Software Engineering 2: PowerEnjoy Project Planning

Andrea Pace, Lorenzo Petrangeli, Tommaso Paulon

22nd January 2017

Contents

1 Introduction **2**

2 Project size, cost and effort estimation **2**

2.1 Size estimation: function points 2

2.1.1 Internal Logic Files 2

2.1.2 External Interface Files 2

2.1.3 External Input 3

2.1.4 External Output 3

2.1.5 External Inquiry 3

2.1.6 Complexity level of Function Points 4

2.1.7 UFP to SLOC conversion 4

2.1.8 Count of FPs, by type and weight 4

2.2 Cost and effort estimation: COCOMO II 7

2.2.1 Scale Drivers 8

2.2.2 Cost Drivers 9

3 Schedule **17**

3.1 Schedule introduction 17

3.2 Activities and tasks identification 17

4 Resource allocations **18**

5 Risk management **18**

6 Hours of work **19**

1 Introduction

The Project Plan documents aims at estimating the effort and the cost related to the development of the PowerEnjoy project. We will estimate the size of our project through Function Points(FP) and then we will use COCOMO II to estimate the cost and effort of the development. Then we will define a schedule for the expected tasks and the distribution of the work for the components of our team. Eventually a brief analysis on risks and related strategies will be shown.

2 Project size, cost and effort estimation

2.1 Size estimation: function points

2.1.1 Internal Logic Files

An Internal Logic File (ILF) is an *homogeneous set of data used and managed by the application.*

In our system the ILFs are the following:

- Operators
- Users
- Cars
- PowerStations
- Reservations

2.1.2 External Interface Files

An External Interface File (EIF) is an *homogeneous set of data used by the application but generated and maintained by other applications.*

The only EIF of the system is the interface with the map provider (from now we will call this EIF “Maps”).

2.1.3 External Input

An External Input (EI) is an *elementary operation to elaborate data coming from the external environment.*

The EIs of the system are:

- User registration
- User login
- Payment info change

2.1.4 External Output

An External Output (EO) is an *elementary operation that generates data for the external environment.*

The EOs of the system are:

- Final bill notification
- Confirm reservation notification
- Confirm registration email
- Time expired notification

2.1.5 External Inquiry

An External Inquiry (EQ) is an *elementary operation that involves input and output, without significant elaboration of data from logic files.*

The EQs of the system are:

- Car unlock
- Car reservation
- Debts payment
- Money saving option

2.1.6 Complexity level of Function Points

Tables below shows how to calculate the weights of Function Points (from [1]).

	Data Elements		
Record Elements	1-19	20-50	51+
1	Low	Low	Avg
2-5	Low	Avg	High
6+	Avg	High	High

Table 1: FP counting weights for ILFs and EIFs

	Data Elements		
File Types	1-5	6-19	20+
0-1	Low	Low	Avg
2-3	Low	Avg	High
4+	Avg	High	High

Table 2: FP counting weights for EOs and EQs

	Data Elements		
File Types	1-4	5-15	16+
1	Low	Low	Avg
2-3	Low	Avg	High
3+	Avg	High	High

Table 3: FP counting weights for EIs

2.1.7 UFP to SLOC conversion

The multiplicator to convert the Unadjusted Function Points(UFP) to Source Lines of Code(SLOC) for Java is **53** ([1]).

2.1.8 Count of FPs, by type and weight

The tables below display the computed weights for the system.

Functions	**Weights**
Operator	Low
User	Avg
Car	Avg
PowerStation	Low
Reservation	Low

Table 4: Computed weights for ILFs

Operator, *PowerStation* and *Reservation* are entities with a simple structure (small number of fields) so it is reasonable to adopt the low weight; *User* and *Car* are more complex (more information) so the average weight is a better choice.

Functions	**Weights**
Maps	High

Table 5: Computed weights for EIFs

Maps is an entity with a complex structure, so we can adopt the high weight.

