MY TAXI SERVICE

Project Plan

Assignment 5

Amos Paribocci (854818); Lorenzo Pinosa (852231)

2/2/2016 - Version 1

Table of contents

1. Introduction

1.1 Revision history

No revision has been produced yet.

1.2 Purpose and scope

This document aims to describe a project plan for MyTaxiService by providing a detailed schedule of the project development phases, thanks to an estimation of cost and effort.

The schedule comes together with resource allocation and risk analysis, with the purpose of offering a complete view of the development of the project.

MyTaxiService platform scope has already been detailed in RASD and DD documents.

1.3 List of definitions and abbreviations

- User: any person who interacts with the system;
- Visitor: any user who has not registered an account or has not logged in;
- Customer/passenger: any user that will exploit the service in order to request a taxi to pick him up;
- Taxi driver: user to which customer requests are forwarded selectively by the system, and is expected to fulfill them in practice;
- Taxi zone: area of the logical city subdivision, to be used for taxi selecting and queueing management while serving a customer request;
- App: abbreviation for “mobile application”;
- System/backend: the part of the platform that will hold all the data and do all the processing on it, as well as handle instant communication and notification tasks. Mobile apps and the web service will rely upon this to work;
- MyTaxiService components: separate entities which cooperate to get MyTaxiService working and useful. They are the System Backend, the Customer Mobile App, the Driver Mobile App and the Web Service.
- Platform: what all MyTaxiService components build up;
- RASD: Requirements Analysis and Specification Document;
- DD: Design Document;
- ITPD: Integration Test Plan Document.

1.4 List of reference documents

- “Assignment 5 – Project Plan” document, provided in the Software Engineering 2 course context;
- MyTaxiService RASD (Requirements Analysis and Specifications Document);
- MyTaxiService DD (Design Document);
- MyTaxiService ITPD (Integration Test Plan Document);
- [COCOMO] COCOMO II Model Definition Manual (link);
- [IMECS 2011] - Top Ten Lists of Software Project Risks, March 16 – 18, 2011, Hong Kong (link).
- [UFP to SLOC] http://www.qsm.com/resources/function-point-languages-table

2. Effort and cost estimation

To estimate the required cost and effort to develop MyTaxiService we are going to use two techniques: Function points and COCOMO. With the first we will get an estimation of the total lines of source code written (SLOC - Source lines of code) and with the latter an estimation of the Person Months required to complete the platform development. COCOMO will also provide an estimation of the development duration: that information will be used in section 3 to allow a better activity schedule.

2.1 Function points approach

Function Points assign to each entry of function type a value based on the following table:

Function Types	**Simple**	**Medium**	**Complex**
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6
Internal Logic File	7	10	15
External Interchange File	5	7	10

2.1.1 Internal Logical File

This section reports an analysis for each entity that we plan to use in MyTaxiService database.

2.1.1.1 Simple entities

We consider the following entities as simple, since they contain only plain text data or numerical data:

User data, Notification data, Location point.

We have a total of 3 entities: 7 x 3 = 42 FP

2.1.1.2 Medium complexity entities

The following entities are dealing with geospatial data and thus we consider them of medium complexity: Shape descriptor (for taxi zones).

Total 1 entity: 10 x 1 = 10 FP

2.1.1.3 Complex entities

The remaining entities are considered to be complex because they interact and have relations with many others and they deal with dynamic changing data: Taxi information, Taxi Ride data, Taxi Request details, Taxi Booking details.

Total 4 entities: 15 x 4 = 60 FP

Total: 112 FP.

2.1.2 External Interface File

MyTaxiService needs to interface with Mobile Apps and a Web Service. Designing this communication is considered an activity of maximum complexity. We have thus 10 FP for each of: Customer Mobile App, Taxi Driver Mobile App, and Web Service

Total: 10 x 3 = 30 FP.

