Politecnico di Milano

A.A. 2015-2016

Software Engineering 2 Project – “myTaxiService”

Prof. Raffaela Mirandola

Project **P**lanning **D**ocument

Version 1.0

Christian Zichichi (mat. 840565), Luigi Marrocco (mat. 854884)

February 2, 2016

Table of Contents

1. INTRODUCTION

1.1 Revision History

Version	Date	Authors	Summary
1.0	02-02-2016	Christian Zichichi Luigi Marrocco	First Release (Delivery)

1.2 Purpose and Scope

This document contains the project plan for the myTaxiService system. The project team is composed by two people, Christian Zichichi and Luigi Marrocco. Function Points and COCOMO II will be applied to estimate the project size, effort and cost. The project, then, will be divided in tasks and a schedule will be planned for them. The members of the team will be allocated to the tasks taking into account the actual availability for the project. Eventually, project risks will be defined, assessed by relevance and associated to proper recovery actions.

1.3 List of Definitions and Abbreviations

- **RASD:** Requirements Analysis and Specification Document
- **DD:** Design Document
- **ITPD:** Integration Test Plan Document
- **PPD:** Project Planning Document
- **API:** Application Programming Interface

1.4 List of Reference Documents

This document directly refers to the following documents:

- myTaxiService – **R**equirement **A**nalysis and **S**pecification **D**ocument; **D**esign **D**ocument; **I**ntegration **T**est **P**lan **D**ocument

2. FUNCTION POINTS ANALYSIS

The Function Points (FP) is a technique to assess the effort needed to design and develop custom software applications.

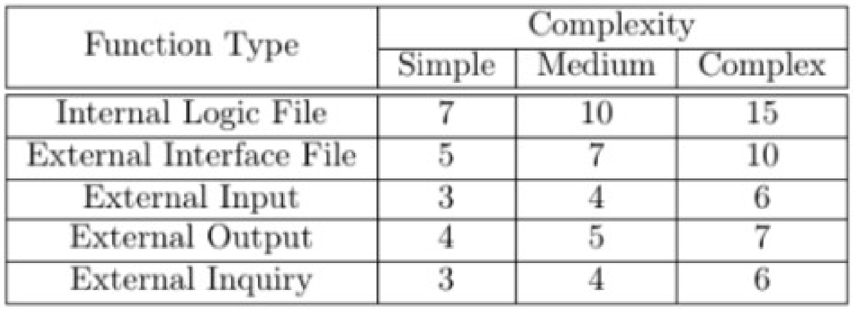
The estimation is based on a combination of program characteristics like:

- **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.

A weight is associated with each of these FP counts. The total is computed by multiplying each raw count by the weight and summing all partial values:

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

The following table shows the coefficients to be used in the UFP (Unadjusted Function Points) computations:



Function Type	Complexity		
	Simple	Medium	Complex
Internal Logic File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

INTERNAL LOGIC FILES (ILF)

The system stores the information about:

- **Passengers**
- **Taxi Drivers**
- **Queues**
- **Zones**
- **Locations**
- **Ride Requests / Reservation**

Each of these entities has a simple structure as it is composed of a small number of fields. Thus, we can decide to adopt for all the six the SIMPLE weight.

*ILF Function Points = 6*7 = 42*

EXTERNAL LOGIC FILES (ELF)

External GPS data: this entity could have a structure of average complexity. So, we can decide to adopt the MEDIUM weight.

*ELF Function Points = 1*7 = 7*

EXTERNAL INPUTS (EI)

The application interacts with the customer to allow him/her to:

- **User Login/Logout:** these are simple operations, so we can adopt the SIMPLE weight for them.
- **User Registration:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Ride Request creation:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Ride Reservation creation:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Ride Reservation cancellation:** this operation involves two entities, the ride and the passenger. It can still be considered simple, so, again we adopt the SIMPLE weight.

- **Taxi Driver ride response:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Taxi Driver availability switch:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Polling request:** this is a simple operation, but it is periodic and very frequent so we can adopt the MEDIUM weight for it.

