Politecnico di Milano

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Computer Science and Engineering

Software Engineering 2 project

myTaxiService
Project Plan Document

Version 1

Prof. Elisabetta Di Nitto

Andrea Autelitano	Matr. 849869
Marco De Cobelli	Matr. 858360
Matthew Rossi	Matr. 858880

Academic year 2015-2016

TABLE OF CONTENTS

1 Introduction

1.1 Revision History

Version	Date	Author(s)	Description
1.0	02-02-2016	Andrea Autelitano Marco De Cobelli Matthew Rossi	Project Plan Document v1.0

1.2 Purpose and Scope

1.2.1 Purpose

This document describes the plan for all phases of the development of the project "myTaxiService". An estimation of effort shall be executed using both the function points and COCOMO II methods. After that, the project will be divided into tasks and each task will be allocated to one or more members of the development team. Finally, risks for each phase of the development process will be considered.

1.2.2 Scope

The aim of this project is to optimize the taxi service of a large city thanks to the simplification of the access to the service for the users and the fair management of taxi queues.
The registered users will be able to request or book a taxi from either a mobile application or a web site and Taxi drivers will use the mobile application to accept the requests of the users.
The city is divided into zones of approximately 2km^2, each of these with a different queue of taxis.
The system will choose the taxi from the zone that is nearest to the user that has made the request/reservation. Usually, it is their own zone.
Furthermore, when a taxi becomes available, either because its taxi driver logs into the system or the related taxi driver reports the termination of a ride, its identifier will be added at the end of the queue of the current zone in which he is located.
When the first taxi driver in a queue receives a request, they can freely choose whether to accept it or to refuse it. In the first case, a notification is sent to the client saying that one taxi driver has accepted their request, in the second case the request is sent to the second taxi driver in the queue and so on.
Moreover, there is the possibility to reserve a taxi from a certain origin to a specific destination within two hours before the ride; in this case, 10 minutes before the ride the system will allocate it to the first taxi driver in the queue.
Finally, besides the traditional interfaces, programmatic interfaces are provided in order to enable the development of additional services like the taxi sharing option. This option allows users to share the cost of a ride with other people after specifying the starting point, the destination of the ride, the number of people to carry on and the availability to share the ride.
In any case, the system automatically calculates the route of the taxi toward the destinations using Google Maps and it provides the fee for each person in the taxi and notifies the users and the Taxi driver who has accepted the ride before the departure.

1.3 List of Definitions and Abbreviations

1.3.1 Definitions

Client	A person that uses the application in order to request rides
Taxi driver	A person that uses their taxi in order to provide the ride
User	Both a Client and a Taxi driver
Route	The ride that has been booked by a Client
Ride	The ride that has been allocated to a Taxi driver and that can be composed of a single route or in the case of a shared ride, more than one route
Tasks	Activities which must be completed to achieve the project goal
Milestones	Points in the schedule against which you can assess progress (e.g. the handover of the system to be tested)
Risk	A potential problem that might happen or not

1.3.2 Abbreviations

UI	User Interface

1.4 List of Reference Documents

1.4.1 External Documentation

ID	Name of the document	Date
[A0]	AA 2015-2016 Software Engineering 2 - Project goal, schedule and rules	October 2015
[A1]	Software Engineering 2 Project, AA 2015-2016 Assignments 1 and 2	October 2015
[A4]	Software Engineering 2 Project, AA 2015-2016 Assignment 4, Test Plan	January 2016
[A5]	Software Engineering 2 Project, AA 2015-2016 Assignment 5, Project Plan	January 2016

1.4.2 Internal Documentation

ID	Name of the document	Version	Date
[RASD]	myTaxiService - Requirements Analysis and Specification Document	2.0	December 2015
[SDD]	myTaxiService – Software Design Description	1.0	December 2015
[ITPD]	myTaxiService – Integration Test Plan Document	1.0	January 2016

2 Function points method

2.1 Introduction

A function point is a unit of measurement to express the amount of business functionalities a software application provides to a user. They measure software size and the cost (in dollars or hours) of a single unit that is calculated from past projects.
The approach on which the function points are based consists in summing the number of elements of a given type multiplied by their weights and then summing all the partial values obtained. Note that the function point count is modified by the complexity of the project.
The steps to be executed in the function points approach are:

1. Determine function counts by type.

   There are five types of function points on which we will perform the analysis:
   - **Internal Logical File (ILF).** Homogeneous set of data used and managed by the application.
   - **External Interface File (EIF).** Homogeneous set of data used by the application but generated and maintained by other applications.
   - **External Input.** Elementary operation to elaborate data coming from the external environment.
   - **External Inquiry.** Elementary operation that involves input and output. Without significant elaboration of data from logic files.
   - **External Output.** Elementary operation that generates data for the external environment. It usually includes the elaboration of data from logic files.

2. Determine complexity levels of each function counts using the following table.

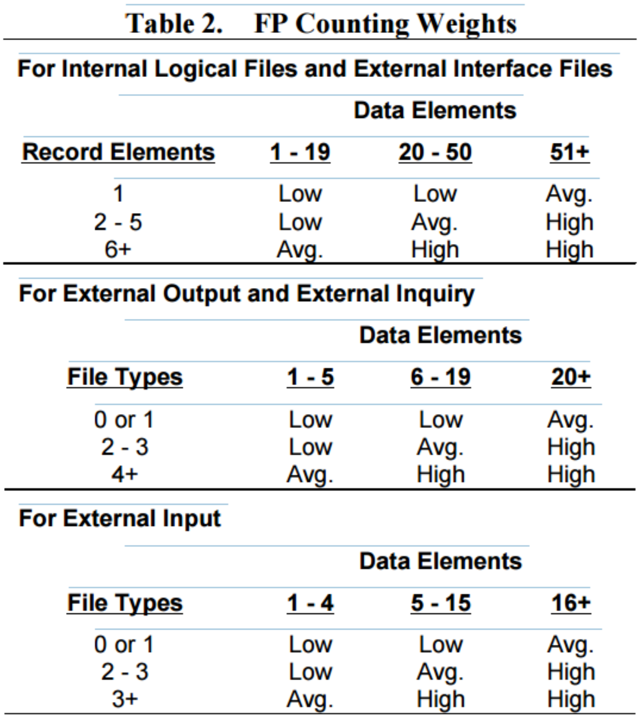


Table 2. FP Counting Weights

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	**1 - 19**	**20 - 50**	**51+**
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

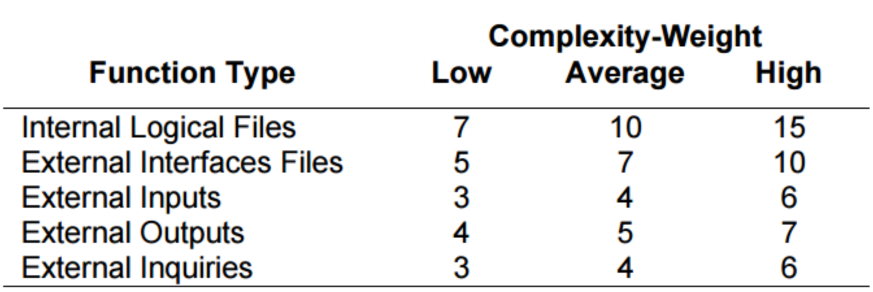
	Data Elements		
File Types	**1 - 5**	**6 - 19**	**20+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	**1 - 4**	**5 - 15**	**16+**
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

