A.Y. 2015/2016

POLITECNICO DI MILANO

COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING 2

MyTaxiService

Project Plan Document

Authors

SARA PISANI – 854223
LEONARDO TURCHI – 853738

02/02/2016

MyTaxiService-ProjectPlan.pdf · V 1.0

Table of contents

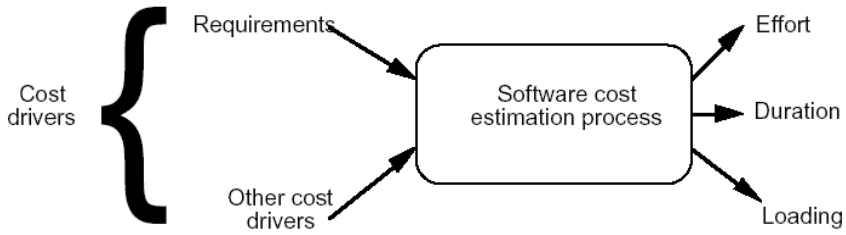
1. Preface

1.1 Scope

The *Project Plan Document* (PPD) is based on an analysis made with two different **metrics**:

- **[M1] Function Points (FP)**
- **[M2] Constructive Cost Model (COCOMO)**

The first one is used to estimate the code size; the second is used to estimate the efforts required in the development of a project.
The COCOMO is composed by taking in account: characteristics of people, products and process.
For this project is used the COCOMO II, that is better of the predecessor for the modern software cost estimation.



1.2 Problem Description

[MAIN ASSIGNMENT RECAP]
The government of a large city aims at optimizing its taxi service.
In particular, it wants to:

- simplify the access of passengers to the service
- guarantee a fair management of taxi queues.

Passengers can request a taxi either through a web application or a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call. The system guarantees a fair management of taxi queues. In particular, the city is divided in taxi zones (approximately 2 km2 each). Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones based on the GPS information it

receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone.
When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue.
Besides the specific user interfaces for passengers and taxi drivers, the system offers also programmatic interfaces to enable the development of additional services (e.g., taxi sharing) on top of the basic one.

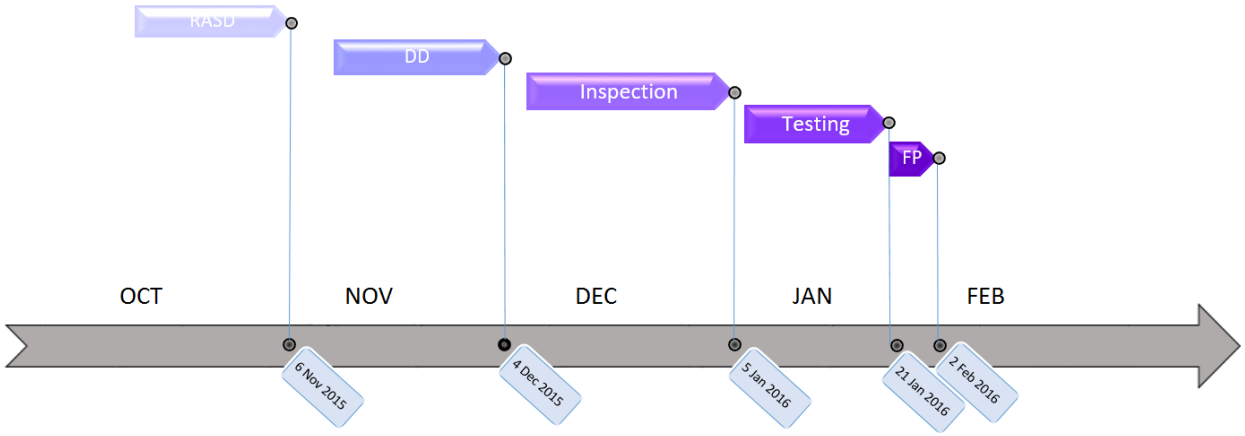
A user can reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user.

A user can enable the taxi sharing option. This means that he/she is ready to share a taxi with others if possible, thus sharing the cost of the ride. In this case the user is required to specify the destination of all rides which he/she wants to do.

