



POLITECNICO MILANO 1863

Politecnico di Milano
A.A. 2015/2016
Software Engineering 2
Assignment 5: Project Plan
Version 1.0

Alessandro Baldassari (mat. 841561)
Alberto Bendin (mat. 841734)
Francesco Giarola (mat. 840554)

February 1, 2016

Contents

	Page
1 Introduction	1
1.1 Purpose and Scope	1
1.2 List of Definitions and Abbreviations	1
2 Function Point analysis	2
2.1 Introduction	2
2.2 FP types estimation	2
3 COCOMO II analysis	5
3.1 Introduction	5
3.2 Parameters choice and estimation	5
3.3 Second analysis with automated tool	6
4 Tasks identification	8
5 Task scheduling and Resources allocation	8
6 Risk planning and management	17
7 References	19
8 Appendix	19
8.1 Software and tools used	19
8.2 Hours of work	19

1 Introduction

1.1 Purpose and Scope

The main purpose of the project plan is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. A failure to adequately plan reduces the project's chances of successfully accomplishing its goals.

Project planning generally consists of:

- Identifying deliverables and creating the work breakdown structure;
- Identifying the activities needed to complete those deliverables and networking the activities in their logical sequence;
- Estimating the resource requirements for the activities;
- Estimating time and cost for activities;
- Developing the schedule;
- Developing the budget;
- Resource allocation (organization of work loads);
- Risk planning.

This document is based on an analysis made with two different algorithmic metrics of the system for *myTaxiService*. The first one is the Function Points (FP), which is used to estimate the software dimension (code size), which is directly used to evaluate the cost. The second is the COCOMO II that is used to estimate the efforts required in the development of a project by taking in account: characteristics of people, products and process.

1.2 List of Definitions and Abbreviations

The following acronyms are used in this document:

- FP: Function Points
- COCOMO: COConstructive COst MOdel
- ILF: Internal Logic File
- EIF: External Interface File
- PM: Person-Months
- SLOC: Source Lines of Code
- KSLOC: Thousands of SLOC
- SD: Scale Drivers
- EAF: Effort Adjustment Factor
- GUI: Graphical User Interface

The following definitions are used in this document:

- Deliverables: are work that are delivered to the customer, e.g. a requirement document for the system.

2 Function Point analysis

2.1 Introduction

The Function Point estimation approach is based on the amount of functionalities in a software and their complexity. Indeed the effort to develop a software project grows with the number of external inputs and outputs, user interactions, files and interfaces used by the system; therefore a weight is associated to all these functionalities and the total effort is computed summing all the partial values.

The parameters used to perform this estimation are summarized in the following tables, taken from COCOMO II, Model Definition Manual at:

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

The schema below defines the weights assigned to every level of complexity for all the FP types.

Table 3. UFP Complexity Weights

Function Type	Complexity-Weight		
	Low	Average	High
Internal Logical 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

Here is a brief explanation of the FP types:

- Internal Logic File: homogeneous set of data used and managed by the application
- External Interface File: homogeneous set of data used by the application but generated and maintained by other applications
- External Input: elementary operation to elaborate data coming from the external environment
- External Output: elementary operation that generates data for the external environment (it usually includes the elaboration of data from logic files)
- External Inquiry: Elementary operation that involves input and output (without significant elaboration of data from logic files)

2.2 FP types estimation

The assignation of the complexity to each functionality is based on the number of fields and interactions with other components required to implement it.

- ILF

Functionalities	Complexity	FP Count
Users	Simple	7
Ride	Complex	15
Request	Medium	10
Zone	Medium	10
Taxi	Simple	7
Payment-Receipts	Simple	7
Total:		56

- EIF

Functionalities	Complexity	FP Count
Payment Data	Medium	7
Google Maps	Medium	7
Push notification metadata	Simple	5
Total:		19

- Ext. INPUT

Functionalities	Complexity	FP Count
Login	Simple	3
SignUp	Simple	3
Create request	Simple	3
Pay	Medium	4
Delete request	Complex	6
Set taxi availability	Simple	3
Accept/reject ride request	Simple	3
Update taxi position	Simple	3
Manage taxi drivers	Medium	4
Create ride	Complex	6
Allocate taxi	Complex	6
Total:		44

