SE2 Project Plan Document

Edoardo Giacomello Mattia Fontana

February 2, 2016

Contents

1 Introduction

1.1 Revision History

1.2 Purpose and Scope

1.2.1 Purpose

The purpose of this document is to provide an estimation of the size, the costs and the possible risks in the development context for the MyTaxiServiceSystem.

1.2.2 Scope

The scope of this document is the planning and allocation of resources designated at the MyTaxiServicesoftware development. This document represent a starting point in cost and effort estimation.

1.3 List of Definitions and Abbreviations

FP - Function Points : Function points measure a software project by quantifying the information processing functionality associated with major external data or control input, output, or file types.

RET - Record Element Type : A RET is user recognizable sub group of data elements within an ILF or an EIF.

DET - Data Element Type A DET is a unique user recognizable, non-recursive (non-repetitive) field. A DET is information that is dynamic and not static. A dynamic field is read from a file or created from DETs contained in a FTR. Additionally, a DET can invoke transactions or can be additional information regarding transactions. If a DET is recursive then only the first occurrence of the DET is considered not every occurrence.

EI - External Input : Every unique user data or user control input type that enters the external boundary of the software system being measured

EO - External Output : Every unique user data or control output type that leaves the external boundary of the software system being measured.

ILF - Internal Logical File : Every major logical group of user data or control information that is generated, used or maintained by the software system

EIF - External Interface File : Files that are passed or shared between different software systems.

EQ - External Inquiry Every unique input-output combination, where input causes and generates an immediate output.

1.4 List of Reference Documents

- Assigment 1: Project Description
- MyTaxiServiceRequirement and Specification Analysis Document
- MyTaxiServiceDesign Document
- COCOMO Model Definition Manual:
  http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

2 Project Size and Cost Estimation

2.1 Project Size: Function Points

2.1.1 Complexity Weights

In the evaluation of the complexity weights we referred to this table:

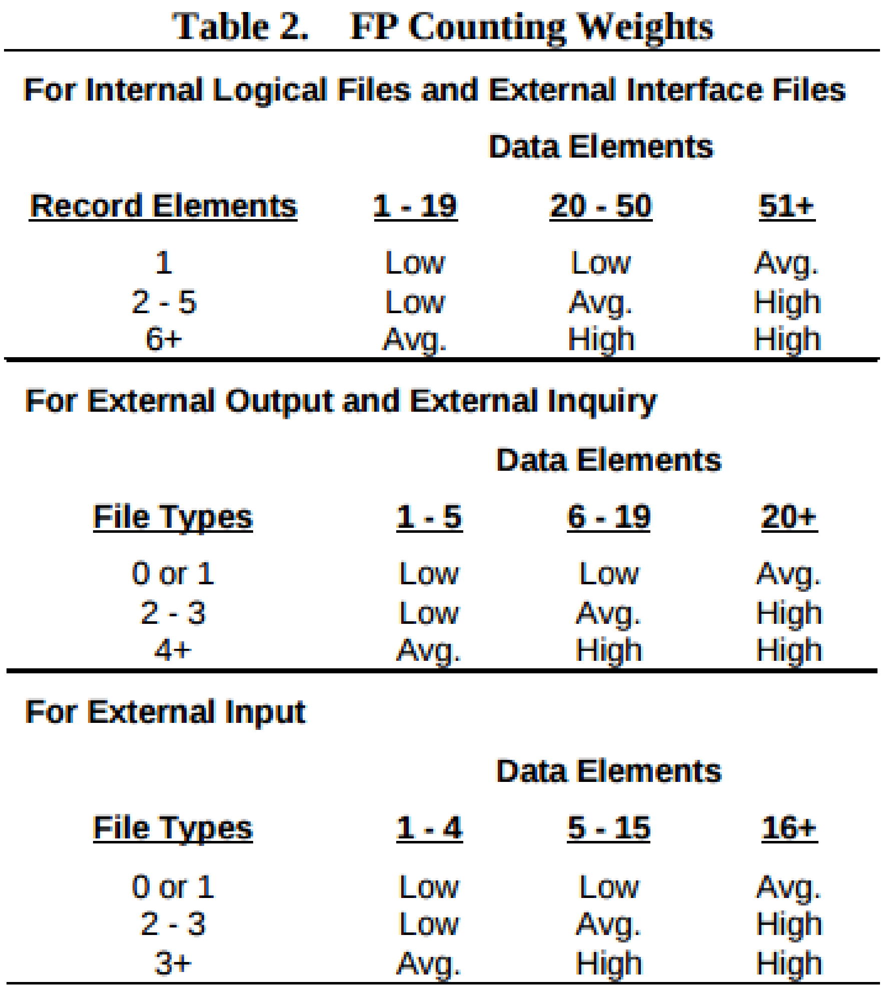


Table 2. FP Counting Weights

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

2.1.2 ILF: Internal Logical Files

The data model resides in the database, which is typically counted as a single ILF which is structured as follow:

Name	RET	DET	Complexity
Zone	1	6	Low
Location	1	3	Low
Taxi	1	5	Low
Requests and Reservations	1	8	Low
Account	1	8	Low
Logs	1	≤ 4	Low

