Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Project Plan myTaxiService

Student: Federico Gatti [Matricola: 852377]
Student: Luca Fochetta [Matricola: 792935]

AA 2015–2016

Contents

1 Introduction **1**

1.1 Purpose . . . 1
1.2 Scope . . . 1
1.3 Definitions, Acronyms, Abbreviations . . . 2
1.4 List of Reference Documents . . . 2
1.5 Document Structure . . . 2

2 Function Point **3**

2.1 Function Point . . . 3

3 COCOMO II **7**

3.1 Number of line Estimation . . . 7
3.2 Effort Estimation . . . 7
3.3 Duration Estimation . . . 8
3.4 Cost Driver . . . 8
3.4.1 Effort and Duration using nominal cost . . . 9
3.4.2 Scale Factors . . . 9
3.4.2.1 Precedentedness (PREC) . . . 10
3.4.2.2 Development Flexibility (FLEX) . . . 10
3.4.2.3 Architecture / Risk Resolution (RESL) . . . 10
3.4.2.4 Team Cohesion (TEAM) . . . 10
3.4.2.5 Process Maturity (PMAT) . . . 10
3.4.3 Effort Multipliers . . . 12
3.4.3.1 Product Factors . . . 12
3.4.3.2 Platform Factors . . . 12
3.4.3.3 Personnel Factors . . . 13
3.4.3.4 Project Factors . . . 14
3.4.4 Effort and Duration using nominal cost . . . 16
3.5 Result Consideration . . . 16

4 Resource Allocation **17**

4.1 Task Scheduling . . . 17

4.1.1 Time Scheduling 18
4.1.2 Gantt 19
4.2 Resource Allocation 20

5 Risks **23**
5.1 Project Risks 24
5.2 Implementation Risks 25
5.3 Technical Risks 26
5.4 Business Risks 27

Chapter 1

Introduction

1.1 Purpose

This document describes the **project plan** for the *myTaxiService* applications. The objective of a project plan is to define the approach to be used by the Project team to deliver the intended project management scope of the project.

The project manager creates the project management plan following input from the project team and key stakeholders. The plan should be agreed and approved by at least the project team and its key stakeholders.

1.2 Scope

The aim of the project is to create a software, called *myTaxiService*, which can manage the queue of the taxi requests in a city. The system is composed by 4 main parts:

- 2 front-end applications used by the passengers called PMA and PWA
- 1 front-end application used by the Taxi Drivers called TMA
- 1 back-end system called QTM

The system will be able to suggest the best distribution of the taxi in the city zone to maximize passengers' satisfaction and the quality of the service. A passenger who sends a request, using a web application or mobile application, can see from the application the waiting time and the code of the taxi that accepted the request. The passenger position can be determinate from GPS or if GPS information are incorrect, or aren't available, the passenger can insert manually the information of the location. An authenticated passenger can reserve a taxi by specifying the origin and the destination of the ride. In this case the passenger must place the reservation at least two hours before the ride. The reservation request will be sent like a normal request ten minutes before the specified time. An authenticated user can always delete his reservation or modify it. A Taxi driver can accept or reject a request.

1.3 Definitions, Acronyms, Abbreviations

- FP: Function Point
- COCOMO: Constructive Cost Model

1.4 List of Reference Documents

1. MeteoCall Project Plan
2. Project Plan Example
3. Wikipedia

1.5 Document Structure

The document is composed by five sections.

1. The first section, this one, defines the goal of the document and a general idea of *myTaxiService* functionalities
2. The second section describes the precondition for the integration, the test integration strategy used and justify the choices, also provides the steps to perform all the integration tests.
3. The third section defines the test sets and test cases starting from the steps described in section two
4. The fourth section lists the software tools used to perform the test integration, also described the case when automated test is impossible
5. The fifth section described the program drivers, stubs and special test data required for each integration step.