*EI Function Points = 7*3+1*4 = 25*

EXTERNAL OUTPUT (EO)

- **Polling output:** this is a complex operation due to the use of more entities, especially the cronjob, so we can apply the weight for the COMPLEX cases.

*EO Function Points = 1*7 = 7*

EXTERNAL INQUIRY (EI)

The application allows customers to request information about:

- **Passenger List of Booked Rides view:** this is an operation of average complexity due to the retrieval of the history of all the rides and the possibility of cancelling the reserved rides up to 10 minutes before, so we can adopt the MEDIUM weight for it.
- **Passenger Notification view:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Taxi Driver Notification view:** this is a simple operation, so we can adopt the SIMPLE weight for it.
- **Taxi Driver in-ride view:** this is an operation of average complexity due to the Google Maps APIs invocations that involve this entity, so we can adopt the MEDIUM weight for it.

*EI Function Points = 2*3+2*4=14*

TOTAL FUNCTION POINTS (UFP) = 42+7+25+7+14 = 144

3. COCOMO II ANALYSIS

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 that are derived from historical project data and current as well as future project characteristics.

Elements of the COCOMO II model:

- **Source Lines of Code (SLOC):** The COCOMO II calculations are based on estimates of a project's size in Source Lines of Code (SLOC).
- **Scale Drivers:** In COCOMO II, some of the most important factors contributing to a project's duration and cost are the Scale Drivers (SD). SDs determine the *exponent* used in the Effort Equation. There are 5 Scale Drivers.
- **Cost Drivers:** COCOMO II has 17 Cost Drivers (CD) multiplicative factors that determine the effort required to complete the software project.
- **The Effort Equation:** COCOMO II model makes its estimates of required effort based primarily on the estimate of the software project's size:

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

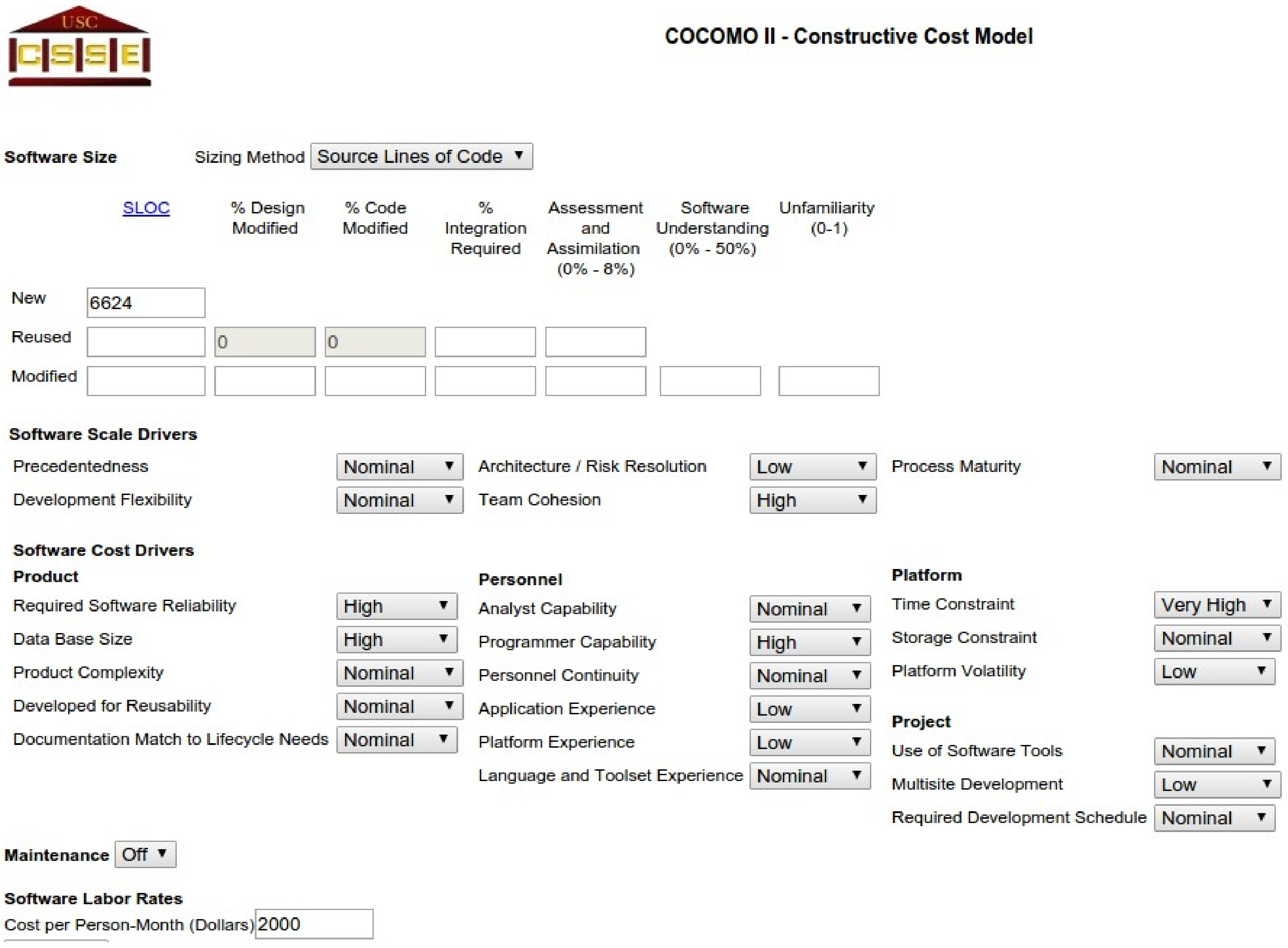
  Where *EAF* (Effort Adjustment Factor) is derived from Cost Drivers, while *E* (Exponent) from Scale Drivers.
- **The Schedule Equation:** COCOMO II schedule equation predicts the number of months required to complete a software project. The duration of a project is based on the effort predicted by the effort equation:

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

  Where *Effort* is the one derived before and *SE* (Schedule Equation exponent) is derived from the five Scale Drivers.

Assuming that the programming language used for our project will be Java EE, the conversion factor between the total Function Point counts (UFP) and the SLOC is 46. Hence, the SLOC number is estimated as:

$$SLOC = 144 * 46 = 6624$$



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	6624						
Reused		0	0				
Modified							

Software Scale Drivers

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

Software Cost Drivers

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

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars) 2000

Results

Software Development (Elaboration and Construction)

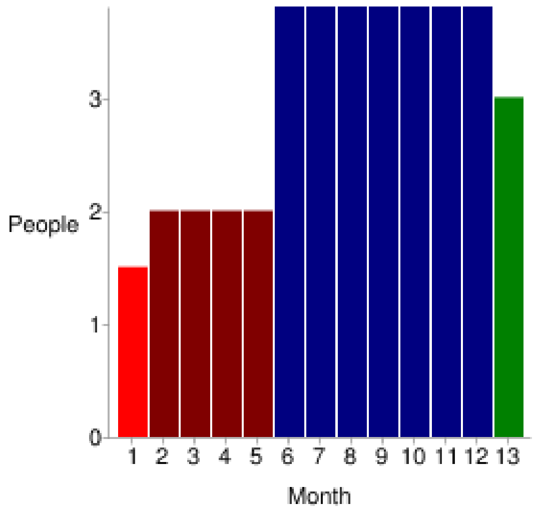
Effort = 38.3 Person-months
Schedule = 12.2 Months
Cost = $76567

Total Equivalent Size = 6624 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	2.3	1.5	1.5	$4594
Elaboration	9.2	4.6	2.0	$18376
Construction	29.1	7.6	3.8	$58191
Transition	4.6	1.5	3.0	$9188

