POLITECNICO DI MILANO

Project Plan Document (PPD)

Computer Science and Engineering (CSE)

Software Engineering 2 Project

Year 2015/16

Date: 22/01/2016 – Version: 1.0

STUDENTS:

*Martino Andrea (**788701**)*

*Marchesani Francesco (**852444**)*

PROFESSOR:

Mirandola Raffaela

1. Introduction

1.0. Table of contents

1. Introduction 2
1.0. Table of contents 2
1.1. Revision History 3
1.2. Purpose and Scope 3
1.3. Glossary 4
1.4. List of reference documents 4
2. Function Points and COCOMO II estimations 5
2.1. Function Points technique 5
2.2. Function Points estimation 7
2.3. COCOMO II technique 14
2.4. COCOMO II estimation 14
3. Project tasks and Schedule 17
3.1. Project tasks 17
3.2. Project schedule 18
4. Resources Allocation 19
5. Risks Analysis of the project 19
5.1. Risks identification, relevance and recovery actions 19
6. Appendix 21
6.1. Hours of work 21

1.1. Revision History

We will keep the **revision history** of the **Project Plan Document** (**PPD**) in this chapter.

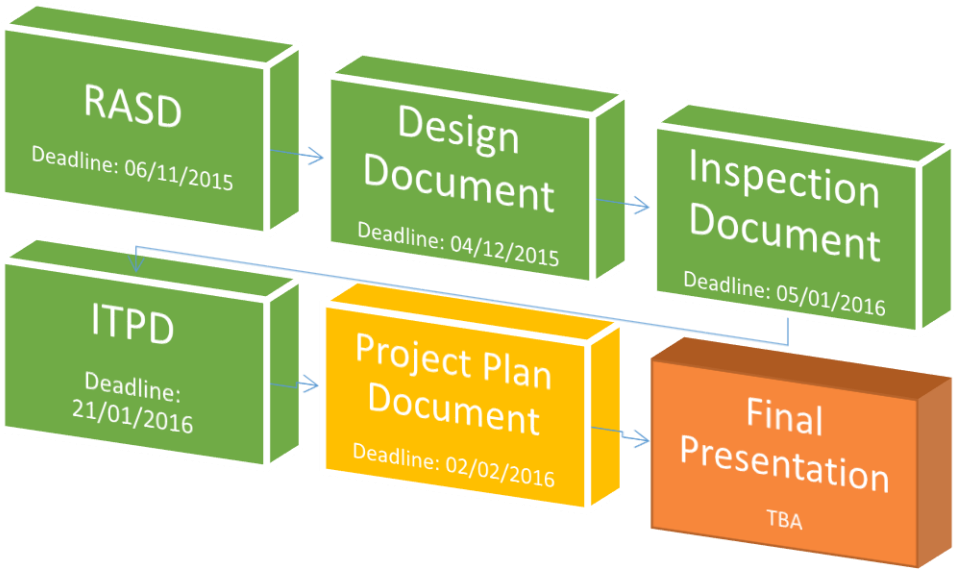
Version	Date	Author(s)	Summary
1.0	22/01/2016	Andrea Martino, Francesco Marchesani	Document Creation

1.2. Purpose and Scope

This **Project Plan Document (PPD)** contains information about several topics about project planning. We will discuss these topics in detail in the next chapters, with respect to the standards. We will do some assumptions in order to clarify specific points of the project (e.g. the implementation phase, not covered in the real course).

This **document** is coherent with *the official guidance template* of the project on the *Beep platform* with some additional chapters for the sake of completeness.
As we said for the past documents, it is important to underline that some parts of this document may evolve in the future (this may occurs for several causes).
Anyway, we will try to maintain coherence as much as possible.

Here is a resume of the steps of the project, with the related deadlines (in green documents already delivered, in yellow the current document):



The main scope of this **PPD** (*Project Plan Document*) is to give an overall guidance to the **project-planning phase** of the **project**, which is *myTaxiDriver* (**Software Engineering 2 project** of year 2015/16 - **Politecnico di Milano**).

We will focus in particular on the **algorithmic methods** used in practice (*Function points* and *COCOMO II*), on the **tasks identification**, on the **project scheduling**, on the **resources allocation** and on the **risk analysis**.

