Politecnico di Milano

Software Engineering II

myTaxiService
Project Plan

Authors
Giovanni Beri 852984
Biagio Festa 841988

Monday 25th January, 2016

Contents

1 FP Estimation

1.1 Brief Introduction

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

Figure 1: Table 2.

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

Figure 2: Table 3.

1.2 Internal Logic Files

The system includes some ILF, used to store persistent information about the user accounts information, the realized *RideReservations*, and the geographical distribution of the *TaxiZones*. We are now going to analyze more specifically, for each of them, what data are going to be actually stored:

- **Users Account**: 5 different tables are expected, listed below:
  - **Account myTaxiDriver**: ID (key), email address, encrypted password, name, surname, taxi license information, isAccepted (boolean).
  - **Account myTaxiUSer**: ID (key), email address, encrypted password.
  - **Fake Requests**: signaled *myTaxiUser* ID, signaler *myTaxiDriver* ID, date of the *fake request*.

  - **Account Admin**: ID (key), email address, encrypted password.
  - **App Keys**: ID (key), linked email address, encrypted password.

- ***RideReservation***: a single table is used, which fields are: ID of sender *myTaxiUser*, time, GPS start location, GPS end location.

- ***TaxiZones***: a single table is used, which fields are: ID (key), and a BLOB (*i.e.* Binary Large OBject) type containing information about geographical distribution.

ILF	**Complexity**	**FP**
User Account	Avg	10
RideReservation	Low	7
TaxiZones	Low	7

1.3 External Interface Files

Our system, in order to accomplish its goals, manages data generated by some external application; we list below those external application, and the data handles by *myTaxiService*:

- **Google Maps API**: data necessary to renderize maps of the desired areas and locations.

- **Google Places API**: data concerning the presence and location of remarkable places in the proximity of a *myTaxiUser*.

EIF	**Complexity**	**FP**
Google Maps API	Low	5
Google Places API	Low	5

1.4 External Input

This *Transactional Function* point allows a user to maintain Internal Logical Files (*ILFs*) through the ability to add, change and delete the data.

We have identified, between the functional requirements listed in the *RASD* document, the following functionalities that we have labelled as External Input.

- Registration to *myTaxiService* as *myTaxiUser*:

- Send request in order to become *myTaxiDriver*

- Login into registered user's account

- Update account data

- Send *RideRequest* manually inserting the desired address

- Send *RideRequest* selecting the address from a list of points of interest

- Cancel *RideRequest* or *RideReservation*

- Send *RideReservation*

- Accept or decline incoming *RideRequest*
- Cancel confirmation of already accepted ride
- Signal *myTaxiDriver* state
- Confirm a valid ride
- Create administrator account
- Activate *myTaxiDrivers* accounts

EI	**Complexity**	**FP**
myTaxiUser registration	Low	3
New *myTaxiDriver* request	Low	3
Login	Low	3
Update Account Data	Low	3
RideRequest with manual address insertion	Low	3
RideRequest with selection of the address	Low	3
Cancel of a ride	Low	3
Send *RideReservation*	Avg.	4
Accept or decline incoming request	Low	3
Cancel an accepted ride	Low	3
Signal state	Low	3
Confirm a ride	Low	3
Create administrator account	Low	3
Activate *myTAxiDrivers* accounts	Low	3

1.5 External Output

That *Transactional Function* point gives the user the ability to produce outputs. The results displayed are derived using data that is maintained and data that is referenced. In function point terminology the resulting display is called an *External Output* (EO).

In accordance to the *RASD* documents, we have identified these points:

- Produce output *procedure* for password recovery.
- Produce a list of remarkable locations placed nearby the coordinates got by GPS signal.
- Produce an estimated time of arrival, calculated by the system in function of the traffic condition and the position of the arriving taxi.
- Produce an external programmatic interface which allows application to authenticate itself and access some internal services.