Functions	**Weights**
User registration	Low
User login	Low
Payment info change	Low

Table 6: Computed weights for EIs

User registration, *User login* and *Payment info change* are simple operations (they involve one or two entities) so low weight is the best choice.

Functions	**Weights**
Final bill notification	Avg
Confirm reservation notification	Low
Confirm registration email	Low
Time expired notification	Low

Table 7: Computed weights for EOs

The EOs of our system are very simple operations then it is reasonable to adopt the low complexity weight for them.

Functions	**Weights**
Car unlock	Avg
Car reservation	Low
Debts payment	Low
Money saving option	Avg

Table 8: Computed weights for EQs

Car reservation and *Debts payment* are simple operations, whereas *Car unlock* and *Money saving option* are more complex due to the position check.

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

Table 9: Scores of function points by type and weight

Function	**Type**	**Complexity**	**Weight**
Operator	ILF	Low	7
User	ILF	Avg	10
Car	ILF	Avg	10
PowerStation	ILF	Low	7
Reservation	ILF	Low	7
Maps	EIF	High	10
User registration	EI	Low	3
User login	EI	Low	3
Payment info change	EI	Low	3
Final bill notification	EO	Avg	5
Confirm reservation notification	EO	Low	4
Confirm registration email	EO	Low	4
Time expired notification	EO	Low	4
Car unlock	EQ	Avg	4
Car reservation	EQ	Low	3
Debts payment	EQ	Low	3
Money saving option	EQ	Avg	4

Table 10: Weights computed for all the functions

Function type	**Complexity-Weight**			**Total Points**
	Simple	Medium	Complex	
External Input	$3 \cdot 3$	$0 \cdot 4$	$0 \cdot 6$	9
External Output	$3 \cdot 4$	$1 \cdot 5$	$0 \cdot 7$	17
External Inquiry	$2 \cdot 3$	$2 \cdot 4$	$0 \cdot 6$	14
Internal Logic File	$3 \cdot 7$	$2 \cdot 10$	$0 \cdot 15$	41
External Interface File	$0 \cdot 5$	$0 \cdot 7$	$1 \cdot 10$	10
Total Points	48	33	10	**91**
SLOC	x53			**4823**

Table 11: Count of Function Points

2.2 Cost and effort estimation: COCOMO II

In this section we will use the COCOMO II approach to estimate the effort an the cost required to develop PowerEnjoy

2.2.1 Scale Drivers

We need to evaluate the values of the scale drivers in order to compute the E parameter in the Effort equation.
We refer to the following table:

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	6.20	**4.96**	3.72	2.48	1.24	0.00
Flexibility	5.07	**4.05**	3.04	2.03	1.01	0.00
	(20%)	(40%)	(60%)	(75%)	(90%)	(100%)
Risk Resolution	7.07	5.65	4.24	**2.83**	1.41	0.00
Team cohesion	5.48	4.38	3.29	2.19	**1.10**	0.00
	Level 1 Lower	Level 1 Upper	Level 2	Level 3	Level 4	Level5
Process maturity	7.80	6.24	4.68	**3.12**	1.56	0.00

For each scale factor a short explanation of the chosen value is provided.

- Precedentedness: our group is not expert in the project's field but we have considerable understanding of the product objectives and we don't need innovative algorithm or architectures: this value will be low
- Flexibility: we have strict constrains about the functionalities that we have to develop and there is a need for software conformance with external interfaces like the payment handler, but no premium for early completion of the system, so this value will be low
- Risk Resolution: Our risk analysis has been conducted in an accurate way, while not much of the whole development time was dedicated to this estimation and we have valid tool supports to develop and verify architectural specs. Thus the level of uncertainty in key architectural drivers such as user interfaces, COTS, hardware and performance requirements is little. We therefore choose a high value
- Team cohesion: for our team the value is very high
- Process maturity: we chose to set this factor to Level 3: we plan to have a set of defined and documented standard processes already established and possibly subject to improvement in our project. These processes can be used to assess the consistency of the project performance

