POLITECNICO
MILANO 1863

Computer Science and Engineering

SOFTWARE ENGINEERING 2

A.A. 2015 / 2016

MyTaxiService

Project Plan

February 2nd, 2016

REFERENCE PROFESSOR:

Mirandola Raffaela

AUTHORS:

Polidori Lucia

Regondi Chiara

Table of contents

1 Introduction

1.1 Purpose and scope

In this document we are going to describe the project plan for our “MyTaxiService” application, supposing that the project team is simply composed by the two of us.
The purpose is to evaluate the size of the project and estimate the effort and the cost by applying two of the main algorithmic estimation techniques: Function Points and COCOMO II.
This techniques allow us to compute the project effort based on estimates of product attributes, such as the size, the project characteristics, and so on.
However, the goal of this document is not simply reporting the cost estimation results, but it’s also to illustrate the project planning phase. In particular, we are going to identify the main tasks and their schedule and how each member of the group has been allocated to each one of them.
Diagrams describing these tasks with their durations and dependencies, activities and staff allocation will be provided.

1.2 Glossary

Here’s a small glossary containing the definitions of some words and concepts we will use in this document:

Cost Driver: a number representing the multiplicative factor that needs to be used in order to compute the effort required to complete a software project.

DD: the *Design Document*, a document containing the description of the design and the architecture of our My Taxi Service system.

Deliverable: a work product that has to be delivered to the customer at the end of the project or within a specific deadline.

External Input: elementary operation used to elaborate data coming from the external environment.

External Output: elementary operation that generates data for the external environment.

External Inquiry: elementary operation that involves inputs and outputs of the system.

External Logic File: homogeneous set of data used by the application but generated and maintained by other applications.

Function Points: a technique used to evaluate the effort needed to design and develop custom software applications.

Internal Logic File: homogeneous set of data used and managed by the application itself.

ITDP: the *Integration Test Plan Document*, a document containing the description of the integration testing process performed on our software project.

Project Planning: phase of the Software Project Management that aims at identifying tasks, estimating and scheduling the project development and assigning people to tasks.

RASD: the *Requirements Analysis and Specification Document*, a document containing the description of the requirements and specifications analysis phase of our software development process.

Risk: potential problem that can happen or not happen.

Scale Driver: factor that contributes to the effort and duration estimation of a project.

Source Lines of Code: number of lines of code written, in a specific programming language, for the implementation of a software application.

Task: an activity that must be completed in order to achieve the project goal.

Weight: the numeric value of every function used in the Function Points algorithm in order to compute the number of each function type. It can be simple, medium or complex.

1.3 Reference documents

We wrote this document referencing the project description provided by our professor, the RASD document, the Design Document and the ITPD document we have developed in the previous months and drawing from old documents of previous projects.

1.4 Document structure

We have decided to structure our Project Plan Document as follows:

- *Section 1*: Introduction
  this first section gives an introduction to the Project Plan Document, specifying the purpose and the scope of the document, the glossary and the documents we referred to in order to develop this paper.
- *Section 2*: Function Points
  in this second section we describe the Function Points approach we used in order to estimate the size of our project.
- *Section 3*: COCOMO II
  in this section we report the results of the COCOMO approach we used in order to estimate the effort and the cost of our project.

- *Section 4*: Project Planning
  this section contains the description of the project planning phase, with the identification of all the tasks and activities, the project scheduling and the resources involved.
- *Section 5*: Project Risks
  this section contains the definition of the risks for the project, their relevance and the associated recovery actions.
- *Section 6*: Conclusions
  this section contains some final considerations on what we have described in the previous sections.
- *Section 7*: Used Tools
  in this section, all the tools we have used in order to develop this document are listed.
- *Section 8*: Working Hours
  this sections contains the result of our effort, quantified in the number of hours we have needed in order to develop this document.

2 Function Points

In this chapter we are going to apply the Function Points algorithm in order to estimate the effort that is required for the design and the implementation of our project.
In particular, we use this approach to evaluate the size of the project on the base of the functionalities of our application that we have already described in the RASD document.
Then, starting from the result of this algorithm, we will estimate the size in terms of “lines of code” and proceed with the estimation of the effort and the duration applying the rules defined by the COCOMO II approach.

