POLITECNICO DI MILANO
Computer Science and Engineering
Project of Software Engineering 2

myTaxiService

Project Plan Document
Ver. 1.1
Release date: February 2, 2016

Authors: Simone Rosmini (853949)
Vincenzo Viscusi (858689)
Matteo Zambelli (776162)

Reference Professor: Mirandola Raffaela

TABLE OF CONTENT

Change history:

v. 1.1: corrected some grammatical errors

1. PROJECT SIZE, EFFORT AND COST ESTIMATION

1.1 FUNCTION POINTS

The function point analysis is a method to estimate the cost of the development process but in our particular case we will use the result of the analysis as starting point to apply the COCOMO II model. So at the end we will calculate the UFP value (Unadjusted Function Point) that will be used in the COCOMO model to estimate the number of lines of source code.

To apply the function point analysis we will use some tables extracted from the COCOMO Model Definition Document:

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

The *Table 2* is used to estimate the level of complexity for each Function Type considering the number of data elements, of record elements and of referenced files.

Table 3. UFP Complexity Weights

	Complexity-Weight		
Function Type	**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

The *Table 3* is used to associate to the level of complexity estimated with the table 2 a complexity weight for each function type.
Using the 2 tables above we can obtain the weight corresponding to each function type that will be used to calculate the UFP.

Internal Logic Files

The application uses some ILFs to store information about: *Users*, *Taxi Drivers*, *Taxis* and *Zones.* Each of these files has a simple structure, with a small number of data elements and record elements, so we have decided to adopt the lowest complexity weight.
The application also stores information about *Calls* and *Rides*, which are two entities with a higher number of fields and record elements than the previous four but it isn't necessary to adopt the highest level of complexity so they ca be considered of average complexity.

ILF	Complexity	FP
Users	Low	7
Taxi Drivers	Low	7
Taxis	Low	7
Zones	Low	7
Calls	Average	10
Rides	Average	10
Total:		**48**

External Logic Files

The application uses only the external logic file relative to the map used for the Google Api, this is a very complex file which contains a high number of data elements and of record elements, so we can adopt the highest level of complexity.

ELF	Complexity	FP
Map	High	15
Total:		**15**

External Inputs

With respect to the design document, we can identify some external inputs analysing the following interfaces: *UserActions*, *TaxiActions* and *AdminInformations*. So we can analyse the function points for each single interface.

UserActions:

This interface is used by the users to request the following operations:

- *Login/Logout*, *Subscription*, *Update user information*. All of these are simple operations which involve not more than one ILF.

- *Make new Call, Modify Call.* These two operations involve at least three entities and they use a high number of data elements, so they can be considered of high complexity.

UserActions - EI	Complexity	FP
User Login/Logout	Low	3+3
Subscription	Low	3
Update user information	Low	3
New Call	High	6
Modify Call	High	6
Total:		**24**

TaxiActions:

This interface is used by the TaxiDrivers to ask the application to perform the following operations:

- *Login/Logout:* as for the user these are simple operations so we adopt the lowest level of complexity.
- *Passenger ON:* this is not a complex operation because it's used by the application only to know if a passenger is on the taxi or not.
- *Passenger OFF:* this operation can be considered more complex than the previous because it requires the update of the information about the call, the ride and the user history. We consider this operation of medium complexity.
- *Accept Ride:* this operation is quite complex because it requires information from both ILFs an ELFs, so we can use the highest level of complexity
- *Refuse Ride:* it doesn't involve any internal or external file so we can use a low complexity weight.
- *Finish Ride:* This operation involves a lot of ILs so we can adopt the highest level of complexity.
- *Set Taxi State:* it's a quite simple operation so it can be used a low level of complexity

TaxiActions - EI	Complexity	FP
Taxi Login/Logout	Low	3+3
Passenger ON	Low	3
Passenger OFF	Average	4
Accept Ride	High	6
Refuse Ride	Low	3
Finish Ride	High	6
Set Taxi State	Low	3
Total:		**31**

AdminInformations:

This interface is used by the Admin to set the information about the zones, about the taxis or about the taxi drivers. All of them can be considered of low complexity.

AdminInformations - EI	Complexity	FP
SetZoneInfo	Low	3
SetTaxiInfo	Low	3
SetTaxiDriverInfo	Low	3
Total:		**9**

External Inputs Total:

Summing the function point estimated above we obtain from the three interfaces a total FP value for the external inputs of **64**.