2. Gantt Diagram

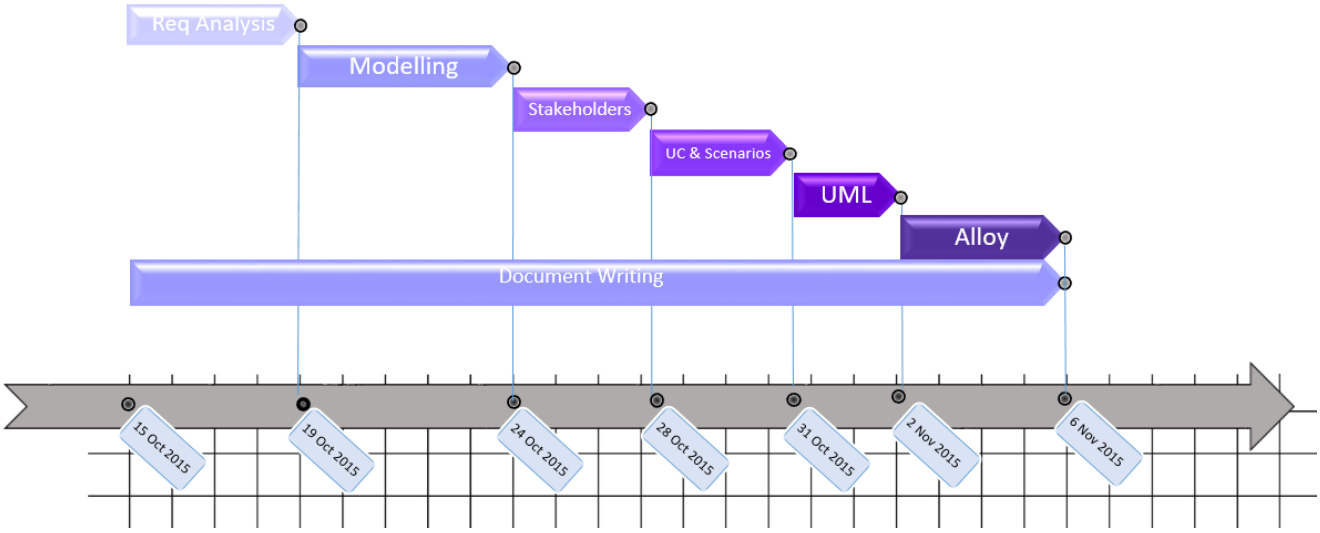
A Gantt chart is a type of bar chart, that illustrates a project schedule: here are indicated the start and finish dates of the terminal elements and summary elements of *MyTaxiService* project.

The tables aid to show all the task that have been necessary to submit each document.



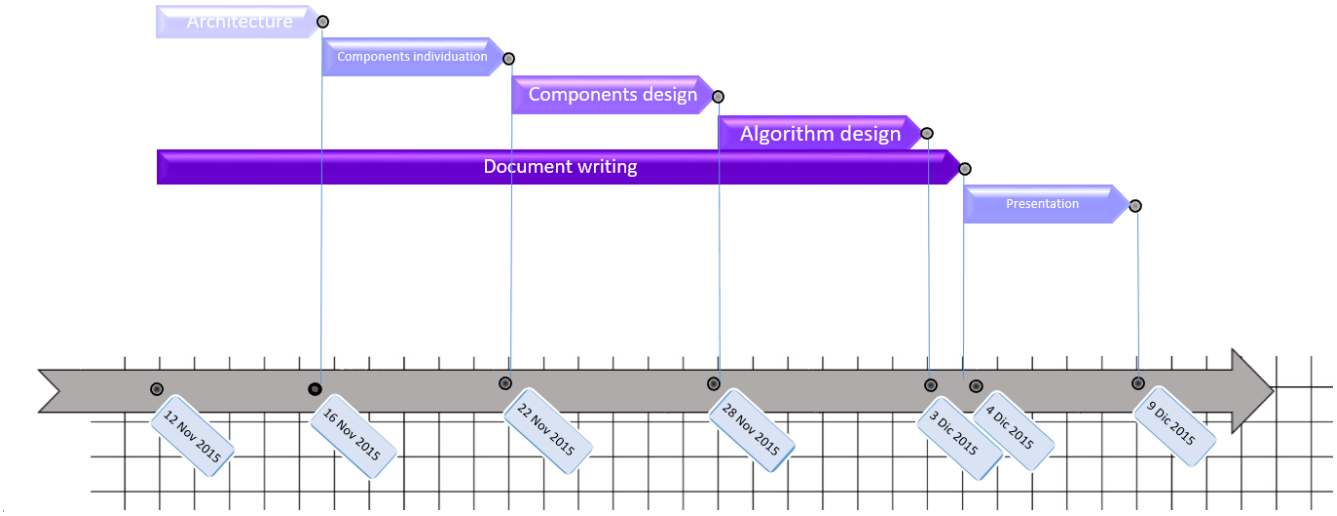
2.1.1 Gantt diagram of RASD

	RASD
Task 1.1	Requirements analysis
Task 1.2	Modelling
Task 1.3	Stakeholders individuation
Task 1.4	uses cases and scenarios
Task 1.5	UML
Task 1.6	Alloy
Task 1.7	Document writing