Chapter 2

Function Point

2.1 Function Point

The Functional Point approach is a technique that allows to evaluate the effort needed for the design and implementation of a project. We have used this technique to evaluate the application dimension basing on the functionalities of the application itself. The functionalities list has been obtained from the RASD document and for each one of them the realization complexity has been evaluated. The functionalities has been groped in:

- **Internal Logic File** (ILFs): it represents a set of homogeneous data handled by the system.
- **External Interface File** (EIFs): it represents a set of homogeneous data used by the application but handled by external application
- **External Imput** (EIs): elementary operation that allows input of data in the system.
- **External Output** (EOs): elementary operation that creates a bitstream towards the outside of the application.
- **External Inquiry** (EIQs)**:** elementary operation that involves input and output operations.

The following table outline the number of Functional Point based on functionality and relative weight for each level of complexity:

Function Type	Simple	Medium	Complex
ILFs	7	10	15
EIFs	5	7	10
EIs	3	4	6
EOs	4	5	7
EIQs	3	4	6

Table 2.1: Weight Table

- **ILFs**: The applications include several ILFs, that corresponds to the table of the Database subsystem. The ILFs are the *passenger, taxi, taxi driver, zone, queue, request, reservation* and *status.* All of these entities have a simple structure as they are composed by a small number of fields, so we decide to give a simple complexity to all for this point: $7 \times 8 = 56$

- **EIFs**: The only data managed by external system but used in the application is the position data of the users and taxis using obtained using API. We decide to adopt a simple weight because this information results only one entity with a very simple structure: $2 \times 5 = 10$

- **EIs**: The application interacts with the users and allow them to perform:

1. *Sign In/Logout:* these are very simple operations, we can adopt the simple weight: $2 \times 3 = 6$

2. *Sign Up:* this is a very simple operation, we can adopt the simple weight: $1 \times 3 = 3$

3. *Place Request:* this is a complex operations because involves *user, taxi, request* and *zone*:$1 \times 6 = 6$

4. *Place Reservation*: this is a bit more complex than request that is already a complex operations so : $1 \times 6 = 6$

5. *Modify Reservation:* this is a complex operations because involves *user, taxi, reservation* and *zone*: $1 \times 6 = 6$

6. *Obtain Position:* this is a simple operations that requires call a method already implemented in an API: $1 \times 3 = 3$

7. *Accept/Reject a request:* these are a medium complex operations because involve *taxi and request*: $2 \times 4 = 8$

8. *Modify Status:* this is a medium operations because involves *taxi* and *status:* $1 \times 4 = 4$

9. *Waiting time and taxi ID after a passenger request:* this is a complex operations because involves *user, taxi* and *location:* $1 \times 6 = 6$

- **EOs**

1. *New taxi distribution notification*: this is a very complex operations, maybe the more complex in the entire system, ans involves *taxi, queue* and *status.* $1 \times 7 = 7$

2. *New taxi status notification:* this is medium complex operation because involves *taxi, queue* and *zone:* $1 \times 5 = 5$

- **EIQs**: The application allows customers to request information about:

1. *Their profiles:* this is a simple operation because involves only *user* or *taxi:* $1 \times 3 = 3$
2. *Taxi Status:* this is a simple operation because involves only *taxi* and *status:* $1 \times 3 = 3$
3. *Information about the reservations:* this is a simple operation because involves only *taxi* and *reservation:* $1 \times 3 = 3$

This table summarizes the FPs previously calculated

Function Type	**FPs**
ILFs	58
EIFs	10
EIs	48
EOs	12
EIQs	9
Total	**137**

Total FP number and summary: In summary, we have computed the following value for the unadjusted FPs: 137.

Chapter 3

COCOMO II

3.1 Number of line Estimation

To pass from FP to SLOC we use an average conversion factor of 46 as described at http://www.qsm.com/resources/function-point-languages-table, an updated version that adds J2EE of the table included in official manual (http://sunset.usc.edu/research/COCOMOII/Docs/modelr

$137 FPs \times 46 = 6302$ SLOC

3.2 Effort Estimation

In COCOMO II effort is expressed as Person-Months (PM). A person month is the amount of time one person spends working on the software development project for one month. The effort is defined:

$PM{=}effort = A * EAF * KSLOC^{E}$

Where:

- $A \rightarrow$ 2.94 in COCOMO II
- $EAF \rightarrow$Effort Adjustment Factor derived from Cost Drivers (product of the effort multipliers corresponding to each of the cost drivers for your project) equals to $\prod_{i=1}^{n} EM_i$
- $E \rightarrow$Exponent derived from Scale Drivers. It is calculated as: $B{+}0.01{*} \sum_{j=1}^{5} SF_j$where $B = 0.91$in COCOMO II
- $KSLOC \rightarrow$Estimation of the size of the software modules that will constitute the application program. In our case is equal to *6.302* lines

3.3 Duration Estimation

The initial version of COCOMO II provides a simple schedule estimation capability similar to those in COCOMO 81 and Ada COCOMO. The initial baseline schedule equation for the COCOMO II Early Design and Post-Architecture stages is:

$$Duration = [C \times (PM)^{(D+0.2\times(E-B))}]$$

where $C = 3.67$,$D = 0.28$, $B = 0.91$

Time to Develop, TDEV, or Duration is the calendar time in months between the estimation end points of LCO and IOC for MBASE/RUP or SRR and SAR for Waterfall lifecycle model. For the waterfall model, this goes from the determination of a product's requirements baseline to the completion of an acceptance activity certifying that the product satisfies its requirements.

After that we can estimate the number of people needed to complete the project with the following formula:

$$N_{people} = Effort/Duration$$

3.4 Cost Driver

Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project. A cost driver is a model factor that "drives" the cost (in this case Person-Months) estimated by the model. All COCOMO II cost drivers have qualitative rating levels that express the impact of the driver on development effort. These ratings can range from Extra Low to Extra High. Each rating level of every multiplicative cost driver has a value, called an effort multiplier (EM), associated with it. This scheme translates a cost driver's qualitative rating into a quantitative one for use in the model. The EM value assigned to a multiplicative cost driver's nominal rating is 1.00. If a multiplicative cost driver's rating level causes more software development effort, then its corresponding EM is above 1.0. Conversely, if the rating level reduces the effort then the corresponding EM is less than 1.0.

The rating of cost drivers is based on a strong rationale that they would independently explain a significant source of project effort or productivity variation. The difference between the Early Design and Post-Architecture models are the number of multiplicative cost drivers and the areas of influence they explain. There are seven multiplicative cost drivers for the Early Design model and seventeen multiplicative cost drivers for the Post-Architecture model. Each set is explained with its model later in the manual.

It turns out that the most significant input to the COCOMO II model is Size. Size is treated as a special cost driver in that it has an exponential factor, E. This exponent is an aggregation of five scale factors. These are discussed next.

3.4.1 Effort and Duration using nominal cost

Considering all Nominal Factor for the scale and cost drivers, that are $E = 1.0997$ and $EAF = 1.0$, we obtain a Nominal Effort equal to:

$$NominalEffort = 2.94 * 1.0 * 6.302^{1.0997} = 22.26 \frac{Person}{Months}$$

Considering Nominal cost driver $(D + 0.2 \times (E - B)) = 0.31794$

$$Duration = [3.67 \times (22.26)^{0.31794}] = 9.84 Months$$

so

$N_{people} = \frac{22.26 \frac{Person}{Months}}{9.84 Months} = 2.26$ Number of person needed in the team

3.4.2 Scale Factors

The exponent **E** is an aggregation of five scale factors (SF) that account for the relative economies or diseconomies of scale encountered for software projects of different sizes.

The Table scale Factor values is the following:

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprece-dented 6.20	largely un-precedented 4.96	somewhat unprece-dented 3.72	generally familiar 2.48	largely familiar 1.24	thoroughly familiar 0.00
FLEX	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL	little (20%) 7.07	some (40%) 5.65	often (60%) 4.24	generally (75%) 2.83	mostly (90%) 1.41	full (100%) 0.00
TEAM	very difficult interactions 5.48	some difficult interactions 4.38	basically cooperative interactions 3.29	largely cooperative 2.19	highly cooperative 1.10	seamless interactions 0.00
PMAT	Level 1 Lower 7.80	Level 1 Upper 6.24	Level 2 4.68	Level 3 3.12	Level 4 1.56	Level 5 0.00

Table 3.1: Scale Factor Values, SFj, for COCOMO II Models

For all the factor that we can't give

Now we consider a more detailed value for the previous driver

3.4.2.1 Precedentedness (PREC)

If a product is similar to several previously developed projects, then the precedentedness is high.