3. Apply complexity weights using the following table.

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

4. Compute Unadjusted Function Points (UFPs) summing all the weighted functions counts.

All the tables above are taken from *COCOMO II, Model Definition Manual:* http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

2.2 Function points estimation

2.2.1 Internal Logical Files (ILFs)

The application includes a number of ILFs that we will use to store information about:

- **Clients.** The system saves a lot of information about clients (i.e. email, password, name, surname, gender, address and date of birth).
- **Taxi drivers.** The system saves a lot of information about taxi drivers (i.e. email, password, name, surname, gender, address, date of birth, taxi license, personal license and IBAN).
- **Routes.** Since they are the core object of the system, a very large amount of data will be saved about routes (i.e. origin, destination, time of reservation, time of departure, date of departure, date of arrival, km, price, number of people, current status of the route, shared route or not).
- **Rides.** They represent a collection of routes for the same taxi and they are characterized by an origin and a destination.
- **Taxis.** The system saves information about plate and the number of seats.
- **Credit cards.** The system saves information about the number of credit cards, the owner name and surname and the expiry date.

From the description above, the resulting complexity and function points for each ILF are:

Internal Logical File (ILF)	**Complexity**	**Function points**
Clients	Average	10
Taxi driver	Average	10
Routes	High	15
Rides	Low	7
Taxis	Low	7
Credit cards	Low	7
Total		56

2.2.2 External Interface Files (EIFs)

The application manages the interaction with external entities for acquiring information about:

- **Geographical information for the position of taxi drivers and the calculation of the routes**. It is acquired from Google Maps APIs using the current and the destinations position in coordinates or addresses.
- **A public authority** (e.g. *Ministero delle Infrastrutture e dei Trasporti*) allows to verify the existence and validity of the data inserted by a taxi driver relative to their taxis information.
- **Public authorities that manages taxi licenses and their owners** allow to verify the correctness of the data about taxi licenses inserted by taxi drivers.
- **Credit cards' companies** to verify correctness of credit cards' data inserted by clients and that provide the support for the payments of the rides to the taxi drivers.

The system will convert all the received information in a correct format in order to be able to elaborate them.

From the description above, the resulting complexity and function points for each EIF are:

External Interface File (EIF)	**Complexity**	**Function points**
Google maps	Low	5
Ministero delle Infrastrutture e	Low	5

dei Trasporti		
Public authorities that manages taxi licenses and their owners	Low	5
Credit cards' companies	Low	5
Total		20

2.2.3 External Inputs

The application interacts with clients and taxi drivers to allow the following operations:

- **Registration.** A simple operation involving the insertion in the system of the data related to the new user.
- **Login/Logout.** Simple operations that need only to create/close a session.
- **Password recovery.** A very simple operation that consists of sending an email to the user that has requested a new password after they have inserted their email. It involves only the entity related to the current specific kind of user.
- **Modification of personal information.** It is a simple operation because it consists only of the insertion of some new data that replace the older data in a single entity (Taxi driver or Client). For example, a Taxi driver can modify their driving license and IBAN.
- **Addition/deletion of a credit card.** They are operations of average complexity. Clients can add more than one credit card to their account and delete a credit card when they want. These operations use the Client and the Credit cards entities.
- **Addition/modification/deletion of a taxi.** They are activities of average complexity because they involve the taxi and the taxi driver entities.
- **Taxi driver working-state modification.** It is an activity of average complexity because it involves operations like zone identification or the insertion in/removal from a queue of taxi drivers.
- **Ride request/modification/cancelation/interruption (both shared and not shared).** It is an operation of average complexity because it involves client, rides, routes and taxi driver.
- **Ride payment.** A complex operation that involves the management of security and client, taxi driver, credit cards and route entities.
- **Conclusion of a ride.** When a taxi driver inserts the input data that allow the conclusion of a ride, this operation involves the taxi driver, client, route, ride entities: therefore, its complexity has to be considered high.

In the following table the results of the above analysis are summarized.

External input	**Complexity**	**Function points**
Registration	Low	3
Login/Logout	Low	2 x 3
Password recovery	Low	3
Modification of personal information	Low	3
Addition/deletion of a credit card	Average	2 x 4
Addition/modification/deletion of a taxi	Average	3 x 4
Taxi driver working-state modification	Average	4
Ride request/modification/cancelation/interruption (both shared and not shared)	High	4 x 6
Ride payment	High	6

Conclusion of a ride	High	6
Total		75

2.2.4 External Inquiries

The application allows taxi drivers and clients to requests these pieces of information:

- **User profile information.** They can see their profile information and this is a simple activity that involves the usage of a single entity.
- **List of their pending/concluded rides.** It is an operation of average complexity because it involves the usage of entities like Taxi Driver/Client, Rides and Routes.
- **List of shared rides available.** This is a very complex activity because it involves the research of a large amount of data in the system to find compatible routes with that of the client who is searching and the communication with external interfaces (Google Maps).

In the following table the results of the above analysis are summarized.

External inquiry	**Complexity**	**Function points**
User profile information	Low	3
List of their pending/concluded rides	Average	2 x 4
List of shared rides available	High	6
Total		17

2.2.5 External Outputs

The application will notify the users with a push notification on their smartphones in the following main cases:

- **Acceptance/Refusal of a taxi request (push notification for the taxi driver).** This is received by the Taxi driver and sent from the system after it has elaborated a new client request. This request involves the Taxi Driver, Client and Route and Ride entities and is to be considered of high complexity.
- **Acceptance of a taxi request (push notification for the client).** This notification is sent to the client when their ride has been accepted by a Taxi driver. It involves Client, Taxi Driver and Route entities and has an average complexity.
- **Interruption of a ride (push notification for the client).** It is sent to the clients of the interrupted ride for some problems on a taxi or other reasons. The complexity is high and it involves the Client, Taxi Driver, Ride and Route entities.
- **Successful payment with credit card (push notification).** It is sent to the client and the taxi driver when a payment is executed. The complexity is average and involves Client, Taxi driver and Route entities.

In the following table the results of the above analysis are summarized.

External output	**Complexity**	**Function points**
Acceptance/Refusal of a taxi request (push notification for the taxi driver)	High	7
Acceptance of a taxi request (push notification for the client)	Average	5

Interruption of a ride (push notification for the client)	High	7
Successful payment with credit card (push notification)	Average	5
Total		24

2.2.6 Total number of Function Points (UFPs)

By summing all the numerical total values obtained from the 5 types of function points that we have considered above, we obtain an estimation of the number of UFPs.
In COCOMO II we will use this value to obtain the estimation of the source lines of code (SLOC).
In the following table our estimations are summarized.

Function type	**Value**
Internal Logical Files (ILFs)	56
External Interface Files (EIFs)	20
External inputs	75
External inquiries	17
External outputs	24
Total	192

3 COCOMO II method

3.1 Introduction

The COCOMO II method is an algorithmic method used to estimate the effort required for the development of a software. The effort PM is calculated in person-months using the following formula:

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

where the parameters have the following values:

- A = 2.94, as determined in COCOMO II Model Definition Manual;
- EAF = Effort Adjustment Factor, obtained as

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

  where EM_i are the Effort Multipliers corresponding to each Cost Driver (see section 3.2 below);
- KSLOC = thousands of Source Lines Of Code, estimated using the Function Points method;
- E = Exponential obtained using the following formula:

$$E = B + 0.01 * \sum_{J=1}^{5} SF_j$$

  where B = 0.91, as determined in COCOMO II Model Definition Manual and SF_j are the Scale Factors determined in section 3.3.