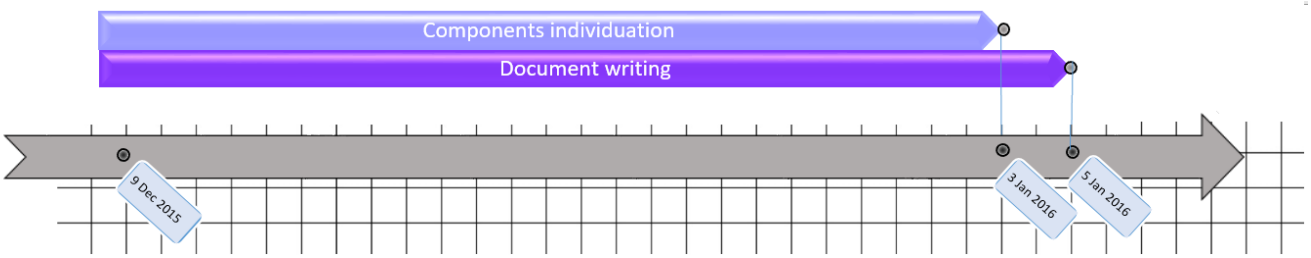
2.1.2 Gantt diagram of DD

	DD
Task 2.1	Architectural design definition
Task 2.2	Components individuation
Task 2.3	Component design
Task 2.4	Algorithm design
Task 2.5	Document writing



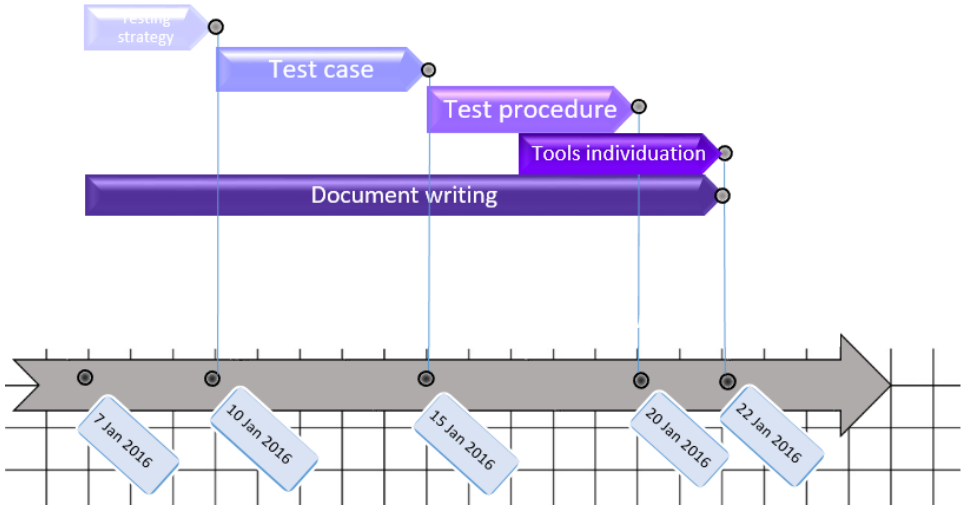
2.1.3 Gantt diagram of Inspection

	Inspection
Task 3.1	Code analysis
Task 3.2	Document writing



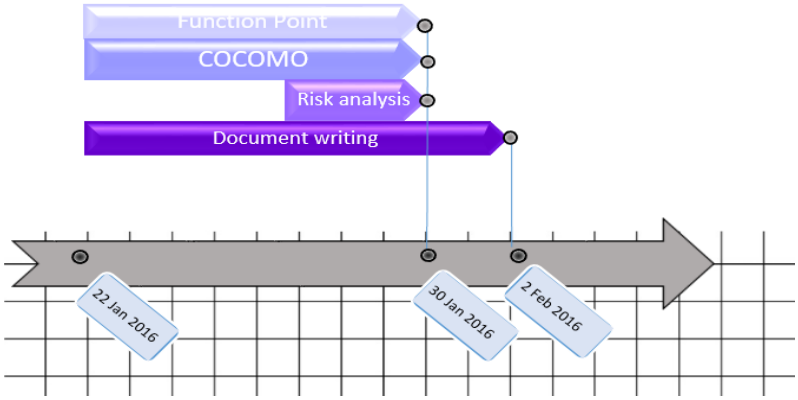
2.1.4 Gantt diagram of Test Plan

	Test Plan
Task 4.1	Testing strategy individuation
Task 4.2	Test cases individuation
Task 4.3	Procedure ideation
Task 4.4	Useful tools and software individuation
Task 4.5	Document writing



2.1.5 Gantt diagram of Project Plan

	Project Plan
Task 5.1	Function Point estimation
Task 5.2	COCOMO estimation
Task 5.3	Risk analysis
Task 5.4	Document writing



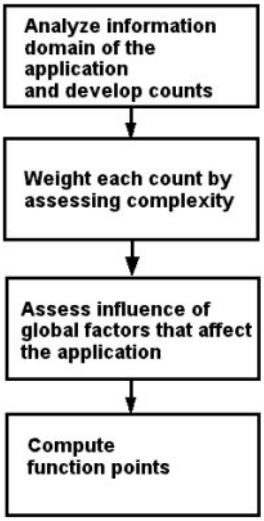
3. [M1] Function Points Estimation

3.1 Introduction

The **Function Point Estimation** is a technique to assess the effort needed to design and develop custom software applications.