1.3. Glossary

- **RASD**: *Requirements Analysis and Specification Document*
- **DD**: *Design Document*
- **ITPD**: *Integration Test Plan Document*
- **PPD**: *Project Plan Document*
- **FPs**: *Function Points*
- **UFPs**: *Unadjusted Function Points*
- **GSCs**: *General System Characteristics*
- **COCOMO**: *Constructive Cost Model*
- **mTS**: *myTaxiService*
- **SE**: *Software Engineering*
- **Task:** *Activities which must be completed to achieve the project goal*
- **Schedule:** *A listing of a project's milestones, tasks, and deliverables*
- **Milestone**: *Points in the schedule against which you can assess progress*
- **Deliverable**: *Work product that is delivered to the customer*

Note: *for the full Glossary may be helpful to see also the paragraph 1.5 of the RASD 2.0, paragraph 2.3 of DD and paragraph 1.3 of ITPD.*

1.4. List of reference documents

Here is a list of the **reference documents** for the current *Project Plan Document (PPD)* of *myTaxiService*:

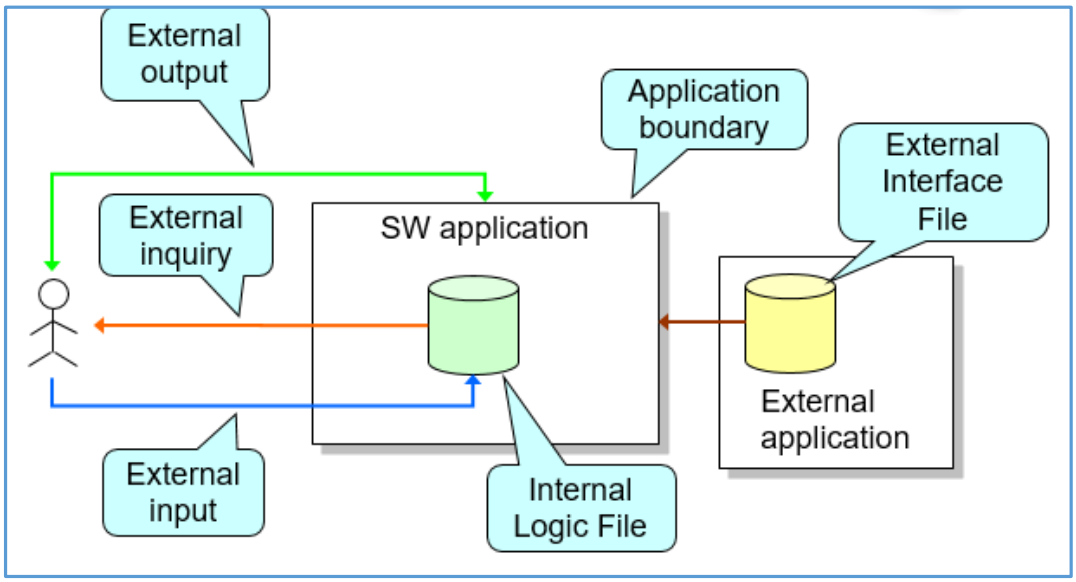
- **Project Description** (from *Beep* platform)
- **RASD 2.0 [RASD Revision]** (hosted on *GitHub Repository*)
- **Design Document [DD]** (hosted on *GitHub Repository*)
- **Integration Test Plan Document [ITPD]** (hosted on *GitHub Repository*)

2. Function Points and COCOMO II estimations

2.1. Function Points technique

Historically, **Allan Albrecht** (from **IBM**) introduced **function points** in **1979** in his work "*Measuring Application Development Productivity*". The functional user requirements of the software are grouped in **five macro-categories:**

- **Internal Logical File (ILF):** homogeneous set of data used and managed by the application.
- **External Interface File (EIF)**: homogeneous set of data used by the application but generated and maintained by other external applications.
- **External Input**: elementary operation to elaborate data coming from the external environment (from users).
- **External Output**: elementary operation that generates data for the external environment. It usually includes the elaboration and a proper representation of data from logic files.
- **External Inquiry**: elementary operation that involves input and output (e.g. specific requests of the user in order to visualize his/her information). This category does not implies significant elaboration of data from logic files.