2.1.3 EIFs (External Interface Files)

The system does not rely on files that resides on external systems

2.1.4 EIs (External Inputs)

Name	RET	DET	Complexity
Login / Logout / Registration	1: Account	$\leq$8	Low
TaxiProbe	1: Location	3	Low
TaxiReservation input and processing	2 Account, Reservation, Zones, Taxi	16	High
DriverResponse	1	3 (ReqId + TaxiId + Accepted)	Low
DriverNotification	1	3 (ReqId + TaxiId + Completed)	Low
LocationUpdate	1: Location	3	Low
BackupDatabase	n.a.		Avg
RestoreDatabase	n.a		Avg

2.1.5 EIQs (External Inquiries)

Name	**RET**	**DET**	**Complexity**
ShowProfile	1	8	Low
ShowAccounts	1	8	Low
ShowTaxiList	2	16	Avg
ShowLogs	n.a.		Avg

2.1.6 EOs (External Outputs)

Name	**RET**	**DET**	**Complexity**
TaxiProbeResponse	1	5(Taxi + Location)	Low
Confirmation	1Reservation	8	Low
Notification	1Reservation	8	Low
DriverRequest	1Reservation	8	Low

2.1.7 UFPs (Un-adjusted Function-Points)

Name	**Low**	**Avg**	**High**	**Total**
ILF	6*7	0	0	42
EIFs	0	0	0	0
EIs	5*3	2 *4	1*6	29
EIQs	2*3	2*4	0	14
EOs	4*4	0	0	16
UFP	79	16	6	**101**

2.2 Effort and Cost Estimation: COCOMO

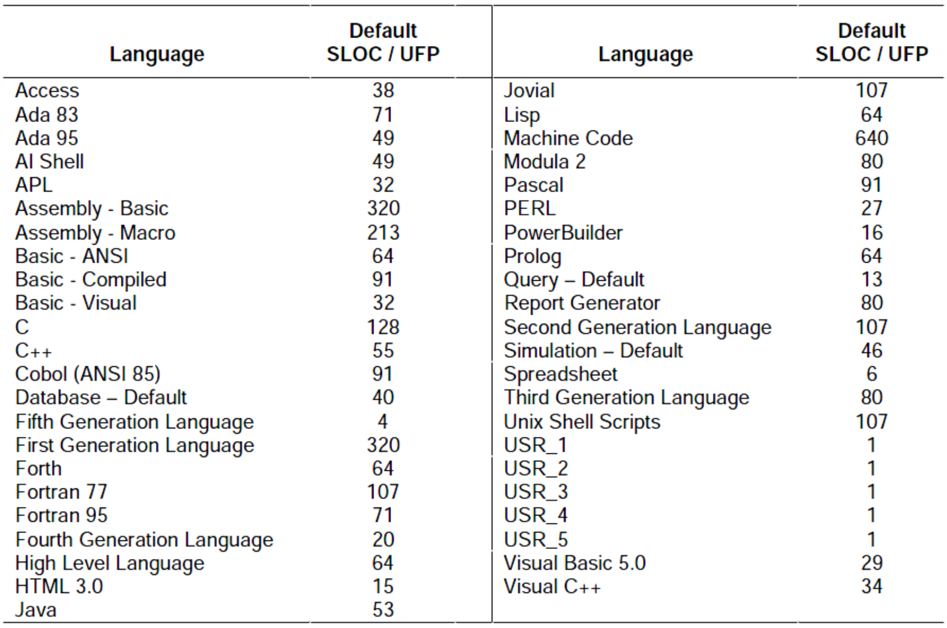
All the tables used in this analysis have been taken from COCOMO II, Model Definition Manual at:
http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf
Elements of the COCOMO II model:

- Source Lines of Code (SLOC)
- Scale Drivers
- Cost Drivers
- The Effort Equation
- The Effort Adjustment Factor
- The Schedule Equation
- The SCED (Schedule Constraints) Cost Driver

