Politecnico di Milano
A.A. 2015-2016

Software Engineering 2: myTaxiService

Project **P**lan

version 1.0

Matteo Bulloni (852676), Marco Cannici (852527)

2 February 2016

Contents

1 Introduction

In this document we make a series of estimates about the time and resources needed to develop the MyTaxyService application project. The estimates will be made according to the analysis of project's Function Points and by using COCOMO software. Then, further predictions will be made about the schedule of the project and its fulfillment, providing a suitable team resources allocation. Finally, a brief project risk evaluation will be carried, to analyze the main risks that may concern the development of this specific project.

2 Function Points and COCOMO

2.1 Function Points

Function Points estimation is a technique that aims to estimate the effort needed to design and implement a software project. It comes useful since it's based on informations that are available early in the project life cycle, and thus allows to estimate at the beginning how much time and resources are going to be needed for the project development. Each of the following subsections treats the estimation of the FPs of one specific category of functions, identifying and grouping them basing on the description of the system provided in the RASD and DD. In the last subsection the different contributions will be summed up, and proper conclusions will be stated according to the results.
Here is the FP weights table we used to associate to each function the appropriate number of FPs, depending on the category it belongs to and on its complexity:

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

2.1.1 Internal Logic Files

Internal Logic Files (ILFs) include the homogeneous set of data used and managed by MyTaxiService application, which is basically any kind of entity that is stored in the server's database.

Internal Logic File	Complexity	Function Points
Account	Medium	10
Request	High	15
Taxi	Low	7
City Zone	Low	7
Total		**39**

2.1.2 External Interface Files

External Interface Files (EIFs) represent the set of homogeneous data used by MyTaxiService application but handled by external applications. This includes the data obtained through the Google Maps' API whenever a conversion between GPS coordinates and actual addresses (or vice versa) is needed, and the estimated travel times that are asked to the API in order to provide an estimated waiting time to any passenger waiting for his taxi to come.

External Interface File	Complexity	Function Points
Travel time estimation	Low	5
Coordinates conversion	Low	5
Total		**10**

2.1.3 External Inputs

External Inputs (EIs) include all the elementary operation to elaborate data that can come from the external environment, which is represented here by the application's clients. The EIs taken into account in the following table are thus based on the functionalities that are offered to the two different kinds of clients (Driver and Passenger), which can be retrieved from section 3.5 "Functional Requirements" of the application's RASD.

External Inputs	Complexity	Function Points
User registration / creation	High	6
User deletion	Low	3
User login / logout	Low	3
User profile update	Medium	4

Ride/reservation request	High	6
Delete reservations	Medium	4
Driver's status change	Medium (considering consequent queue updating)	4
Driver's "complete ride"	Medium	4
Total		**34**

2.1.4 External Inquiries

External Inquiries (EIQs) include all the operations that can be requested from the clients to the server (or vice versa) in order to ask for data that can be retrieved without significant elaboration of data from logic files. The EIQs considered in the following table are again referring to the functionalities of this kind that are offered to the two different kinds of clients (Driver and Passenger), which can be retrieved from section 3.5 "Functional Requirements" of the application's RASD.

External Inquiries	**Complexity**	**Function Points**
User profile	Low	3
Reservation history	Medium	4
GPS coordinates retrieval from Driver clients	Low	3
Administrator querying	Medium	4
Total		**14**

2.1.5 External Outputs

External Outputs (EOs) include all the operations that, possibly including elaboration of data from logic files, are performed to generate data for the external environment, i.e. the clients here. The EOs considered in the following table are thus referring to the main operations performed by the server to generate data to be sent to either Passenger or Driver clients, again as stated in both the DD and RASD's section 3.5.

External Output	**Complexity**	**Function Points**
Ride request forwarding to driver user	Medium	5
Reservation request	High	7

forwarding to driver user		
Ride request confirmation request (in case of no available drivers)	Medium	5
Accepted ride request notification	High	7
Accepted reservation request notification	Low	4
Generic error notification	Low	4
Total		**32**

2.1.6 Function Points Analysis result

The following table resumes the results we have obtained by applying Function Points Analysis.