The general idea is to give a **weight** for each category, with respect to a "*difficulty range*" (Simple, Medium and Complex). Here it is the related table:

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

Then it is possible to compute the amount of weights: the total number obtained (**UFPs**, *Unadjusted Function Points*) is an indicator of the project size.

$$\text{UFPs} = \sum(\text{number of elements of given type}) \times (\text{weight})$$

Optionally we can compute also the **Function Points** with respect to an adjustment based on **14 General System Characteristics [GSCs]** that can give a final correction of *±35%*. These GSC are:

1. *Data Communication*	8. *Online update*
2. *Distributed data processing*	9. *Complex processing*
3. *Performance*	10. *Reusability*
4. *Heavily used configuration*	11. *Installation ease*
5. *Transaction rate*	12. *Operational ease*
6. *Online data entry*	13. *Multiple sites*
7. *End user efficiency*	14. *Facilitate change*

The formula for the **Function Points** computation is the following:

$$FP = UFP \times \left(0.65 + 0.01 \times \sum_{i=1}^{14} F_i \right)$$

Where $\mathbf{F_i}$ are the coefficients for **GCSs** $\in \{0, 1, 2, 3, 4, 5\}$ depending from the specific weight.

2.2. Function Points estimation

After the introduction done in the previous chapter, let us apply the **Function Points estimation** method to the specific *myTaxiService* project, starting from the **computation of UFPs** for each **macro-category** (with respect to the corresponding weight).

2.2.1. ILF (Interior Logical Files)

ILF (Interior Logical Files)	Complexity	Weight
➢ The application stores information about customers and their position in simple structures.	Simple	7
➢ The application stores information about taxi drivers and their position in simple structure.	Simple	7
➢ The application stores information about the administrator(s) of the system. Once again, data structures are simple.	Simple	7
➢ The application stores information about valid addresses for the reservation in Maps Manager with simple data structures.	Simple	7

➢ The application stores information about the reservations made by customers in simple data structures.	Simple	7
➢ The application stores information about queues dynamically. The management is difficult, but the data structures are simple again.	Simple	7
➢ **TOTAL NUMBER OF ILF UFPs**	//	**42**

2.2.2. EIF (External Interface Files)

EIF (External Interface Files)	**Complexity**	**Weight**
➢ *myTaxiService* **does not** use data generated and maintained by other applications. It only communicates with the external APIs. Therefore, we have not UFPs for this category.	//	//
➢ **TOTAL NUMBER OF EIF UFPs**	//	**0**

2.2.3. External Inputs

External Inputs	**Complexity**	**Weight**
➢ The application must allow customer login/logout that can be considered a simple operation.	Simple	3
➢ The application must allow taxi driver login/logout that can be considered a simple operation.	Simple	3

➢ The application must allow SysAdmin login/logout that can be considered a simple operation.	Simple	3
➢ The application must allow the customer to make/delete a reservation. This operation is not easy, because involves the Customer, the Reservation Manager and the System Manager.	Medium	4
➢ The application must allow the system admin to manage part of the system in case of need. Due to the complexity of the system, we can estimate a medium complexity.	Medium	4
➢ The application must handle the taxi driver distribution. It is a simple operation managed by a specific algorithm.	Simple	3
➢ This application must allow a taxi driver to accept or delete a specific reservation. It is a medium operation as complexity, because there is the interaction between Taxi Driver, System Manager and Reservation Manager.	Medium	4
➢ **TOTAL NUMBER OF External Inputs UFPs**	//	**24**

2.2.4. External Outputs

External Outputs	**Complexity**	**Weight**
➢ The application must show an acknowledge to the costumer after	Complex	7

the decision about his/her reservation. This operation is complex because several entities are involved.		
➢ The application must show the maps on Taxi Diver's GPS. With the support of Google Maps, it is a medium operation.	Medium	5
➢ The application must show the requests to the Taxi Driver. A notification operation like this one involves several entities as seen before.	Complex	7
➢ The application must give the customer details about the reservation. This is a medium operation in terms of complexity.	Medium	5
➢ **TOTAL NUMBER OF External Outputs UFPs**	//	**24**