We use this table to evaluate the weight for every function:

FUNCTION TYPES	WEIGHTS		
	Simple	***Medium***	***Complex***
N. Internal Files	7	10	15
N. External Files	5	7	10
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiries	3	4	6

In particular, for our application we have:

Internal Logic File (*entities of the system*):
The system has to store and manage information about User, Reservation, Request, Payment, Taxi Driver, Taxi, Street and Notification. The tables have a different number of columns, but nothing will be very large. They can be considered *simple* structures.
So, we obtain:

$$\text{ILFs}: \ 7 * 7 = 49$$

External Logic Files (*communication between different software*):
The system localizes the distribution of the taxis in each zone using a GPS locator. We can consider it of *medium* complexity.
So, we have:

$$\text{ELFs}: \ 1 * 7 = 7$$

***External Inputs*:**

- **Login**: *simple*. It only requires username and password.
- **Registration**: *simple*. It requires to fill more fields than login, but anyway simple.
- **Update profile information**: *medium*. It requires modifying some fields plus the profile images. The system has to update the database too.
- **Request a taxi (filling out the related form)**: *simple*. It requires only few information.
- **Reserve a taxi (filling out the related form)**: *simple*. It requires only few information.
- **Delete a reservation**: *simple*. This operation involves two entities: User and Reservation, but it's simple.
- **Receive information about taxi driver availability**: *simple*.
- **Receive information about the calls the taxi driver takes care of**: *medium*. The system has to update the database.
- **Retrieve password**: *simple*. The user has simply to click on "forgot password".
- **Logout**: *simple*.

So, we obtain:

$$\text{External Inputs}: \ 8 * 3 + \ 2 * 4 = 40$$

External Outputs (*things that are created from the system for the user*):

- **Retrieve password**: *simple*. The system sends a new password through an email notification.
- **Notification for confirming a request**: *medium*. The system sends an email notification with the confirmation and a notification in the profile page's notifications box. It requires three entities: User, Request and Notification, so we can consider it of medium complexity.
- **Notification for confirming a reservation**: *medium*. The system sends an email notification with the confirmation and a notification in the profile page's notifications box. It requires three entities: User, Reservation and Notification, so we can consider it of medium complexity.
- **Notification for deleting a reservation**: *simple*. The system simply shows a new notification in the profile page's notifications box.
- **Forward request notifications to drivers**: *complex*. The system forward the request to the first taxi in queue. This operation requires more entities: Taxi, Taxi Driver, Reservation/Request and Notification.

So, we obtain:

$$\text{External Outputs}: \ 2 * 4 + \ 2 * 5 + 1 * 7 = 25$$

External Inquiries (*actions that required data retrieval from the database*):

- **Changing settings and showing user profile page**: *simple*. This two operations concern the user, so we could consider them as a single operation.
- **Visualize the reservations**: *medium.* It could show a lot of reservations.
- **Check payments**: *complex*. There could be a lot of payments to be checked.

So, we obtain:

$$\text{External Inquiries}: \ 1 * 3 + \ 1 * 4 + 1 * 6 = 13$$

In summary, we have computed the following number of Function Points:

$$49 + 7 + 40 + 25 + 13 = \mathbf{134\ FPs}$$

This table is a summary of the number of the function points that we have found in our project, each one multiplied by its own weight.

FUNCTION TYPES	WEIGHTS			
	Simple	***Medium***	***Complex***	
N. Internal Files	7*7	0*10	0*15	**49**
N. External Files	0*5	1*7	0*10	**7**
N. Inputs	8*3	2*4	0*6	**40**
N. Outputs	2*4	2*5	1*7	**25**
N. Inquiries	1*3	1*4	1*6	**13**
TOT				**134**

The value obtained, representing the total number of function points, will be used in the following chapter in order to estimate the size of the project in terms of Source Lines of Code.

3 COCOMO II

In this chapter we are going to estimate the effort (expressed in person-months), the number of people required and the total time (expressed in number of months required to complete the project).
In order to do that, we use the COCOMO II approach.
COCOMO is a software cost estimation model used for estimating effort, cost and schedule of software projects. The version that we are going to use is COCOMO II, the successor of the basic and first published COCOMO 81.