3.4.2.2 Development Flexibility (FLEX)

It reflects the degree of flexibility in the development process.

3.4.2.3 Architecture / Risk Resolution (RESL)

Reflects the extent of risk analysis carried out.

3.4.2.4 Team Cohesion (TEAM)

The Team Cohesion scale factor accounts for the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others. These difficulties may arise from differences in stakeholder objectives and cultures; difficulties in reconciling objectives; and stakeholders' lack of experience and familiarity in operating as a team.

3.4.2.5 Process Maturity (PMAT)

This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model.

Scale Factors	Factor	Value
Precentedness • Organizational understanding of product objectives → Thorough • Experience in working with related software systems → Considerable • Concurrent development of associated new hardware and operational procedures → Moderate • Need for innovative data processing architectures, algorithms → Some	High	2.48
Development Flexibility • Need for software conformance with pre-established requirements → Full • Need for software conformance with external interface specifications → Basic • Combination of inflexibilities above with premium on early completion → Medium	Nominal	3.04
Architecture / Risk Resolution • Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR or LCA → Some • Schedule, budget, and internal milestones through PDR or LCA compatible with Risk Management Plan → Some • Percent of development schedule devoted to establishing architecture, given general product objectives → 17 • Percent of required top software architects available to project → 60 • Tool support available for resolving risk items, developing and verifying architectural specs → Some • Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance → Significant • Number and criticality of risk items → 5	Nominal	4.24
Team Cohesion • Consistency of stakeholder objectives and cultures → Some • Ability, willingness of stakeholders to accommodate other stakeholders' objectives → Some • Experience of stakeholders in operating as a team →	Low	4.38 11

With this value $B + 0.01 * \sum_{j=1}^{5} SF_j = 0.91 + 0.01 * 18.82 = 1.0982$

3.4.3 Effort Multipliers

3.4.3.1 Product Factors

Product factors account for variation in the effort required to develop software caused by characteristics of the product under development. A product that is complex, has high reliability requirements, or works with a large testing database will require more effort to complete. There are five product factors, and complexity has the strongest influence on estimated effort.

RELY This is the measure of the extent to which the software must perform its intended function over a period of time.

DATA This cost driver attempts to capture the effect large test data requirements have on product development.

CPLX Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.

RUSE This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.

DOCU Several software cost models have a cost driver for the level of required documentation. In COCOMO II, the rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its lifecycle needs.

3.4.3.2 Platform Factors

The platform refers to the target-machine complex of hardware and infrastructure software (previously called the virtual machine). The factors have been revised to reflect this as described in this section. Some additional platform factors were considered, such as distribution, parallelism, embeddedness, and real-time operations.

TIME This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource.

STOR This rating represents the degree of main storage constraint imposed on a software system or subsystem. Given the remarkable increase in available processor execution time and main storage, one can question whether these constraint variables are still relevant.

PVOL "Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. If the software to be developed is an operating system then the platform is the computer hardware.

3.4.3.3 Personnel Factors

After product size, people factors have the strongest influence in determining the amount of effort required to develop a software product. The Personnel Factors are for rating the development team's capability and experience – not the individual. These ratings are most likely to change during the course of a project reflecting the gaining of experience or the rotation of people onto and off the project.

ACAP Analysts are personnel who work on requirements, high-level design and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. The rating should not consider the level of experience of the analyst; that is rated with APEX, LTEX, and PLEX.

PCAP Current trends continue to emphasize the importance of highly capable analysts. Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. The experience of the programmer should not be considered here; it is rated with APEX, LTEX, and PLEX.

PCON The rating scale for PCON is in terms of the project's annual personnel turnover.

APEX The rating for this cost driver (formerly labeled AEXP) is dependent on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

PLEX The Post-Architecture model broadens the productivity influence of platform experience, PLEX (formerly labeled PEXP), by recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.

LTEX This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. Software

development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, planning and control, etc. In addition to experience in the project's programming language, experience on the project's supporting tool set also affects development effort.

3.4.3.4 Project Factors

Project factors account for influences on the estimated effort such as use of modern software tools, location of the development team, and compression of the project schedule.