Using the value of PM, the amount of calendar time for the development of the software TDEV can be estimated using the following formula:

$$TDEV = C * PM^{F}$$

where the parameters have the following values:

- C = 3.67, as determined in COCOMO II Model Definition Manual;
- PM = the value calculated at the previous step;
- F = determined using the formula

$$F = D + 0.2 * (E - B)$$

  where D = 0.28, as determined in COCOMO II Model Definition Manual.

Finally, the average number of staff NS required for the development of the project will be estimated using the following formula:

$$NS = \left\lceil \frac{PM}{TDEV} \right\rceil$$

In the following sections, all parameters will be estimated referencing to the tables taken from

COCOMO II, Model Definition Manual:
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

3.2 Source Lines of Code

The source lines of code can be simply estimated multiplying the function points calculated in Chapter 2 by the factor calculated for J2EE in the table found here: http://www.qsm.com/resources/function-point-languages-table
The average factor for J2EE is 46, therefore the estimated number of SLOC is

$$SLOC = UFP * FPL = 168 * 46 = 7728 \ \cong 7.7\ KSLOC$$

3.3 Cost Drivers

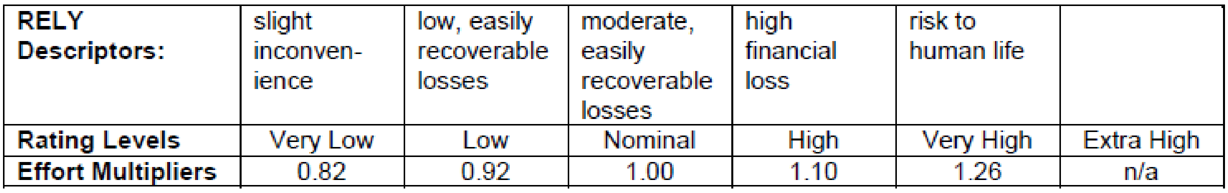
For the evaluation of the Cost Drivers, we will apply the Post-Architecture Model. This decision was made since it is the most detailed model and can be applied using the SLOC that we have estimated in Chapter 3.2, and when a software life-cycle architecture has been developed, which is our case.

3.3.1 Product Factors

3.3.1.1 Required Software Reliability (RELY)

This is the measure of the extent to which the software must perform its intended function over a period of time. In [RASD, Chapter 3] we have stated that it shall have a reliability of 99.99%, therefore it is set to High.

Table 17. RELY Cost Driver

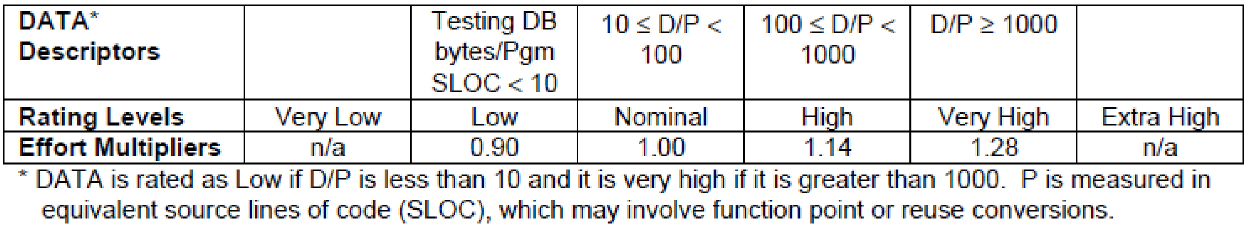


RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

3.3.1.2 Data Base Size (DATA)

This is the measure of the effect large test data requirements have on product development using the ratio D/P, where D is the size (in bytes) of the testing database and P is the number of SLOC. We do not have the exact value of D, but we can imagine that the ratio will be between 10 and 100, therefore its value will be set to Nominal.

Table 18. DATA Cost Driver



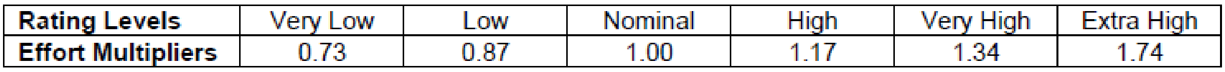
DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

* DATA is rated as Low if D/P is less than 10 and it is very high if it is greater than 1000. P is measured in equivalent source lines of code (SLOC), which may involve function point or reuse conversions.

3.3.1.3 Product Complexity (CPLX)

This is the measure of the complexity of the final product, divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. The average complexity can be considered High.

Table 20. CPLX Cost Driver

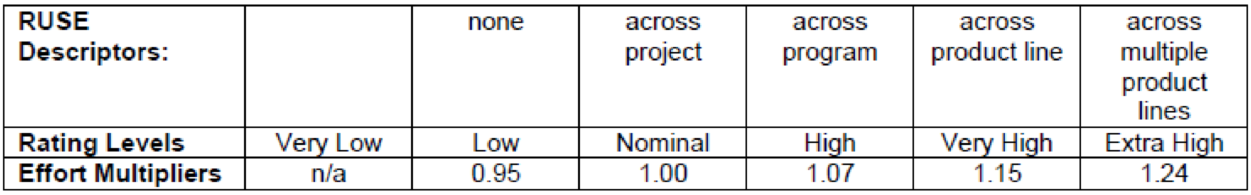


Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.73	0.87	1.00	1.17	1.34	1.74

3.3.1.4 Developed for Reusability (RUSE)

This is the measure of the additional effort needed to construct components intended for reuse on current or future projects. As stated in [RASD, Chapter 3], few modules will be developed to be reused, therefore its value will be set to Nominal.

Table 21. RUSE Cost Driver

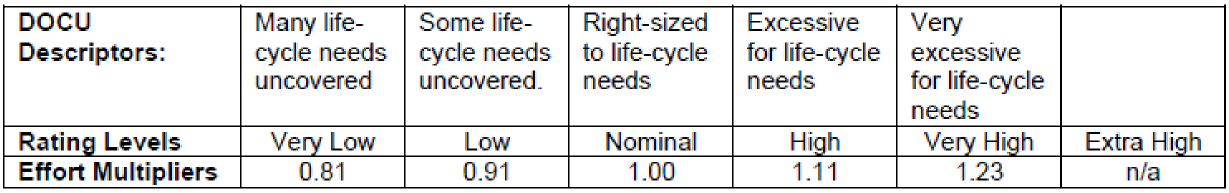


RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

3.3.1.5 Documentation Match to Life-Cycle Needs (DOCU)

This is the measure of the level of required documentation. It must be extremely thorough in order to avoid additional costs in a future maintenance scenario; therefore its value will be set to High.