COCOMO II model is based on the following elements:

- ★ **Source Lines of Code (*SLOC*)**

- ★ **Scale Drivers (*SD*)**
  they represent very important factors that contribute to the project's duration and cost. Their values determine the exponent used in the Effort Equation.
  The possible values for each scale driver are: *Very low*, *Low*, *Nominal*, *High*, *Very high*.
  Example of relevant Scale Drivers can be:
  - Precedentedness
  - Development Flexibility
  - Team Cohesion
  - Project Maturity

- ★ **Cost Drivers**
  they represent multiplicative factors that determine the effort required to complete the software project. They can assume one of the following values: *Very low*, *Low*, *Nominal*, *High*, *Very high*. "Nominal" means always 1, that is no change on the total effort.
  Example of relevant Cost Drivers can be:
  - Required Software Reliability
  - Product Complexity
  - Required Reusability
  - Analyst Capability
  - Programmer Capability
  - Personnel Capability

- ★ **Effort Equation**
  it estimates the effort, measured in Person-Months (number of months per person), that is required in order to complete the project.

$$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^{\text{E}}$$

where:

EAF = ***E**ffort **A**djustment **F**actor* derived from the Cost Drivers

E = exponent derived from the values of the Scale Drivers

★ **Schedule Equation**

it predicts the number of months that are required to complete the software project.

$$\text{Duration} = 3.67 * (\text{Effort})^{\text{SE}}$$

where:

Effort = the result of the Effort Equation

SE = schedule equation exponent derived from the values of the Scale Drivers

Now we can perform the analysis on our "MyTaxiService" project using the COCOMO model describe above.

First of all, starting from the Function Points value found in the previous chapter, we have to provide an estimate of the Source Lines of Code in order to compute the effort.

As we have already stated in the Integration Test Plan Document, we supposed that our project is developed using Java EE. So, we use the average conversion factor for J2E language that is equal to 46.

$$134 \text{ FPs} * 46 = 6164 \text{ SLOC}$$

Having estimated the number of lines of code of our project, we can now proceed computing the effort, the duration and the number of people required, applying the formulas described before.

We consider our project as a project with all "*Nominal*" Scale Drivers and Cost Drivers: this means that we have a value of **EAF = 1.00** and exponents **E = 1.0997** and **SE = 0.3179**.

We obtain:

$$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^{\text{E}} = 2.94 * (1.00) * (6.164)^{1.0997} = 21.72 \textit{ Person-Months}$$

$$\text{Duration} = 3.67 * (\text{Effort})^{\text{SE}} = 3.67 * (21.72)^{0.3179} = 9.76 \textit{ Months}$$

Finally, using the two results obtained, we can compute the number of people needed to complete the project as the ratio between the Effort and the Duration:

$$N_{\text{people}} = \text{Effort} / \text{Duration} = 21.72 / 9.76 = 2 \text{ people}$$

In conclusion, we can see that the COCOMO II prevision are greater that the real project, especially in terms of Effort and Duration (see the “Conclusions” chapter).
This is due to the fact that this estimate takes into account possible statics error and that maybe we have worked harder than expected.
In general, we are quite satisfied with the result we have obtained.

4 Project Planning

4.1 Tasks identification and schedule

In this chapter we are going to identify all the tasks for our project, that is all those activities that we had to complete in order to achieve the project goal.

We can identify the following tasks for our project:

T1. **RASD**

T1.1 Read carefully the project specification and provide a general introduction on the requirements analysis phase

T1.2 Identify the goals and the domain properties of the project

T1.3 Identify the actors involved in the application

T1.4 Identify the functions and all the requirements of the project

T1.5 Identify some of the possible scenarios

T1.6 Design the UML diagrams representing the application

T1.7 Design the Alloy model of the project

T2. **Design Document**

T2.1 Provide a general introduction on the design phase

T2.2 Describe the architecture of the system (component, deployment, runtime views)

T2.3 Describe the selected architectural styles and patterns

T2.4 Describe some of the main algorithms

T2.5 Define the user interface and provide some mockups that illustrate it

T2.6 Provide the mapping between architectural components and requirements

T3. **Integration Testing**

T3.1 Provide a general introduction on the integration testing phase

T3.2 Define the integration strategy

T3.3 Describe the steps that have to be performed during the integration testing

T3.4 Provide a description of the tools and the test data required to perform the testing