TOOL Software tools have improved significantly since the 1970s' projects used to calibrate the 1981 version of COCOMO. The tool rating ranges from simple edit and code, very low, to integrated life-cycle management tools, very high.

SITE Determining its cost driver rating involves the assessment and judgement-based averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia).

SCED This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in the earlier phases to eliminate risks and refine the architecture, more effort in the later phases to accomplish more testing and documentation in parallel.

Considering the tables included in the COCOMO II manual we have found this values:

Product Factors	Factor	Value
Required Software Reliability (RELY)	High	1.10
Data Base Size (DATA)	Nominal	1.00
Product Complexity (CPLX)	Nominal	1.00
Developed for Reusability (RUSE)	Very High	1.15
Documentation Match to Life-Cycle Needs (DOCU)	High	1.11
Subproduct		1.392

Platforms Factors	Factor	Value
Execution Time Constraint (TIME)	Nominal	1.00
Main Storage Constraint (STOR)	Nominal	1.00
Platform Volatility (PVOL)	Low	0.87
Subproduct		0.87

Personnel Factors	Factor	Value
Analyst Capability (ACAP)	High	0.85
Programmer Capability (PCAP)	Nominal	1.00
Personnel Continuity (PCON)	Nominal	1.00
Applications Experience (APEX)	Low	1.10
Platform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Nominal	1.00
Subproduct		0.935

Project Factors	Factor	Value
Use of Software Tools (TOOL)	High	0.90
Multisite Development (SITE)	High	0.93
Required Development Schedule (SCED)	High	1.00
Subproduct		0.837

Figure 3.2: Cost Drvier Table

Remembering that $EAF = \prod_{i=1}^{n} EM_i$

Product Factors	1.392
Platforms Factors	0.87
Personnel Factors	0.935
Project Factors	0.837
Total	**0.948**

Figure 3.3: Total Sum

3.4.4 Effort and Duration using nominal cost

Using the value calculated previously we can obtain the new Effort and Duration value:

- $E = B + 0.01 * \sum_{j=1}^{5} SF_j = 0.91 + 0.01 * 18.82 = 1.0982$
- $EAM = \prod_{i=1}^{n} EM_i = 0.948$
- $KSLOC = 6.302$

$Effort = A * EAF * KSLOC^E = 2.94 * 0.948 * 6.302^{1.0982} = 21.04 \frac{Person}{Months}$

- $(D + 0.2 \times (E - B)) = 0.31764$ where $C = 3.67$,$D = 0.28$, $B = 0.91$

$Duration = [C \times (PM)^{(D+0.2\times(E-B))}] = [3.67 \times (22.26)^{0.31764}] = 9.83 Months$

so

$N_{people} = \frac{21.04 \frac{Person}{Months}}{9.83 Months} = 2.14\ Person$

3.5 Result Consideration

We can see that the new values are very similar with the values calculated using the nominal cost estimation; so we can say that for medium project, considering the fact that we don't have same cost for some driver, the nominal cost is a good approximation for COCOMO II.

Chapter 4

Resource Allocation

Our project will last for eleven months, and here we will describe how we are going to allocate our time for the different phases of the project.

4.1 Task Scheduling

To have a better time scheduling, we decided to produce a Gantt where we insert all the activity we are going to perform during the project.