Staffing Profile

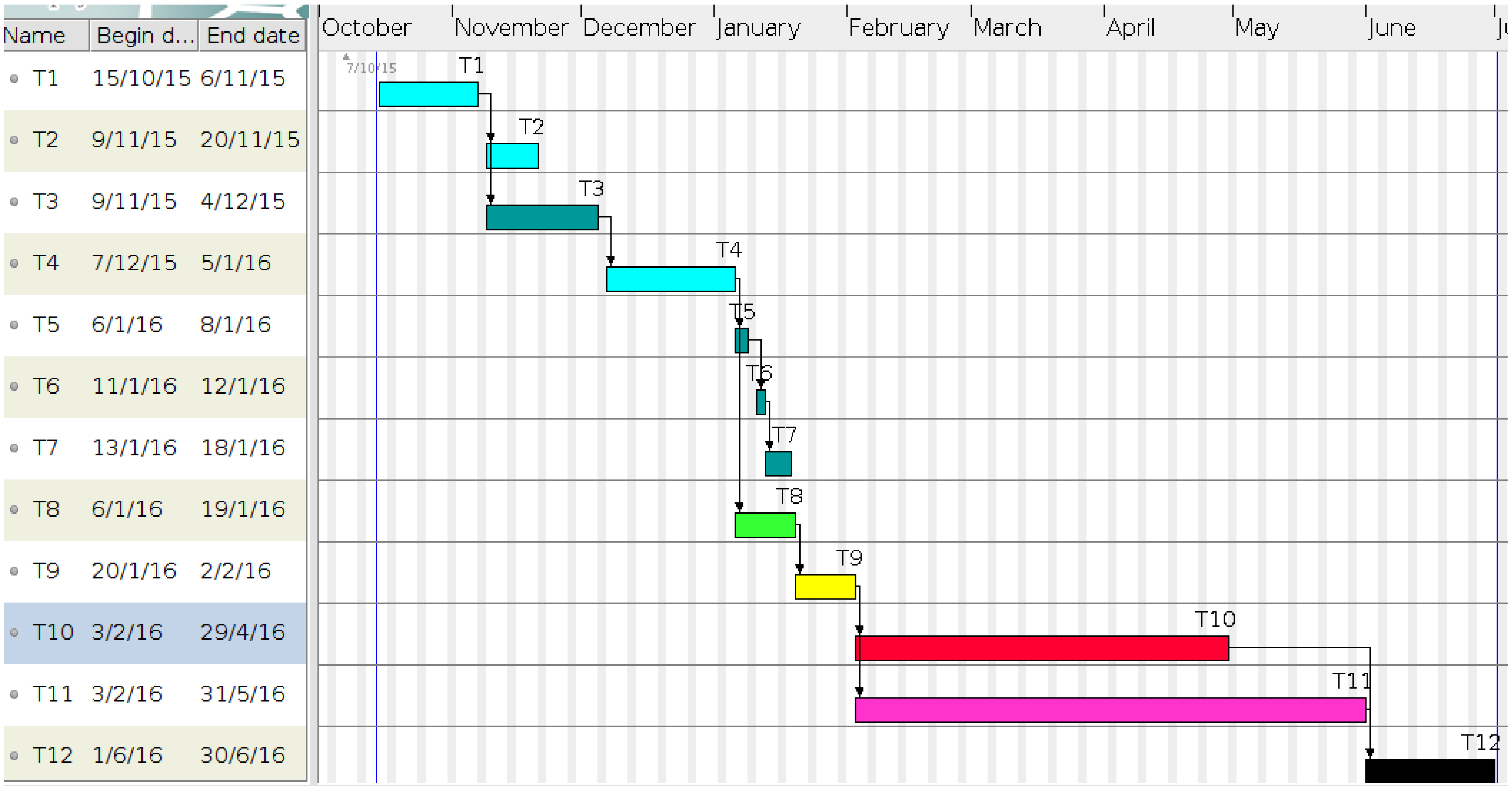


Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.3	1.1	2.9	0.6
Environment/CM	0.2	0.7	1.5	0.2
Requirements	0.9	1.7	2.3	0.2
Design	0.4	3.3	4.7	0.2
Implementation	0.2	1.2	9.9	0.9
Assessment	0.2	0.9	7.0	1.1
Deployment	0.1	0.3	0.9	1.4