Table 22. DOCU Cost Driver



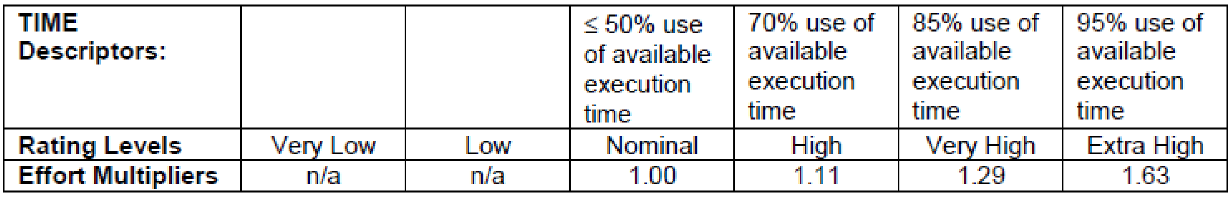
DOCU Descriptors:	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

3.3.2 Platform Factors

3.3.2.1 Execution Time Constraint (TIME)

This is the measure of the degree of execution time constraint imposed upon a software product. The rating is expressed in terms of available execution time expected to be used. It will usually be less than 50%, therefore its value will be set to Nominal.

Table 23. TIME Cost Driver

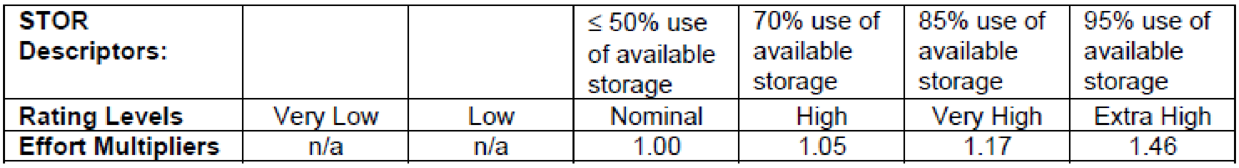


TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

3.3.2.2 Main Storage Constraint (STOR)

This is the measure of main storage constraint imposed on a software system or subsystem. It will usually be less than 50%, therefore its value will be set to Nominal.

Table 24. STOR Cost Driver

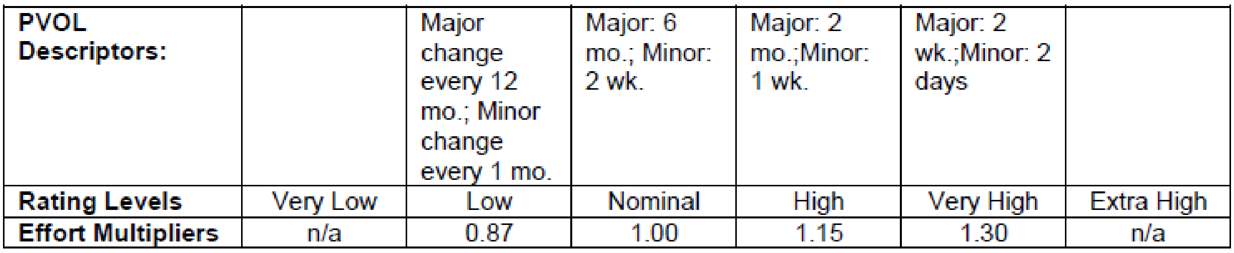


STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

3.3.2.3 Platform Volatility (PVOL)

This is the measure of the number and level of changes that will be applied to the platform (e.g. hardware, software…). Since our system will be deployed mostly on Amazon Web Services, we do not expect an elevated number of changes, therefore its value will be set to Low.

Table 25. PVOL Cost Driver



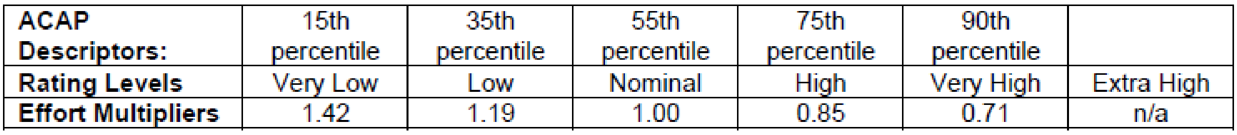
PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

3.3.3 Personnel Factors

3.3.3.1 Analyst Capability (ACAP)

This is the measure of analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate of the analysts. Its value will be set to High.

Table 26. ACAP Cost Driver

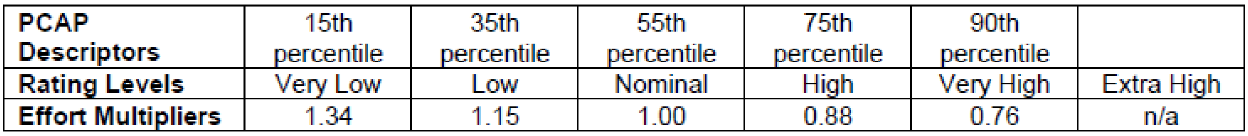


ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

3.3.3.2 Programmer Capability (PCAP)

This is the measure of ability, efficiency and thoroughness, and the ability to communicate and cooperate of the programmers, seen as a team. Its value will be set to High.

Table 27. PCAP Cost Driver

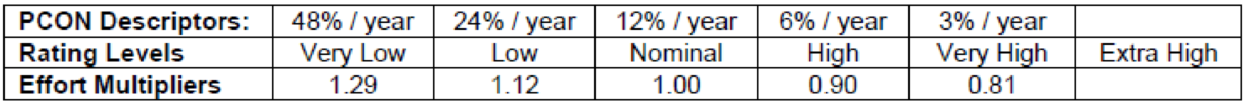


PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

3.3.3.3 Personnel Continuity (PCON)

This is the measure of the project's annual personnel turnover: since it will not change its value can be set to Very High.

Table 28. PCON Cost Driver

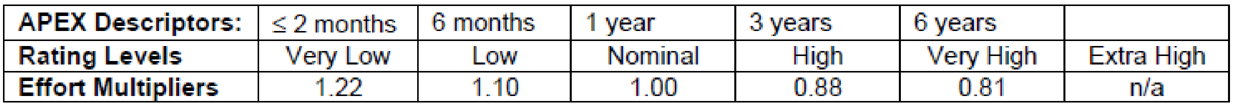


PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

3.3.3.4 Applications Experience (APEX)

This is the measure of the level of experience of the project team in developing this type of application. Since we have very few previous experiences in the development of such an application, its value will be set to Low.

Table 29. APEX Cost Driver

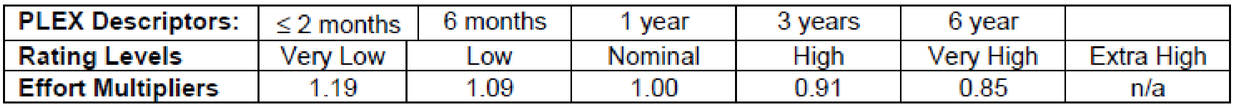


APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

3.3.3.5 Platform Experience (PLEX)

This is the measure of understanding the use of more powerful platforms, such as more graphic user interface, database, networking, and distributed middleware capabilities. We have been studying these kinds of platforms for about a year, therefore its value will be set to Nominal.