- Ext. INQUIRY

Functionalities	Complexity	FP Count
Request history	Simple	3
Manage profile	Simple	3
Ride history	Simple	3
Public API	Medium	4
DB	Medium	4
Total:		17

- Ext. OUTPUT

Functionalities	Complexity	FP Count
SMS	Simple	4
Email	Simple	4
PushNotification	Simple	4
Monitor Zone	Medium	5
Total:		17

- Total and conclusions

FP types	Value
ILF	56
EIF	19
Ext. INPUT	44
Ext. INQUIRY	17
Ext. OUTPUT	17
Total:	153

The total number of FPs is then 153.

We can use this number to estimate the number of lines of code needed for the project; to do this a conversion factor is used: 46.

This value is obtained from the average value for J2EE language from the table at <http://www.qsm.com/resources/function-point-languages-table>.

Thus the estimate of SLOC will be:

$$153FPs * 46 = 7038 \text{ SLOC}$$

This result is the starting point for the next evaluation technique, the COCOMO approach.

3 COCOMO II analysis

3.1 Introduction

This estimation is achieved through a complex, statistical model that takes in account the characteristics of the product but also of people and process. The result of this technique is the estimation of Person-Months required to develop the project.

The COCOMO II calculations are based on the estimated of the software dimension in source lines of code (SLOC). Two very important metrics are used for this evaluation:

- Scale Drivers: the 5 SD determine the exponent used in the “effort equation”
- Cost Drivers: the 17 Cost Drivers represent the multiplicative factors that determine the effort required to complete the project

The result, the so-called “*Effort equation*” is:

$$\text{Effort} = 2.94 * EAF * (KSLOC)^E \text{ [Person-Months]}$$

Where:

- *EAF*: Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for the project)
- *E*: Exponent derived from SD
- *KSLOC*: SLOC measured in thousands

Once the effort has been computed it is possible to calculate the number of months required to complete the project with the duration equation:

$$\text{Duration} = 3.67 * (\text{Effort})^E \text{ [Months]}$$

Where:

- *Effort*: is the effort computed above
- *E*: is the schedule equation exponent derived from the five Scale Drivers

When both the *Effort* and the *Duration* are available then the number of people required to complete the project is:

$$N_{\text{people}} = \lceil \text{Effort} / \text{Duration} \rceil \text{ [People]}$$

3.2 Parameters choice and estimation

A first estimation is carried out considering a project with all nominal Cost Drivers and Scale Drivers: this means an *EAF* of 1.00 and exponent *E* of 1.0997. The effort equation thus becomes:

$$\text{Effort} = 2.94 * 1.0 * (7.038)^{1.0997} = 25.13 \text{ Person-Months}$$

Proceeding with the computation of the schedule equation the formula with the nominal parameter of *E* = 0.3179 becomes:

$$\text{Duration} = 3.67 * (25.13)^{0.3179} = 10.23 \text{ Months}$$

The size of the team is then:

$$N_{\text{people}} = \lceil 25.13 / 10.23 \rceil = \lceil 2.45 \rceil = 3 \text{ People}$$

The analysis gives the following result: it takes about 10 months of 3 people-work to finish this project.


3.3 Second analysis with automated tool

A second analysis is here presented; it has been done with the help of an online tool (<http://csse.usc.edu/tools/COCOMOII.php>), where it is easier and immediate to evaluate the difference in effort and scheduling as parameters of drivers change.

In particular the variations with respect to the nominal case are the following:

- Precedentedness: LOW, because the team has not much experience on large scale Java Enterprise projects with such high quality and reliability requirements
- Architecture/Risk Resolution: HIGH, since the team wants to avoid mistakes that can be raised during the development at a late stage that could involve the feasibility of the project itself; this is achieved through a thorough analysis of the architecture and the possible risks.
- Required Software Reliability: HIGH, since the expected quality of the system is very high, and on it depend a lot of processes (passenger clients, drivers, the public transportation for whole cities...)