The **objectives** of the analysis of function points are:

- ✓ to measure the capabilities that the user requires and receives.
- ✓ to measure the results of the development and / or maintenance of the software regardless of the used technology.
- ✓ to provide a measure that is consistent between projects and different manufacturers.



We have used this technique to evaluate the application dimension basing on the **functionalities** of the application itself, grouped in:

- **Internal Logic File:** represents a set of data handled by the system.
- **External Interface File:** represents a set of data used by the application but handled by external application.
- **External Input:** operation that allows input of data in the system.
- **External Output:** operation that creates a stream to the outside of the application. Are considered as external output, only outputs that need a mathematical formula to be produced, that create information drivated by ILFs, or that change the system behaviour.
- **External Inquiry:** operation that involves input and output operations.

To perform this estimation, we've based our parameters on the following tables, taken from COCOMO II, Model Definition Manual at:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

- [**T2-1**] Complexity Level
  (allow to classify each functionality into Low, Average and High complexity levels)

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

- [**T2-2**] Complexity Weights
  (defines the weights values that we've to use to perform the FP value)

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

- [**T2-3**] Language Gearing Factor

QSM SLOC/FP Data

Language	Average	Median	Low	High
JEE	46	49	15	67

source: http://www.qsm.com/resources/function-point-languages-table

3.2 Estimation

3.2.1 Internal Logic Files

The ILFs are used to store information about users, queues, taxis etc.
Entities as taxi, city, zone and queue have a simple structure as it is composed of a small number of fields, instead entities like passenger, driver, ride and shared ride have a more complex structure. Thus, in the first case will be adopted a simple complexity weight and average complexity weight for all other entities.

ILF	*Complexity*	*FP*
Passengers	Average	10
Drivers	Average	10
Taxis	Low	7
Cities	Low	7
Zones	Low	7
Queues	Low	7
Rides	Average	10
Shared Rides	Average	10

3.2.2 External Interface Files

The application features also only one EIF to manage the interaction with google Map APIs. This information results in only one entity with a simple structure. Thus, a simple weight will be adopted.

EIF	*Complexity*	*FP*
Map API Controller	Low	5

3.2.3 External Inputs

Login, Logout and Registration are simple operations, so a simple weight is adoptable for them.
Making a reservation, both immediate and delayed, is not a simple operation, because several entities (e.g. passenger, driver, taxi, zone etc.) are involved, thus a complex weight will be adopted.

EInputs	*Complexity*	*FP*
Registration	Low	3

Login	Low	3
Logout	Low	3
Ride Request	High	6
Ride Booking	High	6

3.2.4 External Outputs

To warn a driver that must leave the current zone to cover another area, the system must use algorithm responsible of the analysis of the taxis distribution.
For this reason the complexity grows.

EO	*Complexity*	*FP*
Migration Order	High	7

3.2.5 External Inquiries

Giving information about a reservation, both immediate and delayed, is a simple operations, even if several entities (e.g. passenger, driver, taxi, zone etc.) are involved, thus a medium weight will be adopted.

EInquiries	*Complexity*	*FP*
Booked Ride List	Average	4
Ride Details	Average	4
Ride Notification	Average	4

3.2.6 Resuming

The following table resumes our estimations:

Function Type	Value
Internal Logic Files	(4x10)+(4x7) = 68FPs
External Interface Files	(1x5) = 5FPs
External Inputs	(3x3)+(2x6) = 21FPs
External Output	(1x7) = 7FPs
External Inquiries	(3x4) = 12FPs
TOTAL	**113FPs**

Lines of Code
The total estimation value is used to get the estimation of the number of lines of code.

With the calculus

$$113\ FPs \cdot\ 46\ =\ 5198\ LOC\ =\ 5.198\ KLOC$$

will be obtained the value of the lines of code estimation for the entire project.

4. [M2] COCOMO Estimation

4.1 Introduction

The **Constructive Cost Model** (COCOMO) is an algorithmic software cost estimation model developed by *Barry W. Boehm*.
This **estimation** is achieved through a non linear model considering the characteristics of the product and also people and processes.

To perform this estimation, we've based our parameters on the following tables, taken from COCOMO II, Model Definition Manual at:
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

4.2 Scale Drivers

- [**T3-1**] 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

Where we can evaluate:

- **PREC - Precedentedness**
  Value: LOW
  Description:
  reflects the previous experience of the team, and this is our first experience.

- **FLEX - Development flexibility**
  Value: VERY HIGH
  Description:
  reflects the degree of flexibility in the development process, and the professors let us to interpret the general specifications with a quietly wide degree of freedom.