2.1.3 External Input

External inputs are operations that involves elaboration of an input without producing substantial output from the system. The MyTaxiService features that require such operations are the following:

- Location data received by taxis: it will update the taxi position without doing much other work. Considered as simple: 3 FP.
- Login data: receive user login data in order to authenticate him. Medium complexity: 4 FP
- Registration of a new user: a complex operation, since it involves complex data validation in addition to database insertion. 6 FP

Total: 13 FP

2.1.4 External Output

External Outputs are elaborations that process data in order to generate information needed from other components. For sure the following elaborations will be needed during MyTaxiService execution:

- Taxi selection to handle a specific request: a very complex operation. 7 FP
- Taxi retrieval based on geographic query: not simple, though not complex operation. 5 FP
- Notification sending: a specific protocol needs to be implemented. Maximum complexity: 7 FP
- Taxi booking issuing a taxi request when needed: medium complexity (involves task scheduling) 5 FP

Total: 24 FP

2.1.5 External Inquiry

Almost all main functionalities provided by MyTaxiService will be based on external inquiry. They are:

- Taxi Request: needs to elaborate a lot of data, maximum complexity. 6 FP
- Taxi Booking inserting, deleting, retrieval, editing: simple operations, 3 FP each = 12 FP

Total: 18 FP

2.1.6 Results

The sum of all FP is 197.

To estimate the number of SLOC, the following formula is used:

$$SLOC = UFP * SLOC/FP$$

Where SLOC/FP parameter is taken from [UFP to SLOC] page, for the C# programming language: 54.

The result is 10638 SLOC = 10,638 KSLOC.

2.2 Effort and cost estimation with COCOMO

In this section we will refer to section 3.1 and 3.2 of [COCOMO] document to list and assign a value to all COCOMO Scale Drivers and Cost Drivers.

2.2.1 Scale Drivers

2.2.1.1 Precedentedness (PREC)

We have no experience in developing a complex platform, though we have some experience in developing Mobile Apps. We thus decided for a conservative pick: Very Low, value 6.20.

2.2.1.2 Development Flexibility (FLEX)

The MyTaxiService platform commission was given with only general goals. We picked indeed “General Goals”, value 0.0.

2.2.1.3 Architecture / Risk Resolution (RESL)

We have done a Risk study in section 5. We are rather confidential with its accuracy, but even here we wanted to make a conservative choice and selected “often”. Value: 4.24.

2.2.1.4 Team Cohesion (TEAM)

We will work in tight and constant collaboration: seamless interaction. Value: 0.0

2.2.1.5 Process Maturity (PMAT)

To compute the Process Maturity we have first compiled the Key Process Area Questionnaire to get an Estimated Process Maturity Level (EPML), and compared it to Table 15 in [COCOMO] to get PMAT Rating.

Once compiled the KPA Questionnaire, the following formula has been used to calculate EPML, as stated in [COCOMO]:

$$EPML = 5 * \left(\sum_{i=1}^{n} \frac{KPA\%}{100} \right) x \frac{1}{n}$$

The result is 3.45, implying an PMAT rating of High. Value: 3.12.

2.2.1.6 Total

The total sum of the Scale Drivers is 13.56.

2.2.2 Cost Drivers

2.2.2.1 Required Software Reliability (RELY)

Reliability for MyTaxiService is important, because a platform malfunction can lead to financial losses of Taxi companies. It’s still not a “high financial loss” if the service is down for a limited time. We choose Nominal, value: 1.00.

2.2.2.2 Data Base Size (DATA)

The test database will store some mock data of Taxi Ride history, in addition to Taxi Requests and bookings. The size will be around 100Kb, with a D/P ratio of 10. We have chosen Nominal rating, value: 1.00.

2.2.2.3 Product Complexity (CPLX)

Examining Table 19 in [COCOMO], we have decided that Nominal rating is the best that describes MyTaxiService platform. Value: 1.00.

2.2.2.4 Developed for Reusability (RUSE)

We plan to reuse much code that we are going to write for MyTaxiService for other projects that we will develop in future. We have chosen Extra High, with a value of 1.24.

2.2.2.5 Documentation Match to Life-Cycle Needs (DOCU)

Since MyTaxiDriver platform is expected to be extended with plugins, we have to make sure that the documentation is really well done and easy to understand. High rating level was chosen, value: 1.11.

2.2.2.6 Execution Time Constraint (TIME)

Once MyTaxiService platform will be deployed, it is expected to be widely used and almost replace the actual way to request taxis. Thus, the hardware running the system backend will be under high load. We considered a High rating level, value: 1.11.

