Politecnico di Milano

Project Plan Document

"myTaxiService"

Nicolas Tagliabue (matr. 853097), Matteo Pagliari (matr. 854353)

February 2, 2016

Versione 1.0

Contents

1 Introduction

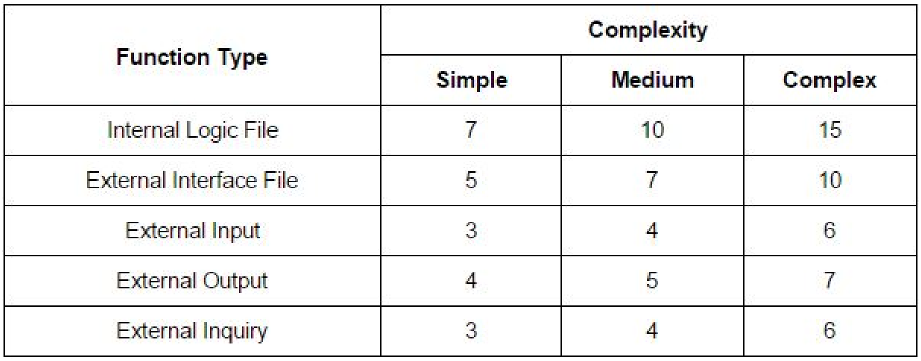
The aim of this document is to evaluate the resources needed to develop the myTaxiService project, mainly human and time resources. In order to do it, Functional Point analysis and the COCOMO model were used.

2 Function Points Analysis

Function points is "*a technique to evaluate the effort needed to design and develop custom software applications*". This method is used to do a precise prevision about the dimension of the application starting from the functionalities of software. The functionalities to be evaluated are listed into the RASD document. For each functionality, it will be calculated the complexity of the realization. The functionality are grouped in:

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

We can estimate the Functional points of our application following the table below that contains for each functionality the relative complexity.



Function Type	Complexity		
	Simple	Medium	Complex
Internal Logic File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

The number of FPs can be computed as the weighted sum of function types using the coefficient of the table above. We perform the calculation step by step:

Internal Logic File The application has stored information about taxi drivers, customers, taxi cabs, zones, addresses, request and reservation. Zones, addresses and taxi cabs have a simple structure and they have a small number of data fields. Taxi drivers, reservations and notification have a lot of fields and we decided to declare them as complex. Indeed, customer and reservations have a quite number of fields and for them we decided to adopt medium weights. Concluding ILFs, the result of this part is 3*7+2*10+3*15=87 FPs.

External Interface File This feature of the application has to interact also with Google Maps to visualize the positions of a route. To manage this functionality only one component of the applications is sufficient, hence we gave this part a simple weight. As a result we get 5 FPs.

External inputs Signup for TaxiDriver and Customer: this a simple operation, so we can choose a simple weight: 1*3=3 FPs.
Login/logout: these are simple operations, so we can choose a simple weight: 2*3=6 Fps.
Request: this is complex operation that involves several components and entities: Customer, Request, Address, Zone, TaxiDriver, so we can adopt a complex weight: 1*6=6FPs.
Reservation: this is complex operation that involves several components and entities: Customer, Request, Address, Zone, TaxiDriver, so we can adopt a complex weight: 1*6=6 FPs.
Modify reservation: this is an operation that involves a medium number of entities: Customer and Reservation but it's not a very difficult operation, so we can choose a medium weight: 1*4=4 FPs.
Cancel reservation: this is an operation that involves two entities, Customer and Reservation, and so we can adopt a medium weight: 1*4=4FPs.
Inform the system about availability: this is a simple operation that involves only one big data structure, TaxiDriver, so we can choose a simple weight: 1*3=3 Fps.
Confirm the system request/reservation call: this operation involves two entities, TaxiDriver and Request but it's not a very easy operation, so we can adopt a medium weight: 1*4=4 FPs.

