PROJECT PLAN DOCUMENT

MYTAXISERVICE

Reference Professor: Elisabetta Di Nitto

POLITECNICO DI MILANO
Computer Science and Engineering
Project of Software Engineering 2

Andrea Binelli
858235

Summary

1. Introduction

1.1 Revision History

Document Title	Project Plan Document
Document Identification	Deliveries/PPD_v1.pdf
Document Version	1.0
Date Of Creation	29/01/2016
Changes	Creation of the document

1.2 Purpose and Scope

The purpose of this document is to create a project plan for the myTaxiService, described and designed in the RASD, DD and ITPD.

The project team is composed by me (the project manager) and I imagine another guest to work with (a colleague of the course Software Engineering 2) in order to create a plan that have sense.

1.3 List of Definitions and Abbreviations

1.3.1 Definitions

COCOMO	It's a model based on the statistical analysis of real projects, periodically updated with new data and cost coefficients. From 1997 we have COCOMO2
Function Point	A method used to estimate the size of a project, it is based on a combination of program characteristics and weights for each type of FPs.

1.3.2 Abbreviations

DD	Design Document
EI	External Input
EIF	External Interface File
EIQ	External Inquiries
EO	External Output
FP	Function Point
ILF	Internal Logic File
ITPD	Integration Test Plan Document
PPD	Project Plan Document
RASD	Requirement Analysis and Specification Document
UFP	Un-adjusted Function Point

1.4 List of Reference Documents

- RASD, Requirement Analysis and Specification Document, myTaxiService, 2015 Version 2
- DD, Design Document, myTaxiService, 2015 Version 2
- ITPD, Integration Test Plan Document, 2015 Version 1

2. Function Points (FPs)

In this section I will estimate the software dimension with the Function Point Methods, and I did it by valuating the weight of the functionalities that the software has to provide. The value that I obtain is then transformed to Lines of code to verify the level of complexity of the application.

Every functionality has a weight assigned to it according to his category and complexity. The following table is used to calculate the weight of every function.

Function Types	Weight		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. ILF	7	10	15
N. EIF	5	7	10

2.1 Internal Logical Files (ILFs)

ILFs are homogeneous set of data used and managed by the application.

Evaluate ILFs:

The application stores the information about:

- Users
- Taxis
- Areas
- Addresses
- Requests

Area has the bigger data structure (it incorporates both Areas and Queues) so I decide to adopt for it the medium weight.

Each of the others entities has a smaller number of fields so I decide to adopt for them the lower weight.

Thus, 1 x 10 + 4 x 7 = 10 + 28 = 38 FPs concerning ILFs.

2.2 External Interface File (EIFs)

EIFs are homogeneous set of data used by the application but generated and maintained by other applications.

Evaluate ELFs:

The application myTaxiService, in particularly the Taxi Client, use the coordinates provided from the GPS module of the mobile device. Are simple data (latitude and longitude) so 5 FPs concerning EIFs.

2.3 External Inputs (EIs)

EIs are elementary operation to elaborate data coming from the external environment.

Evaluate EIs:

The application interacts with the user/taxi driver as follows:

- Signup/Change Profile Data: these are medium operations (because it requires more fields and verifications than login), so I can adopt the medium weight for it. 2 X 4 = 8 FPs.
- Login/Logout: these are simple operations, so I can adopt the simple weight for them. 2 x 3 = 6 FPs
- Turn Availability On/Off: these are simple operations, so I can adopt the simple weight for them. 2 x 3 = 6 FPs

In summary, 8 + 6 + 6 = 20 FPs concerning EIs.

2.4 External Output (EOs)

EOs are elementary operation that generates data for the external environment, it usually includes the elaboration of data from logic files.

Evaluate EOs:

The application has to handle the functions:

- Manage Taxi Position. It's a medium operation, it involves the entity Taxi and Area and it include some elaboration of data about coordinates. 1 x 5 = 5 FPs
- Request Log: It's a simple operation that involves only the entity request. 1 x 4 = 4 FPs
- Manage Request: it's a complex operation (that include more than one sub-functions like parse the address and calculate wait time). 1 x 7 = 7 FPs

In summary, 5 + 4 + 7 = 16 FPs concerning EOs.

2.5 External Inquiries (EIQs)

EIQs are elementary operation that involves input and output, without significant elaboration of data from logic files.

The application allows user to:

- Make a request and receive messages: it's a medium operation. 1 x 4 = 4 FPs

The application allows taxi to:

- Answer for a request: it's a simple operation (only to click yes or no). 1 x 3 = 3 FPs

In summary, 4 + 3 = 7 FPs concerning EOs.