4. TASK SCHEDULE

Here below, a Gantt diagram represents the schedule of the tasks in which the project is divided with the relative duration (weekends are not considered as workdays):



Name	Begin d...	End date
T1	15/10/15	6/11/15
T2	9/11/15	20/11/15
T3	9/11/15	4/12/15
T4	7/12/15	5/1/16
T5	6/1/16	8/1/16
T6	11/1/16	12/1/16
T7	13/1/16	18/1/16
T8	6/1/16	19/1/16
T9	20/1/16	2/2/16
T10	3/2/16	29/4/16
T11	3/2/16	31/5/16
T12	1/6/16	30/6/16

TASK LEGENDA:

- **T1:** RASD v1.0
- **T2:** RASD v2.0
- **T3:** DD v1.0
- **T4:** RASD v3.0
- **T5:** DD v2.0
- **T6:** DD v3.0
- **T7:** ITPD v1.0
- **T8:** DD v4.0
- **T9:** PPD v1.0
- **T10:** Front-end development
- **T11:** Back-end development
- **T12:** Acceptance Testing

5. RESOURCE ALLOCATION

5.1 RASD

Together [20 hours including revision]	• Introduction • Functional Requirements • Specifications • Use Case description
Christian Zichichi [17 hours including revision]	• Actors • Non Functional Requirements • Use Case diagram • State Chart diagrams • Class diagram
Luigi Marrocco [18 hours including revision]	• Scenarios • Sequence diagrams • Alloy

5.2 DD

Christian Zichichi [12 hours including revision]	• Introduction • Overview • Deployment view • Architectural styles and patterns • Other design decision • User Interface Design
Luigi Marrocco [11 hours including revision]	• Component view • Component interfaces • Runtime view • Algorithm Design
Together [7 hours including revision]	• High level components and their interactions • Requirements Traceability

5.3 ITPD

Christian Zichichi [0.5 hours]	• Introduction • Tools and Test Equipment Required
Luigi Marrocco [0.5 hours]	• Program Stubs and Test Data Required
Together [8 hours]	• Integration Strategy • Individual Steps and Test Description