2.2.1 Source Lines of Code (SLOC)

101 FPs * 46 = 4646 SLOC
Where 46 is found from this table :

Table 4. UFP to SLOC Conversion Ratios

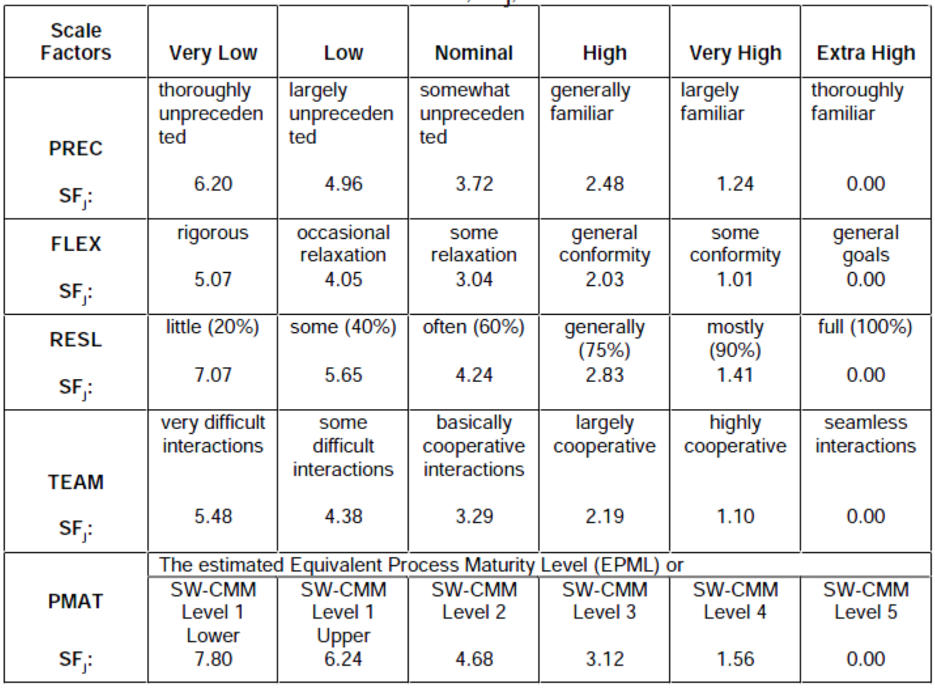


Language	Default SLOC / UFP	Language	Default SLOC / UFP
Access	38	Jovial	107
Ada 83	71	Lisp	64
Ada 95	49	Machine Code	640
AI Shell	49	Modula 2	80
APL	32	Pascal	91
Assembly - Basic	320	PERL	27
Assembly - Macro	213	PowerBuilder	16
Basic - ANSI	64	Prolog	64
Basic - Compiled	91	Query – Default	13
Basic - Visual	32	Report Generator	80
C	128	Second Generation Language	107
C++	55	Simulation – Default	46
Cobol (ANSI 85)	91	Spreadsheet	6
Database – Default	40	Third Generation Language	80
Fifth Generation Language	4	Unix Shell Scripts	107
First Generation Language	320	USR_1	1
Forth	64	USR_2	1
Fortran 77	107	USR_3	1
Fortran 95	71	USR_4	1
Fourth Generation Language	20	USR_5	1
High Level Language	64	Visual Basic 5.0	29
HTML 3.0	15	Visual C++	34
Java	53		

2.2.2 Scale Drivers

We use this table :

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

for create this results:

Scale Driver	**Factor**	**Value**
Precedentedness	Low	4.96
Development Flexibility	High	2.03
Architecture / Risk Resolution	Very High	1.41
Team Cohesion	Nominal	3.29
Process Maturity	High	3.12
Total :	/	14.81

Analisy of the results :

- Precedentedness : Low because this is the first enterprise project the team develops.
- Development Flexibility : High because only general requirements are given by the client, and the team can design the system with a large degree of flexibility.
- Architecture / Risk Resolution : Reflects the extent of risk analysis carried out, most of risks were considered in the requirements.

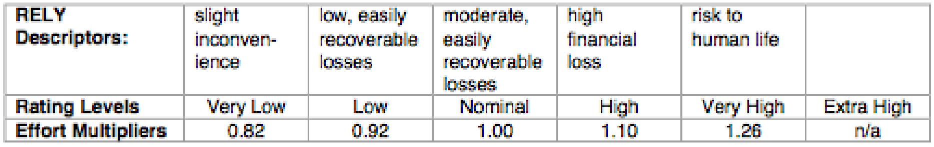
- Team Cohesion : Reflects how well the development team know each other and work together, this is Nominal because this is the first project that we do together but each team member knows the capabilities of other members.
- Process Maturity : This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model.

2.2.3 Cost Drivers

Cost Driver	Factor	Value
Required Software Reliability	Very Low	0.82
Data Base Size	High	1.14
Product Complexity	High	1.17
Required Reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	High	1.11
Main Storage Constraint	Nominal	1.00
Platform Volatility	Very Low	n/a
Analyst Capability	High	0.85
Programmer Capability	High	0.88
Application Experience	Low	1.10
Personnel continuity	Very Low	1.29
Platform Experience	Low	1.09
Language and Tool Experience	Low	1.09
Usage of Software Tools	Nominal	1.00
Multisite development	Extra High	0.80
Required development schedule	High	1.00
Total :	/	16.41/17=0.96

For defining this table we used the information contained in these tables : **Required Software Reliability :** Very Low some slight inconvenience aren't crucial.

Table 17. RELY Cost Driver

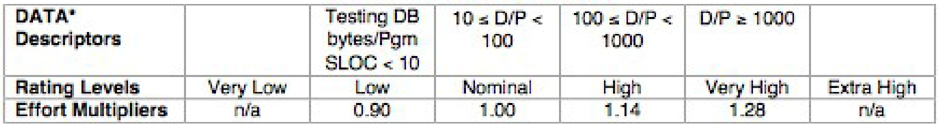


RELY Descriptors:	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

Data Base Size :
P=4646 SLOC
D= 640 KB

D/P=137.75
This result show that the value is High.

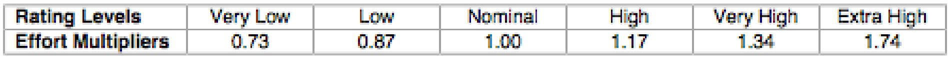
Table 18. DATA Cost Driver



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

Product Complexity: high according to the new COCOMO II CPLEX rating scale.
Control operations, computational operations,device-dependent operations, data management operations and user interface management operations are the five areas in this complexity is organized.