External Inquiries

The application allows the users to get information about the User account or about the Call History, it also allows to request the details of a single call. These are all simple operations which involve a small number of entities, so we use the lowest level of complexity.
There are also operations like Get route information or Get local Address which involve a higher number of entities so they can be considered of medium complexity.
Finally the application allows the admin to request information about the zones, the taxi driver and the taxi, that can be all considered simple operations.

EQ	Complexity	FP
View Call History	Low	3
Get Call Details	Low	3
Get User information	Low	3
Get Route information	Average	4
Get Local Address	Average	4
Get Zone Info	Low	3
Get Taxi Info	Low	3
Get Driver Info	Low	3
Total:		**26**

External Outputs

About the outputs the application uses two methods, one to send the notification to the user and another one to send the requests for Rides to the taxis. The first operation involves more ILFs but contains a medium number of data elements, so it can be considered of medium complexity. The second operation involves a high number of ILfs and ELFs so we adopt the highest level of complexity.

EO	Complexity	FP
Send Notifications	Average	4
Send Request	High	5
Total:		**9**

Unadjusted Function Point

Finally we can calculate the UFP value by making the sum of the weighted function points obtained above:

Function Type	Value
Internal Logic Files	48
External Logic Files	15
External Inputs	64
External Inquiries	26
External Outputs	9
UFP:	**162**

At this point of the Function Point Analysis it could be possible to calculate the Adjusted Function Point by considering the degree of influence of some Application Characteristics, like, for example, the influence of the Performance or of the Transaction rate, obtaining at the end a Technical Complexity Factor used to adjust the UFP indicator.
Nevertheless in our situation it is sufficient to stop with the UFP value that will be used to apply the COCOMO II.

COCOMO II

To execute the COCOMO II model and estimate the final effort for the project development we need, first of all, to estimate the number of lines of source code (SLOC). To do that we will use the UFP calculated in previous section. After that we will estimate all the effort rates concerning both the Scale Factors and the Cost Drivers by using the reference tables of the COCOMO II Model Definition Document. Finally we will use all the estimations done to apply the effort equation obtaining in this way the final estimation of the effort required for our project.

Estimation of the SLOC

The number of lines of source code can be estimated using the UFP considering the ratio SLOC/UFP which depends on the specific programming language used for the implementation of the software. For the J2EE language the ratio SLOC/UFP is equal to 46, this means that each UFP is equal, on average, to 46 lines of J2EE code.
So we can simply obtain the SLOC by multiplying our UFP value by 46:

SLOC = 162 * 46 = 7452 **KLOC** = 7,452

Scale Factor

Table 10. Scale Factor Values, SF_j, for COCOMO II Models

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

Precedentedness:

This factor is aimed to identify the grade of novelty of the software that we are developing. Since this is our first experience with this kind of project, we assign to this factor a *Very Low* rate which corresponds to a value of ***6.20***.

Development Flexibility:

In the project assignments we didn't have so much constraints about software requirements or about some specific external interface to use, but we just had the goals of the application and some conformity that had to be respect, so we assign to this factor a *Very High* rate with a consequent scale factor value of ***1.01***.

Architecture/Risk resolution:

We think say that by the architecture analysis that has been done the percentage of significant risks has been reduced by around the 60%. In fact probably some risks are still in the software but they were not considered. We assign to this point a *Nominal* rate with a value of ***4.24.***

Team Cohesion:

We have to consider that actually this is the first time that the team components work together. Considering also the various difficulties due to the different university schedules and the different cities from where the components come from, we have to assign to this point a *Low* rate, with a scale factor value of ***4.38***.

Process Maturity:

This Factor is aimed at evaluating the maturity level of the organization in the project development. As our first experience we are learning during the development all the necessary concepts concerning the software engineering, so it's difficult for us to achieve a high level of maturity. We think that a correct level could be the *Upper Level 1,* with a consequent value of **6.24**.

Cost Drivers

Product Factors

Reliability Required:

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

The nature of the service provided to the users, makes the reliability factor quite relevant. In fact let's think for example to a user that decides to make a reservation with the MyTaxiService to arrive at the airport at a precise hour of a certain day, well, if a taxi doesn't arrive at destination because of a failure of the system then the user could probably charge the MyTaxiService for the flight ticket. This makes us think that a reliability problem may cause a high financial problem, still without the possibility of human risks. We finally assign a *high rating level* with an effort multiplier of ***1.10***

DataBase Size:

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

We don't think that the database size is relevant in the estimation of the effort for our application so we assign it a *Nominal* level with an effort multiplier of ***1.00***.