5.4 PPD

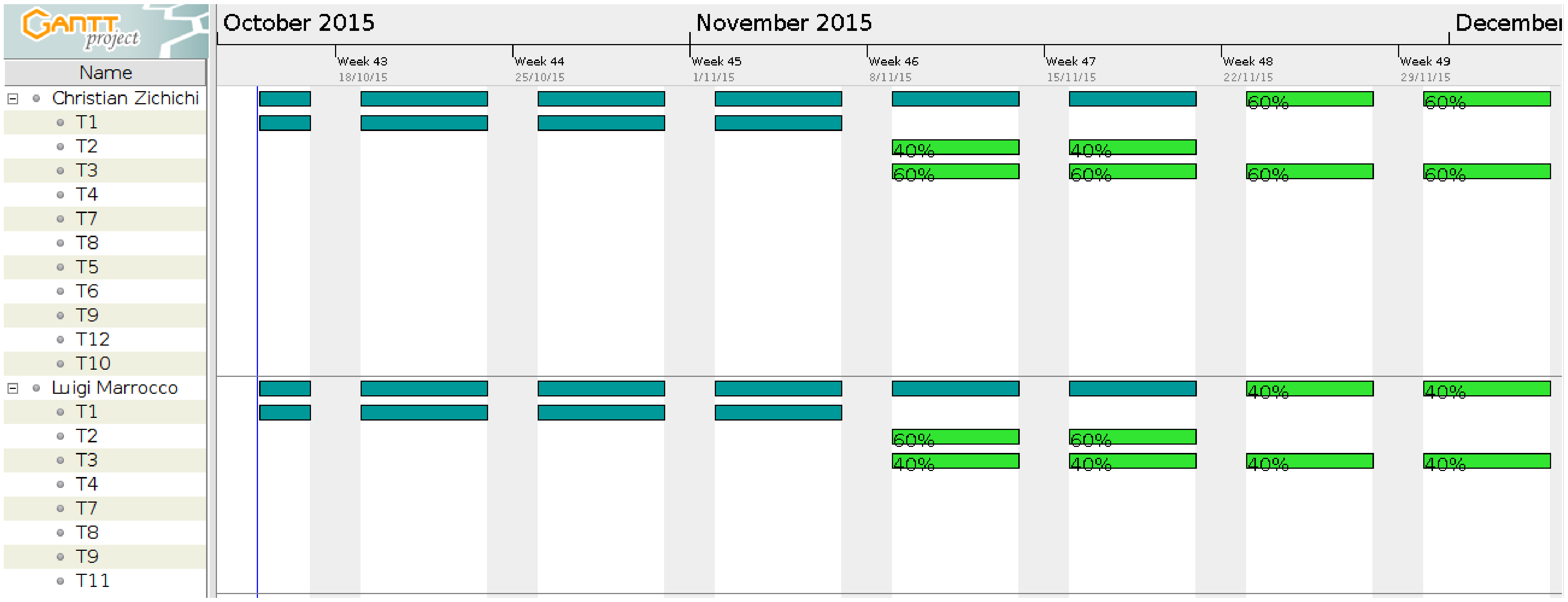
Christian Zichichi [4 hours]	• Introduction • Task Schedule • Resource Allocation • Risk Management
Luigi Marrocco [3 hours]	• Function Points analysis • COCOMO II analysis