Function Type	Value
Internal Logic Files	39
External Interface Files	10
External Inputs	34
External Inquiries	14
External Outputs	32
Total	**129**

Because in the design phase of the project the language that has to be used to develop the application has not been fixed, to compute the final SLOCs value we have decided to refer to some of the Object Oriented languages listed in "QSM Function Points Languages Table" **[1]** as a reference.

Language	SLOC/FP	SLOC
Java	53	6837
C++	50	6450
C#	54	6966
J2EE	46	5934
Average SLOC value		**6547**

We will use the average value **6547** in the following COCOMO analysis.

2.2 COCOMO

2.2.1 Scale Drivers

Scale Drivers are factors that contribute on defining the exponent E of the Effort equation. Please refer to the "COCOMO II Model Definition Manual" **[2]** for the list of values associated with each factor.

Here is the list of values we have identified in our analysis:

Scale Drivers	Factor	Value
Precedentedness	Low	4.96
Development Flexibility	High	2.03
Risk Resolution	Nominal	4.24
Team Cohesion	Very High	1.10
Process Maturity	High	3.12
Total		15.36

The Process Maturity parameter has been obtained by considering the Key Process Areas (KPAs) of the SEI Capability Maturity Model. We report here the results we have obtained:

Key Process Area (KPA)	Value
Requirements Management	Almost Always
Software Project Planning	Almost Always
Software Project Tracking and Oversight	Frequently
Software Subcontract Management	Does Not Apply
Software Quality Assurance	Frequently
Software Configuration Management	About Half
Organization Process Focus	Frequently
Organization Process Definition	Does Not Apply

Training Program	Does Not Apply
Integrated Software Management	Does Not Apply
Software Product Engineering	Frequently
Intergroup Coordination	Almost Always
Peer Reviews	Frequently
Quantitative Process Management	About Half
Software Quality Management	About Half
Defect Prevention	Rarely if Ever
Technology Change Management	Rarely if Ever
Process Change Management	Occasionally
EMPL	**3.04**

The equivalent process maturity level (EPML) is computed as five times the average compliance level of all n rated KPAs for a single project (Does Not Apply and Don't Know are not counted which sometimes makes n less than 18). After each KPA is rated, the rating level is weighted (100% for Almost Always, 75% for Frequently, 50% for About Half, 25% for Occasionally, 1% for Rarely if Ever) **[2]**.

2.2.2 Cost Drivers

Cost Drivers are effort multipliers used to adjust the nominal effort (Person/Months) to reflect the software product under development. **[2]**

Cost Drivers	Factor	Value
Required Software Reliability	Nominal	1.00
Data Base Size	Nominal	1.00
Product Complexity	High	1.17
Developed of Reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	High	1.11
Main Storage Constraint	Nominal	1.00
Platform Volatility	Nominal	1.00
Analyst Capability	High	0.85

Programmer Capability	High	0.88
Application Experience	Low	1.10
Platform Experience	Nominal	1.00
Language and Tool Experience	Nominal	1.00
Personnel Continuity	Very High	0.81
Usage of Software Tools	Nominal	1.00
Multisite Development	Nominal	1.00
Required Development Schedule	High	1.00
Total (Product)		0.926

2.2.3 COCOMO Analysis results

The estimated effort of the project (Person/Months) can be calculated by applying the following equation [2]:

$$Effort := A * EAF * KSLOC^{E}$$

A := 2.94
EAF := 0.926 Effort Adjustment Factor derived from Cost Drivers
KSLOC := 6.547 Estimated lines of code measured as thousand of SLOC
*E := B + 0.01 * sum{i}SF[i] = 0.91 + 0.01 * 15.35 = 1.0636*
B := 0.91

With the previous parameters we obtain:

$$\textbf{\textit{Effort}} := 2.94 * 0.926 * 6{,}547^{1.0636} = \textbf{\textit{20.086 [Person/Months]}}$$

We can also calculate the number of months required to complete the software application by applying the following equation:

$$Duration := 3.67 * Effort^{F}$$
$$F := 0.28 + 0.2 * (E - B) = 0.28 + 0.2 * (1.0636 - 0.91) = 0.31072$$