T4. **Project Plan and Reporting**

T4.1 Provide a general introduction on the project planning and cost estimation

T4.2 Describe and perform the Function Points algorithm to estimate the size of the project

T4.3 Apply the COCOMO model in order to evaluate the effort and duration of the project

T4.4 Identify the main tasks, their schedule and the allocation of the needed resources

T4.5 Identify and describe the potential project risks

So, we have four main tasks that coincides with the deliverables of the project:

T1: Provide the *Requirements Analysis and Specification Document*

T2: Provide the *Design Document*

T3: Provide the *Integration Test Plan Document*

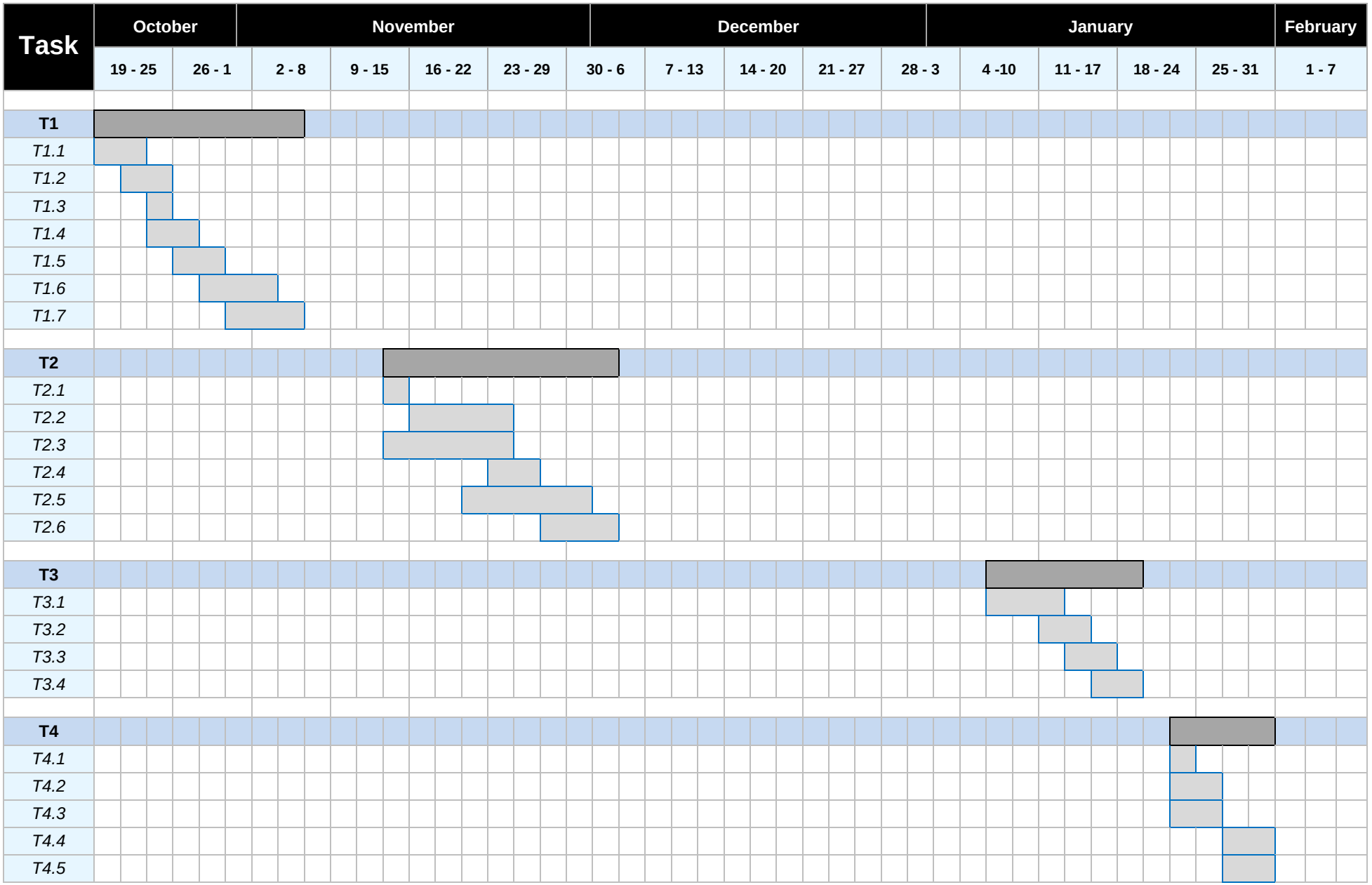
T4: Provide the *Project Plan Document*

and other sub-tasks that needs to be completed in order to satisfy all the main project tasks.

We provide here a simple table that reports all the main tasks with the corresponding duration, effort and dependencies:

TASK	START DATE END DATE	DURATION *(days)*	WORKING PEOPLE	HOURS per PERSON	DEPENDENCIES
***T1*: RASD**	**18.10.2015 04.11.2015**	**18**	**2**	**48**	**-**
T1.1: Read carefully the project specification and provide a general introduction on the requirements analysis phase					
T1.2: Identify the goals and the domain properties of the project					
T1.3: Identify the actors involved in the application					
T1.4: Identify the functions and all the requirements of the project					
T1.5: Identify some of the possible scenarios					
T1.6: Design the UML diagrams representing the application					