Table 30. PLEX Cost Driver

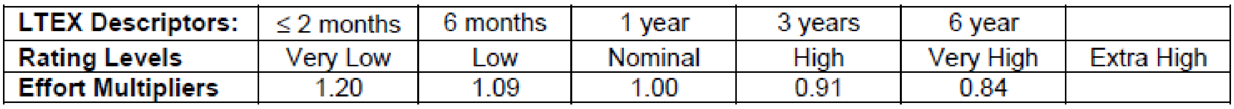


PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

3.3.3.6 Language and Tool Experience (LTEX)

This is the measure of the knowledge of the used programming language and tools. The project will be developed using Java Enterprise Edition using NetBeans as IDE, tools that we have been studying for less than 6 months, therefore its value will be set to Low.

Table 31. LTEX Cost Driver



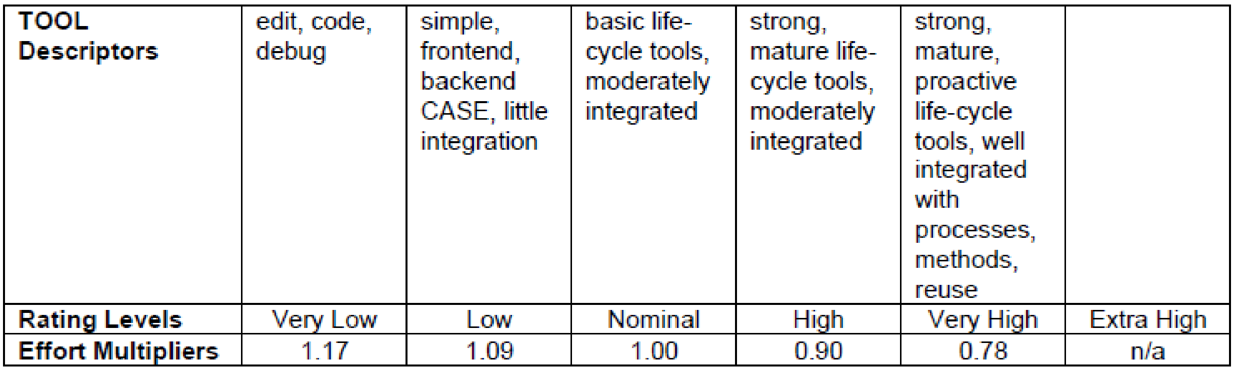
LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

3.3.4 Project Factors

3.3.4.1 Use of Software Tools (TOOL)

This is the measure of the level of the used software tools. The tools we have used have been on the market for a long time, therefore its value will be set to High.

Table 32. TOOL Cost Driver

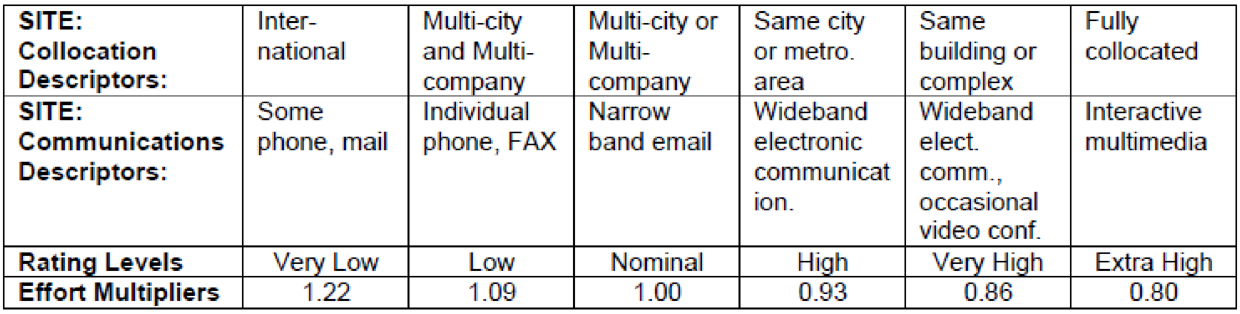


TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

3.3.4.2 Multisite Development (SITE)

This is the measure of the physical distance among the members of the team. Although we live in the same city (and have been able to work in the same room), sometimes we had to use instant messaging applications and occasional Skype calls in order to coordinate our work, therefore its value will be set to High.

Table 33. SITE Cost Driver

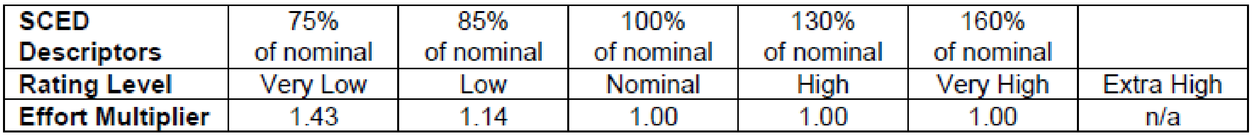


SITE: Collocation Descriptors:	Inter-national	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communication.	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

3.3.4.3 Required Development Schedule (SCED)

This is the measure of the schedule constraint imposed on the project team. We have always been able to respect the deadline but using more than the expected time, therefore its value will be set to High.