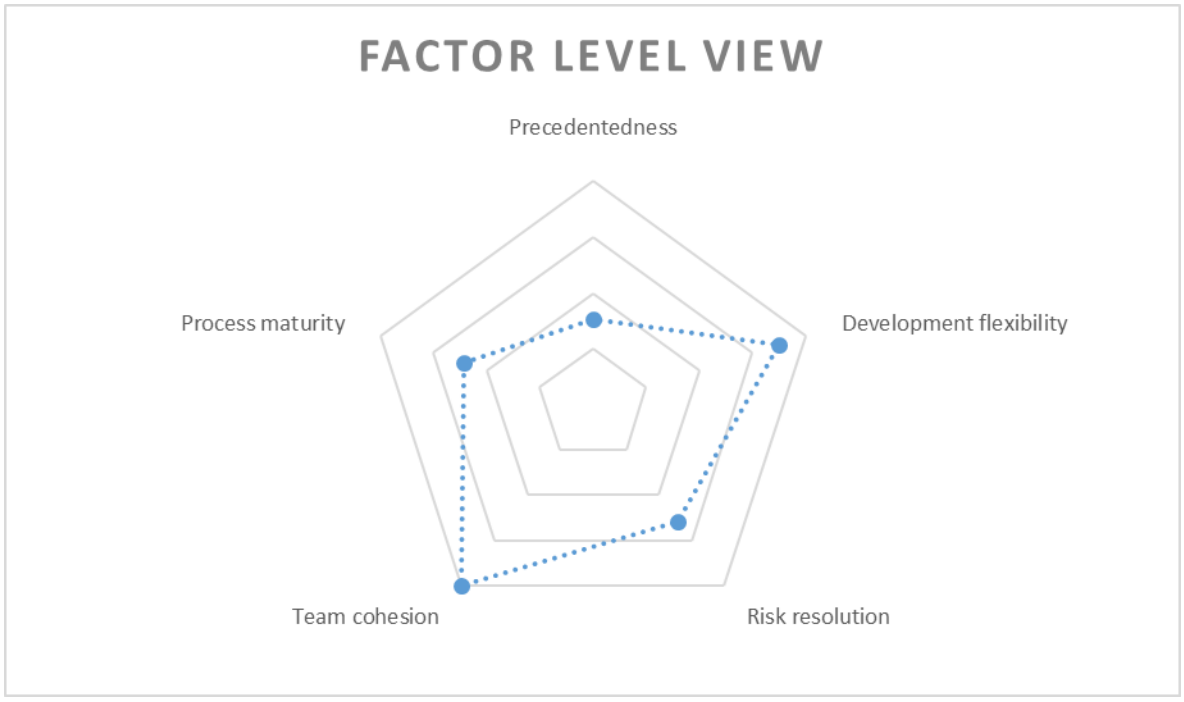
- **RESL - Risk resolution**
  Value: HIGH
  Description:
  reflects the width of the risk analysis; with the use of security access in union with a specific architecture of the system, mostly of the risks were eliminated.

- **TEAM - Team cohesion**
  Value: EXTRA HIGH
  Description:
  reflects how well the development team know each other and work together, and we had no problem to work and synchronize our documents.

- **PMAT - Process maturity**
  Value: HIGH
  Description:
  reflects the process maturity of the organization and was evaluated using the 18 Key Process Area (KPAs) questionnaire in the SEI Capability Model. We think the goals were consistently achieved.

❖ [**T3-2**] Resulting Table

SCALE DRIVER	FACTOR	VALUE
PREC - Precedentedness	Low	4.96
FLEX - Development flexibility	Very High	1.01
RESL - Risk resolution	High	2.83
TEAM - Team cohesion	Extra High	0.00
PMAT - Process maturity	High	3.12
TOTAL >		**11.92**



4.3 Cost Drivers

We can evaluate:

- **RELY - Required Software Reliability**
  Value: NOMINAL
  Description:
  Software failures don't have critical consequences and a failure may be fixed in several minutes. A failure may interest the booking procedure, and may compromise the ability to momentarily request a taxi, with a loss of profit.

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

- **DATA - Data Base Size**
  Value: NOMINAL
  Description:

  APPROXIMATE SIZE OF VARIABLES
  Reference: http://dev.mysql.com/doc/refman/5.5/en/storage-requirements.html
  - ***VARCHAR**: 32+ Bytes*
    (it depends by length of string: medium size is 150 chars)
  - ***BIGINT**: 8 Bytes*
  - ***FLOAT**: 4 Bytes*
  - ***DATETIME**: 8 Bytes*
  - *Database Tuple Header: 250 Kb*

  APPROXIMATE NUMBER OF RECORDS
  - *1 City (10 Zones) : 1 Row x 15 Columns*
  - *100 Taxis (100 Drivers) : 100 Rows x 20 Columns*
  - *1000 Users (personal data) : 1000 Rows x 10 Columns*
  - *500 Website Data : 50 Rows x 10 Columns*
  - *500 Temp Data : 250 Rows x 2 Columns*
    TOTAL = 13.015 Fields