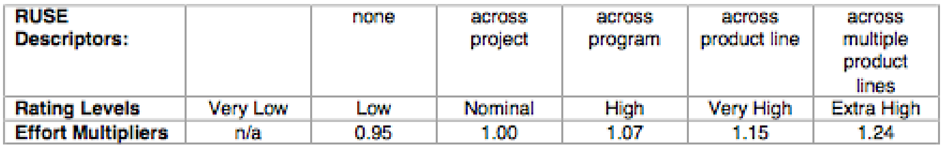
Table 20. CPLX Cost Driver



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

Required Reusability: Across program could apply to reuse across multiple financial applications projects for a single organization.

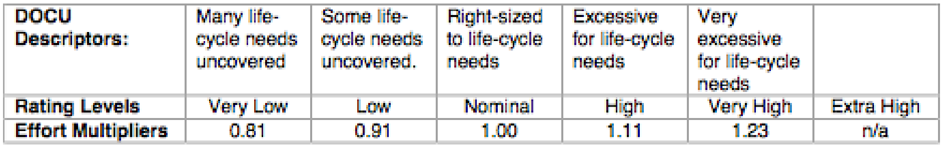
Table 21. RUSE Cost Driver



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

Documentation match to life-cycle needs: This is Nominal because we defined all information necessary to understand the system in the RASD and DD document.

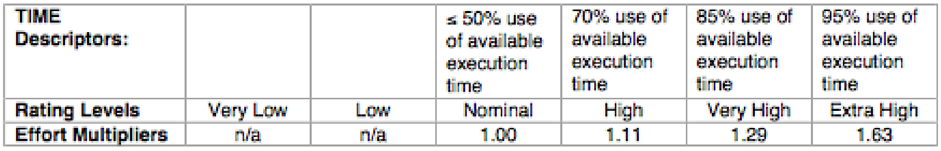
Table 22. DOCU Cost Driver



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

Execution Time Constraint: is High because the system must be available the most of the time.

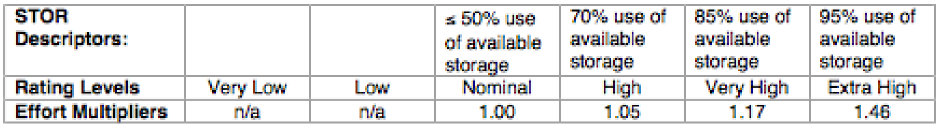
Table 23. TIME Cost Driver



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

Main Storage Constraint: is Nominal because the system doesn't use all the information storage

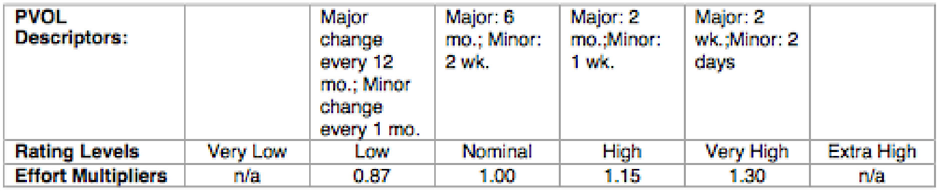
Table 24. STOR Cost Driver



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

Platform Volatility: is Very Low because the hardware and software structure don't change very often in normal conditions.

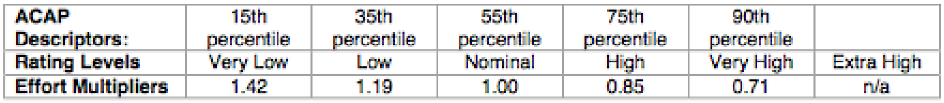
Table 25. PVOL Cost Driver



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

Analyst Capability: is High because this project is based on analysis of the problem of MytaxiService.

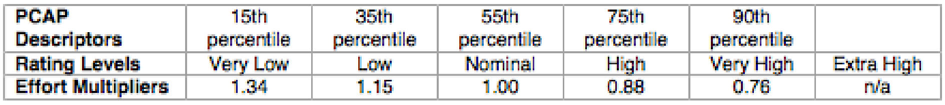
Table 26. ACAP Cost Driver



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

Programmer Capability: is High because this is an ability that is very important to communicate and collaborate.

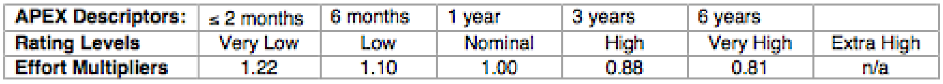
Table 27. PCAP Cost Driver



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

Application Experience: is low because we works to this project since less than 6 months.

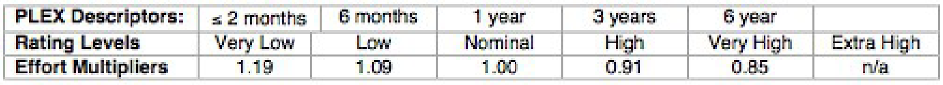
Table 29. APEX Cost Driver



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

Platform Experience: is Low because the platforms that we used in this project were new to us.