2.6 Total Function Points and Lines of Code

- ILFs: 38 FPs
- EIFs: 5 FPs
- External Inputs: 20 FPs
- External Outputs: 16 FPs
- External Inquiries: 7 FPs

- Total Estimate: 38 + 5 + 20 + 16 + 7 = 86 FPs

I use the table below to transform the function point to lines of code.

Language	Project			
	Low	Average	Median	High
J2EE	15	46	49	67

I can consider that my project is of medium complexity so:

46 x 86 = 3956 LOC

3. COCOMO II

In this section I will use the COCOMO II (COnstructive COst MOdel) to estimate the months-men that are necessary to realize the software and the duration of developing, expressed in months.

All the tables and the equations of this section are adapted from the official manual of B. Bohem

Source: http://www.dmi.usherb.ca/~frappier/IFT721/COCOMOII.PDF

3.1 Scale Drivers (SDs)

In COCOMO II, some of the most important factors contributing to a project's duration and cost are the Scale Drivers (SD).

Each scale driver has a range of rating levels, from Very Low to Extra High. Each rating level has a weight, W, and the specific value of the weight is called a scale factor. All project's scale factors, Wi, are summed across, and used to determine a scale exponent, B, via the following formula:

$$B = 1.01 + 0.01 \times \sum W_i$$

I adopt the following convention on weights:

- Extra High – W = 0
- Very High – W = 1
- High – W = 2
- Nominal – W =3
- Low – W = 4
- Very Low – W = 5

The following table represents the five scale drivers defined in the post-architecture model with a brief explanation of each level in each driver.

Scale Factors (Wi)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar
FLEX	Rigorous	Occasional Relaxation	Some Relaxation	General Conformity	Some Conformity	General Goals
RESL[a]	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
TEAM	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
PMAT	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

I weight the drivers as follows, taking into account the features of the project and making some reasonably assumptions:

- PREC (Precedentedness): Very Low. This is the first web application ever developed by team's members, so the previous knowledge about platforms and technologies is almost nothing.
- FLEX (Development Flexibility): Nominal. There are not big constraints on requirements and external interface specifications, but the deadlines are fixed and very short.
- RESL (Architecture / Risk Resolution): Nominal. I decide to do not go into details of these drivers, because it takes into accounts aspects that are outside the limited scope of this project.
- TEAM (Team Cohesion): Very High. There is a high cooperation between team members, due to the limited size of the team and to frequent meetings. Furthermore there are no problems of communication with stakeholders.
- PMAT (Process Maturity): Very Low. The official definition of this parameter takes into account the outcomes of CMM Maturity Questionnaires, I do not consider them (because the project is for educational purposes only) and I fix the value to very low making the assumption that our level of maturity in the development of applications is very low, since the team is composed by beginners.

The overall sum of weights becomes 17.

The scale exponent B is 1.18.

3.2 Cost Drivers

COCOMO II has 17 cost drivers, i.e., multiplicative factors that determine the effort (in Person Months) required to complete the software project.

COCOMO II defines each of the cost drivers, and the Effort Multiplier associated with each rating.

They are grouped into four categories: product, platform, personnel and project.

The table below summarizes all of them.

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	Slight inconvenience	Low, easily recoverable losses	Moderate, easily, recoverable losses	High financial loss	Risk to human life	
DATA		DB bytes / Pgm SLOC < 10	10 <= D/P < 100	D/P >= 1000		
CPLX	See table 20					
RUSE		None	Across project	Across program	Across product line	Across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
TIME			<= 50% use of available execution time	70%	85%	95%
STOR			<= 50% use of available storage	70%	85%	95%
PVOL		Major change every mo.; minor change every 1 mo.	Major: 6 mo. ; minor: 2 wk.	Major: 2 mo. ; minor: 1 wk.	Major: 2 wk. ; minor: 2 days	
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	
AEXP	<= 2 month	6 month	1 year	3 year	6 year	
PEXP	<= 2 month	6 month	1 year	3 year	6 year	
LTEX	<= 2 month	6 month	1 year	3 year	6 year	

TOOL	Edit, code, debug	Simple, frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature lifecycle tools, moderately integrated	Strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
SITE Colloc.	International	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro, area	Same building or complex	Fully collocated
SITE Comm.	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication.	Wideband elect. comm., occasional video conf.	Interactive multimedia
SCED	75% of nominal	85%	100%	130%	160%	

3.2.1 Product Factors

Follows a brief explanation of each choice:

- RELY: Low. A failure of the system does not cause great losses for the administrator or the users.
- DATA: Nominal.
- CPLX: Nominal. The overall complexity of the application (on the different areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations) can be considered of medium level.
- RUSE: Nominal. The components are designed and developed for being reused inside the project itself, not for being adopted in other projects.
- DOCU: Nominal.