ASSUMING AN AVERAGE OF

- *10% BIGINT*
- *5% FLOAT*
- *5% DATETIME*
- *80% VARCHAR*

We have for a TESTING DB:

$$13.015 * ((0{,}1 * 8) + (0{,}05 * 4) + (0{,}05 * 8) + (0{,}8 * 150)) + 250.000 = 251.580\ Bytes = \mathbf{251{,}6\ KBytes}$$

So for a test, an approximate database size of 250 KB could be used.
The D/P factor is:

$$\frac{250.000\ (250KB)}{5198\ (SLOC)} = \mathbf{48{,}1}$$

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

- **CPLX - Product Complexity**
  Value: HIGH
  Description:
  set to high according to the new COCOMO II CPLEX rating scale.

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

- **RUSE - Required Reusability**
  Value: NOMINAL
  Description:
  in the system there are only few reusable component (i.e. the algorithm that works both on mobile app and for webpages).

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

- **DOCU- Documentation match to life-cycle needs**
  Value: NOMINAL
  Description:
  describes the relation between the provided documentation and the application requirements. We described in the RASD and in the DD many aspects of the project, but, on the other hand, there is no part of those document unrelated to the actual phase of the development the document is addressed to.

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

- **TIME - Execution Time Constraint**
  Value: EXTRA HIGH
  Description:
  the system must be ready and available at any time.

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

- **STOR - Main Storage Constraint**
  Value: LOW
  Description:
  represents the storage needy of the system, in our case there is a storage of only simple data (no large files).

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

- **PVOL - Platform Volatility**
  Value: LOW
  Description:
  considering as platform the server, the OS and the DBMS, this coefficient establishes the necessity of periodical changes, and our platform shouldn't change too often.

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

- **ACAP - Analyst Capability**
  Value: HIGH
  Description:
  we dedicated a lot of time in the analysis of the system and its documentation and its potential integration in a real word scenario. In particular, not only we can grant that the requirements have been correctly studied and accomplished, but also that our design makes our application actually useful for an end user, providing each of the basic functionalities he may need.

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

- **PCAP - Programmer Capability**
  Value: NOMINAL
  Description:
  assuming a nominal degree of cooperation of the development team.

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

- **APEX - Application Experience**
  Value: LOW
  Description:
  evaluating this driver considering the previous experience of the team in other similar projects (as described in the 3.2 PREC driver).

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

- **PLEX - Platform Experience**
  Value: HIGH
  Description:
  our average knowledges about databases, UI and server-side development are around 2 years.

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

- **LTEX - Language and Tool Experience**
  Value: HIGH
  Description:
  same of the previous.

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	

- **PCON - Personnel continuity**
  Value: VERY LOW
  Description:
  we have less time than an half-year to complete the project.

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	

- **TOOL - Usage of Software Tools**
  Value: NOMINAL
  Description:
  Assuming the use of Eclipse or NetBeans with Maven to manage all the dependencies of the project.

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

- **SITE - Multisite development**
  Value: HIGH
  Description:
  our team is in the same city, and we work together or we synchronize our progresses through Skype or Git.