2.2.5. External Inquiries

External Inquiries	**Complexity**	**Weight**
➢ The application must allow the customer to request information about his/her reservations. It is a medium-complexity operation (just a query with selection on the customer).	Medium	4
➢ The application must allow the customer to request information about his/her profile. In addition, this one is a medium-complexity	Medium	4

operation too (just a query with selection on the customer).		
➢ The application must allow the taxi driver to request information about the next reservations. The complexity is medium as before.	Medium	4
➢ The application must allow the System to request information about Maps and Payment from the external APIs. It is a medium complexity operation, because the querying process with these APIs is not difficult.	Medium	4
➢ **TOTAL NUMBER OF External Inquiries UFPs**	//	**16**

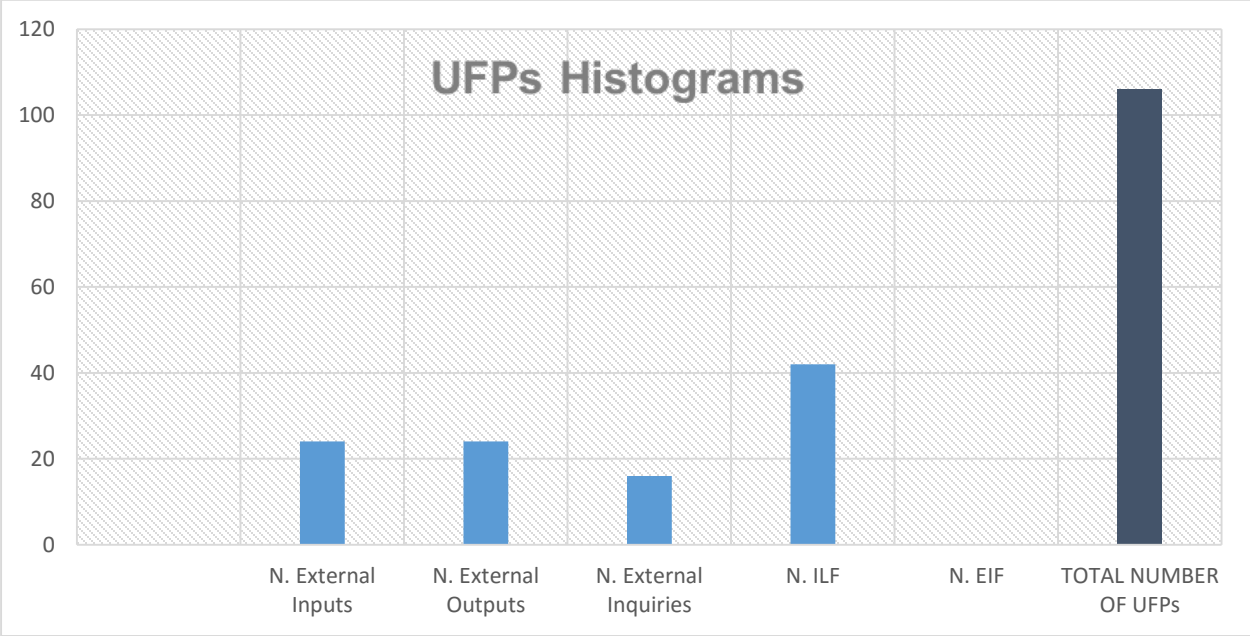
2.2.6. Total UFPs Computation

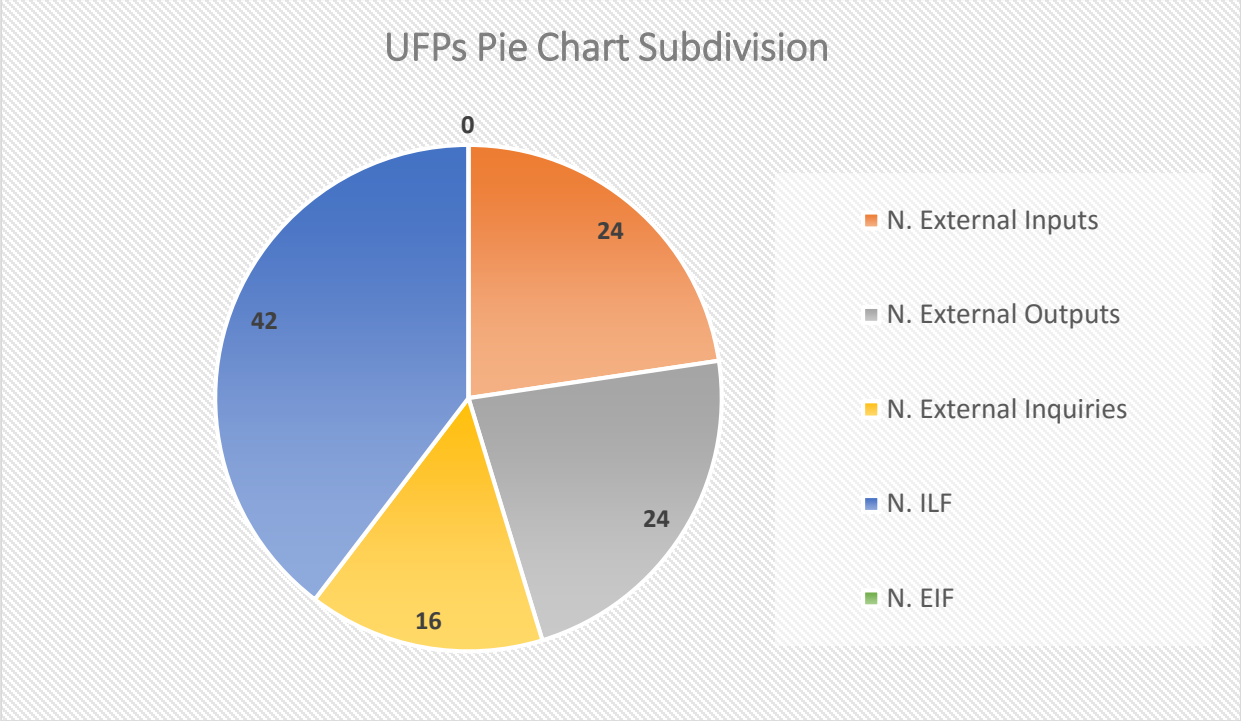
mTS Function Types	**Total weight**
➢ N. External Inputs	24
➢ N. External Outputs	24
➢ N. External Inquiries	16
➢ N. ILF	42
➢ N. EIF	0
➢ **TOTAL NUMBER OF UFPs**	**106**

As we can see, the **Unadjusted Function Points** (**UFPs**) number is **106**.

2.2.7. Graphical representations of UFPs

These are two **graphical representation** of **UFPs** based on *two different representation models* (**histograms** and **pie chart**):





2.2.8. Adjusted FP Calculation