3.2.2 Platform Factors

The platform refers to the target-machine complex of hardware and infrastructure software. Since in my project I don't considered many of these aspects, I can assume that all of them are nominal, so:

- TIME: Nominal.
- STORE: Nominal.
- PVOL: Nominal.

3.2.3 Personnel Factors

Follows a brief explanation of each choice:

- ACAP: Low. Team's members are not so expert in design and requirements.
- PCAP: Nominal. Team's members have a previous experience of programming in-group, but this is not enough to be considered as high.
- PCON: Nominal.
- AEXP: Nominal.
- PEXP: Low. Platforms (e.g. JEE) are completely new for team's members.

- LTEX: Nominal. Team's members have few years of experience with the programming language (Java) and on the tools (github, office, UML drawers…).

3.2.4 Project Factors

Follows a brief explanation of each choice:

- TOOL: Very High. A very powerful IDE is used for the realization of the project (like NetBeans), with a set of functionalities that increase considerably the productivity of the team.
- SITE: High. Team members belong to the same regional area.
- SCED: High. This rating measures the schedule constraint imposed on the project team developing the software. Since the deadlines are very strict, the schedule stretch-out is about 130%, this corresponds to a value of high for the parameter.

3.2.5 Overall Contribution

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	1.15	1.39	
DATA		0.93	1.00			
CPLX	0.75	0.88	1.00	1.15	1.30	1.66
RUSE		0.91	1.00	1.14	1.29	1.49
DOCU	0.89	0.95	1.00	1.06	1.13	
TIME			1.00	1.11	1.31	1.67
STOR			1.00	1.06	1.21	1.57
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.50	1.22	1.00	0.83	0.67	
PCAP	1.37	1.16	1.00	0.87	0.74	
PCON	1.24	1.10	1.00	0.92	0.84	
AEXP	1.22	1.10	1.00	0.89	0.81	
PEXP	1.25	1.12	1.00	0.88	0.81	
LTEX	1.22	1.10	1.00	0.91	0.84	
TOOL	1.24	1.12	1.00	0.86	0.72	
SITE	1.25	1.10	1.00	0.92	0.84	0.78
SCED	1.29	1.10	1.00	1.00	1.00	

In order to find the adjustment factor due to cost drivers, I compute a multiplication of all the weights.

The overall contribution is 0.88 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.50 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 0.72 * 0.92 * 1.00 = 0,874368.

3.3 Effort Estimation

I apply the fundamental equation of COCOMO for the estimation of the effort (person-month).

$$PM = A \times [Size]^B \times \prod_{i=1}^{17} EM_i$$

Where:

- PM: the effort
- A: constant, set to 2.94
- Size: the number of KLOC, equals to 3.956
- B: scale factors exponent, equals to 1.18
- EMi: multiplier of i-th cost factor, notice that I've previously computed the overall contribution, which is 0,874368

Given this figures, it turns out that the Effort is: 2.94 * $3.956^{1.18}$ * 0.874368 = 13.03 ≈ 13 person-month.

3.4 Duration Estimation

I apply the following equation to compute the estimation of the time (month) needed to complete the project:

$$TDEV = \left[A \times PM^{(0.33+0.2\times(B-1.01))}\right]$$

Where:

- TDEV: time of development, so the total duration
- A: constant, set to 3.67
- PM: the effort, previously computed
- B: scale factors exponent, equals to 1.18

Given this figures, it turns out that the Development Duration is: 3.67 * $13.03^{(0.33 + 0.2 * (1.18 - 1.01))}$ = 9.34 months

Given the values of effort "PM" and duration "TDEV" for the project based on COCOMO II, the number of required people "NRP" is:

$$NRP = \frac{PM}{TDEV}$$

Given this figures, it turns out that the Number of Required people is: 13.03 / 9.34 = 1.40

I round to the upper integer number, which is 2.

3.5 Conclusions

The following table summarizes the results.

I can conclude that obtained results are for sure a good estimation.

The real time is about 100 hours (summing the time written at the end of every deliveries), at that must be summed the time of the other parts not done in this project (implementations, unit testing, final testing...). That may be other 100 hours, thus 200 hours.