Below are the settings of this analysis.



COCOMO II - Constructive Cost Model

Software Size Sizing Method Source Lines of Code ▾

[SLOC](#)

	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	<input type="text" value="7038"/>					
Reused	<input type="text"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text"/>		
Modified	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Software Scale Drivers

Precedentedness	Low ▾	Architecture / Risk Resolution	High ▾	Process Maturity	Nominal ▾
Development Flexibility	Nominal ▾	Team Cohesion	Nominal ▾		

Software Cost Drivers

Product	Personnel	Platform			
Required Software Reliability	High ▾	Analyst Capability	Nominal ▾	Time Constraint	Nominal ▾
Data Base Size	Nominal ▾	Programmer Capability	Nominal ▾	Storage Constraint	Nominal ▾
Product Complexity	Nominal ▾	Personnel Continuity	Nominal ▾	Platform Volatility	Nominal ▾
Developed for Reusability	Nominal ▾	Application Experience	Nominal ▾	Project	
Documentation Match to Lifecycle Needs	Nominal ▾	Platform Experience	Nominal ▾	Use of Software Tools	Nominal ▾
		Language and Toolset Experience	Nominal ▾	Multisite Development	Nominal ▾
				Required Development Schedule	Nominal ▾

Maintenance Off ▾

Software Labor Rates

Cost per Person-Month (Dollars)

Below is the output of the analysis.

Results

Software Development (Elaboration and Construction)

Effort = 27.6 Person-months

Schedule = 11.0 Months

Cost = \$220459

Total Equivalent Size = 7038 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.7	1.4	1.2	\$13228
Elaboration	6.6	4.1	1.6	\$52910
Construction	20.9	6.9	3.1	\$167549
Transition	3.3	1.4	2.4	\$26455



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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.8	2.1	0.5
Environment/CM	0.2	0.5	1.0	0.2
Requirements	0.6	1.2	1.7	0.1
Design	0.3	2.4	3.4	0.1
Implementation	0.1	0.9	7.1	0.6
Assessment	0.1	0.7	5.0	0.8
Deployment	0.0	0.2	0.6	1.0

The slight variation due to the different parameters does not really change the estimation:

- Duration: from about 10 to 11 months.
- Size of the team: always 3 people required ($N_{people} = \lceil 27.6/11 \rceil = \lceil 2.5 \rceil = 3 \text{ People}$)

This obvious result is a direct consequence of the increase in risk control and high-quality-software objectives prefixed with the parameters.

4 Tasks identification

The “Waterfall Model” of the software life-cycle is used as a guideline for the choice of the tasks and their order of execution.

ID	Task	Effort [Person-Days]	Duration [Days]	Dependencies
T1	Stakeholders identification	5	2	
T2	Actors identification	5	2	T1
T3	Goals identification	12	4	T2
T4	Requirements identification	18	6	T3
T5	Use Case and Scenarios	11	4	T4
T6	Class diagram	18	6	T5
T7	Alloy model	10	5	T6
T8	Components and interfaces	24	12	T7
T9	Deployment architecture	17	9	T8
T10	Runtime elements	15	8	T9
T11	Algorithms design	28	10	T8
T12	Sequence diagrams	19	7	T8
T13	User interface mock-up	25	25	T5
T14	Client manager development	30	15	T10, T11, T12, T13
T15	Account manager development	32	16	T10, T11, T12
T16	Request manager development	65	22	T10, T11, T12
T17	Ride manager development	38	19	T10, T11, T12
T18	Zone manager development	73	25	T10, T11, T12
T19	Taxi manager development	15	15	T10, T11, T12
T20	Notification manager development	18	18	T10, T11, T12
T21	Payment manager development	18	18	T10, T11, T12
T22	Integration testing strategy	35	12	from T14 to T21
T23	Integration testing execution	40	14	T22
T24	Software deployment	10	4	T23

The effort measured in Person-Months can be converted in Person-Days taking into account that in a months there are about 21 working days, so the 27.6 Person-Months are equivalent to $27.6 * 21 = 580$ *Person-Days*. The sum of the column “Effort” of the table above should match that figure.