Let us go on with the **adjustment of UFPs** based on the GSCs coefficients (**F_i**) estimated for this specific process. Note that there is a certain grade of freedom in the coefficients selection based on the personal interpretation of the given characteristics. We will give a **weight** from **0~5** as in the standard:

GSC	F_i coefficient	GSC	F_i coefficient
1. Data Communication	4	8. Online Update	2
2. Distributed Data Processing	1	9. Complex Processing	3
3. Performance	5	10. Reusability	1
4. Heavily Used Configuration	2	11. Installation Ease	2
5. Transaction Role	3	12. Operational Ease	3
6. Online Data Entry	2	13. Multiple Sites	0
7. End-User Efficiency	3	14. Facilitate Change	2

Therefore, we can calculate:

$$\sum_{i=1}^{14} Fi = 33$$

Then:

$$\textbf{FPs} = \text{UFPs} \times (0.65 + 0.01 \times \sum_{i=1}^{14} Fi) =$$

$$106 \times (0.65 + 0.33) =$$

$$106 \times 0.98 = \mathbf{103.88\ (\sim 104)}$$

We can easily conclude that the number of **Adjusted Function Points** is **104**.

2.3. COCOMO II technique

The **Constructive Cost Model** (**COCOMO**) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic **regression formula** with parameters obtained from historical project data and current as well as future project characteristics.
We will focus on **COCOMO II**, which is the improved version of the original model.
By the way, we will use this approach to estimate **effort** and **cost** of *mTS* project.

From a practical point of view, we will use the **COCOMO II – Constructive Cost Method** generator (*http://csse.usc.edu/tools/COCOMOII.php*) that takes into account all the **scale drivers** of this technique.