T1.7: Design the Alloy model of the project					
***T2*: Design Document**	**13.11.2015 03.12.2015**	**20**	**2**	**25**	**T1**
T2.1: Provide a general introduction on the design phase					
T2.2: Describe the architecture of the system					
T2.3: Describe the selected architectural styles and patterns					
T2.4: Describe some of the main algorithms					
T2.5: Define the user interface and provide some mockups that illustrate it					
T2.6: Provide the mapping between components and requirements					
***T3*: Integration Testing**	**07.01.2016 20.01.2016**	**13**	**2**	**15**	**T1, T2**
T3.1: Provide a general introduction on the integration testing phase					
T3.2: Define the integration strategy					
T3.3: Describe the steps to be performed during the integration testing					
T3.4: Provide a description of the tools and the test data required in order to perform the testing					
***T4*: Project Plan and Reporting**	**22.01.2016 31.01.2016**	**10**	**2**	**10**	**T1, T2, T3**
T4.1: Provide a general introduction on the project planning and cost estimation					
T4.2: Describe and perform the Function Points algorithm to estimate the size of the project					
T4.3: Apply the COCOMO model in order to evaluate the effort and duration					
T4.4: Identify the main tasks, their schedule and the allocation of the resources					
T4.5: Identify and describe the potential project risks					

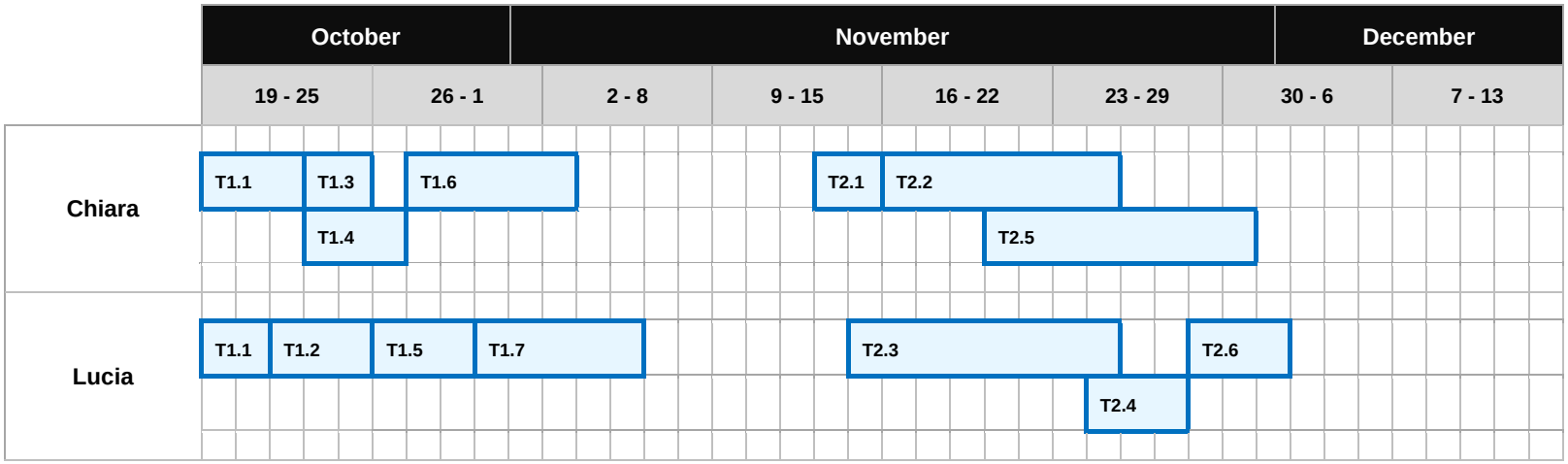
In the following page a detailed “activity bar chart” is reported. It indicates when each tasks has been developed.