5 Task scheduling and Resources allocation

























Below are shown the Gantt diagrams representing the guideline for the scheduling of the project which helps respecting the deadlines and deliverables.

First is presented an overall view of the whole project, later on the focus is on the particular of every section of the project.

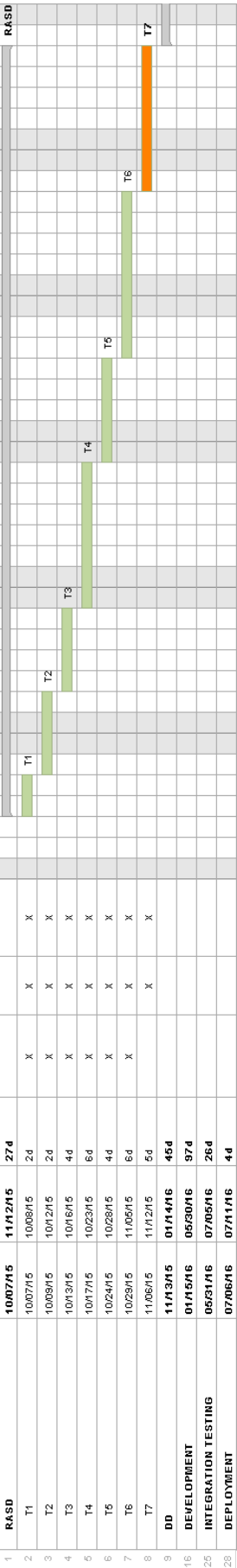
Be aware that in the charts the US date format is used!

The meaning of the colors is:

- GREEN: all the 3 members of the team are working on the task
- ORANGE: two member of the project are working on the task
- PURPLE: just one member of the group is working on that task.

Task Name	Start Date	End Date	Duration	Alessandro	Alberto	Francesco	Q4				Q1				Q2				Q3			
							Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep				
1	R4SD	10/07/15	11/12/15	27d																		
2	T1	10/07/15	10/08/15	2d	X	X																
3	T2	10/09/15	10/12/15	2d	X	X																
4	T3	10/13/15	10/16/15	4d	X	X																
5	T4	10/17/15	10/23/15	6d	X	X																
6	T5	10/24/15	10/28/15	4d	X	X																
7	T6	10/29/15	11/05/15	6d	X	X																
8	T7	11/06/15	11/12/15	5d	X	X																
9	DD	11/13/15	01/14/16	45d																		
10	T8	11/13/15	11/30/15	12d	X	X																
11	T9	12/01/15	12/11/15	9d	X	X																
12	T10	12/12/15	12/22/15	8d	X	X																
13	T11	12/23/15	01/05/16	10d	X	X																
14	T12	01/06/16	01/14/16	7d	X	X																
15	T13	11/13/15	12/17/15	25d		X																
16	DEVELOPMENT	01/15/16	05/30/16	97d																		
17	T14	01/15/16	02/04/16	15d	X																	
18	T15	03/03/16	03/24/16	16d	X																	
19	T16	03/25/16	04/25/16	22d	X																	
20	T17	02/05/16	03/02/16	19d	X																	
21	T18	04/26/16	05/30/16	25d	X																	
22	T19	03/05/16	03/24/16	15d	X																	
23	T20	01/15/16	02/09/16	18d	X																	
24	T21	02/10/16	03/04/16	18d	X																	
25	INTEGRATION TESTING	05/31/16	07/05/16	26d																		
26	T22	05/31/16	06/15/16	12d	X	X																
27	T23	06/16/16	07/05/16	14d	X	X																
28	DEPLOYMENT	07/06/16	07/11/16	4d																		
29	T24	07/06/16	07/11/16	4d	X	X																