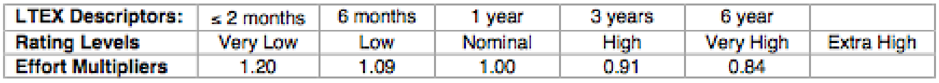
Table 30. PLEX Cost Driver



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

Language and Tool Experience: is Low because the language that tools we used in this project were new to us.

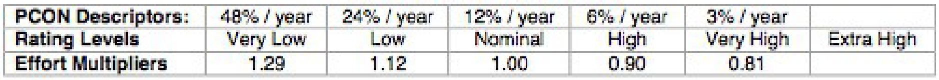
Table 31. LTEX Cost Driver



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	

Personnel continuity: is Very Low because this project cover a period of time that is very small.

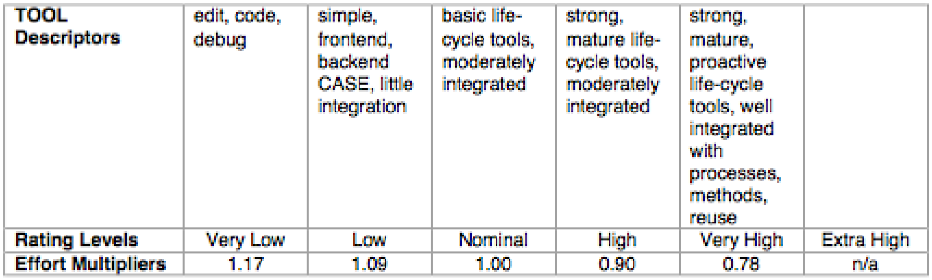
Table 28. PCON Cost Driver



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	

Usage of Software Tools: is Nominal because for creating this project we use basic tools.

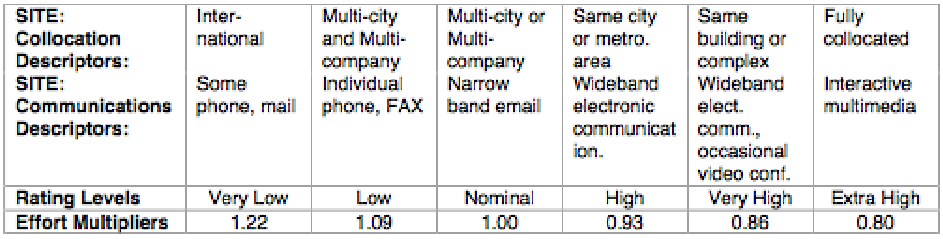
Table 32. TOOL Cost Driver



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

Multisite development: is Extra High because for doing this project we used different instruments: phone, mail and drive.

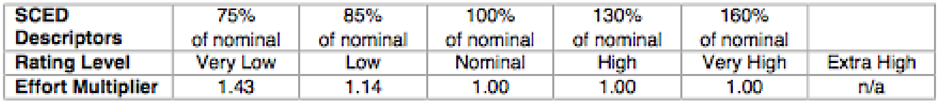
Table 33. SITE Cost Driver



SITE: Collocation Descriptors: SITE: Communications Descriptors:	Inter-national Some phone, mail	Multi-city and Multi-company Individual phone, FAX	Multi-city or Multi-company Narrow band email	Same city or metro. area Wideband electronic communication.	Same building or complex Wideband elect. comm., occasional video conf.	Fully collocated 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

Required development schedule: is High because our efforts were well distributed over the available development time in order to deliver the project in time for the deadlines.

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

2.2.4 Effort Equation

$Effort = A * EAF * (KSLOC)^{E}$
Where:

- A = 2.94
- EAF = product of all the cost drivers, equal to : 0.96 ;
- E = exponent derived from Scale Drivers. Is calculated as:
  B + 0.01 * sum{i} SF[i] := B + 0.01 *14.81 = 0.91 + 0.1481 = 1.0581;
  in which B is equal to: 0.91 for COCOMO.2000 .
- KSLOC = estimated lines of code using the FP analysis: 4.646