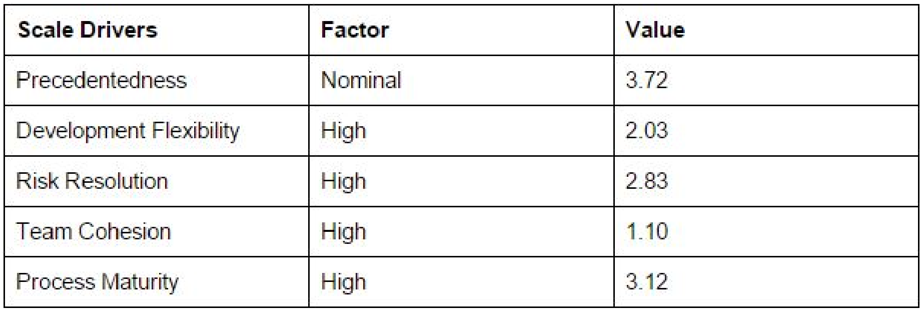
External Output After login, the application informs the customer about the found taxi: the operation involves 4 entities: Customer, TaxiDriver, Request/Reservation and Notification; so we can choose a complex weight: 1*7=7 FPs.
After login, the application informs a taxi driver about a request call: the operation involves 4 entities: Customer, TaxiDriver, Request/Reservation and Notification; so we can adopt a complex weight: 1*7=7 FPs.

External Inquiries The application allow customers to request informations about his/her reservations: this operation involves only Customer and Reserva-

tion, and it is a simple operations. We decided to give a simple weight: 1*3=3 FPs.
Total FP number: for our application, we obtained the following value of the unadjusted FPs: 142.

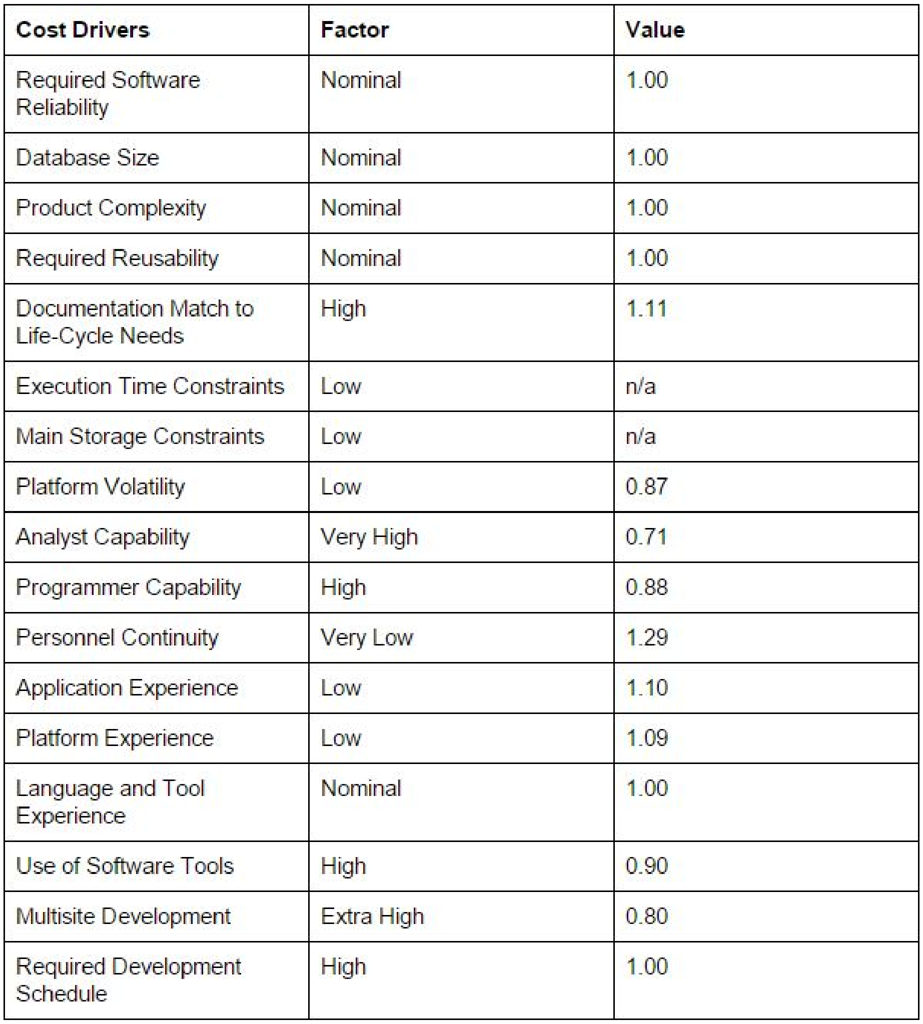
SLOC (Source Line of Code) = 142*46=6532

3 COCOMO



Scale Drivers	Factor	Value
Precedentedness	Nominal	3.72
Development Flexibility	High	2.03
Risk Resolution	High	2.83
Team Cohesion	High	1.10
Process Maturity	High	3.12

Precedentedness: it reflects the previous experience of the organisation with this type of project. Very low means no previous experience, Extra high means that the organisation is completely familiar with this application domain.
Development Flexibility: it reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals.
Risk Resolution: it reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.
Team Cohesion: it reflects how well the development team know each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems.
Process Maturity: it reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level from 5.