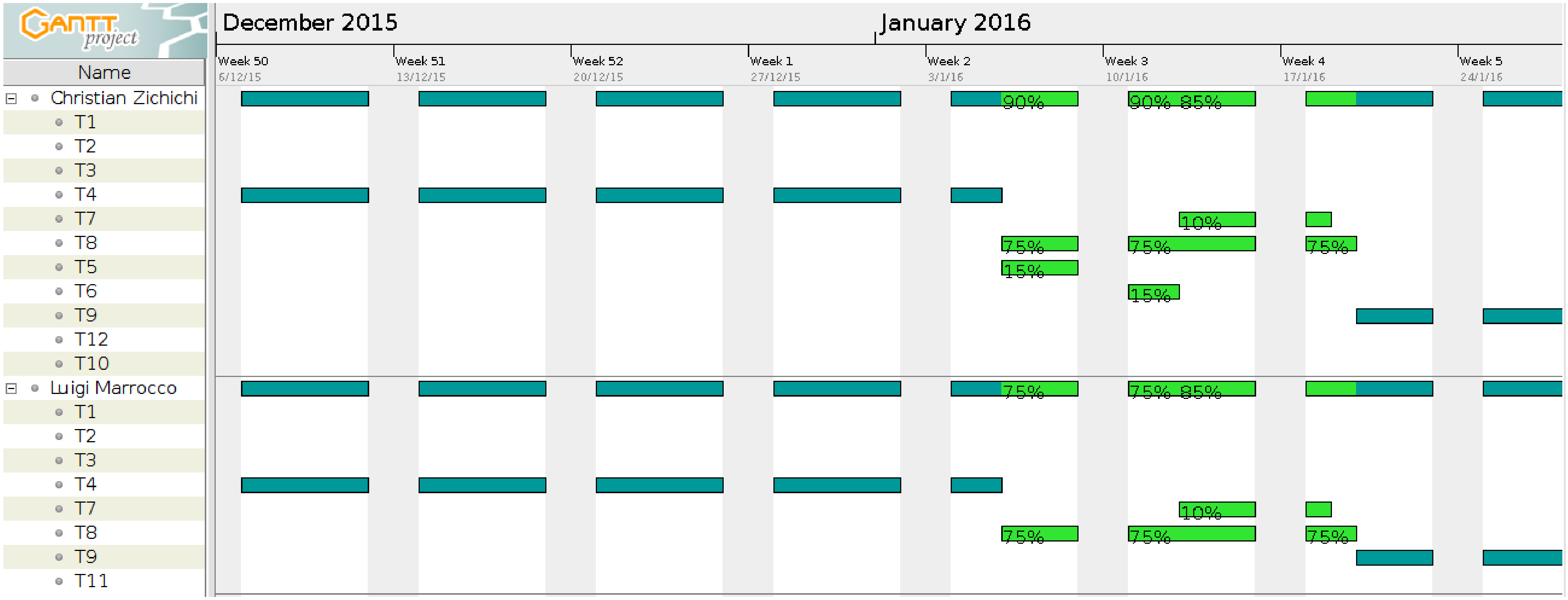
5.5 DEVELOPMENT AND ACCEPTANCE TESTING

Christian Zichichi [5 months – ESTIMATION]	• Front-end (Web and Mobile) development • Acceptance Testing
Luigi Marrocco [5 months – ESTIMATION]	• Back-end development

RESOURCE ALLOCATION DIAGRAM

For each task, the work time allocated to each team member is indicated in percentage with respect to the total amount of work time available (a blue row indicates 100%). For example, at week 46 the 100% of the work time of Christian Zichichi is allocated to myTaxiService project. In particular, 40% of his time is dedicated to task T2, while the remaining 60% to task T3. At week 48, instead, only 60% of the total work time of Christian Zichichi is allocated to myTaxiService project (task T3 in particular), while the remaining 40% is dedicated to other activities (e.g., Code Inspection Document). Tasks T10, T11, T12 (Development and Acceptance Testing phases) are not represented here due to space constraints, but it is assumed that they would occupy the 100% of the work time of each team member.




Name
Christian Zichichi
T1
T2
T3
T4
T7
T8
T5
T6
T9
T12
T10
Luigi Marrocco
T1
T2
T3
T4
T7
T8
T9
T11
December 2015
January 2016
Week 50 6/12/15
Week 51 13/12/15
Week 52 20/12/15
Week 1 27/12/15
Week 2 3/1/16
Week 3 10/1/16
Week 4 17/1/16
Week 5 24/1/16
90%
90% 85%
10%
75%
15%
75% 85%

6. RISK MANAGEMENT

Here below, a list of possible risks that can occur during the project:

1. *Geolocation system (Google Maps) stops working*

   Probability: Very Low

   Effect: Catastrophic

   Recovery Strategy: If the problem is temporary, use another external service to let the system work again.

2. *Temporary unavailability of personnel involved in critical tasks*

   Probability: Moderate

   Effect: Serious

   Recovery Strategy: Reorganize the resource allocation so that there is more overlap of work and people therefore understand each other's jobs. Another aspect to consider is to select those people that are not already executing other critical activities.

3. *Misunderstanding followed by structural changes to the project's requirements*

   Probability: Moderate

   Effect: Critical

   Recovery Strategy: Redefine the requirements and try to adapt the older ones with the new ones as much as possible.

4. *The database used in the system cannot process as many transactions per second as expected*

   Probability: Moderate

   Effect: Serious

   Recovery Strategy: Investigate the possibility of buying a higher-performance database.

5. *Underestimated development time*

   Probability: Moderate

   Effect: Serious

   Recovery Strategy: Investigate buying-in components; investigate the use of a program generator.

7. USED TOOLS

- **LibreOffice Writer 5.0:** to redact and format the document
- **GanttProject:** to draw Task Gantt Diagram and Resource Allocation diagram

8. HOURS OF WORK

Since we are neighbors, we have worked **together almost all the time** at each other's home and **equally shared all the tasks and efforts**. We have worked on this document **for a total of 8 hours**.

9. REFERENCES

Here is a short list of other references for this document:

- Slides of the Software Engineering 2 course (from the Beep Platform)