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

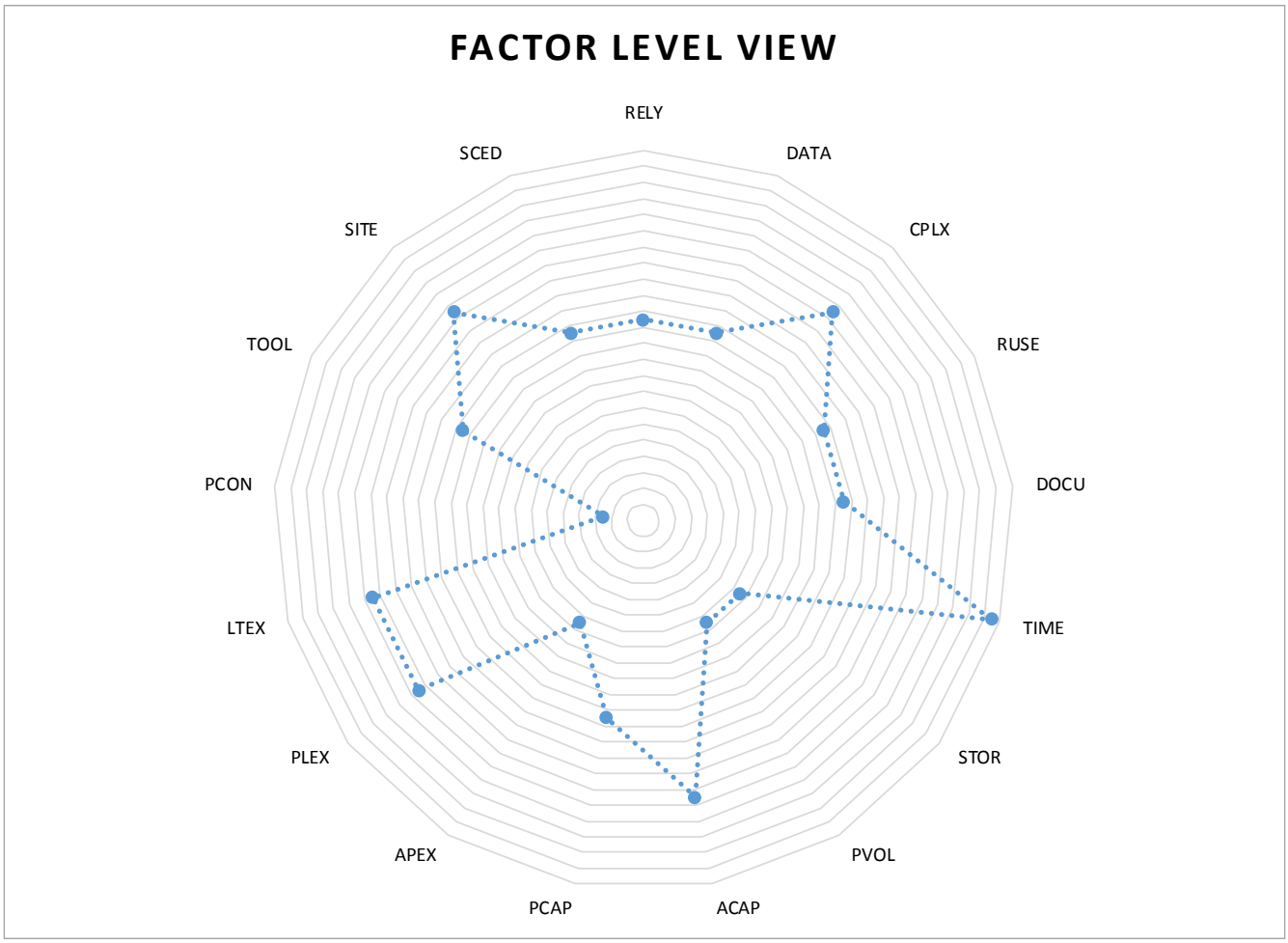
- **SCED - Required development schedule**
  Value: NOMINAL
  Description:
  assuming a distributed and balanced developing time.

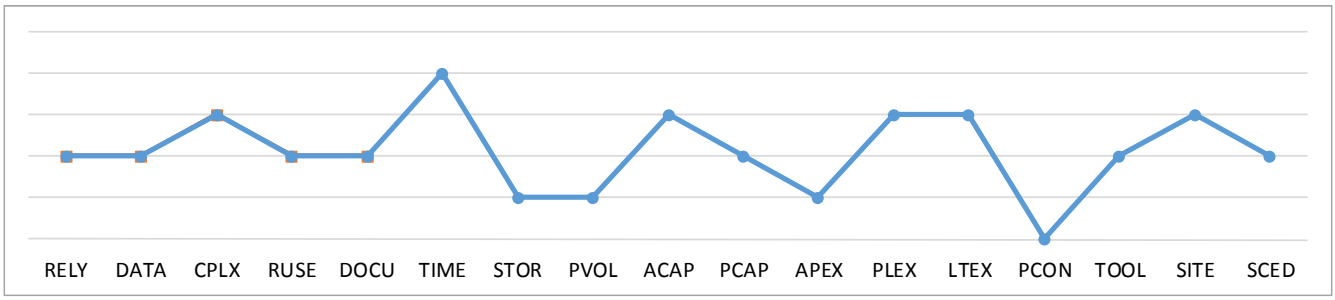
Table 34. SCED Cost Driver

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

❖ [**T3-3**] Resulting Table

SCALE DRIVER	FACTOR	VALUE
RELY - Required software reliability	Nominal	1.00
DATA - Data base size	Nominal	1.00
CPLX - Product complexity	High	1.17
RUSE - Required reusability	Nominal	1.00
DOCU - Documentation match to life-cycle needs	Nominal	1.00
TIME - Execution time constraint	Extra High	1.63
STOR - Main storage constraint	Low	-
PVOL - Platform volatility	Low	0.87
ACAP - Analyst capability	High	0.85
PCAP - Programmer capability	Nominal	1.00
APEX - Application experience	Low	1.10
PLEX - Platform experience	High	0.91
LTEX - Language and tool experience	High	0.91
PCON - Personnel continuity	Very Low	1.29
TOOL - Usage of software tools	Nominal	1.00
SITE - Multisite development	High	0.93
SCED - Required development schedule	Nominal	1.00
PRODUCT >		**1.54**