$Effort = 2.94 * 0.96 * 4.646^{1.0581} = 14.33689 Person - months$

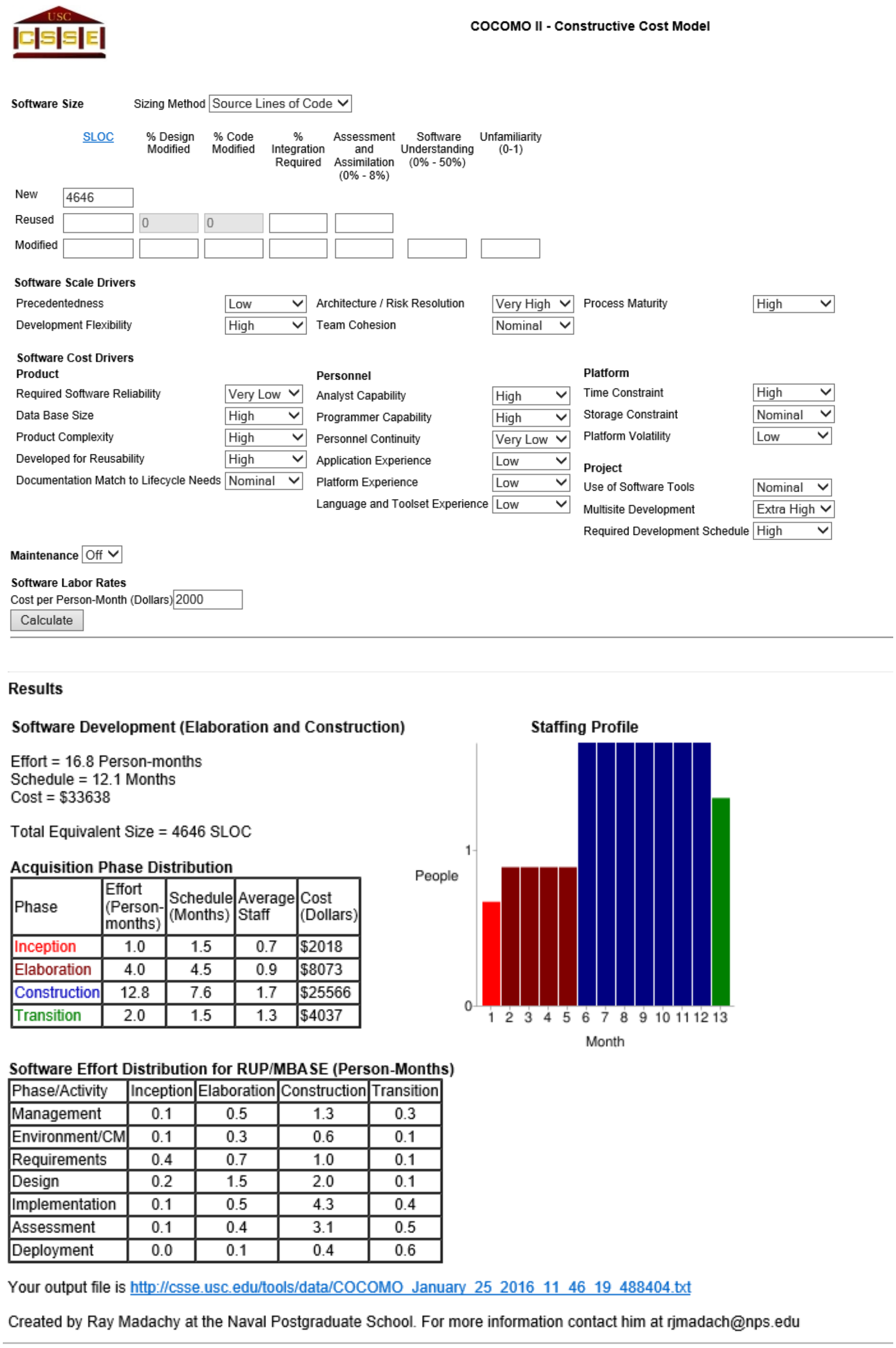
2.2.5 Schedule Estimation

$Duration := 3.67 * Effort^{F}$
Where:

- F := 0.28 + 0.2 * (E - B) = 0.28 + 0.2*(1.0581 - 0.91) = 0.30962

$Duration = 3.67 * 14.33689^{0.30962} = 8.37 = 8 Months$
As for the required number of people the estimation is:
$P = Effort/Duration = 14.33689/8 = 1.79 = 2 People$

We want to give a more precise estimation adjusting some Scale Driver. To evaluate the COCOMO II and determine the effort required to complete the software project we also use an online tool (http://csse.usc.edu/tools/COCOMOII.php). This is the result of the online tool:



COCOMO II - Constructive Cost Model

Software Size Sizing Method Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	4646						
Reused		0	0				
Modified							

Software Scale Drivers

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

Software Cost Drivers

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

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars) 2000

Calculate

Results

Software Development (Elaboration and Construction)

Effort = 16.8 Person-months
Schedule = 12.1 Months
Cost = $33638

Total Equivalent Size = 4646 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.0	1.5	0.7	$2018
Elaboration	4.0	4.5	0.9	$8073
Construction	12.8	7.6	1.7	$25566
Transition	2.0	1.5	1.3	$4037

Staffing Profile

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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.5	1.3	0.3
Environment/CM	0.1	0.3	0.6	0.1
Requirements	0.4	0.7	1.0	0.1
Design	0.2	1.5	2.0	0.1
Implementation	0.1	0.5	4.3	0.4
Assessment	0.1	0.4	3.1	0.5
Deployment	0.0	0.1	0.4	0.6

Your output file is http://csse.usc.edu/tools/data/COCOMO_January_25_2016_11_46_19_488404.txt

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

3 Tasks and Schedule

3.1 RASD :

- Identify the project objectives.
- Describe the structure and functions of the project.
- Define the specific requirements.
- Identify the use cases and scenarios.
- Representing UML diagrams.
- Representing the model in Alloy.
- Latex document.

3.2 DD :

- Describe the architecture of the system.
- Describe component view and deployment view.
- Describe runtime view and component interfaces.
- Describe algorithm design.
- Define user interface design.
- Latex document.

3.3 Planning:

- Define Function points.
- Define effort and cost estimation : Cocomo.
- Describe tasks for the project and the schedule.
- Define the risks.
- Latex document.

3.4 Development

- Web Service
- Mobile Application
- Back-end
- Database
- Documentation

3.5 Unit Testing

- Web Service
- Mobile Application
- Back-end
- Database
- Documentation

3.6 Integration Testing:

- Describe integration strategy.
- Define individual steps and test description.
- Define program stubs.
- Latex document.