EO	**Complexity**	**FP**
Password Recovery	Low	4
List of nearby places	Low	4
Estimation Time of Arrival	Low	4
Programmatic Interface	Low	4

1.6 External Inquiries

The final capability provided to users through a computerized system addresses the requirement to select and display specific data from files. To accomplish this a user inputs selection information that is used to retrieve data that meets the specific criteria. In this situation *there is no manipulation of the data*. It is a direct retrieval of information contained on the files. These transactions are referred to as *External Inquiries* (EQ).

In accordance to the *RASD* documents, we have identified these points:

- Display *GPS locations* and maps.
- Display ride informations.
- Visualize a map showing the *CityZone* partition of the city, and the distribution of *myTaxiDrivers* in the zones.
- Visualize informations about position and state of all *myTaxiDrivers*.

EQ	**Complexity**	**FP**
Display maps	Low	3
Display Ride Informations	Low	3
Display Taxi Distribution	Low	3
Visualize taxi drivers' informations	Low	3

1.7 Final Estimation

Now, let's show an overview of the *Function Points* assigned in the paragraphs before.

Function Type	**Value**
Internal Logic Files	24
External Interfaces Files	10
External Input	43
External Inquiries	12
External Output	16
Total	105

Lines of Code Now, we are going to estimate the number of *SLOC* (*i.e.* Source Line of Code) necessary to build our software, retrieving the *AVC* (*i.e.* average number of LOC per FP for a given language) at the URL specified below. This parameter's value is equal to

$$46\frac{SLOC}{FP}$$

, so the expected number of *SLOC* is:

$$\texttt{LOC} = 105 \cdot 46 = \mathbf{4830}$$

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

2 Cocomo

2.1 Scale Drivers

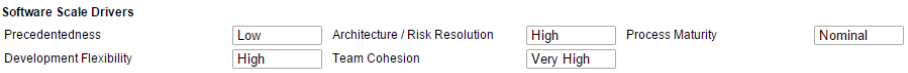


Figure 3: Scale Drivers.

Precedentedness
: It represents the previous experience that we had with that kind of systems. Since this is our first experience using that framework, and methods, the value will be *low*.

Development flexibility
: It reflects the degree of flexibility in the development process. This value will be *hight* because our work is based on a single page goals document.

Architecture/Risk Resolution
: Reflects the extent of risk analysis carried out. Since we've made a risk evaluation this value will be *hight*.

Team Cohesion
: The Team Cohesion scale factor accounts for the sources of project turbulence and entropy due to difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others. These difficulties may arise from differences in stakeholder objectives and cultures; difficulties in reconciling objectives; and stakeholder's lack of experience and familiarity in operating as a team. Our value will be *very hight* because the low number of person in the team.

Process maturity
: This was evaluated around the 18 Key Process Area (KPAs) in the SEI Capability Model. We've selected *CMM Level 2*, then the value will be *nominal*. That's because we've tried to achieve all objectives but your inexperience on real project will affect the team's maturity.

2.2 Cost Drivers

Figure 4: Cost Drivers.

Required Software Reliability

This is the measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only slight inconvenience then reliability is low. If a failure would risk human life then reliability is very high.

Our value will be *low* because a failure don't have critical consequences.

Data Base Size

This measure attempts to capture the affect large data requirements have on product development.

The reason the size of the database is important to consider it because of the effort required to generate the test data that will be used to exercise the program.

The value will be *low*.

Product Complexity

In accordance with the new COCOMO II CPLEX rating scale, this value will be *high*.

Required Reusability

This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.

We've chosen a *nominal* value because the architecture is quite dependent to the goals required.

Documentation match to life-cycle needs

Several software cost models have a cost driver for the level of required documentation. In COCOMO® II, the rating scale for the DOCU cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs. The rating scale goes from Very Low (many life-cycle needs uncovered) to Very High (very excessive for life-cycle needs).