4.1.1 Time Scheduling

Phases of the project	Start Date	End Date
RASD Requirement Identifying	15/10/15	23/10/15
RASD Typical Scenario Identifying	26/10/15	29/10/15
RASD Alloy Model	30/10/15	04/11/15
RASD Final Checkout	05/11/16	06/11/15
RASD Delivery	06/11/15	06/11/15
DD Architectural Design	09/11/15	20/11/15
DD User Interface Design	16/11/15	20/11/15
DD Algorithm Design	16/11/15	20/11/15
DD Requirement Traceability	23//11/15	30/11/15
DD Final Checkout	01/12/15	04/12/15
DD Delivery	04/12/15	04/12/15
Unit Test Plan Document	07/12/15	05/01/16
ITPD Integration Strategy Definition	06/01/16	15/01/16
ITPD Stubs,Driver and Data Test Identification	11/01/16	20/01/16
ITPD Individual Step: Definition & Description	18/01/16	25/01/16
ITPD Tools Choices	26/01/16	29/01/16
ITPD Final Checkout	01/02/16	02/02/16
ITPD Delivery	02/02/16	02/02/16
Application Logic Implementation	03/02/16	13/04/16
Dispatcher Implementation	13/04/16	05/05/16
PA Implementation	06/05/16	26/05/16
TMA Implementation	06/05/16	30/05/16
TMA Unit Test	31/05/16	23/06/16
Application Logic Unit Test	31/05/16	01/07/16
PMA Unit Test	23/06/16	11/07/16
Dispatcher Unit Test	04/07/16	22/07/16
TMA Integration Test	25/07/16	15/08/16
PMA Integration Test	25/07/16	15/08/16
Application Logic Integration Test	16/08/16	16/09/16
Dispatcher TMA Integration Test	16/08/16	31/08/16
Dispatcher PMA Integration Test	31/08/16	16/09/16
Applcation Logic Database Integration Test	16/09/16	30/09/16
Application Logic Dispatcher Integration Test	16/09/16	30/09/16
All Sub-System Integration Test	03/10/16	24/10/16