3.7 Inspection:

- Define functional rules.
- Describe assignment checklist.
- Latex document.

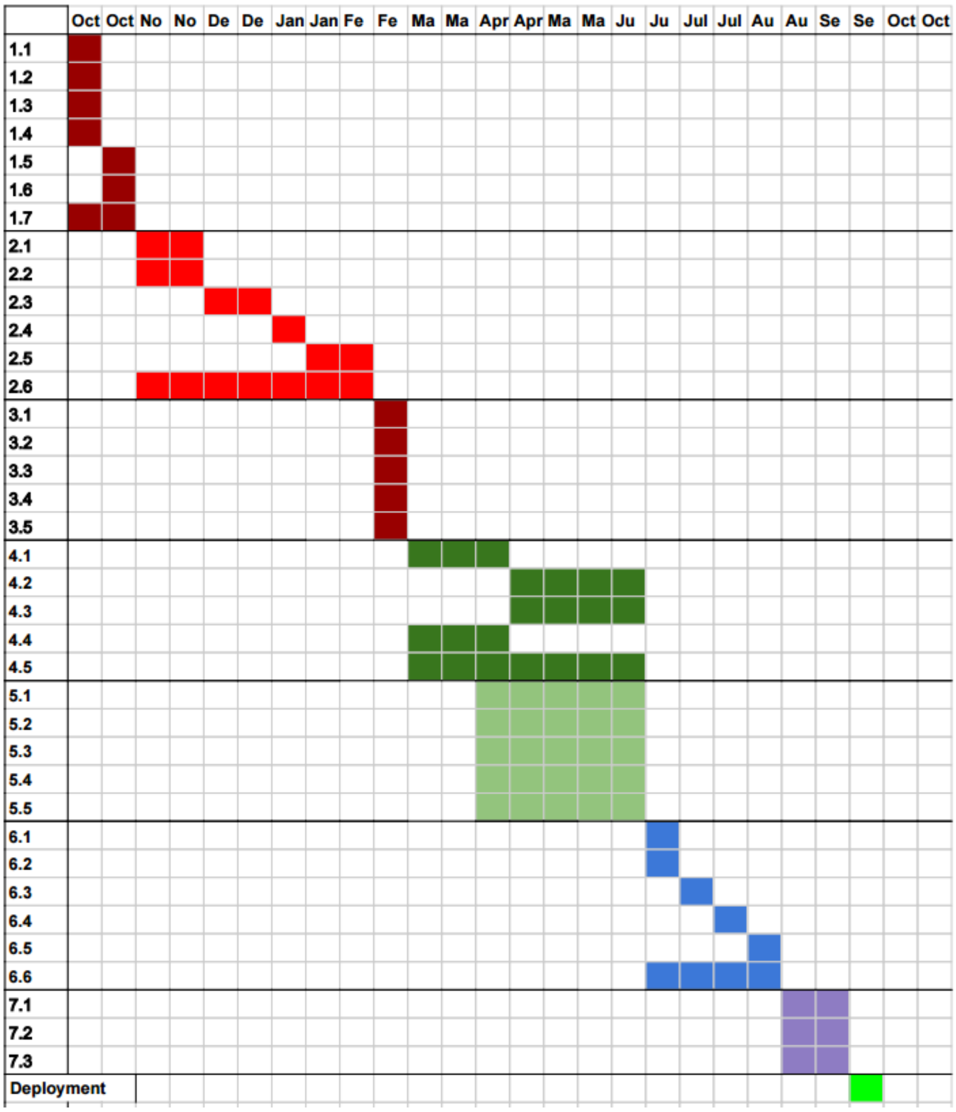
4 Resources Allocation and Schedule

Based on the results provided by COCOMO, a possible schedule has been produced.
Given the statistical nature of COCOMO, this schedule may vary during development according to the actual effort and project state.

Here is presented the resource (team members) allocation with the relative deadline for each task. If a deadline is not present it means that that task is not assigned to the team member.

Task ID	Description	Giacomello	Fontana
1.1	Identify the project objectives.	15/10/2015	15/10/2015
1.2	Describe the structure and functions of the project.	15/10/2015	
1.3	Define the specific requirements.	15/10/2015	
1.4	Identify the use cases and scenarios.		15/10/2015
1.5	Representing UML diagrams.	30/10/2015	
1.6	Representing the model in Alloy.		30/10/2015
1.7	Documentation	30/10/2015	30/10/2015
DESIGN DOCUMENT			
2.1	Describe the architecture of the system	30/11/2015	30/11/2015
2.2	Describe component view and deployment view	30/11/2015	
2.3	Describe runtime view and component interfaces		30/12/2015
2.4	Describe algorithm design	15/01/2016	
2.5	Define user interface design		15/02/2016
2.6	Documentation	15/02/2016	15/02/2016
PLANNING			
3.1	Define Function points	30/02/2016	
3.2	Define Cocomo		30/02/2016
3.3	Describe task for the project and the schedule	30/02/2016	30/02/2016
3.4	Define the risks	30/02/2016	30/02/2016
3.5	Documentation	30/02/2016	30/02/2016