Thus we obtain:

$$\textbf{\textit{Duration}} := 3.67 * 20.086^{0.31072} = \textbf{\textit{9.3129 [Months]}}$$

The number of people required can be calculated as:

$$\textbf{\textit{N}}_{\textbf{\textit{people}}} := Effort / Duration = 20.086 / 9.3129 = \textbf{\textit{2.15 [People]}}$$

3 Tasks Identification and Resources Allocation

3.1 Tasks Identification

Here we present a table listing the main tasks of which MyTaxiService's project development is composed, listed from first to last in the order they are going to be dealt with, up to the integration testing phase, which is the last one that we ourselves as a team are going to be taking care of. The following phases of the development, which include different phases of integrated system testing, performance testing and all the other phases that will come before the application distribution, won't be analyzed in this description, being out of our team's scope.

Task	Main Subtasks	Scheduled	Available Working Time
RASD	1. Actors, goals, domain properties and constraints definition 2. Requirements definition 3. Scenarios identifying and use cases definition 4. Alloy model development	from *21/10/2015* to *6/11/2015*	~2 weeks
DD	1. Architectural design definition 2. Component view definition 3. Runtime view definition 4. Algorithms design 5. User interface design	from *12/11/2015* to *4/12/2015*	~3 weeks
Integration Testing Plan	1. Integration strategy definition 2. Tests definition and description	from *08/01/2016* to *21/01/2016*	~2 weeks
Project Plan	1. Function points estimation 2. COCOMO analysis 3. Tasks definition and allocation 4. Project risks evaluation	from *21/01/2016* to *02/02/2016*	~2 weeks
Software Implementation	1. Server application implementation 2. Client applications implementation 3. Code inspection and unit testing	to be defined	to be defined
Integration Testing	1. Component integrations 2. Subsystems integrations	to be defined	to be defined

3.2 Resources Allocation

Here we provide a table showing, for each task and subtask of the project, how it has been or will be allocated in terms of team members taking care of it. Then, according to the confrontation of some estimated parameters (see below for details) with the actual ones that can be derived from the project schedule presented in the above table, for each task it is estimated if it is going to be completed in time, according to the given deadline, or not.

Task	**Subtask**	**Team members allocation**	**Estimated individual member availability** (hours per week)	**Estimated working time needed** (hours of work and days/weeks/ months estimate)	**Estimated deadline fulfillment** (is estimated working time >= available working time?)
RASD	1.	Both	~12 hours/week	~ 70 hours ~3 weeks	NO
	2.	Bulloni			
	3.	Both			
	4.	Cannici			
DD	1.	Both	~12 hours/week	~ 70 hours ~ 3 weeks	YES
	2.	Bulloni			
	3.	Both			
	4.	Both			
	5.	Cannici			
Integration Testing Plan	1.	Both	~7 hours/week	~ 25 hours ~ 2 weeks	YES
	2.	Both			
Project Plan	1.	Both	~3 hours/week	~ 25 hours ~ 4 weeks	NO
	2.	Cannici			
	3.	Bulloni			
	4.	Both			

Software Implementation	*1.*	*Both*	-	~ 10 months (COCOMO estimate for a 2 people team)	-
	2.	*Both*			
	3.	*Both*			
Integration Testing	*1.*	*Both*	-	-	-
	2.	*Both*			

For each task, depending on the period of the year in which it has to be faced (i.e. considering the amount of workload already present for each team member in that period of the year, which dramatically increases as an exam session approaches), it is provided an *estimated individual member availability (EIMA)* parameter, expressed in hours/week of working time available to each team member, and an *estimated working time needed (EWTN)* parameter, which expresses the time (in hours) we believe it is needed to complete that specific task. The corresponding *estimated working time needed* in weeks is obtained by computing