Table 34. SCED Cost Driver



SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

3.4 Scale Drivers

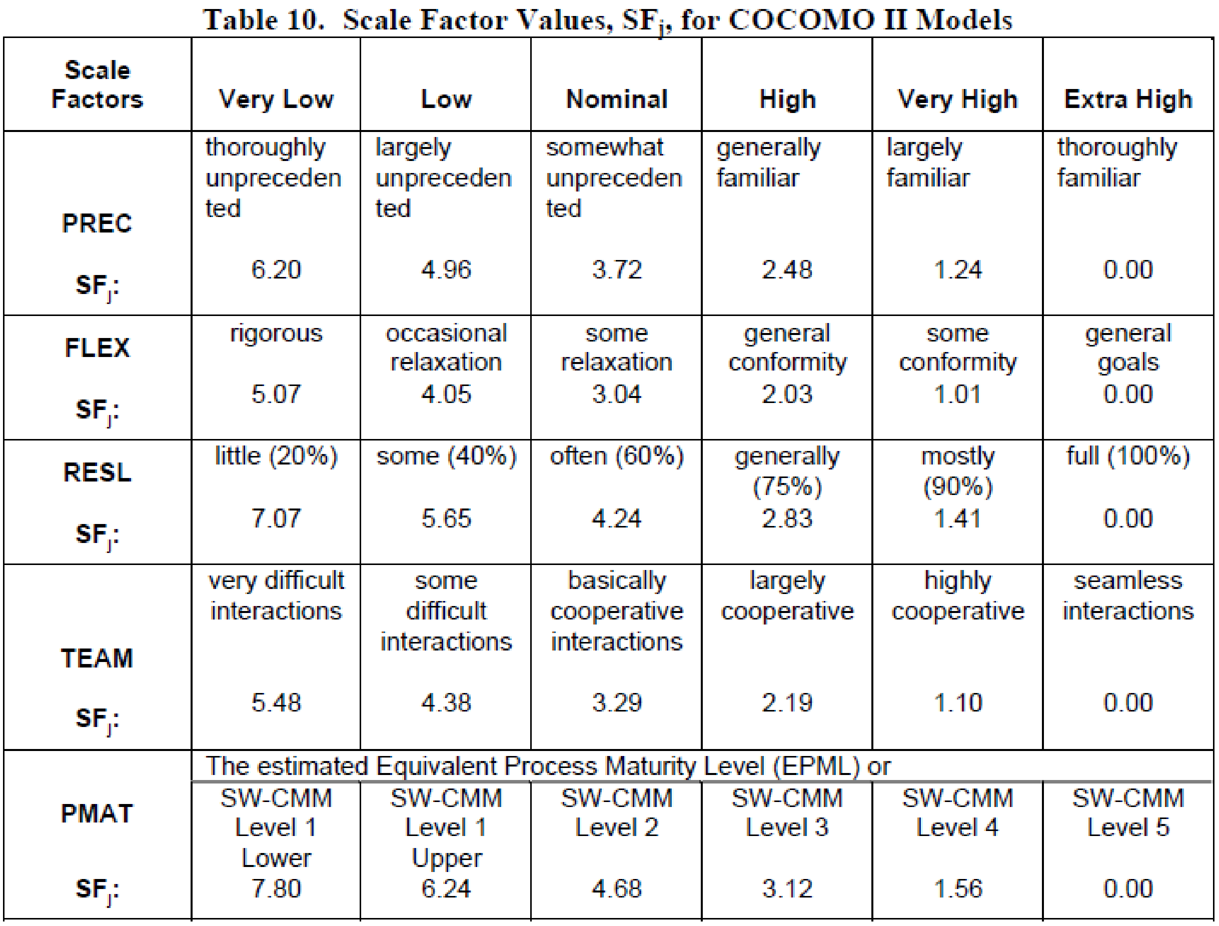


Table 10. Scale Factor Values, SF$_j$, for COCOMO II Models

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC SF$_j$:	thoroughly unprecedented 6.20	largely unprecedented 4.96	somewhat unprecedented 3.72	generally familiar 2.48	largely familiar 1.24	thoroughly familiar 0.00
FLEX SF$_j$:	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL SF$_j$:	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 SF$_j$:	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	The estimated Equivalent Process Maturity Level (EPML) or					
SF$_j$:	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	SW-CMM Level 2 4.68	SW-CMM Level 3 3.12	SW-CMM Level 4 1.56	SW-CMM Level 5 0.00

3.4.1 Precedentedness (PREC)

It expresses the similarity to projects previously developed by the team. Since it is the first time we have worked on such a project, its value will be set to Low.

3.4.2 Development Flexibility (FLEX)

It expresses the flexibility of the requirements and usage of external interfaces. Since the specifications were not excessively detailed and we did not have any restriction in the choice of technologies, its value will be set to Very High.

3.4.3 Architecture / Risk Resolution (RESL)

It expresses the thoroughness in the analysis of risks and definition of the software architecture. Since both processes were extremely thorough, its value will be set to Very High.

3.4.4 Team Cohesion (TEAM)

It expresses the sources of project turbulence and entropy because of difficulties in synchronizing the project stakeholders' activities. Since we did not face particular problems in the synchronization of our activities, its value will be set to Very High.

3.4.5 Process Maturity (PMAT)

It expresses the maturity of the development process and it is calculated from the following 18 Key Process Areas (KPAs):

KPA	Factor	Value
Requirements Management	Frequently	75%
Software Project Planning	About Half	50%
Software Project Tracking and Oversight	Frequently	75%
Software Subcontract Management	About Half	50%
Software Quality Assurance (SQA)	Frequently	75%
Software Configuration Management (SCM)	Frequently	75%
Organization Process Focus	Frequently	75%
Organization Process Definition	Frequently	75%
Training Program	Rarely if Ever	1%
Integrated Software Management	About Half	50%
Software Product Engineering	Frequently	75%
Intergroup Coordination	Frequently	75%
Peer Reviews	Almost Always	100%
Quantitative Process Management	About Half	50%
Software Quality Management	About Half	50%
Defect Prevention	Frequently	75%
Technology Change Management	Frequently	75%
Process Change Management	Frequently	75%

Using the following formula, we can calculate the value of the equivalent process maturity level EMPL, which can be linked to PMAT:

$$EMPL = 5 * \frac{1}{n} * \sum_{i=1}^{n} \frac{KPA\ \%_i}{100} = 3.2\overline{6}$$

which is equivalent to High.

3.5 Final count

Scale Driver	Factor	Value
Precedentedness (PREC)	Low	4.96
Development Flexibility (FLEX)	Very High	1.01
Architecture / Risk Resolution (RESL)	Very High	1.41
Team Cohesion (TEAM)	Very High	1.10
Process Maturity (PMAT)	High	3.12
Total:		11.6

Therefore, the value of E will be 1.026.

Cost Driver	Factor	Value
Required Software Reliability (RELY)	High	1.10
Data Base Size (DATA)	Nominal	1.00
Product Complexity (CPLX)	High	1.17
Developed for Reusability (RUSE)	Nominal	1.00
Documentation Match to Life-Cycle Needs (DOCU)	High	1.11
Execution Time Constraint (TIME)	Nominal	1.00
Main Storage Constraint (STOR)	Nominal	1.00
Platform Volatility (PVOL)	Low	0.87
Analyst Capability (ACAP)	High	0.85
Programmer Capability (PCAP)	High	0.88
Personnel Continuity (PCON)	Very High	0.81
Applications Experience (APEX)	Low	1.10
Platform Experience (PLEX)	Nominal	1.00
Language and Tool Experience (LTEX)	Low	1.09
Use of Software Tools (TOOL)	High	0.90
Multisite Development (SITE)	High	0.93
Required Development Schedule (SCED)	High	1.00
Product:		0.76

Applying the formulas described in Chapter 3.1, we obtain the following results:

$$PM = A * EAF * KSLOC^{E} = 2.94 * 0.76 * 7.7^{1.026} = 18.14\, person\ months$$

$$TDEV = C * PM^{F} = 3.67 * 18.14^{0.3032} = 8.84\, months\ \cong 9\, months$$

$$NS = \left\lceil \frac{PM}{TDEV} \right\rceil = \left\lceil \frac{18.14}{8.84} \right\rceil = 3\, persons$$