2.2.2.7 Main Storage Constraint (STOR)

The same considerations of the previous have been made for this one. Here we have to consider even the big amount of data that will be stored by the backend: all taxi ride, requests and bookings history. Rating level chosen: Extra High, 1.46.

2.2.2.8 Platform Volatility (PVOL)

Considering in particular the Mobile Apps, running on mobile OSes frequently updated, we have chosen a Nominal rating level. Value: 1.00.

2.2.2.9 Analyst Capability (ACAP)

Considering our performance in the past projects, we have decided to pick a conservative rating level: low. Value: 1.19.

2.2.2.10 Programmer Capability (PCAP)

We consider ourselves skilled in problem solving and algorithm elaborating. Experience has not to be considered in this point. Value chosen: High, 0.88

2.2.2.11 Personnel Continuity (PCON)

We are very continuative, since none of our team has quitted yet. A Very High value is appropriate here: 0.81.

2.2.2.12 Applications Experience (APEX)

We have poor experience in this type of application: a Very Low rating level is appropriate. Value: 1.22.

2.2.2.13 Platform Experience (PLEX)

We have some experience in some platform we are going to use, then a Low value seemed correct to us. Value: 1.09.

2.2.2.14 Language and Tool Experience (LTEX)

We will use tools and an IDE we are familiar with. We have an experience of 1 year and we picked Nominal rating level. Value: 1.00.

2.2.2.15 Use of Software Tools (TOOL)

We will use advanced tools to help us in MyTaxiService developing process. A Very High value is appropriate here. Value: 0.78.

2.2.2.16 Multisite Development (SITE)

We will work always together, in the same office. We have picked Extra High rating level, value: 0.80.

2.2.2.17 Required Development Schedule (SCED)

We have no needs to stretch or accelerate the schedule of the process. Nominal rating level is correct, value: 1.00.

2.2.3 Resulting Project Effort and Duration

Now that all parameters for COCOMO have been chosen, we can apply the equations 11 and 12 of [COCOMO] to calculate an estimated Project Effort.

Effort Manager (EM) values are the values specified in section 2.2.2:

$$\prod_{i=1}^{n} EM = 1.57002$$

Parameter E is calculated using the Scale Factor values:

$$E = B + 0.01 * \sum_{j=1}^{5} SFj = 1.0456$$

Lastly, the Person-Months estimated effort for MyTaxiService is:

$$PM = A * (KSLOC)^{E} * \prod_{i=1}^{n} EM = 54.6937\ PersonMonths$$

COCOMO provides even a way to estimate the development duration:

$$t_{DEV} = C * PM^{F}$$

Where C is a constant (C=3.67) and F is calculated by the following formula:

$$F = D + 0.2 * (E - B) = 0.30712$$

(D and B are other constants).

t_{DEV} becomes: 12.537 months, about a year.

The number of people needed according to that estimation would be:

$$P = \frac{PM}{t_{DEV}} = 4.55$$

2.3 Final comments

As described in section 4.1 – Team composition – the development team consists in 2 people, while the number of people that is needed in order to meet the development time would be 4: in a real-world scenario, it would be advisable to allocate some budget and hire 2 more developers (before the actual start of the project, avoiding unnecessary communication overheads).

In this project's case, though, we are asked to consider 2 people only: from now on, we will consider the project development duration to be doubled, consisting in 2*12.537 months = ~25 months = ~750 days.

3. Task identification and scheduling

3.1 Project tasks

In order to define the project tasks, we are adopting a retrospective point of view, supposing a waterfall model is followed. At the project start date, 15/10/2015, the project duration described in chapter 2.2 is adopted, for a total of ~750 days, and the time allocation for the phases is the following:

- Requirements Analysis and Specification → 10% (75 days);
- Product Design → 15% (~112 days);
- Product Programming → 50% (375 days);
- Integration and Testing → 25% (~187 days).

The reported percentages are being esteemed following the COCOMO II Model Manual reference document. A list of the tasks corresponding to each phase, together with the related time allocation, will follow.

3.1.1 Requirements Analysis and Specification (75 days)

- 3.1.1.1 Redaction of the RASD → 100% (75 days).

