Project Plan

Beretta Carolina 852650
Brizzolari Cecilia 852399

February 1, 2016

Contents

1 Project size, effort and cost

1.1 Functional Points

The parameters used for the following estimations were taken from tables in the COCOMO II, Model Definition Manual[1]

1.1.1 Internal Logic Files

The applications stores the information about users, ride request and the city.

In particular users can be of two types: drivers or passenger each one of them having their specif information stored in the database. The system must manage ride request, either call and reservation, that are mapped to several entities: each request has in fact a driver, a passenger, a zone and other specific fields associated to it. In order to assign the ride request to drivers, the system must store in the database a representation of the city, and in particular its division into zones. Moreover each zone has an associated queue of available drivers.

ILF	**Complexity**	**FP**
Passenger	low	7
Driver	average	10
Request	high	15
Zones	high	15
Queues	high	15
Total		62

1.1.2 External Logic Files

The system has to interact an acquired data with two external service: Google Maps and the service of the city for drivers' license validation.

The interaction with Google Maps will be used to obtain the ETA and for support in the geolocalization of the drivers, thus will require many instances. On the other hand the interaction with the city's service will only be done after a new driver registration and the returned data will be a simple Boolean value.

ELF	**Complexity**	**FP**
Google Maps	high	10
City's Service	low	5
Total		15

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

1.1.3 External Inputs

The system interacts with the user to allow him/her:

1. *Sign Up*: the process requires the user to input an average amount of data that needs to be validated. The system then proceeds to create a dummy account, waiting for the activation link, thus augmenting the overall complexity of the process. Moreover if the registration is done for a driver account, the validation of the license requires an additional step.

2. *Log In*: the process is very simple, as it's just a checking of two string values

3. *Manage profile*: the update process requires the user to insert data, the system has to validate before performing the actual update of the tuple in the database. For the deletion the user has to insert the password to confirm the operation. Both operation are considered as average

4. *Request a taxi, for either a call or a reservation* (passenger only): these operations involve many entities such as the zones, for queue managing, and drivers and trigger the execution of the algorithm to find a taxi, thus having a high complexity

5. *Change Status* (driver only): drivers can only change their status from available to offline and vice-versa, thus making the operation a very simple one

6. *Accept or decline a ride* (driver only): drivers needs to respond to the ride request by just either confirm it or decline it through the given buttons, thus making the operation a very simple one

EI	**Complexity**	**FP**
Sign Up	average	4
Log In	low	3
Manage profile (update and delete)	average	2x4
Request taxi (call and reservation)	high	2x6
Change status	low	3
Accept/Decline a ride	low	3
Total		33

1.1.4 External Inquires

The system allows users to request information about their profile and for passengers to see the details of their active request. Both operations are simple, as they just need to retrieve basic data form the database, thus having a low complexity

EI	Complexity	FP
User Profile	low	3
Active Request	low	3
Total		6

1.1.5 External Outputs

EO	Complexity	FP
Ride Notification	low	4
Total		4

1.1.6 Final Estimation

The total number of functional points is:

Table 1: recap of FP values

Function type	value
Internal logic files	62
External logic files	15
External inputs	33
External inquires	6
External outputs	4
Total	120

This value can be used as a basis to estimate the size of the project in lines of code:

$$SLOC = 120 * 46 = 5520\, lines$$

where the factor 46 represents the language selected for the implementation of the project, which is J2EE[2]. In previous assignments, in particular in the DD, we tried to present our documents and our design decision as independent as possible from the programming language or architecture. However in order to estimate the SLOC we were forced to make a decision, and we chose the J2EE as it was the one presented during the lectures.

[2]http://www.qsm.com/resources/function-point-languages-table

1.2 COCOMO

1.2.1 Scale Drivers

The values of scale drivers were evaluated as follows:

1. **Precedentedness**: since we do not have any previous experience with similar projects, this value is set to low

2. **Development flexibility**: the specification of the project provided by the professor were quite general on how to implement and manage issues of the application. As this was done on purpose to allow us to design different solutions, this value is set to high.