If 200 hours are divided by 5 hours per day we obtain 40 days, we divide again 40 days by 5 working days per week and we obtain 8 week, about 2 months. This value is consistently smaller than the estimated one.

A possibly reason is that some assumptions of the model does not hold for the real project. For example the testing activity does not cover the entire project, the design is not accomplished by a high quality team, and so on.

Another reason can be that the assumptions in the RASD documents are very optimistic

Furthermore the strict deadlines on deliveries have led to a stretching of the development time in order to build a sufficiently complete project on time.

4. Tasks

4.1 Identify activities

I've identified these tasks, some of them actually made (the CID wasn't made with the code of this project, because there is no code).

- T1: Creation of the Requirement Analysis and Specification Document (RASD)
- T2: Creation of the Design Document (DD)
- T3: Creation of the Integration Test Plan Document (ITPD)
- T4: Creation of the components (code writing)
- T5: Creation of the code inspection document (CID)
- T6: Unit Testing
- T7: Application of the ITPD
- T8: System testing

4.2 Tasks schedule

Assumptions:

- I imagine to invite one colleague, called Guest, to share the development effort with me and use these new resources in the assignment.
- For simplicity I kept the number of hours from the deliveries, assuming that the adding of a member didn't made the tasks shorter (but the effort less intensive).
- I have invented the number of hours of the tasks that were not actually made.
- There are 5 working hours in a day, I made this underestimation in order to prevent time lost in pauses, delays, illness and other types of unproductive time.
- The project has started in October 2015 (the week after the release of the first assignment) but I didn't follow the real schedule of the deliveries from that point (like it was a real project).
- All the days are used except for weekends (Saturday and Sunday) and holidays (in this case All Saints' Day on November 1 and Feast of the Immaculate on Tuesday 8).

Task	Members	Duration (Hours - Days)	Dependencies	Start Date	End Date
T1	Andrea, Guest	30 – 6	-	19/10/2015	26/10/2015
T2	Andrea	30 – 6	T1	27/10/2015	03/10/2015
T3	Guest	15 – 3	T1	27/10/2015	29/10/2015
T4	Andrea, Guest	50 – 10	T2, T3	04/11/2015	17/11/2015
T5	Andrea, Guest	15 – 3	T4	18/11/2015	20/11/2015
T6	Andrea, Guest	30 – 6	T5	23/11/2015	30/11/2015
T7	Andrea, Guest	10 – 2	T6	01/12/2015	02/12/2015
T8	Andrea, Guest	20 - 4	T7	03/12/2015	09/12/2015

5. Risks

A risk is a potential problem and I want to prevent my project from them.

For this reason I have choose a Proactive Risk strategy in order to avoid them and to have a plan if something goes wrong.

5.1 Identify risks

I have identified some of the possible risks (what can go wrong).

For each of the risks I have estimated the probability (Low, Moderate and High) that it will do if it does occur and the impact. The impact can be: negligible, marginal, serious, critical and catastrophic.

Risk	Probability	Effects
R1: It is impossible to recruit a member with the skills required for the project.	Low	Marginal
R2: Staff members are ill at critical times in the project	Moderate	Serious
R3: Changes to requirements/assumptions that require rework are proposed	Low	Critical
R4: The database used in the system cannot process the load of queries/transactions as expected	Moderate	Serious
R5: The algorithm that calculate the waiting time for taxi is too simply/inaccurate	High	Marginal
R6: The algorithm that parse the addresses is too simply/inaccurate	Moderate	Serious
R7: Taxis change zones too often and data may be inconsistent	Moderate	Serious
R8: Too much denials from taxis can cause loops in the queues	Low	Marginal

5.2 Strategies to help manage risks

For each of the risks identified above I have developed a contingency plan to follow in case something of them will occur.

Risk	Strategy
R1	Use the forum online to find out an inactive colleague with the proper skills
R2	Reorganize team according to the schedule
R3	Maximize information hiding in the design
R4	Upgrade the database server with extra hardware components
R5	Re-develop the algorithm with new parameters, like information about the traffic condition in the areas
R6	Re-develop the algorithm with a more sophisticate intelligence that recognize better abbreviations or different type of writing an address; Investigate the possibility to introduce a function that replace the address of the user with his GPS position like taxis
R7	Reduce the number of areas or develop a more sophisticated algorithm that manage crossings of the borders between two zone
R8	Reduce the timeout; add a new state of the taxi: inactive that means that a taxi is available but it has denied a request recently

For redacting and writing this document I spent about 15 hours.