Analyst Capability

Analysts are personnel that work on requirements, high level design and detailed design. The major attributes that should be considered in this rating are Analysis and Design ability, efficiency and thoroughness, and the ability to communicate and cooperate.

Programmer Capability

This parameter is evaluated according to our degree of cooperation, due to some small problems on it, this value it's set to high.

Personnel Continuity

The rating scale for Continuity is in terms of the project's annual personnel turnover: from 3%, very high, to 48%, very low.

We've selected a value *very high* because our continuity is estimated very high.

Applications Experience
This rating is dependent on the level of applications experience of the project team developing the software system or subsystem.

Our value is very low because of our few experiences.

Platform Experience
The Post-Architecture model broadens the productivity influence of platform experiences, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.

Our value is very low because of our few experiences.

Language and Tool Experience
This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem.

Execution Time Constraint
This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource.

Main Storage Constraint
This rating represents the degree of main storage constraint imposed on a software system or subsystem.

Platform Volatility
It represents the volatility of the platform. Because the principal users are mobile clients, we know this kind of technology is continually updating. Then the value is setted as *high*.

Use of Software Tools
In accordance with COCOMO® table, we've setted as *very low*.

Multisite Development

Required Development Schedule

2.3 Effort Estimation

Results

Software Development (Elaboration and Construction)

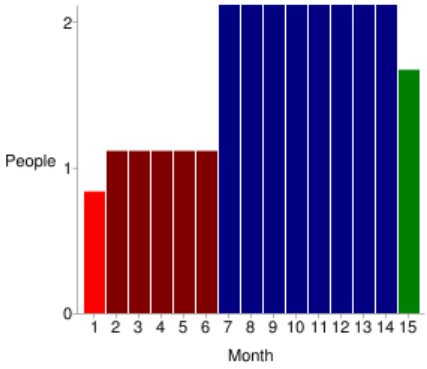
Effort = 23.5 Person-months
Schedule = 13.5 Months
Cost = $58857

Total Equivalent Size = 4830 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.4	1.7	0.8	$3531
Elaboration	5.7	5.1	1.1	$14126
Construction	17.9	8.5	2.1	$44731
Transition	2.8	1.7	1.7	$7063

Staffing Profile



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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.7	1.8	0.4
Environment/CM	0.1	0.5	0.9	0.1
Requirements	0.5	1.0	1.4	0.1
Design	0.3	2.0	2.9	0.1
Implementation	0.1	0.7	6.1	0.5
Assessment	0.1	0.6	4.3	0.7
Deployment	0.0	0.2	0.5	0.8

Figure 5: COCOMO Results.