The final result is the following:

Scale driver	Factor	Value
Precedentedness	Low	4.96
Flexibility	Low	4.05
Risk Resolution	High	2.83
Team Cohesion	Very High	1.10
Process maturity	Level 3	3.12
Total		16.06

2.2.2 Cost Drivers

We are in an early design phase, we don't have clear information on the architecture to be developed. The cost drivers will be chosen according to this choice. So we will estimate the drivers for the post-architecture approach and then calculate the early design drivers. A brief description of each driver and an explanation of each choice of the values is provided as follows. For every cost driver we refer to the official COCOMO II tables:

- ACAP: this value assess the analists' design and analysis ability, regardless of their experience. We think that our analysis has to be conducted in a rigorous way so we set this parameter to high

ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.42	1.19	1.00	0.85	0.71

- PCAP: this value assess programmers' capability as a team, regardless of their individual experience. Since we are in an early design phase and we have never worked together we can roughly estimate this parameter and we set this parameter to high

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.34	1.15	1.00	0.88	0.76

- PCON: this value refers to the project's personnel turnover: in our case we can't work continously through the project development so this value will be set to low. We use six months as unit of time instead of the year

PCON Descriptors	48%/year	24%/year	12%/year	6%/year	3%/year
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.29	1.12	1.00	0.90	0.81

- RELY: this value is about the effect of a system failure. The worst possible problem in our project can arise if the system locks the car with people onboard, so this value will be set to low.

RELY Descriptors	slight inconvenience	low, easy, recoverable loss	moderate, easily recoverable losses	high financial losses	risk to human life
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	0.82	0.92	1.00	1.10	1.26

- DATA: this value is defined as the ratio of bytes in the testing database to SLOC in the program: the size of the database is impossible to know in advance but we assume to have something like 100 cars and 30 users to perform the tests. The amount of storage required can't be more than 2 MB. The D/P ratio is then 414 and a High value is chosen

DATA Descriptors	SLOC < 10	$10 \leq$ D/P < 100	$100 \leq$ D/P < 1000	D/P ≥ 1000
Rating levels	Low	Nominal	High	Very High
Effort multipliers	0.90	1.00	1.14	1.28

- CPLX: this value is a subjective weighted average of five area ratings: complexity of control operations is low, and so is complexity of computational operators, since we have to evaluate simple expressions. For device dependant operation the value is set to low thanks to the isolation provided by API calls that are probably going to be used. Data management operation are also very simple, so we choose a low value. We also need a simple graphic user interface, that means a low value. The overall value is low

- DOCU: this value reflects the need for a complete documentation in order to ensure that it is suitable for the project's life-cycle needs. For our project this label is set to nominal, since a right-sized documentation is always a good practice

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
Effort multipliers	0.81	0.91	1.00	1.11	1.23

- RUSE: this value represents the additional effort needed to build components intended for reuse on different levels. We don't plan to reuse our components across modules in complex and larger products, so we choose the nominal value

RUSE Descriptors	none	across project	across program	across product line	across multiple product line
Rating levels	Low	Nominal	High	Very High	Extra High
Effort multipliers	0.95	1.00	1.07	1.15	1.24

- TIME: this is a measure of the percentage of the available execution time expected to be used by our system. Since our application is not extremely demanding we estimate that about 70% of the available execution time will be used. That means a High value

TIME Descriptors		<= 50% use of available execution time	70%	85%	95%
Rating levels	Low	Nominal	High	Very High	Extra High
Effort multipliers		1.00	1.11	1.29	1.63

- STOR: this is a measure of the main storage constraint imposed on our server: since we can easily afford a database with considerable capacity and we can rely on cloud computing, we choose the nominal value

STOR Descriptors	$\leq$ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating levels	Nominal	High	Very High	Extra High
Effort multipliers	1.00	1.05	1.17	1.46