Product Complexity:

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

Doing the weighted average among the 5 different areas reported in the Model Definition Document (Control Operations, Computational Operations, Device-Dependent Operations, Data Management Operations, User Interface Operations), We decided to assign a *high level rating* to the cost complexity cost. So finally we have an effort multiplier of ***1.17***.

Required Reusability:

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

We don't think that the system should have any particular constraint in terms of reusability, in fact this factor doesn't even appear in the requirements of the project. So we can set it to a *low level,* with an effort multiplier of ***0.95***.

Documentation needs:

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

Probably the effort in terms of documentation for this project has been quite high, but this is in particular due to the specific software life cycle used for the development process. The Waterfall approach in fact requires intrinsically a heavy effort in documentation than, for example, using some Agile methodologies. So at the end we can say that the number of documents produced is in line with the lifecycle needs and we can assign to this factor the *Nominal* rating level, with an effort multiplier of ***1.00***.

Platform Factors

Execution-Time Constraints:

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

We think that this parameter does not affects positively or negatively the effort so we decided to assign it the *Nominal* rating level, with an effort multiplier of ***1.00.***

Main Storage Constraints:

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

We can assume that the usage of the available storage is under the 50% so we assign it a *Nominal* rate level with an effort multiplier of ***1.00***.

Platform Volatility:

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

We think that considering the type of the application the platform deployed will not be subject to a high updating frequency, so we can assign to this factor a *Low* rate level, with a consequent multiplier of ***0.87***.

Personnel Factors

Analyst capability:

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

About this factor we have to say that this is the first time that we deal with the analysis in software development, and even if during the project we have probably increased our capability as software analysts, it has still required a higher effort than the case in which we already knew all the needed software engineering

concepts. At the end we decided to set this cost factor to a *low* rating level, with a consequent effort multiplier of ***1.19***.

Programmer capability:

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

We didn't implement the project so we think that the effort required about this factor should influence in a positive way the total effort, for this reason we assign a *Very High* rating level, with a multiplier of ***0.76***.

Personnel Continuity:

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	

We think that viewing the circumstances of the developing of our software this factor should not affect positively or negatively the final effort, so we assign it a *Nominal* rate level, with a multiplier of ***1.00***.

Application Experience:

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

Since this is our first experience with this type of application, the rating level about this factor is set to *Low* with an effort multiplier of ***1.10***.

Platform Experience:

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

Since we didn't implement the project this point has influenced neither negatively nor positively the effort, so the correct rate to assign to this factor is a *Nominal* value, with a multiplier of ***1.00***.

Language and tool Experience:

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	

About this factor we can say that most of the tools and languages used during the software lifecycle were completely new for us, for example we didn't know the alloy language used in the requirement document. Nevertheless we don't think that this has increased in a significant way the effort for the development, so we assign don't assign to this point a very low level but just a *Low* rate, with a consequent effort multiplier of ***1.09***.

Project Factors

Use of software Tools:

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

We think that the number of tools used during the project doesn't exceed the natural needs of the Waterfall life cycle, for this reason we assign to this point a *Nominal* rate level, with a multiplier of ***1.00***.

Multi-Site development:

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 communicat ion.	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

Considering that the development group is formed by three people coming from three different cities and with different time schedules for the university courses, this factor has to be under the nominal value. At the end, even though we don't have different international collocations this has probably been the one of the most significant factor in effort required for the project development, so we assign to this factor a *Very Low* rate level, with an effort multiplier of ***1.22.***

Required development schedule:

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

Considering the number of people in the software development group, the time constraints imposed by the university schedules and the time constraints concerning the project assignment, we can say, keeping in mind the considerations about the multi-site development, that at the end the project development has been accelerated with respect to the nominal time requirements. So we assign to this factor a *low* rate level, with an effort multiplier of ***1.14***.