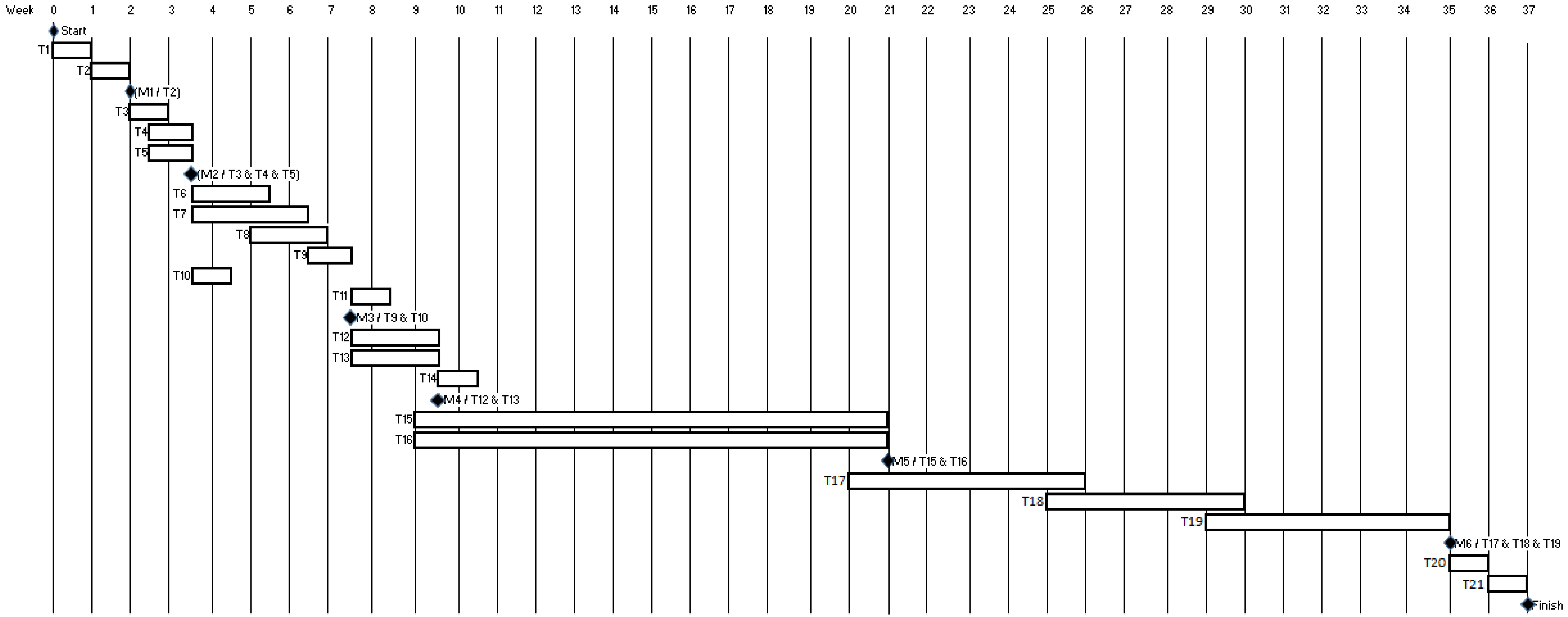
4 Tasks and schedule

The following table shows how the project has been divided into tasks and the dependencies among different tasks.

In most cases, dependencies are weak precedencies where only part of the “entry” tasks must have been completed in order to execute the current one.

For each task we have also specified the estimated time and resources needed to accomplish it; in order to be as precise as possible we did these estimation considering the actual availability of the team members, that is **3 hours per day, 5 days per week**.

Tasks		**Effort (person-days)**	**Duration (days)**	**Dependencies**
ID	**Name**			
	Requirements analysis and specification			
T1	Meetings with the client	3	5	
T2	Feasibility analysis	12	5	T1
T3	Functional requirements definition	4	5	T2
T4	Non-functional requirements definition	4	5	T2
T5	Interfaces definition	8	5	T2
T6	System test plan	18	10	T3, T4, T5
	Architectural design			
T7	Deployment architecture design	20	15	T3, T4, T5
T8	High level components design	8	10	T7
T9	High level components interaction design	5	5	T8
T10	Data base design	6	5	T3, T4, T5
T11	Integration test planning	13	5	T8, T9
	Detailed design			
T12	Detailed front-end components design	8	10	T9, T10
T13	Detailed back-end components design	18	10	T9, T10
T14	Unit test planning	7	5	T12, T13
	Coding			
T15	Front-end components implementation	50	60	T12, T13
T16	Back-end components implementation	100	60	T12, T13
	Testing			
T17	Unit testing	70	30	T14, T15, T16
T18	Integration testing	60	25	T11, T17
T19	System testing	70	30	T6, T18
	Delivery			
T20	Installation	15	5	T19
T21	Acceptance testing	15	5	T20



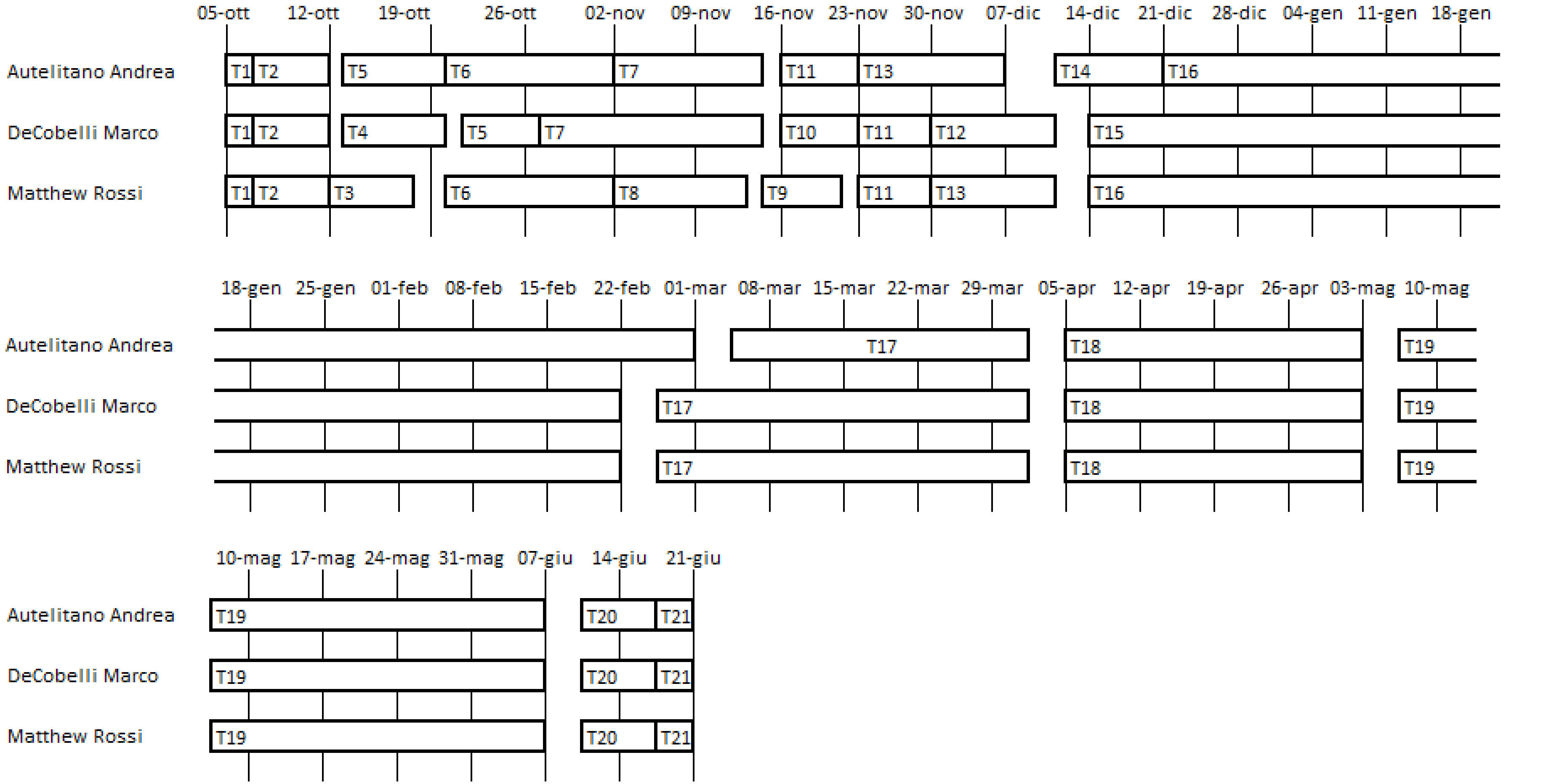
♦: is the graphical representation of a milestone

M1: the project plan has been completed
M2: the domain and the requirements have been defined
M3: the architecture has been designed

M4: the components have been designed in details
M5: software development has been completed
M6: unit, integration and system testing have been completed successfully

5 Resource allocation

The following bar charts show the team members' allocation to the tasks defined at paragraph 4 considering the actual availability of the team members, that is 3 hours per day, 5 days per week.