- PVOL: this is a measure of the rate of major updates to our complex of hardware and software. We don't expect to change the main functionality of our system more often than twice a year, so we choose the nominal value

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	Low	Nominal	High	Very High
Effort multipliers	0.87	1.00	1.15	1.30

- APEX: this is a measure of the experience of the team developing the project. Our experience is limited to a six-month project about a Java application, so we choose a low value

APEX Descriptors	<= 2 months	6 months	1 year	3 years	6 years
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.22	1.10	1.00	0.88	0.81

- PLEX: refers to the developers acquaintance of the platform that is going to be used. We don't have experience with the Java EE platform or database development so we set this value to low

PLEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.19	1.09	1.00	0.91	0.85

- LTEX: this driver reflects the project team knowledge of the programming language and the sofware tools(used throughout every phase from requirement analysis to development) to be used. Our knowledge is limited to the tools and the programming language highlighted in the PLEX driver, so we choose a low value

LTEX Descriptors	$\leq$ 2 months	6 months	1 year	3 years	6 years
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.20	1.09	1.00	0.91	0.84

- TOOL: this drivers reflects the quality of the software tools used to develop this project: our application environment is a well integrated one and will be used through the whole project's life cycle, so we choose a high value

TOOL Descriptors				strong, mature life-cycle tools moderately integrated	
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.17	1.09	1.00	0.90	0.78

- SITE: this driver reflects the communication issues caused by the site collocation of the team members and by the technology used to communicate. Our group is located in the same city and we use efficient applications to communicate, so the value will be set to high

SITE Collocation Descriptors	International	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro area	Same or c
Communications Descriptors	Some phone, email	Individual phone, fax	Narrow band email	Wideband electronic communication	Wideb comm occasio con
Rating levels	Very Low	Low	Nominal	High	Ver
Effort multipliers	1.22	1.09	1.00	0.93	(

- SCED: this driver is a measure of the schedule compression/expansion for the whole project: our effort has been distributed thoroughout this semester, even if we will devote a consistent amount of time to analysis and design first and testing/inspection at the end. The value will be set to high

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal
Rating levels	Very Low	Low	Nominal	High	Very High
Effort multipliers	1.43	1.14	1.00	1.00	1.00

Now we can derive the Early Design counterparts according to the following table:

Early design Cost Driver	*Post-architecture counterparts*
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

Table 12: Post- architecture and early design counterparts

For each driver calculated following the post-architecture method, we assign a value to each choice. We refer to the following table:

Level	Value
Very Low	1
Low	2
Nominal	3
High	4
Very High	5
Extra High	6

The tables used to retreive the value of the early design cost drivers ca be found in the document linked in the References. The values obtained for each choice will be summed and the result will determine the value of the related early design cost driver:

- PERS: we have High for ACAP, High for PCAP and Low for PCON, so the result is 10. This means a High value: 0.83
- RCPX: we have Low for RELY, High for DATA, Low for CPLX and Nominal for DOCU, so the result is 11. This means a Low value: 0.83
- RUSE: same as in the post-architecture, the value is 1.00
- PDIF: we have High for TIME, Nominal for STOR and Nominal for PVOL, so the result is 10 and the overall value is High: 1.29
- PREX: we have Low for APEX, PLEX and LTEX, so the result is 6. This means a Very Low value: 1.33
- FCIL: we have High fot TOOL and SITE so the result is 8. Tis means a High value: 0.87
- SCED: same as in the post- architecture, the value is 1.00

Now we can compute the final product of the Effort Multipliers:

Effort Multiplier	**Value**
PERS	0.83
RCPX	0.83
RUSE	1.00
PDIF	1.29
PREX	1.33
FCIL	0.87
SCED	1.00
Result	1.029

We have now all the elements to compute the estimated amount of PM(Person Month) for this project:

$$PM = A \cdot Size^E \cdot \prod_i EM_i$$

where:

- $A = 2.94$

- $Size$ is the estimated size of the project in KSLOC. In our case this value is 4.823
- E is the aggregation of the five scale factors
- EM_i are the effort multipliers