Effort Equation

Summarizing the previous results we obtain the following tables for the scale factors and for the cost drivers:

Scale Factors	**Rating**	**Value**
Precedentedness	Very Low	6,20
Development flexibility	Very High	1,01
Architecture/Risk Resolution	Nominal	4,24
Team Cohesion	Low	4,38
Process Maturity	Low	6,24

Cost Drivers	**Rate Level**	**Effort Multiplier**
Reliability Required	High	1,10
Database Size	Nominal	1,00
Product Complexity	High	1,17
Reusability Required	Low	0,95
Documentation Needs	Nominal	1,00
Execution-Time Constraints	Nominal	1,00
Main Storage Constraints	Nominal	1,00
Platform Volatility	Low	0,87
Analyst Capability	Low	1,19
Programmer Capability	Very High	0,76
Personnel Continuity	Nominal	1,00
Application Experience	Low	1,10
Platform Experience	Nominal	1,00
Language and Tool Experience	Low	1,09
Use of Software Tools	Nominal	1,00
Multi-Site Development	Very Low	1,22
Required Development Schedule	Low	1,14

Now we have all the necessary data to apply the Effort Equation:

$$EFFORT = 2{,}94 * EAF * KLOC^{E}$$

Where:

EAF → Effort Adjustment Factor derived from Cost Drivers.

$$EAF = \prod cost\ driver = 1{,}604$$

E → exponent derived from Scale Drivers.

$$E = 0.91 + 0.01 * \sum scale\ factor = 0.91 + 0.01 * 22.07 = 1.13$$

Finally we obtain:

$$EFFORT = 2{,}94 * 1{,}604 * 7{,}452^{1{,}13} = 45\ PM$$

The result found is maybe quite high, this is probably due to the fact that the COCOMO model is based on statistical data picked up in a business environment, while in our case we are talking about an educational environment. In fact in our analysis the factors that deeply increase the required effort are the ones relative to the experience of the organization in this kind of project, so the final effort value is correct in a business environment only if we think of an organization with no previous experiences and where all the employee are students that are already studying the fundamental concepts of software engineering and that have no time to meet for the project developing.

2. TASKS AND THEIR SCHEDULE

2.1 TASKS

We have taken the main activity during the development of our project and split them into tasks in the following order :

Tasks	NAME
T1	Requirement elicitation
T2	Design
T3	DataBase Controller Impl
T4	Clients + AdminHandler Impl
T5	UserHandler + Call Impl
T6	TaxiHandler + ZoneController Impl
T7	RideManager Impl
T8	Inspection
T9	DataBase Controller + Clients + AdminHandler unit testing
T10	UserHandler + Call + TaxiHandler + ZoneController unit testing
T11	RideManager unit testing
T12	Integration Testing
T13	System Testing

The first two tasks are focused on the development part, from T3 to T7 the tasks are focused on the implementation part and from T9 to T10 are focused on the testing part of the project.

2.2 SCHEDULE

After the definition we define the effort, duration and dependencies of the various tasks.

Tasks	EFFORT	Duration	Dependencies
T1	3	2 weeks	
T2	3	2 weeks	T1
T3	1	1 week	T2
T4	2	1 week	T2
T5	1	1 week	T2,T4
T6	1	7 days	T2,T4
T7	3	2 weeks	T3,T4,T5,T6
T8	3	1 weeks	T7
T9	1	1 week	T3,T4
T10	1	1 week	T5,T6
T11	1	1 week	T7
T12	3	2 weeks	T8,T9,T10,T11
T13	3	2 weeks	T12

3. RESOURCES ALLOCATION

Here we defined the tasks of a single person in the group based on the schedule table.

The table represent the people that work over a single task, where tasks are coloured with the same colour, different from black, it means that these tasks are executed in parallel w.r.t. the dependencies:

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
Simone	X	X		X		X	X	X	X			X	X
Vincenzo	X	X		X	X		X	X		X		X	X
Matteo	X	X	X		X		X	X			X	X	X

Tasks during the time (from the 25 October 2015 to 2 February 2016)