3.1.2 Product Design (112 days)

- 3.1.2.1 Redaction of the DD → 75% (84 days);
- 3.1.2.2 Redaction of the ITPD → 25% (28 days).

3.1.3 Product Programming (375 days)

- 3.1.3.1 Detailed design → 25% (~93 days);
- 3.1.3.2 System backend implementation and Unit testing → 35% (~131 days);
- 3.1.3.3 Customer mobile app implementation and Unit testing →15% (~56 days);
- 3.1.3.4 Taxi driver mobile app implementation and Unit testing → 15% (~56 days);

- 3.1.3.5 Customer Web service implementation and testing → 10% (~37 days).

3.1.4 Integration and Testing (187 days)

- 3.1.4.1 Integration testing → 25% (~46 days);
- 3.1.4.2 Performance/stress testing → 15% (~28 days);
- 3.1.4.3 Final test → 60% (112 days).

3.2 Task schedule

3.2.1 Dependencies analysis

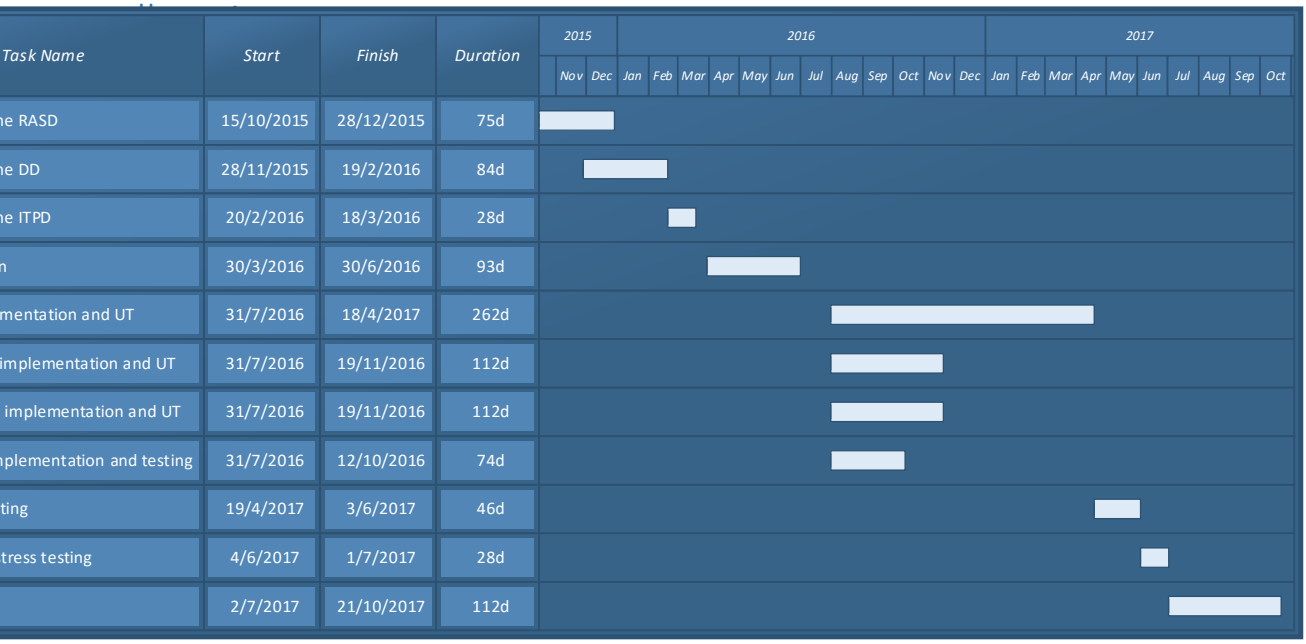
This section reports, for each task that has been listed in section 3.1, the list of dependencies:

- Redaction of the RASD: is the first task of the project (does not depend on any other task);
- Redaction of the DD: requires completion of task 3.1.1.1;
- Redaction of the ITPD: requires completion of tasks 3.1.2.1;
- Detailed design: requires completion of task 3.1.2.1;
- System backend implementation and Unit testing: requires completion of task 3.1.3.1;
- Customer mobile app implementation and Unit testing: requires completion of task 3.1.3.1;
- Taxi driver mobile app implementation and Unit testing: requires completion of task 3.1.3.1;
- Customer Web service implementation and testing: requires completion of task 3.1.3.1;
- Integration testing: requires completion of tasks 3.1.3.2, 3.1.3.3, 3.1.3.4, 3.1.3.5;
- Performance/stress testing: requires completion of task 3.1.4.1;
- Final test: requires completion of task 3.1.4.2.