Cost Drivers	Factor	Value
Required Software Reliability	Nominal	1.00
Database Size	Nominal	1.00
Product Complexity	Nominal	1.00
Required Reusability	Nominal	1.00
Documentation Match to Life-Cycle Needs	High	1.11
Execution Time Constraints	Low	n/a
Main Storage Constraints	Low	n/a
Platform Volatility	Low	0.87
Analyst Capability	Very High	0.71
Programmer Capability	High	0.88
Personnel Continuity	Very Low	1.29
Application Experience	Low	1.10
Platform Experience	Low	1.09
Language and Tool Experience	Nominal	1.00
Use of Software Tools	High	0.90
Multisite Development	Extra High	0.80
Required Development Schedule	High	1.00

This final equation gives us the effort estimation measured in Person-Months:

Effort = A*EAF*(KSLOK)^{E}

Where :

A = 2.94, EAF = 0.67, E = 1.0298

Effort = 2.94*0.67*(6.532)^1.0298 = 13.6068 PM Person/Month

E = B + 0.01 * sum{i} SF[i] =
= 0.91 + 0.01 * sum{i} SF[i] = 0.91 + 0.01*11.98 = 1.0298

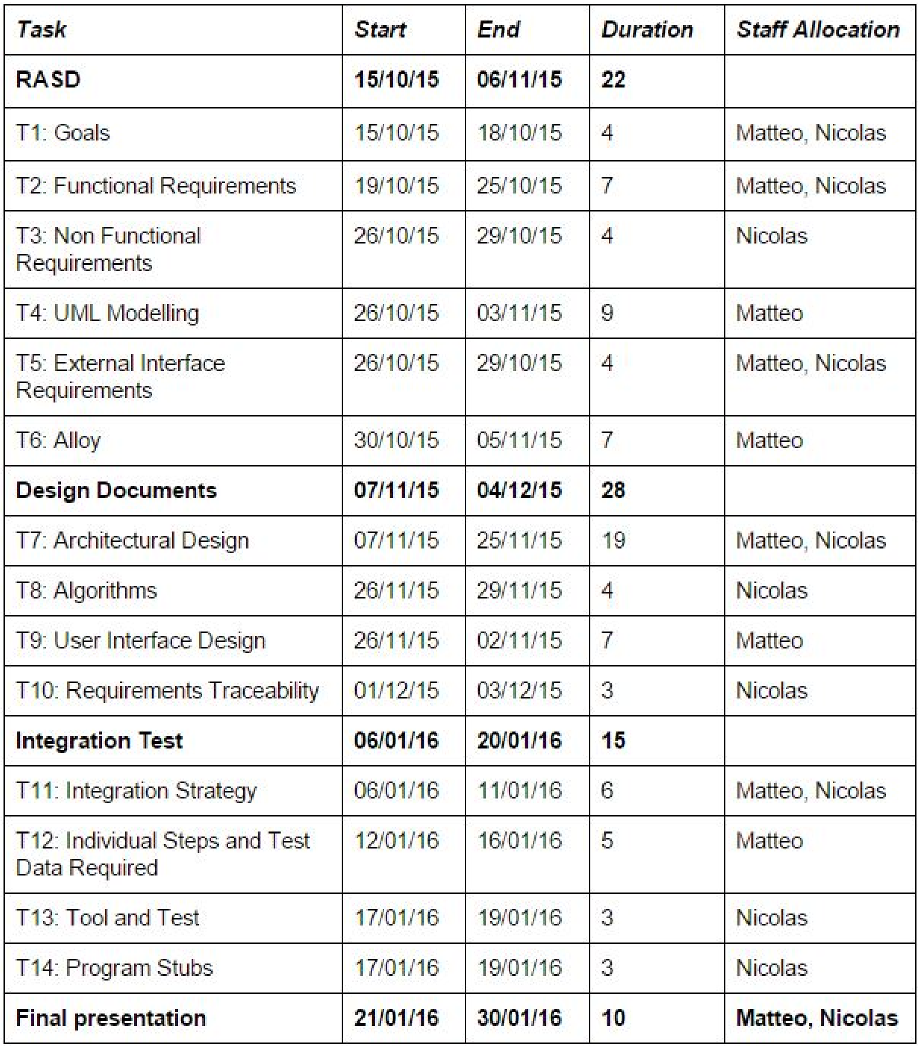
Duration = 3.67 * (effort)^F = 3.67 * (13.6068)^0.3040 = 8.1157 Months