2.4. COCOMO II estimation

In order to perform a *myTaxiService* **COCOMO II estimation** we selected the **Function Points sizing method** with the *Unadjusted Function Points* calculated before (**106**) and Java as **programming language**. We also estimated a salary of 2000 Dollars *per Person-Month* (quite reasonable in practice).

- This is the **Software Scale Drivers** classification:

Software Scale Driver	Priority
Precedentedness	Low
Development Flexibility	Low
Architecture / Risk Resolution	Nominal
Team Cohesion	Very high
Process Maturity	High

- This is the **Software Cost Drivers** classification:

Software Scale Driver	Priority
Required Software Reliability	High
Data Base Size	Nominal
Product Complexity	Nominal
Developed for Reusability	Low
Documentation Match to Lifecycle Needs	High
Analyst Capability	High

Programmer Capability	High
Personnel Continuity	High
Application Experience	High
Platform Experience	Nominal
Language and Toolset Experience	High
Time Constraint	Very High
Storage Constraint	Nominal
Platform Volatility	High
Use of Software Tools	High
Multisite Development	High
Required Development Schedule	High

In the next sub-chapters are presented the **results** in detail of this estimation.

2.4.1. Software Development (Elaboration and Construction)

- **Effort** = 15.3 Person-months
- **Schedule** = 11.7 Months
- **Cost** = $30624
- **Total Equivalent Size** = 5618 SLOC

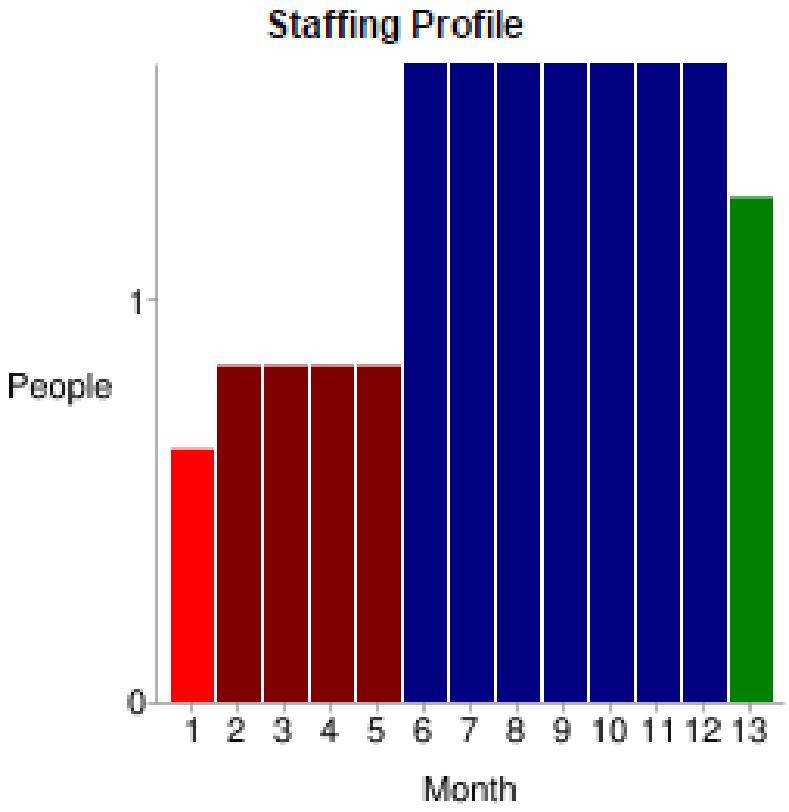
2.4.2. Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.9	1.5	0.6	$1837
Elaboration	3.7	4.4	0.8	$7350
Construction	11.6	7.3	1.6	$23275
Transition	1.8	1.5	1.3	$3675

2.4.3. Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.4	1.2	0.3
Environment/CM	0.1	0.3	0.6	0.1
Requirements	0.3	0.7	0.9	0.1
Design	0.2	1.3	1.9	0.1
Implementation	0.1	0.5	4.0	0.3
Assessment	0.1	0.4	2.8	0.4
Deployment	0.0	0.1	0.3	0.6

2.4.4. Staffing Profile



3. Project tasks and Schedule

In this section, we show you how *mTS* project with respect to the **different tasks**. Every task is then arranged to create a feasible **schedule**. Keep in mind that in our projects every task could be associated to a one or more **milestone(s)** and/or **deliverable(s)**.