3.2.2 Temporal overview and Gantt chart

Following the dependencies between tasks, tasks 3.1.3.2, 3.1.3.3, 3.1.3.4 and 3.1.3.5 may be carried on in parallel. We think, however, that this approach comes with the side-effect of slowing the said tasks: the task duration for those activities is doubled in the following Gantt chart.

	Task ID	Task Name	Start	Finish	Duration
1	3.1.1.1	Redaction of the RASD	15/10/2015	28/12/2015	75d
2	3.1.2.1	Redaction of the DD	28/11/2015	19/2/2016	84d
3	3.1.2.2	Redaction of the ITPD	20/2/2016	18/3/2016	28d
4	3.1.3.1	Detailed Design	30/3/2016	30/6/2016	93d
5	3.1.3.2	Backend implementation and UT	31/7/2016	18/4/2017	262d
6	3.1.3.3	Customer app implementation and UT	31/7/2016	19/11/2016	112d
7	3.1.3.4	Taxi driver app implementation and UT	31/7/2016	19/11/2016	112d
8	3.1.3.5	Web service implementation and testing	31/7/2016	12/10/2016	74d
9	3.1.4.1	Integration testing	19/4/2017	3/6/2017	46d
10	3.1.4.2	Performance/stress testing	4/6/2017	1/7/2017	28d
11	3.1.4.3	Final test	2/7/2017	21/10/2017	112d



4.2 Task allocation

Many of the project tasks are critical and require some discussion between the team members: that is why the following tasks will be performed by both Amos and Lorenzo:

- Task 3.1.1.1 – Redaction of the RASD;
- Task 3.1.2.1 – Redaction of the DD;
- Task 3.1.2.2 – Redaction of the ITPD;
- Task 3.1.3.1 – Detailed design;
- Task 3.1.4.1 – Integration testing;
- Task 3.1.4.3 – Final testing.

The remaining tasks will be assigned to the team members while keeping an even workload distribution, as descripted below (see chapter 3.1 – Project tasks – for information about task duration estimations).

Tasks allocated to Amos:

- Task 3.1.3.3 – Customer mobile app implementation and Unit Testing;
- Task 3.1.3.4 – Taxi driver mobile app implementation and Unit Testing;
- Task 3.1.3.5 – Web service implementation and testing.

Tasks allocated to Lorenzo:

- Task 3.1.3.2 – System backend implementation and Unit Testing;
- Task 3.1.4.2 – Performance/stress testing.

5. Risk analysis

5.1 Risk categories and identification

The following table summarizes some of the most important risks for the project.

The document [IMECS 2011] has been used as reference in order to identify and classify major risk categories in software project management (User, Requirements, Project Complexity, Planning & Control, Team and Organizational Environment).

For each risk type, an estimation of both probability (low, medium or high) and impact on the project (marginal, critical and catastrophic) is given, considering the team composition and experience.

Risk	Probability	Impact
Related to users		
Failure to manage end-user expectations	Medium	Critical
Failure to gain user involvement	Medium	Marginal
Related to requirements		
Unclear system requirements	Medium	Critical
Incorrect system requirements	Low	Critical
Misunderstanding of the requirements	Low	Catastrophic
Late changes to the requirements	Low	Critical
Related to project complexity		
High level of technical complexity	Medium	Marginal
Project involves the use of technology that has not been used before	High	Marginal
Inadequate security features being built into the system	Medium	Critical
Related to planning and control		
Inadequate estimation of required resources	Low	Critical
Unrealistic time and cost estimations	Medium	Critical
Lack of effective project management skills	Medium	Marginal
Changing scope/objectives	Low	Catastrophic
Related to the team		
Inexperienced team members	High	Critical
Temporary personnel shortfall (i.e. illness)	Medium	Critical
Permanent personnel shortfall (i.e. team member leaves the project)	Low	Catastrophic
Related to organizational environment		
Unstable organization environment	Low	Marginal
Lack of top management commitment and support	Low	Marginal
Change of ownership of senior management	Low	Marginal

5.2 Risk values motivation

Here will follow a brief description for each risk which has been reported in the table, explaining how the values for probability and impact have been determined.