Starting from the 25 October 2015 until 8 November 2015 for T1

Starting from the 8 November 2015 until 22 November 2015 for T2

Starting from the 22 November 2015 until 6 December 2015 for T3 and T4

Starting from the 6 December 2015 until 13 December 2015 for T5 and T6

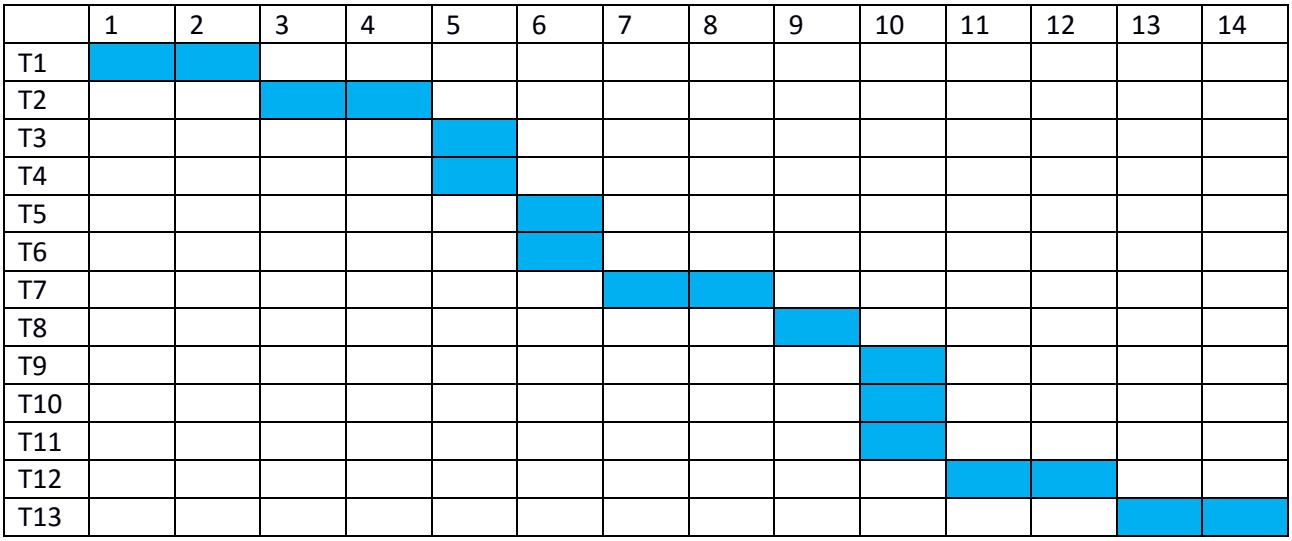
Starting from the 13 December 2015 until 27 December 2015 for T7

Starting from the 27 December 2015 until 3 January 2016 for T8

Starting from the 3 January 2016 until 10 January 2016 for T9 and T10 and T11

Starting from the 10 January 2016 until 24 January 2016 for T12

Starting from the 24 January 2016 until 2 February 2016 for T13



4. RISK OF THE PROJECT

Risks for the project and their relevance

Risk	Probability	Effects
Underestimated development time.	High	Serious
The database used cannot process enough transactions.	Moderate	Serious
Lack of knowledge or information.	High	Serious
Many members of staff are ill at critical times in the project.	Moderate	Catastrophic
It is impossible to recruit staff with the skills required.	Low	Serious
Organizational financial problems force reductions in the project budget.	Low	Serious
Faults in reusable software components have to be repaired before these components are reused.	Moderate	Serious
Changes to requirements cause design rework.	Moderate	Catastrophic
The organization is restructured	High	Serious

and a new management is responsible for the project.		

Risk management plan

Risk	Strategy
Underestimated development time.	Search buying-in components; consider the idea of using a program generator.
The database used cannot process enough transactions.	Investigate the possibility of buying a higher-performance database.
Lack of knowledge or information.	Consider the possibility of consulting an expert of the sector.
Many members of staff are ill at critical times in the project.	Consider the idea of increasing the number of the staff members; reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
It is impossible to recruit staff with the skills required.	Alert customer to potential difficulties and the possibility of delays; Search on job listing sites; investigate buying-in components.
Organizational financial problems force reductions in the project budget.	Prepare a document for the management showing how the project is making a very important role for the business and presenting reasons why cuts to the project

	budget would not be costeffective.
Faults in reusable software components have to be repaired before these components are reused.	Consider the possibility of replacing defective components with bought-in components of known reliability.
Changes to requirements cause design rework.	Derive traceability information to assess requirements change impact; maximize information hiding in the design.
The organization is restructured and a new management is responsible for the project.	Prepare a document for the management showing how the project is making a very important role for the business.

Hours of works

• Rosmini Simone: ~ 10 hours.

• Viscusi Vincenzo: ~ 10 hours.

• Zambelli Matteo: ~ 10 hours.