*EWTN [weeks]= EWTN[hours]/(N. of Team Members * EIMA)*

This parameter represents an estimate on how many weeks of work for the whole team are needed to successfully complete the task. By applying a confrontation with the actual *available working time* (retrievable from table of section 3.1 for each task), that has been calculated by checking how many days the team has to work on the task from the day it is assigned to the deadline day for that task, we can draw a conclusion stating if we believe that the team is able to successfully complete the task in the provided time (i.e. *available working time* >= *estimated working time needed*) or not (opposite). This statement is expressed in the *estimated deadline fulfillment* column of the table. As we can see, 2 out of 4 of the tasks for which we have been assigned an actual deadline, and can thus be examined to make the estimation, turn out to exceed the available working time. This may translate into either trying to complete the task in less time, which certainly leads to a much more approximate valuability of the performed job, together with many other downsides concerning the degree of understanding and learned skills derived from the assigned task, or in missing the deadline fulfillment. Therefore, to ensure a good performance level and to guarantee the usefulness of the understanding and learning process associated to working on a task, we believe that for each single task of the project the available working time should match the estimate of needed working time.
No estimates of this kind could be made for the last two tasks since the team doesn't know yet about those tasks' deadlines.

4 Project Risks

For what concerns business risks, being this project developing an application for a public transportation company, which basically merely answers an innovation need of the clients of the company without modifying the kind of service that was already provided, it will most probably be used by people without even any big need of sponsorship investments, and therefore it seems unlikely that any business risk may threaten the project.
About risks involving the project's development, being the team quite lacking of specific experience in planning, managing and working on this kind of delicate project, it is indeed possible that the predicted schedule won't be respected and that the application, together with the consequent testing, will take more time than what has been estimated. However, being the project's size quite small and thus assuming that the company will provide all the needed resources in terms of developing and testing environments, the development will most probably be carried out without huge losses of time or money in the end.
Regarding risks that should involve a well studied approach during the development, in order to avoid facing them when it's too late and losing huge amounts of time and money to fix them at that point, an important risk to be analyzed for the project might concern the highly variable workload to which the servers of the application are subjected to: it may happen, in case of particular holidays, events, or even just during a specific hour, that huge amounts of contemporary requests are received by the servers, or that all the taxicabs available in a certain window of time become booked for reserved rides, leaving no one to handle ride requests (e.g. on new year's eve). Therefore, in order to avoid the risk of crashes, service unavailabilities or slowdowns, it is very important to apply the most severe and complete performance testing before launching the application, paired with the most accurate data available to produce estimations on the amount of people using the service at each time of each day of the year, to ensure that the servers are ready to face the most request-crowded periods of time.
Moreover, it should also be considered another aspect concerning particularly busy time periods in terms of service usage: the fact that, even with fast servers handling all the requests in time, the number of available taxicabs might not always be enough or well enough placed to answer all the ride and reservation requests in a reasonable amount of time, especially in occasions like the ones listed above as risky in terms of workload, and this issue might get even bigger with the release of the application, since it's possible that the number of average taxicab users increases thanks to this new fast way to use it. However, the management of this logistic issue indeed overcomes the scope of this document and the job of the developing team, and should thus be properly predicted and taken care by the transportation company itself, in parallel with the application's development.
For what concerns issues that have to be taken into account as intrinsic problems of what is specifically being developed, and therefore can't be fully avoided in any way, even with the most accurate planning, the main risk that is likely to be faced after the release of the application is the application's full dependence on internet connection. This is something that can't be avoided, but it's indeed a risk that has to be taken into account. Of course the network of servers of the application shall be supported by the fastest available internet service, same for the mobile devices used by the taxi drivers, but also the users of the application have to be made aware of the fact that the deliverance reliability of any kind of request they make is subject to the reliability of their internet connection. However, the danger concerning this kind of issue is with any chance going to fade away with time, as

internet service providers are constantly improving the performance and reliability of their networks.

5 Appendix

5.1 Hours of work

Here is how long it took to redact this document:

- Matteo Bulloni: ~ 8 hours
- Marco Cannici: ~ 8 hours

References

[1] “QSM Function Points Languages Table”
Version: 5.0
Link: http://www.qsm.com/resources/function-point-languages-table

[2] “COCOMO II Model Definition Manual”
Version: 2.1
Link: http://csse.usc.edu/csse/research/COCOMOII/.../CII_modelman2000.0.pdf