4.1.2 Gantt

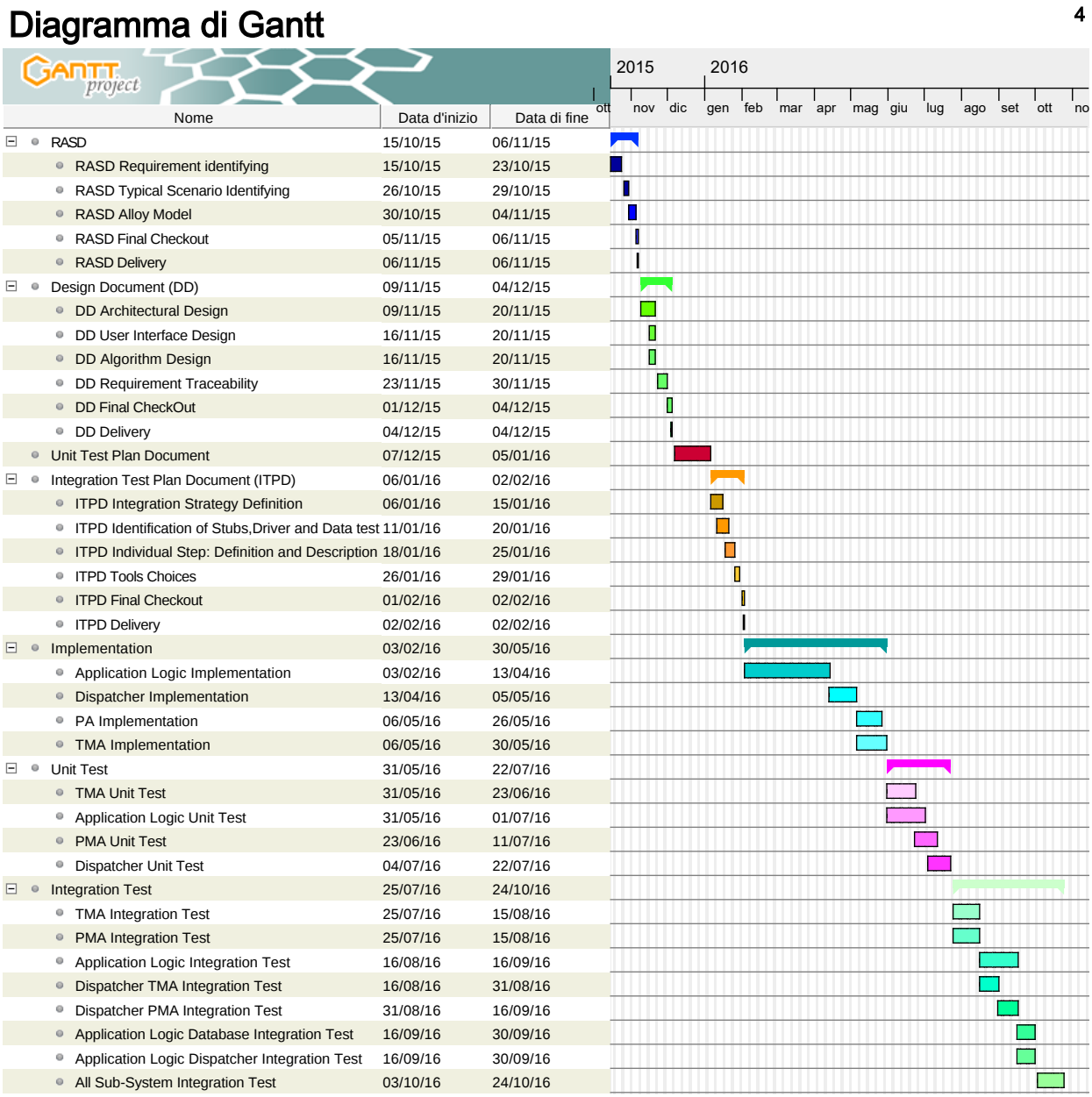
Diagramma di Gantt
4
GANTT project
2015
2016
ott nov dic gen feb mar apr mag giu lug ago set ott no
Nome
Data d'inizio
Data di fine
RASD 15/10/15 06/11/15
RASD Requirement identifying 15/10/15 23/10/15
RASD Typical Scenario Identifying 26/10/15 29/10/15
RASD Alloy Model 30/10/15 04/11/15
RASD Final Checkout 05/11/15 06/11/15
RASD Delivery 06/11/15 06/11/15
Design Document (DD) 09/11/15 04/12/15
DD Architectural Design 09/11/15 20/11/15
DD User Interface Design 16/11/15 20/11/15
DD Algorithm Design 16/11/15 20/11/15
DD Requirement Traceability 23/11/15 30/11/15
DD Final CheckOut 01/12/15 04/12/15
DD Delivery 04/12/15 04/12/15
Unit Test Plan Document 07/12/15 05/01/16
Integration Test Plan Document (ITPD) 06/01/16 02/02/16
ITPD Integration Strategy Definition 06/01/16 15/01/16
ITPD Identification of Stubs,Driver and Data test 11/01/16 20/01/16
ITPD Individual Step: Definition and Description 18/01/16 25/01/16
ITPD Tools Choices 26/01/16 29/01/16
ITPD Final Checkout 01/02/16 02/02/16
ITPD Delivery 02/02/16 02/02/16
Implementation 03/02/16 30/05/16
Application Logic Implementation 03/02/16 13/04/16
Dispatcher Implementation 13/04/16 05/05/16
PA Implementation 06/05/16 26/05/16
TMA Implementation 06/05/16 30/05/16
Unit Test 31/05/16 22/07/16
TMA Unit Test 31/05/16 23/06/16
Application Logic Unit Test 31/05/16 01/07/16
PMA Unit Test 23/06/16 11/07/16
Dispatcher Unit Test 04/07/16 22/07/16
Integration Test 25/07/16 24/10/16
TMA Integration Test 25/07/16 15/08/16
PMA Integration Test 25/07/16 15/08/16
Application Logic Integration Test 16/08/16 16/09/16
Dispatcher TMA Integration Test 16/08/16 31/08/16
Dispatcher PMA Integration Test 31/08/16 16/09/16
Application Logic Database Integration Test 16/09/16 30/09/16
Application Logic Dispatcher Integration Test 16/09/16 30/09/16
All Sub-System Integration Test 03/10/16 24/10/16

4.2 Resource Allocation

During the project, we need to allocate all the resource that we have in order to have a better time scheduling for our resource and to be sure that each of our tasks will be completed at the appointed time.

Our resources will be:

- Federico Gatti (F.G.): Project Manager

- Luca Fochetta (L.F.): Deputy Project Manager

Into this table we will insert all the different phases of the project and the percentages that each resources have to perform for that phase.