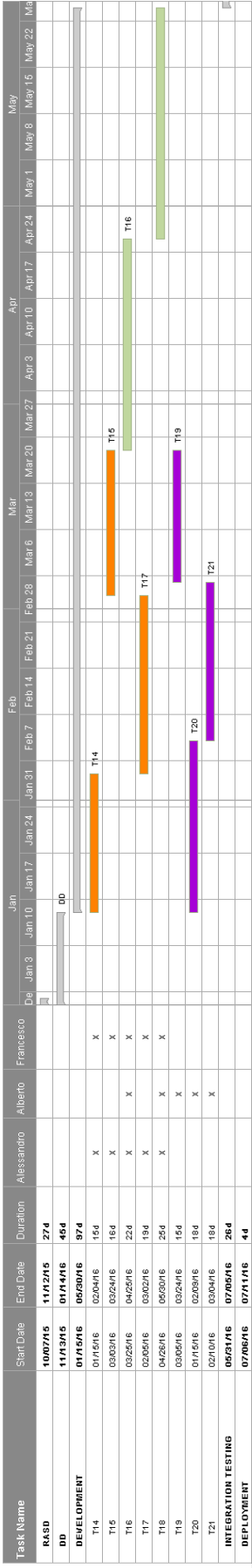
Scheduling and resources allocation for RASD document



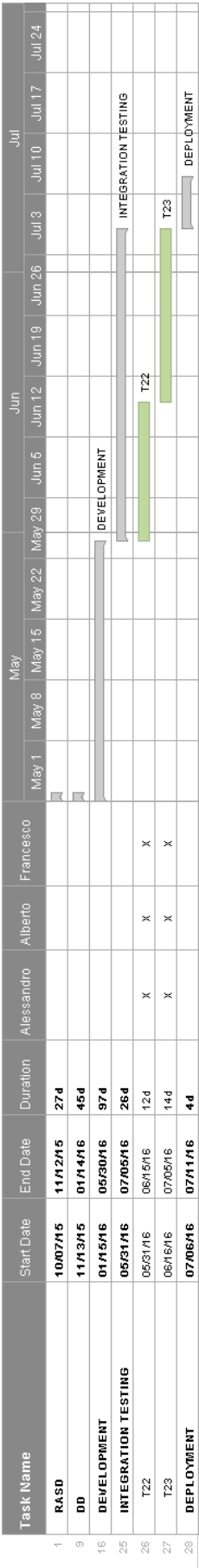
Scheduling and resources allocation for Design document

Task Name		Start Date	End Date	Duration	Alessandro	Alberto	Francesco	Nov			Dec			Jan						
								Nov 1	Nov 8	Nov 15	Nov 22	Nov 29	Dec 6	Dec 13	Dec 20	Dec 27	Jan 3	Jan 10	Jan 17	Jan 24
1	R4SD	10/07/15	11/12/15	27d																
9	DD	11/13/15	01/14/16	45d																
10	T8	11/13/15	11/30/15	12d	X	X														
11	T9	12/01/15	12/11/15	9d	X	X														
12	T10	12/12/15	12/22/15	8d	X	X														
13	T11	12/23/15	01/05/16	10d	X	X	X													
14	T12	01/06/16	01/14/16	7d	X	X	X													
15	T13	11/13/15	12/17/15	26d			X													
16	DEVELOPMENT	01/15/16	06/30/16	97d																
25	INTEGRATION TESTING	05/31/16	07/05/16	26d																
28	DEPLOYMENT	07/06/16	07/11/16	4d																

Scheduling and resources allocation for the development phase



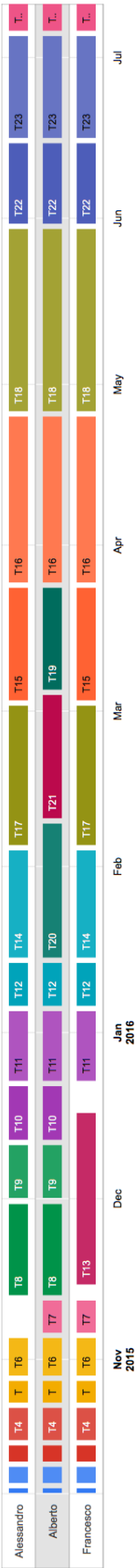
Scheduling and resources allocation for integration testing



Scheduling and resources allocation for deployment of the final product

Task Name	Start Date	End Date	Duration	Alessandro	Alberto	Francesco	Jul 3							Jul 10						
							S	M	T	W	T	F	S	S	M	T	W	T	F	S
1 RASD	10/07/15	11/12/15	27d																	
9 DD	11/13/15	01/14/16	45d																	
16 DEVELOPMENT	01/15/16	05/30/16	97d																	
25 INTEGRATION TESTING	05/31/16	07/05/16	26d																	
28 DEPLOYMENT	07/06/16	07/11/16	4d																	
29 T24	07/06/16	07/11/16	4d	X	X	X														

The following diagram shows the allocation of the three people in the team in a more clear way, with the tasks grouped by person.