3.1. Project tasks

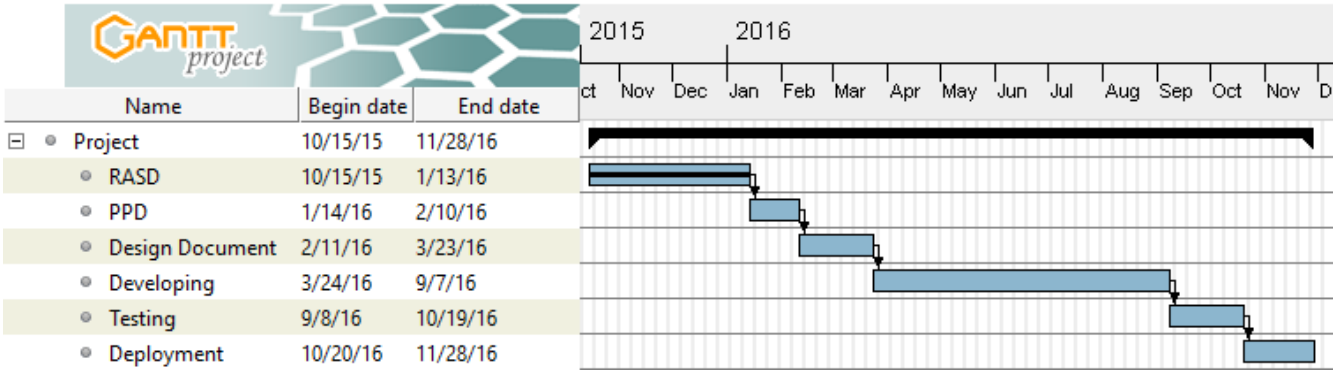
mTS Service project could be divided in these tasks:

Task	Milestone	Deliverable	Description
➢ ***RASD***	M1	RASD*)*	Every activities that belongs to *Requirements Engineering* (stakeholders' identification, requirements elicitation, …) and, therefore, contribute the writing of RASD.
➢ ***PPD***	M2	PPD	Function points identification, COCOMO II analysis and every other activities that contribute to the writing of Project Plan Document
➢ ***DD***	M3	DD	Architectural design, component definition and every other activities that contribute to the writing of Design Document
➢ ***Development***	*	//	Software development
➢ ***Testing***	M4, M5	Testing Document	Software unit testing (M4) and integration testing (M5).
➢ ***Deployment***	M6	First mTS release	Every software components is the deployed and the service becomes accessible by users.

* *Development* task is a complex task that would require several milestones and different intermediate deliveries. They can be decided after the *Design Document* completion. In *PPD*, we will not enter in details.

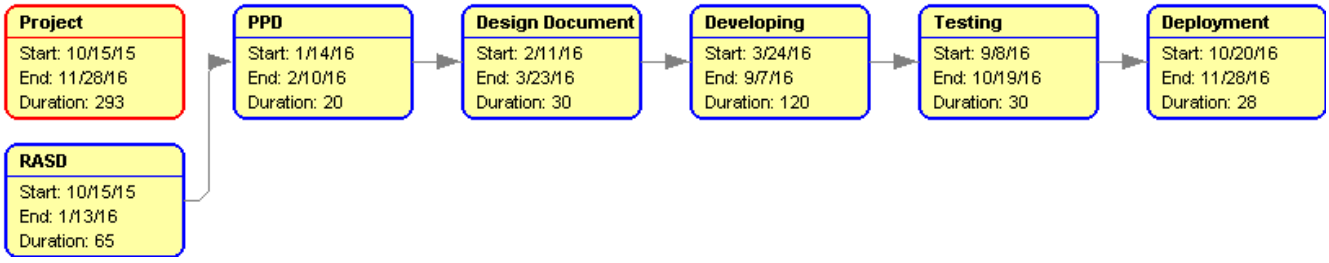
3.2. Project schedule

This is the **Gantt diagram** for the **project schedule**:



Gantt diagram

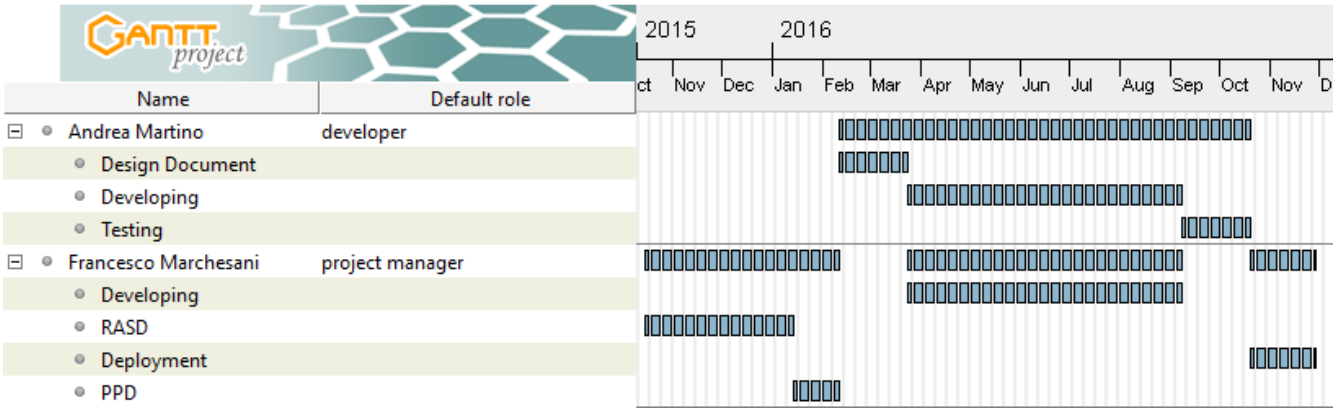
As it is possible to see in the **Gantt diagram,** every task is dependent from the previous one. In fact, *mTS* schedule will follow a ***waterfall-like*** *software development process.* This type of software development lifecycle is not always a good choice. However, we found completely reasonable to use this in *mTS* project because we think that stakeholders defined requirements in a clear and precise way.



PERT chart

4. Resources Allocation

Resources are allocated in order to respect *COCOMO II estimation analysis* as much as possible. This is a *graphical representation* of **resources allocation**:



Resources chart

5. Risks Analysis of the project

5.1. Risks identification, relevance and recovery actions

Here we can see in a compact way the **risk analysis** of myTaxiService project. Note that we will consider the main categories of possible risk. Of course it is impossible to understand at the beginning each possible source of risk (unless the risk estimator is a sort of Cassandra, as in the Greek mythology).

We will give a **relevance** with respect to a **five points scale** (where *0=NEGLIGIBLE, 1=VERY LOW, 2= LOW, 3=MEDIUM, 4=HIGH, 5=VERY HIGH*):

Risk	Relevance	Recovery actions

➢ Organizational finance problems	3/5	Prepare a briefing document for senior management of the project in order to show how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective, (they can only contribute to decrease the income).
➢ Recruitment problems	2/5	Tell the customers about probability of delays and search for pre-developed components that can fit myTaxiService project (without developing similar components from scratch).
➢ Staff illness	4/5	Considering that the probability of illness in winter is high, in case of illness we can perform team re-organization and meetings about work done by ill colleagues (in order to understand each other's job).
➢ Defective components	2/5	Substitute potentially defective components with bought-in components of known reliability, in order to avoid problems.
➢ Requirements changes	3/5	Derive traceability information to assess requirements change impact (it can be catastrophic in certain cases). We also need to perform information hiding in the project design in order to avoid chains of changes in the architecture.
➢ Organizational restructuring	1/5	As said for *Organizational finance problems*, we can prepare a document for a senior management to show that the project is very important for the business of the company and it is important to fulfill respecting the deadlines.
➢ Database performances	2/5	Consider the possibility of a general improvement of the database, with better performances.

➢ Underestimated development time	4/5	Consider the "*make or buy*" tradeoff, searching for suitable components of the project developed externally.

6. Appendix

6.1. Hours of work

- **Andrea Martino**: ~ 12 *Hours*
- **Francesco Marchesani**: ~ 12 *Hours*