3. **Team cohesion**: we did not have any problem in working together as a team, as we had familiarity and experience in working together on a project.

4. **Process maturity**: this parameter was evaluated around the 18 Key Process Area(KPAs) in the SEI capability model. The result we got was 2,988 that was approximated to level 3.

	PREC	**FLEX**	**RESL**	**TEAM**	**PMAT**	**Total**
scale	low	high	high	extra high	level 3	
value	4.96	2.03	2.83	0.00	3.12	12.94

1.2.2 Cost Drivers

After a discussion and analysis on the tables and their values, the following estimation of cost drivers was found:

Cost Driver	scale	value
Required Software Reliability	high	1.10
Database size	nominal	1.00
Product complexity	high	1.17
Reusability	nominal	1.00
Documentation match to life-cycle needs	high	1.11
Execution time constraint	nominal	1.00
Main storage constraint	nominal	1.00
Platform volatility	low	0.87
Analyst capability	nominal	1.00
Programmer capability	very high	0.76
Personnel continuity	very high	0.81
Application experience	very low	1.22
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	nominal	1.00
Product (EAF)		0.88

1.2.3 Effort Equation

The estimation is based on the FPs approach converted to SLOC, and provides the effort estimation in Person-Months (PM):

$$Effort = A * EAF * KSLOC^{E}$$

Where

$$\begin{aligned} A &= 2.94(for\ COCOMO.2000) \\ EAF &= 0.88 \\ E &= B + 0.01 * \sum_{i} SD(i) = 0.91 + 0.01 * 12,94 = 1.0394 \end{aligned}$$

Thus the effort value is:

$$Effort = 2.94 * 0.88 * 5.5520^{1.0394} = 15.27\,PM$$

The duration of the project can then be estimated as:

$$Duration = 3.67 * Effort^F = 3.67 * 15.27^{0.306} = 8.45\, months$$

where the exponent F has been computed as

$$F = 0,28 + 0,2 * (E - 0,91) = 0,306$$

Finally the number of people required to complete the project can be estimated with the formula

$$N_{people} = \frac{Effort}{Duration} = \frac{15.27\, PM}{8.45\, months} = 1.81\, people$$

2 Project plan

This section deals with the identification of the tasks and the creation of a schedule to follow for the creation of the myTaxiService project. Every decision made in this chapter is taken with the support of the Cocomo approach, using the data extracted in chapter 1.

2.1 Task identification

Every software project is divided into four phases, as per the *Rational Unified Process*.

1. **Inception phase**: During this first phase, the primary objective is to scope the system adequately as a basis for validating initial costing and budget, along with the estabilishment of the business case which includes business context, success factors, and financial forecast. To complement the business case, a basic use case model, project plan, initial risk assessment and project description of the core project requirements, constraints and key features are generated.

2. **Elaboration phase**: The elaboration phase is where the project starts to take shape, with the aid of detailed diagrams and descriptions. In this phase the problem domain analysis is made and the architecture of the project gets its basic form.

3. **Construction phase**: This is where the software is actually built: the main focus is on the development of components and other features of the system. This is the phase when the bulk of the coding takes place along with the unit and integration testing, and with its conclusion comes the first external release of the product.

4. **Transition phase**: The last phase means to 'transit' the system from development into production, making it available to and understood by the end user. The activities of this phase include training the end users and maintainers and beta testing the system to validate it against the end users' expectations. The product is also checked against the quality level set in the Inception phase.

Taking this schema into account, the task identification is comprehensive of documentation delivery, implementation and testing. The actual tasks are

- Creation and delivery of the following documentation: *Project plan*, *Requirement analysis and specification document*, *Design document*, *Unit test plan (draft)*, *Integration test plan (draft)*, *Unit test plan (final)*, *Integration test plan (final)*

- Implementation of the components as defined in the documentation, divided into *Data layer*, *Business layer back-end*, *Business layer front-end*, *Presentation layer*

- Testing of the components and their functionalities, as in *unit testing*, *implementation testing*, *alpha testing*, *beta testing*.

