probability of occurrence of this kind of situation we could consider to overestimate the time necessary to carry out the project, taking better account of the inexperience and granting additional time to face technical problems that could arise. Furthermore, whether at the end of the pre-established time the project was not perfectly finished, it could be a good solution to propose the so-obtained system as a first release for the application, assuring to realize a second complete version within a reasonable period.
A team member, assigned to a task, becomes unavailable during the exercise of that task	If such a situation occurs, it is sufficient to replace the unavailable team member, possibly rearranging the schedule of the tasks.
The external system of GPS trackers could not working well	In this case, we have to find and replace the defective components or, if necessary, change completely the external system in use, since this localization system is an essential element for performing the activities of MyTaxiService application. It is also important to consider the costs for acquiring and maintaining such a system; hence, it could be useful to stipulate proper agreements with the external suppliers, as to be protected in case of incompatibility or bad functioning.
The database could not assure suitable performance	In this case, we could consider the possibility of buying an higher performance database.