DEVELOPMENT			
4.1	Web Service		15/04/2016
4.2	Mobile Application		30/04/2016
4.3	Backend	30/04/2016	
4.4	Database	15/04/2016	
4.5	Documentation	30/04/2016	30/04/2016
UNIT TESTING			
5.1	Web Service	30/04/2016	
5.2	Mobile Application	30/04/2016	
5.3	Backend		30/04/2016
5.4	Database		30/04/2016
5.5	Documentation	30/04/2016	30/04/2016
INTEGRATION TESTING			
6.1	Component Stubs	30/06/2016	30/06/2016
6.2	Phase 1	30/06/2016	
6.3	Phase 2		15/07/2016
6.4	Phase 3	30/07/2016	
6.5	Phase 4		15/08/2016
6.6	Documentation	15/08/2016	15/08/2016
INSPECTION DOCUMENT			
7.1	Define functional rules	15/09/2016	
7.2	Describe assignment check list		15/09/2016
7.3	Documentation	15/09/2016	15/09/2016



	Oct	Oct	No	No	De	De	Jan	Jan	Fe	Fe	Ma	Ma	Apr	Apr	Ma	Ma	Ju	Ju	Jul	Jul	Au	Au	Se	Se	Oct	Oct
1.1	■																									
1.2	■																									
1.3	■																									
1.4	■																									
1.5		■																								
1.6		■																								
1.7	■	■																								
2.1			■	■																						
2.2			■	■																						
2.3					■	■																				
2.4							■																			
2.5								■	■																	
2.6			■	■	■	■	■	■	■																	
3.1										■																
3.2										■																
3.3										■																
3.4										■																
3.5										■																
4.1											■	■	■													
4.2														■	■	■	■									
4.3														■	■	■	■									
4.4											■	■	■													
4.5											■	■	■	■	■	■	■									
5.1													■	■	■	■	■									
5.2													■	■	■	■	■									
5.3													■	■	■	■	■									
5.4													■	■	■	■	■									
5.5													■	■	■	■	■									
6.1																		■								
6.2																		■								
6.3																			■							
6.4																				■						
6.5																					■					
6.6																		■	■	■	■					
7.1																						■	■			
7.2																						■	■			
7.3																						■	■			
Deployment																								■		

4.1 Considerations

Since the actual project has not exactly followed this particular schedule, this is to be intended as a simulation of a real situation in which the system would have really developed.
Anyway, in a scenario in which our team had to actually develop the system, it is possible to notice that the theoretical schedule fits the one we followed quite well.
In particular, the develop process should begin on march, since more time has to be allocated to the design document redaction process.
This choice has been made because of the lack of some details in the design document we provided in the given time; in this way the team could spend the month of February to enrich the Design Document while getting acknowledgement from the stakeholders.
Moreover, we consider the development period as quite appropriate to develop the system given its complexity, provided that the developers periodically write and runs unit-tests while writing components.

5 Risks and Management

Risk	Severity	Possible Resolution
Inadequate Infrastructures	High	The client should be informed about the infrastructure costs and the device requirements and an agreement has to be reached.
Frequent Data loss	High	A system administrator should be assigned to investigate the problem. Frequent database backups should be performed anyway.
Excessive Average System Load	High	A code inspection could be performed for analysing the complexity of the code. It it's not sufficient the managers could consider a system upgrade or the implementation of a distributed system, depending on the costs. A cloud hosting is also a feasible possibility.
The system security is proven to be not sufficient	High	A security specialist have to join the team in order to improve overall security of the system
Malfunction of any device used to communicate with the system.	Moderate	Periodic maintenance should be performed on the system infrastructure. If a taxi driver terminal is not functional, it has to be replaced
JEE platform malfunctions	Moderate	The platform has to be updated
Faults in reusable software components have to be repaired before these components are reused.	Moderate	A team will be dedicated to repair the malfunctioning components before reusing.
The database used in the system cannot process as many transactions per second as expected.	Moderate	A system upgrade should be considered, the cost analysis should consider a possible increase in the number of users and requests.

Improper use of application - i.e. Taxi Drivers interact with the application while driving	Moderate	User Interface improvement - It could be possible to add a control that prevents the users to interact with the application while the device is in movement
Delays in project development	Moderate	The schedule could be reorganised to better fit the team members necessities. The precedence could be given to the core functionalities and testing, while the code and user interface inspection could be delayed.
Low Web Interface quality	Moderate	A web specialist could be enrolled in order to improve the interface quality
The organization is restructured so that different managers are responsible for the project.	Low	The responsibles have to be informed about the current state of the project. This documentation has to be kept updated constantly
Personnel Adaptation	Low	For taxi driver users the interface has been kept as simple as possible, so it is difficult to have issues in migrating to the new system. System Administrators should be instructed to accomplish their new tasks as the system is delivered, although the process should not require much time
A new feature is requested	Low	By design it is possible to add a new component to improve the system functionalities
Changes in the user interface.	Low	The modularity of this system should allow modifications on system interface with no or small modification of the system backend, depending on the nature of the modification.

6 Work Hours

- **Edoardo Giacomello**: 23 hours
- **Mattia Fontana**: 18,5 hours