Phases of the project	F.G. (%)	L.F (%)
RASD Requirement Identifying	60	40
RASD Typical Scenario Identifying	10	90
RASD Alloy Model	90	10
RASD Final Checkout	50	50
RASD Delivery	100	0
DD Architectural Design	100	0
DD User Interface Design	10	90
DD Algorithm Design	100	0
DD Requirement Traceability	30	70
DD Final Checkout	50	50
DD Delivery	100	0
Unit Test Plan Document	50	50
ITPD Integration Strategy Definition	100	0
ITPD Stubs,Driver and Data Test Identification	30	70
ITPD Individual Step: Definition & Description	100	0
ITPD Tools Choices	0	100
ITPD Final Checkout	50	50
ITPD Delivery	100	0
Application Logic Implementation	50	50
Dispatcher Implementation	50	50
PA Implementation	100	0
TMA Implementation	0	100
TMA Unit Test	100	0
Application Logic Unit Test	0	100
PMA Unit Test	100	0
Dispatcher Unit Test	50	50
TMA Integration Test	100	0
PMA Integration Test	0	100
Application Logic Integration Test	100	0
Dispatcher TMA Integration Test	0	100
Dispatcher PMA Integration Test	0	100
Applcation Logic Database Integration Test	0	100
Application Logic Dispatcher Integration Test	100	0
All Sub-System Integration Test	50	50

Chapter 5

Risks

During all the phase of the *MyTaxiService* life we have to face many risk that can lead our project in the worst case to failure or to have to invest a large quantity of money that weren't estimated to solve it.

The risk can be classified into three different category:

- Project Risk
- Implementation Risk
- Business Risk
- Technical Risk

The relevance of a problem that will spring out will be classified with three value:

- Negligible
- Normal
- Catastrophic

5.1 Project Risks

Those issues can occur during the different phases of the project.

Project Issue	Description	Effect	Possible Recovery Action
Misunderstood of the Requirement	During the compilation of the RASD document some requirement can be misunderstood by the member of the project.	Catastrophic	New release of RASD document, rearrange of all the related document. Then modify the implementation according to the changes.
Changes of the Requirement	During the building of the project some change on the environment can bring to a modification of some requirement.	Catastrophic	New release of Rasd document, rearrange of all the related document. Then modify the implementation according to the changes.
Missing personnel	Some members of the staff don't participate at the different phase of the project.	Normal	Dismiss those members and hire other personnel.
Delays of deadline	A miscalculation of the time required for each step of the project can lead to a delays of the expected deadlines.	Catastrophic	Hire new personnel in order to recover the time loss for some of the step where we made the miscalculation.

5.2 Implementation Risks

Those issues can occur during the implementation of the project.

Implementation Issue	Description	Effect	Possible Recovery Action
Misunderstood of the Document	During the implementation of the project some software developer can misunderstood some requirement of the document.	Normal	Make a step back during the implementation of the project to a safe position and modify the part where the developer start to go out from the predetermined path.
Fail of some Integration Test	During the implementation some integration test fails.	Catastrophic	Check the modules involved and start to find what cause the failure during the integration test.
Fail of some Unit Test	During the implementation some unit test fails.	Normal	Check the module involved and try to resolve the problem.

5.3 Technical Risks

Those issues can occur after the complete implementation of the system, when MyTaxiService have to start to work on the different machines.

Implementation Issue	Description	Effect	Possible Recovery Action
Congestion of the line	The service isn't able to answer instantly to all the connected user.	Negligible	Try to increase the broadband by asking to the Provider an upgrade.
One machine stop working	One machine that our application use stop working.	Normal	Call the technicial to repair the machine. In the meantime the other server farm's machine will compensate the work of the broken apparatus.
Data Loss	The data can be lost due to an attack or machine problem.	Catastrophic	Make backup very often in order to minimize this problem and, in case of data loss, use those backups to restore the system.
Data Leak	The data can be shared due to an attack from some hacker.	Catastrophic	Reorganize the security system adding some new data flow control and some other firewalls inside the system.

5.4 Business Risks

Those issues can occur after the complete the implementation, and the causes are external to the system.

Implementation Issue	Description	Effect	Possible Recovery Action
New Competitors	Some competitors for the taxi service appear in the city after the delivery of our application.	Negligible	Try to understand what are the difference from the others competitors and, in case, introduce some new functionality.
Bankruptcy	A miscalculation of the cost for the project and for the maintenance of the system can lead to a bankruptcy.	Catastrophic	If the bankruptcy appear, there aren't recovery action.
User not satisfy from the Service	The consumer aren't satisfy from the different service our application provide.	Normal	Provide some surveys about what we can improve on our Service Application.