It is important to notice that in the scheduling of the project and in the allocation of resources the difference between business days and weekends has been taken into account. The workweek is from Monday to Friday, while the weekend is Saturday and Sunday. Days are 8 hours of work time.

For instance the tasks T1 and T2 are both 2-day long, but the latter covers the weekend, so it has been counted up for 4 days instead of 2.

Public holidays have not been considered since this is an approximate schedule, but they could cause up to about 10 days of delay on the proposed schedule.

The proposed schedule matches the 10/11 months expected duration of the project forecast with the COCOMO analysis, in fact the planned work starts October 7, 2015 and ends July 11, 2016.

6 Risk planning and management

The main factors of risk are related to project and technical ones; after an analysis of all the possible risks, those which are ranked higher in the probability-of-occurrence list are:

1. Unrealistic schedule
2. Availability of staff
3. Wrong functionalities
4. Wrong User Interface
5. Bad external components
6. Real-time shortfalls

A *proactive* strategy is employed, thus a contingency plan is developed to handle unavoidable risks in a controlled and effective manner.

Respectively:

1. To avoid delays in the scheduled work, the work plan itself has been drafted with more relaxed time windows for every deadline, taking into account a possible lower pace of work and effort to foresee such problems.
2. This strategy also covers the possibility of delays due to illness of key staff at critical times: being the due-date of the sections stretched a bit more, the other components of the team are able to catch up with the work of the unavailable members. Besides many tasks are carried out together, with the allocation of all the three co-workers, to bypass the overhead time that would be lost when a member has to complete a task not assigned to himself to cover an unavailable colleague. In other situations two members of the team work on high priority and complex tasks, while the third person carries on another task in parallel. Being the tasks assigned to the single individual simpler and with a lower priority (e.g. User Interface mock-ups), if it happens that he is ill, the others can easily continue his work so that possible delays do not have a great impact on the most important outcomes.
3. Many interviews with stakeholders and other managers of the project are organized as soon as the project is assigned to the team. It is in the interest of both the parties (stakeholders and developing team) to correctly understand the target functionalities of the system to-be-produced. Besides often *milestones* are scheduled to be screened with the client, so as to have a constant check on the goals and functionalities that can be assessed and possibly corrected at any moment, in order to avoid sudden changes in requirements that require a major design rework.
4. Mock-ups of the final GUI are drafted at an early stage; these are sent to the client for approval and possibly corrected. During the screening of milestones the client is able to test the GUI and the interactions with the system in order to have a continual assessment of these.
5. The external components on which the team relies to set-up the system (e.g. payment provider, SMS and email services) are high quality ones offered by well known companies that have been in the business for years. This choice has been taken to avoid dependencies with just-created unknown services which could be unreliable or shut-down in the near future. Others are bought-in components of known reliability, easily replaceable.

6. In order to offer an high quality service and fast response-time of the system, the whole service has been calibrated to handle many more connections and higher work-load than the real estimate of future use. All the components have been “underestimated” and tested in the worst conditions of usage.

7 References

Material from Wikipedia

- Project management: https://en.wikipedia.org/wiki/Project_management#Planning

8 Appendix

8.1 Software and tools used

- TeXstudio 2.10.6 (<http://www.texstudio.org/>) to redact and format this document.
- Astah Professional 7.0 (<http://astah.net/editions/professional>)

8.2 Hours of work

The time spent to redact this document:

- Baldassari Alessandro: 12 hours.
- Bendin Alberto: 12 hours.
- Giarola Francesco: 12 hours.