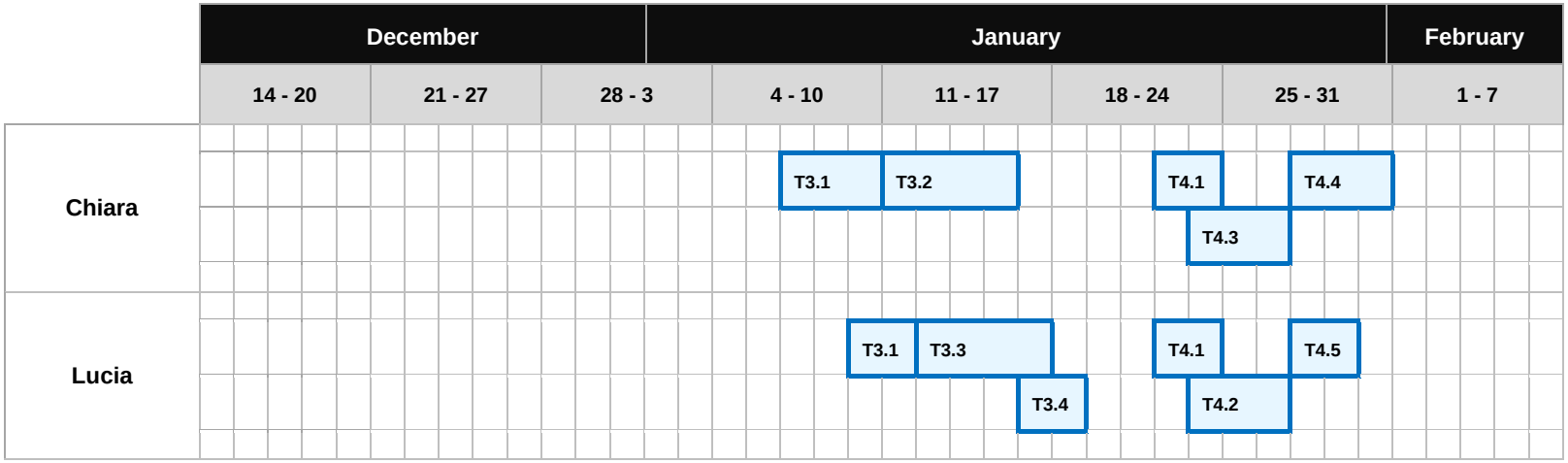

Task
October
November
December
January
February
19 - 25
26 - 1
2 - 8
9 - 15
16 - 22
23 - 29
30 - 6
7 - 13
14 - 20
21 - 27
28 - 3
4 -10
11 - 17
18 - 24
25 - 31
1 - 7
T1
T1.1
T1.2
T1.3
T1.4
T1.5
T1.6
T1.7
T2
T2.1
T2.2
T2.3
T2.4
T2.5
T2.6
T3
T3.1
T3.2
T3.3
T3.4
T4
T4.1
T4.2
T4.3
T4.4
T4.5

4.2 Resources allocation

In this paragraph we report a “staff allocation chart” that represents how the resources have been allocated to each tasks. In particular, it’s indicated who took care of which task and how long.
Obviously, before the delivery of each assignment, each member of the group has checked and reviewed all the tasks performed by the other member and a few hours have been devoted to a general review (performed together) of each document.
The time spent for the reviews has not been considered in the tables below.