3 Scheduling Plan

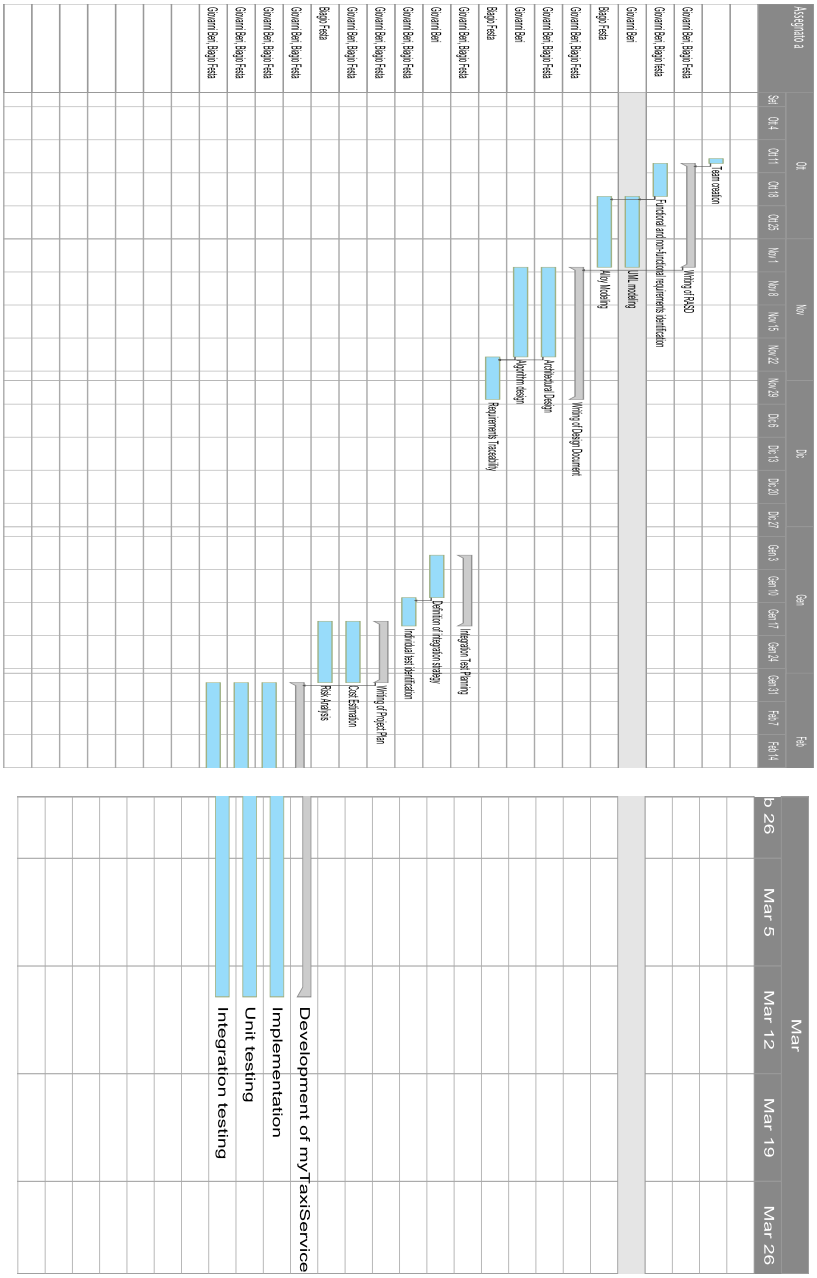


Figure 6: Scheduling.

4 Risks Analysis

4.1 Business Risks

Surveys
Build a good software system without any risks is essentially impossible. In order to avoid bad investments as first approach we suggest to use surveys. That strategy allows us to better understand trend of the market and so to know whether our application could be interesting for the persons. In that way it could be possible estimate an average profit also.

Feedbacks
Another market strategy is about the feedback usage. We suggest to provide to the software a kind of mechanism which allows the users (taxi drivers and passengers) to evaluate the software goodness.

Prototypes
The creation of prototypes may be a optimal solution to evaluate the feedback of the users. In spite of this we must to consider the fact that to build a prototype requires an extra effort. Whether this strategy will be take into account, it's mandatory plan a good design in order to minimize the extra effort and to build a prototypes which can be more as useful as possible.

4.2 Technical Risks

Integration Fails
Because our system is quite complex and it is composed by different components the integration among them may be a problem. That's why the integration test is a essential part in our project.

4.3 Project Risks

We want to prevent, as much as possible, to find out some misunderstanding in the requirements in the late phases of the project. To avoid this, we are going to built, as soon as possible, some really basic prototypes of our system consisting essentially in a GUI, with unimplemented functionalities. This is going to require some extra effort, but it's a fundamental measure in order to avoid very heavy modifications when most of the system has already been implemented.

References

[1] Center for Software Engineering, USC,
COCOMO II, Model Definition Manual,
`http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf`, 2000.

[2] Biagio Festa, Giovanni Beri
myTaxiService, Design Document,
2015.

[3] Biagio Festa, Giovanni Beri
myTaxiService, Requirements Analysis and Specification Document,
2015.

5 Hours of Work

- Beri Giovanni: ~ 15 hours
- Festa Biagio: ~ 15 hours