6 Risk assessment

The following table lists the possible identified risks, the estimate of the probability that it will occur and the impact (i.e., damage) it will have if it does occur.

Risk	Probability	Effects
Executive turnover disrupts project A key executive leaves the company, the resulting disruption becomes a project issue.	Low	Catastrophic
Gold plating inflates scope The project team add their own product features that aren't in requirements or change requests.	Moderate	Marginal
Estimates are inaccurate Inaccurate estimates is a common project risk.	High	Serious
Change management overload A large number of change requests dramatically raises the complexity of the project and distracts key resources.	Low	Catastrophic
Project team misunderstand requirements When requirements are misinterpreted by the project team a gap develops between expectations, requirements and work packages.	Moderate	Serious
Under communication Communication is a challenge that's not to be underestimated. You may need to communicate the same idea many times in different ways before people remember it.	Moderate	Marginal
Resource shortfalls Inability to secure sufficient resources for the project.	Moderate	Serious
Resources are inexperienced Resources who are just out of school or who are new to your industry or profession tend to make more mistakes and be less productive.	High	Marginal
Team members with negative attitudes towards the project Resources who are negative towards the project may actively or passively sabotage project efforts.	Low	Catastrophic
Resource turnover Resource turnover leads to delays and cost overrun.	Low	Serious
Architecture lacks flexibility The architecture is incapable of supporting change requests and needs to be reworked.	Moderate	Serious
Architecture is not fit for purpose The architecture is low quality.	Low	Catastrophic
Design lacks flexibility A poor design makes change requests difficult and costly.	Moderate	Serious
Design is not fit for purpose The design is low quality.	Low	Catastrophic
Technology components aren't fit for purpose Technology components are low quality.	Low	Serious
Technology components have security vulnerabilities Security vulnerabilities are key technology risks.	Moderate	Catastrophic

Technology components lack stability Components that crash.	Low	Serious
Failure to integrate with systems The risk that your product will fail to integrate with existing systems.	Moderate	Serious
Failure to integrate components The risk that product components will fail to integrate with each other.	Low	Serious
Requirements are ambiguous Requirements are unclear and open to interpretation.	Moderate	Serious
Requirements are low quality Requirements aren't fit for purpose.	Low	Serious
Requirements are incomplete You can spot obvious holes in the requirements.	Moderate	Serious
Legal & regulatory change impacts project If your project spans areas that are compliance-sensitive you may want to list regulatory change as a risk.	Low	Catastrophic
User interface doesn't allow users to complete tasks The risk that the user interface doesn't allow users to complete end-to-end tasks.	Low	Catastrophic
User interface is low quality The user interface is buggy, slow or difficult to use.	Low	Serious
Users reject the product The general risk that users will reject your product.	Low	Catastrophic

The following table shows the contingency plan to handle risks in a controlled and effective manner.

Risk	**Strategy**
Executive turnover disrupts project A key executive leaves the company, the resulting disruption becomes a project issue.	Reorganize team so that there is more overlap of work and therefore people understand each other's jobs.
Gold plating inflates scope The project team add their own product features that aren't in requirements or change requests.	Change in the requirements can be apply only to satisfy clients' needs.
Estimates are inaccurate Inaccurate estimates is a common project risk.	Investigate buying-in components; investigate use of a program generator.
Change management overload A large number of change requests dramatically raises the complexity of the project and distracts key resources.	Derive traceability information to assess requirements change impact; maximize information hiding in the design.
Project team misunderstand requirements When requirements are misinterpreted by the project team a gap develops between expectations, requirements and work packages.	Any requirement has to be explained so that the whole project team is on the same page.

Under communication Communication is a challenge that's not to be underestimated. You may need to communicate the same idea many times in different ways before people remember it.	Any common information that has to be communicated to other team members must be written in shared resources in the most clear way, so that no time is wasted for repeating same ideas.
Resource shortfalls Inability to secure sufficient resources for the project.	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Resources are inexperienced Resources who are just out of school or who are new to your industry or profession tend to make more mistakes and be less productive.	In the time estimation take into account a lower productivity of the team members and a longer testing phase.
Team members with negative attitudes towards the project Resources who are negative towards the project may actively or passively sabotage project efforts.	Keep the team members attitudes as positive as possible by avoiding the enforcement of impossible deadlines, too much effort and competition into the team.
Resource turnover Resource turnover leads to delays and cost overrun.	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Architecture lacks flexibility The architecture is incapable of supporting change requests and needs to be reworked.	Use as much as possible architectural patterns of known flexibility.
Architecture is not fit for purpose The architecture is low quality.	Make sure that in the defined architecture all the requirements are satisfied, by taking them into account in each architectural choice.
Design lacks flexibility A poor design makes change requests difficult and costly.	Use as much as possible design patterns of known flexibility.
Design is not fit for purpose The design is low quality.	Make sure that the design is consistent with respect to the defined architecture and detailed components can offer all functionalities.
Technology components aren't fit for purpose Technology components are low quality.	Investigate buying-in components.
Technology components have security vulnerabilities Security vulnerabilities are key technology risks.	Replace potentially insecure components with bought-in components of known security.
Technology components lack stability Components that crash.	Replace potentially defective components with bought-in components of known stability.
Failure to integrate with systems The risk that your product will fail to integrate with existing systems.	Develop components that corresponds to the architecture description so that the integration phase will not be a problem; anyway the integration testing is there to handle this kind of problems.

Failure to integrate components The risk that product components will fail to integrate with each other.	Develop components that corresponds to the architecture description so that the integration phase will not be a problem; anyway the integration testing is there to handle this kind of problems.
Requirements are ambiguous Requirements are unclear and open to interpretation.	Do further meetings with the client in order to resolve ambiguity.
Requirements are low quality Requirements aren't fit for purpose.	Reorganize team so that more time will be devoted to the requirements definition.
Requirements are incomplete You can spot obvious holes in the requirements.	Do further meetings with the client in order to capture requirements that has not been identified in the previous ones.
Legal & regulatory change impacts project If your project spans areas that are compliance-sensitive you may want to list regulatory change as a risk.	Derive traceability information to assess requirements change impact; develop a system as flexible as possible so that if changes are needed those can be applied in time.
User interface doesn't allow users to complete tasks The risk that the user interface doesn't allow users to complete end-to-end tasks.	In the interface definition task stay focused on the functionalities that the system has to guarantee to the users.
User interface is low quality The user interface is buggy, slow or difficult to use.	Reorganize team so that more time will be devoted to the user interface definition.
Users reject the product The general risk that users will reject your product.	Minimize overall changes deriving traceability information to assess change impact; change the product coherently to user directives.

7 Appendix

7.1 Used tools

- Microsoft Word 2013 to write this document
- Microsoft Visio 2013 to create UML diagrams
- Microsoft Excel 2013 to create the graphs

7.2 Hours of work

Each member of the group has worked on this document for 18 hours.