E id obtained using the following formula:

$$E = B + 0.01 \sum_{j=1}^{5} SF_j$$

where $B = 0.91$ and SF_j are the 5 Scale Factors

Therefore E is equal to 1.0706 and PM is equal to 16.29

3 Schedule

3.1 Schedule introduction

The main activities involving the project are: RASD, DD, ITPD, Project Planning, Presentation, Implementation, Testing and Deployment. Some tasks needs to be completed before others can begin, as shown in the **dependency graph** below:

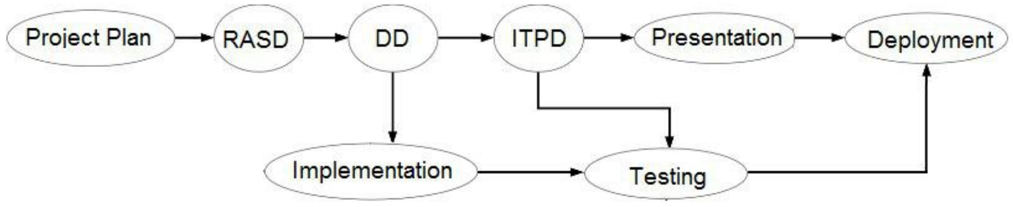

Table 13: Dependency graph

3.2 Activities and tasks identification

The actual diagrams are contained in the Gantt.pdf file provided with this document. Those are the activities and tasks concerning the project. We assume that we won't implement, test and deploy the project, as shown in the Resource Allocation paragraph. We added those parts to give a more realistic description of the project. The period of time in which every task is active is approximate, however the deadlines for the entire activities are mandatory. It is expected that some tasks will be resumed after some time, depending on the development of the project.

4 Resource allocations

In the Gantt.pdf file there is also an overview of how the tasks from **Schedule** section will be divided between the three members of the team. Although each member will be actively involved in every task as a supervisor, we won't assign a task to more than two members for better parallelism. As mentioned in **Schedule** section, some tasks won't be assigned to the members of the team. Those tasks are shown as assigned to **Others**. The periods of time in which every task is active are approximate, and the percentage represents how many tasks are active for each resource (for instance: 25% means that the resourse is allocated on 4 tasks).

5 Risk management

In this section we are going to analyze some risks that could affect the project.

Due to inconveniences during the development of the project the deadlines could be delayed. In that case, we may release a beta version of the application and we may release the stable version later.

Since the team often works remotely, another possible problem could be the lack of communication among team members; the solution is defining explicitly the role of each team member and writing a clear and complete documentation.

A large number of users could cause scalability issue; in that case we may ask a third part provider to host our system. As soon as users will start using the application, bugs in the code may appear: the team will update the system in a small time working on users' warnings.

Another small risk is represented by the fact that our system uses external services such as maps and payments, but the providers are very reliable and this problems are unpredictable and have low chances to occur.

Nowadays data leaks and attacks are a serious issue; to avoid this problems we may encrypt the communication and adopt all the security standards.

Hardware malfunctioning could cause data loss; these problems, even if unpredictable, may be contained with backups and regular maintenance.

Our system doesn't require the driver licences to be really existent. According to the Italian legislation it's entirely up to the driver to have a valid driver license while driving. Changes to this rule could require our system to check the actual existence of the provided driving license numbers.

In case of accident it's up to both the user and the company to refund other people for damages. We can permanently ignore this unpleasant situation by stating in the terms of the service that the client must refund the company for its part and therefore pay the whole sum.

There may be a problem with competitors since it's a new market. In this case we will provide new features and start an appropriate market campaign.

6 Hours of work

- Andrea Pace: 15 hours
- Lorenzo Petrangeli: 12 hours
- Tommaso Paulon: 14 hours

References

[1] COCOMO II - Model Definition Manual, Version 2.1, 1995-2000, Center for Software Engineering, USC. **http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman200**