2.2 Schedule

For the creation of the schedule, we relied on the *Cocomo II Costructive Cost Model*[3], that delivered the histogram shown in figure 1 after being given as input the scale drivers and cost drivers determined in section 1.2 and 2000$ per month as salary.

Software Development (Elaboration and Construction)

Effort = 15.8 Person-months
Schedule = 9.1 Months
Cost = $31633

Total Equivalent Size = 5520 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.9	1.1	0.8	$1898
Elaboration	3.8	3.4	1.1	$7592
Construction	12.0	5.7	2.1	$24041
Transition	1.9	1.1	1.7	$3796

Staffing Profile

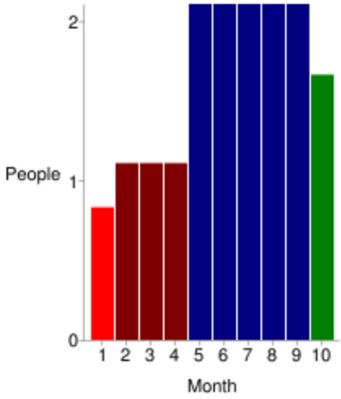

Figure 1: Cocomo II Constructive Cost Model

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

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.5	1.2	0.3
Environment/CM	0.1	0.3	0.6	0.1
Requirements	0.4	0.7	1.0	0.1
Design	0.2	1.4	1.9	0.1
Implementation	0.1	0.5	4.1	0.4
Assessment	0.1	0.4	2.9	0.5
Deployment	0.0	0.1	0.4	0.6

Figure 2: Cocomo II CCM: detailed division

After seeing this model, we decided to make the following adjustements taking into account also the estimation we made manually in the previous chapter:

- the system approximated the duration (9.1 months) as ten, while we had a lower number; so the whole schedule duration is actually of nine months

[3] http://csse.usc.edu/tools/COCOMOII.php

- the system distributed the inception and elaboration phase in four months, having only one person working on it; we decided instead to have both people work on the documentation, so instead of four months we're giving these phases two months
- the last month gained is added to the construction phase; this is considering our own confidence in the implementation of this project, seeing that we're not familiar with the language and with the complexity of the software to be delivered.

The actual rough schedule, then, would be

- October to November: Inception and Elaboration
- December to May: Construction
- June: Transition.

The specific schedule for the delivery of each task, instead, is illustrated in table 2 in temporal sequence. The month indicated in the first column means that that specific task must be ready at the end of the given month. Most task are executed in parallel by the two people.

	Documentation	Implementation	Testing
October 2015	Project plan		
	Requirement analysis and specification document		
November 2015	Design document		
	Unit test plan (draft)		
	Integration test plan (draft)		
January 2016		Data Layer	
		Business layer back-end	
March 2016		Business layer front-end	
		Presentation layer	
	Unit test plan (final)		
	Integration test plan (final)		
April 2016			Unit testing
mid-May 2016			Integration testing
May 2016			Alpha testing
June 2016			Beta testing

Table 2: Task delivery schedule

3 Resource Allocations

The team is composed of two people, each one of them representing a resource to be allocated to different tasks.

3.1 Inception and Elaboration

The allocation for the first part of the project, i.e. the inception and elaboration phase is based on the actual division of the work done for the delivers of the documents:

Table 3: resource allocation phase one and two

	Resource 1	**Resource 2**
PP	all	
RASD	Specific Requirements	Overall description and Alloy
DD	Algorithm and User Interface Design	Architectural Design
ITP	Individual steps, Program stubs and test data	Integration Strategy and Sequence, Tools and Test equipment

The Project Plan document does not have a clear distinction, as it was redacted in parallel by both team members. Since we did not have to write any document for the unit test, we can't define the division of the work on a delivered document. The reduction of the document will be split into two: one person will focus on the draft definition of the unit test of the back end, while the other will focus on the front end and presentation layer.