See the tables in the following page: we used two tables in order to make everything bigger and easier to understand.


October
November
December
19 - 25
26 - 1
2 - 8
9 - 15
16 - 22
23 - 29
30 - 6
7 - 13
Chiara
T1.1
T1.3
T1.6
T2.1
T2.2
T1.4
T2.5
Lucia
T1.1
T1.2
T1.5
T1.7
T2.3
T2.6
T2.4


December
January
February
14 - 20
21 - 27
28 - 3
4 - 10
11 - 17
18 - 24
25 - 31
1 - 7
Chiara
T3.1
T3.2
T4.1
T4.4
T4.3
Lucia
T3.1
T3.3
T4.1
T4.5
T3.4
T4.2

5 Project Risks

In this chapter we are going to define the possible risks of our project, those potential problems that may happen or not happen, their relevance and the associated recovery actions.
There exists many different types of risks, threatening the project plan, the quality of the software to be produced or the feasibility of the project.
Once a possible risks is identified, it's important to analyze it in order to estimate the probability that it will occur and the effect that it will have if it really happens.

With respect to our project, we have identified the following potential risks:

	RISK	***PROBABILITY***	***EFFECT***
1)	The developers are ill at the same time and they can't work on the project, so they will not deliver the software on time.	*Moderate*	*Catastrophic*
2)	There are faults in the reusable software components, so every time the software is reused, the same initial errors will arise.	*High*	*Serious*
3)	Changes to the requirements that require major design rework are proposed.	*Moderate*	*Serious*
4)	The database used in the system cannot process as many TPS as expected.	*Moderate*	*Serious*
5)	Both the database and the backup database break without a dump and log file.	*Low*	*Catastrophic*
6)	Creation of a software far more complex than the one required with the obvious risk of missing deadlines.	*Moderate*	*Serious*
7)	Virus in the system.	*Moderate*	*Serious*

The next step, after identifying all the potential risks, is to develop strategies that can help manage all those risks.
We think that the following strategies could be useful to prevent the risks that we have found:

RISK	***STRATEGY***
1) Developers illness.	Begin to work on the project as soon as possible.
2) Faults in reusable software components.	Hard testing.
3) Changes to requirements.	Derive traceability information to assess requirements change impact and be able to update the system as soon as possible. Design a cheap and reliable architecture that helps future changes.
4) Not as many TPS as expected.	Evaluate the possibility of buying a higher-performance database.
5) Databases breakdown.	Create a dump and a log file and update them periodically.
6) Software more complex than required.	Read carefully the requirements one more time.
7) Virus in the system.	Avoid code injection.

6 Conclusions

As a conclusion of this document, we want to provide a comparison between the results we have obtained with the COCOMO approach and the effective hours of work and project duration.
As previously reported in the Project Planning chapter, the hours that we spent for our "MyTaxiService" project are:

RASD		***DD***		***ITPD***		***Project Plan***	
Chiara	*Lucia*	*Chiara*	*Lucia*	*Chiara*	*Lucia*	*Chiara*	*Lucia*
48	48	25	25	15	15	10	10

So, the total number of hours we spent for developing this project is: $196\ hours$.
This means that, supposing that each one of us can work 25 hours in a week on this project (this means 100 hours in a month), we obtain:

$$196 \text{ hours} / 100 \text{ hours} = 1.96 \text{ Persons-Months} \approx 2 \text{ Person-Months}$$

Comparing the results we have:

	Estimated	***Effective***
Effort	21.72 Person-Months	2 Person-Months
Duration	9.76 Months	3 Months *
Number of people	2	2

* the effective duration of the whole project is actually 4 months, but we are considering here, as we have considered in the whole document, only the time spent for the production of the deliverables concerning the "MyTaxiService" application.
Almost a month, after the Design Document deadline and before the beginning of the Integration Testing assignment, has been devoted to the Code Inspection activity that we have not taken into consideration to develop this document.

7 Used Tools

In this paragraph we are going to list all the tools we have used to create this document:

- *Microsoft Office Word 2010* : to redact and format this document

8 Working Hours

In order to analyze the code, redact and write this document we spent about 10 hours per person.