5.2.1 Failure to manage end-user expectations

This risk is indeed a critical one, since meeting end-users expectations is a key factor for the product success. Since the development team is not particularly experienced, we decided to set the risk probability as medium.

5.2.2 Failure to gain user involvement

Involving users in the development phase is particularly useful for defining the product requirements and the requested functionalities. However, since the product scope is pretty well defined and of simple comprehension, we decided to keep the risk impact as marginal and the probability as medium.

5.2.3 Unclear system requirements

This risk, like pretty much all the risks involving the software requirements, is a critical one. The team is at its first experience with requirements analysis for such a big project, so we set the probability as medium.

5.2.4 Incorrect system requirements

We think that the product scope and mission are simple enough to allow a reasonable degree of correctness in the requirements, so we classify this one as a critical risk with low probability.

5.2.5 Misunderstanding of the requirements

Working on requirements that are different from the expected ones will surely produce catastrophic consequences. Because of the same reason of the previous risk, we set the probability as low.

5.2.6 Late changes to the requirements

A sensible late change to the product requirements is expected to negatively impact the project development in a negative way, slowing down the entire process. Because of the category of product described here, we think the probability of big changes is low.

5.2.7 High level of technical complexity

As described in section 2 – Effort and cost estimation – there are several complex methods and components that need to be developed in order to deliver the final product. We think that such complexity could slow the development and testing phases, so we think this is a marginal risk of medium probability.

5.2.8 Project involves the use of technology that has not been used before

Considering the team prior experience, this project expect indeed the use of new technology. The learning process will impact the development time in a sensible way, so this is a marginal risk of High probability.

5.2.9 Inadequate security features being built into the system

This risk is critical, like all the security-related aspects. The MyTaxiService product features some security element, especially in the communication between the system backend and the rest of the environment: we decided to set the risk probability as medium.

5.2.10 Inadequate estimation of required resources

The team that will be working on the project is of the simplest kind (only two people). That is why we consider this critical risk to be of low probability.

5.2.11 Unrealistic time and cost estimations

Time and cost estimation has been produced with a function points approach and with the usage of COCOMO (see chapter 2 – Effort and cost estimation – for further details). Given the lack of experience with said techniques, we do not exclude some imprecision in the performed estimations: the risk probability is set as medium.

5.2.12 Lack of effective project management skills

Even if it is true that the team members are at their first experience with the management of such a big project, we think such a small team is pretty easy to manage: the risk has been set as marginal with medium probability.

5.2.13 Changing scope/objectives

A change of scope and objectives in the middle of the product development is sure to affect the entire work in a catastrophic way. Since the product mission is well defined and pretty straightforward, we consider this risk to be of low probability.

5.2.14 Inexperienced team members

Because of what has been explained in section 4.1 – Team composition – we think that this critical risk is potentially going to materialize, thus we set the probability as high.

5.2.15 Temporary personnel shortfall (i.e. illness)

There are several reasons why some team member can be unable to work at the project, one of the most common being illness: since the team is only composed of two people, we think this critical risk is of medium probability.

5.2.16 Permanent personnel shortfall (i.e. team member leaves the project)

Given the small size of the team that will be working on the project, one of the team members leaving would undoubtedly cause catastrophic consequences, severely affecting the project development: even replacing the old member with a new one would cause communication overheads and slow down the entire process. We think this risk has low probability, because of the small team size and of the commitment to the project.

5.2.17 Organization-related risks

We do not know much about the organization requesting the product development: we are going to assume its reality is reasonably stable, both in terms of economic support and management commitment. That is why we classified all of the organization-related risks as marginal with low likelihood.

6. Appendix

6.1 Hours of work

This is the time spent in order to redact this document:

- Amos Paribocci: 6 hours;
- Lorenzo Pinosa: 6 hours.