3.2 Construction

The coding of the application could be split as follows:

Table 4: resource allocation phase one and two

Resource 1	**Resource 2**
Business layer back-end	Data layer
Presentation layer	business layer front-end

During the construction phase the team will have to redact the final version of the unit test plan and integration test plan document. Each team member should focus on the definition of the unit test for the parts she is working on. The final integration test plan document on the other hand could be redacted while having the team members complete and revise the part of the document they did not write the draft on. This approach could in fact aid in the identification of any problem in the document itself.

As soon as the final document on testing have been delivered the testing process can begin in these order: unit test, integration test and system test(alpha version). It has to be noted that the unit test process would probably begin in parallel with the coding, as this tests are very simple one.

Since the integration strategy is bottom-up the work could be split along the creation of drivers:

Table 5: resource allocation for integration test

Resource 1	**Resource 2**
S02, S04, S07, S09	S03, S05, S06, S08

3.3 Transition phase

The beta testing could be divided by assign to each resource a type of user: driver and passenger. In this phase the team should also make other tests such as load and stress testing.

4 Project risks

In this section the project risks are identified and defined, according to figure 3[4].

4.1 Risk identification

Evaluating the project and the team abilities, the risks that can possibly arise are the following:

- **Requirements**
  - Reqm2, due to an inexperience of the project managers and a too general description given for the various assignments
  - Reqm3, same as above
- **Planning & Control**
  - P&C2, due to the inexperience of the project managers, the project may not be adequately monitored
  - P&C4, because the managers may not have estimated the size of the project and the according schedule correctly
  - P&C6, due to the fact that the team members have no prior experience for a project of this kind
- **Team**
  - Team1, due to the fact that the team members have no prior experience for a project of this kind
  - Team2, because there was no actual specific training to prepare the team members

The most relevant risk is the Reqm2, since an incorrect identification of requirements may increase the project complexity, creating a huge amount of problems. This is the most problematic situation also because the requirements are specified in the *Requirement Analysis and Specification Document*, that is the second document to be delivered: an error in the first phase of a project will create, by cascade, other problems in the other phases, and recovering from a requirement error when the project is in its last phase will be expensive in terms of time and money.

[4] http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp732-737.pdf

SIX DIMENSIONS OF SOFTWARE RISKS OF WALLICE ET AL.[3]

Risk dimension	Abbrev iation	Software risk
User	User1	Users resistance to change
	User2	Conflicts between users
	User3	Users with negative attitudes toward the project
	User 4	Users not committed to the project
	User5	Lack of cooperation from users
Requirements	Reqm1	Continually changing requirements
	Reqm2	System requirement not adequately indentified
	Reqm3	Unclear system requirements
	Reqm4	Incorrect system requirements
Project complexity	Comp1	Project involves the use of new technology
	Comp2	High level of technical complexity
	Comp3	Immature technology
	Comp4	Project involves the use of technology that has not been used prior project
Planning & Control	P&C1	Lack of effective project manage technology
	P&C2	Project progress not monitored closely enough
	P&C3	Inadequate estimation of required resources
	P&C4	Poor project planning
	P&C5	Project mile stone not clearly defined
	P&C6	Inexperience project managers
	P&C7	Ineffective communications
Team	Team1	Inexperience team members
	Team2	Inadequately trained development team members
	Team3	Team members lack of specialized skill required by the project
Organizational Environment	Org1	Change in organizational management during the project
	Org2	Corporate politics with negative effect on the project
	Org3	Unstable organizational environment
	Org4	Organization undergoing restructuring during
		the project

Figure 3: Risk dimensions

4.2 Risk recovery

To recover from the risks elencated previously, the recovery actions should be

- for the requirement-dimension risks: improve communication and stakeholder management, and eventually redefine the project, i.e. rejustify the problem financially
- for the team-dimension risks: add resources and improve the team preparation through formative courses
- for the planning&control-dimension risks: either replace the project managers or call a consultant to manage the recovery; instead, if an error in the scheduling arises, it may be enough to add resources to the development team.

A Appendix

A.1 Software and tools

1. *Lyx* to redact and format the document
2. *Cocomo II tool* to create the tables and histogram in chapter 2

A.2 Hours of work

- Carolina Beretta ~ 15 h
- Cecilia Brizzolari ~ 15 h