F= 0.28 + 0.2*(E-B) = 0.28 + 0.2*(1.0298-0.91) = 0.3040

Npeople = effort/duration = 13.6068/8.1157 = 1.68 circa 2 People

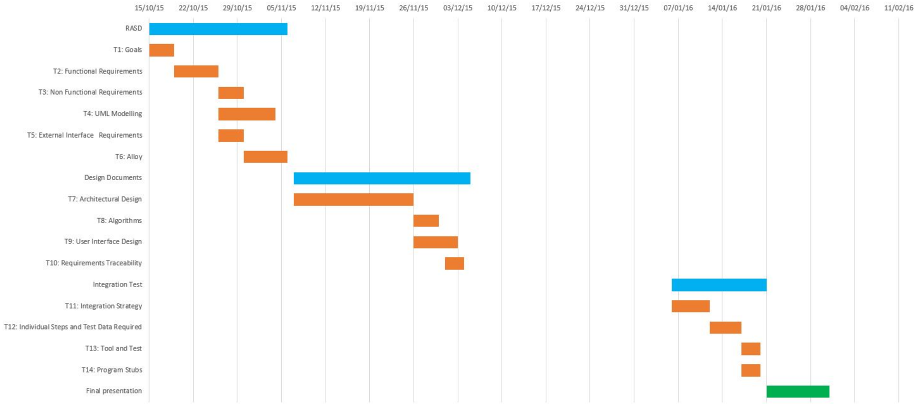
4 Resource Allocation

Here is reported how we divided the work between the two of us:



Task	***Start***	***End***	***Duration***	***Staff Allocation***
RASD	**15/10/15**	**06/11/15**	**22**	
T1: Goals	15/10/15	18/10/15	4	Matteo, Nicolas
T2: Functional Requirements	19/10/15	25/10/15	7	Matteo, Nicolas
T3: Non Functional Requirements	26/10/15	29/10/15	4	Nicolas
T4: UML Modelling	26/10/15	03/11/15	9	Matteo
T5: External Interface Requirements	26/10/15	29/10/15	4	Matteo, Nicolas
T6: Alloy	30/10/15	05/11/15	7	Matteo
Design Documents	**07/11/15**	**04/12/15**	**28**	
T7: Architectural Design	07/11/15	25/11/15	19	Matteo, Nicolas
T8: Algorithms	26/11/15	29/11/15	4	Nicolas
T9: User Interface Design	26/11/15	02/11/15	7	Matteo
T10: Requirements Traceability	01/12/15	03/12/15	3	Nicolas
Integration Test	**06/01/16**	**20/01/16**	**15**	
T11: Integration Strategy	06/01/16	11/01/16	6	Matteo, Nicolas
T12: Individual Steps and Test Data Required	12/01/16	16/01/16	5	Matteo
T13: Tool and Test	17/01/16	19/01/16	3	Nicolas
T14: Program Stubs	17/01/16	19/01/16	3	Nicolas
Final presentation	**21/01/16**	**30/01/16**	**10**	**Matteo, Nicolas**

and here is Gantt chart:



5 Risks

Risks in software project can be divided in these following categories:

Schedule risks could be caused by wrong estimation of the development time but also from a not optimal allocation of human resources. Another possible cause is underestimating the complexity of some tasks, thus running out of time. This kind of risks could lead to project failure and economical repercussions for the company. Since this problems are strongly related to the initial project analysis, one possible recovery solution could be an overhaul of the initial analysis and the increase of amount of work\hour per person.

Budget risks could be caused by wrong estimation of the costs, leaving you with no funds available, but also by some unexpected expenses that could come up during project development. One possible solution could be the search for new investor or some kind of crowdfunding (eg Kickstarter, Indiegogo, ecc).

Operational risks could be caused by: unexpected illness during critical development phases, lower than expected skills of some of the members, lack of communication, project could be too complex and failing to fulfill personal responsibilities. For the communication problem one solution is to use better means to communicate, while for the lack of skills it could be necessary to reallocate the person to different tasks that better reflect his\her professional abilities.

Programmatic risks are those that are beyond control, such as a sudden Government law change. Unfortunately there's little you can do about this issue.