FACTOR LEVEL VIEW
RELY
DATA
CPLX
RUSE
DOCU
TIME
STOR
PVOL
ACAP
PCAP
APEX
PLEX
LTEX
PCON
TOOL
SITE
SCED


RELY DATA CPLX RUSE DOCU TIME STOR PVOL ACAP PCAP APEX PLEX LTEX PCON TOOL SITE SCED

4.4 Effort Equation

This final equation calculates the effort estimation measured in Person-Months (PM)

$$\textbf{Effort} := A * EAF * KLOC^{E}$$

Where:

- *A* → *2.94 (for COCOMO 2000)*
- *EAF* → product of all the cost drivers, equal to: **1.54**;
- *E* → exponent derived from Scale Drivers:
  $B + 0.01 * \mathbf{11.92} = 0.91 + 0.1192 = \mathbf{1.0292}$
  where B=0.91 (for COCOMO 2000)
- *KLOC* → estimated lines of code using the FP analysis: **5.198**

With this parameters, the Effort value is equal to:

$$\textbf{Effort} := 2.94 * 1.54 * 5.198^{1.0292} = \mathbf{24.69487\ PM}$$

4.5 Schedule Estimation

This equation calculates the schedule (duration) of project measured in Months

$$\textbf{\textit{Duration}} := 3.67 * Effort^{F}$$

Where:

- *Effort* → value calculated in the previous paragraph;
- F → exponent calculated as follows
  $\boldsymbol{F} = 0.28 + 0.2 * (E - B) = 0.28 + 0.2 * (1.0292 - 0.91) = \mathbf{0.30384}$

With this parameters, the Effort value is equal to:

$$\textbf{\textit{Duration}} := 3.67 * 24.69487^{0.30384} = \mathbf{9.72289\ Months}$$

This equation, however, calculates the number of people needed to complete the project:

$$Number\ of\ People := \frac{Effort}{Duration}$$

$$\boldsymbol{Number\ of\ People := \frac{24.69487\ PM}{9.72289\ Months} \simeq 2.54\ P}$$

4.6 Results' Analysis

If we look the previous **result** for the **Effort** *(= 24.69487*), is evident that it is a little oversized respect the expected one.

This is because we used, for the cost drivers, some coefficients higher than the nominal value.

- One of the **reasons** is that we set the team's cohesion as nominal (and not high or very high) because this is the first project we are working on together and we thought that is a realistic value.

- Another **reason** is that we imposed an extra high time constraint, in this case because we assumed the service as always-on one and all the resources used by components involve a large part of the entire available time of the system.
  In particular, we thought the service as scalable and, if used in several cities or with different time zones, all the server incoming traffic must be managed at every hour and without interruptions.

- Others **consequences** of this assumption are that the complexity value and the database size value are higher than the strictly required ones (this is even to correctly dispatch many request concurrently) and also this bring the Effort coefficient to an oversized value.

5. Conclusions

Below are listed the real hours of works spent for *MyTaxiService* project:

- Requirements Analysis and Specifications Document:
  – Sara Pisani: ~56 hours.
  – Leonardo Turchi: ~56 hours.

- Design Document
  – Sara Pisani: ~35 hours.
  – Leonardo Turchi: ~35 hours.

- Inspection Document
  – Sara Pisani: ~20 hours.
  – Leonardo Turchi: ~20 hours.

- Test Plan
  – Sara Pisani: ~15 hours.
  – Leonardo Turchi: ~15 hours.

- Project Plan
  – Sara Pisani: ~18 hours.
  – Leonardo Turchi: ~18 hours.

- Final Presentation
  – Sara Pisani: ~4 hours.
  – Leonardo Turchi: ~4 hours.

Our team has spent 296 hours of work to complete all phases of the project.
Assuming that, on average, the weekly hours of work are 40 is possible to calculate the amount of people that has to be monthly employed in this project:

$$\boldsymbol{Effort} := \frac{296\ hours}{40\ hours * 4\ weeks} = \mathbf{1.85\ PM}$$

6. Risk Analysis

In this section we have considered some possible issues that can happen during the software development and some possible solution to these.

RISK	MITIGATION
Team member has to leave the project	Ask the professor to downgrade the work from a two people team to a one people team
Team member is temporarily unable to work on the project	Others will cover if is a short period, and that member will have a heavier workload after his return. If too long will considered as risk 1 happened
Problems with the drafting of the documents	Ask the tutors if is a known issue or search on the Internet
Too low hours allocated, the project is late	We kept some free day before the deadlines
Github has issues	Send a mail to the prof to warn her and to find a solution

7. Hours of works

Here is the time spent for redact this document:

[sum of hours spent by team's members]
+13h (25/01)
+8h (26/01)
+4h (30/01)
+4h (31/01)
+4h (01/02)
+3h (01/02)

TOTAL ~ 36 hours

- Leonardo Turchi: ~ 18 hours
- Sara Pisani: ~ 18 hours