Politecnico di Milano, A.A. 2016/2017

Software Engineering 2: PowerEnJoy
Project Plan

Binosi Lorenzo 876022

January 19, 2017

Contents

1 Introduction

1.1 Revision History

Version of this document: 1.0
Last update: x

1.2 Purpose

This document is the Project Plan Document (PPD) for the PowerEnJoy project. Its aim to estimate which are the sizes, the costs and the efforts that should be faced in order to realize the project. According to the estimation made, it will define the required budget, the resources allocation and the schedule of the activities.

The document is written for customers, project managers, developers, testers and all the stakeholders involved in the project implementation.

1.3 Scope

PowerEnJoy is a digital management system for a car-sharing service that exclusively employs electric car. This car rental system provides an alternative solution to public transport, thus being not only eco-friendly, but also simple and reliable. Its implementation should be achieved in a clear and defined way in order to meet the established deadlines, and avoid any further cost.

1.4 Definitions, acronyms, and abbreviations

1.5 Reference Documents

This document refers to the following documents:

- Software Engineering 2 project [1].
- PPD assignement [2].
- RASD of the PowerEnJoy project [3].
- DD of the PowerEnJoy project [4].
- ITPD of the PowerEnJoy project [5].
- COCOMO II - Model Definition Manual [6].

2 Project size, cost and effort estimation

This section describes which are the main functionalities of PowerEnJoy that allowed us to estimate the expected size, the cost and the required effort of the project.

For the size estimation we will use the **Function Points** approach. It allows to estimate the correspondent amount of lines of code to be written in Java, according to the main functionalities of PowerEnJoy. We won't consider the functionalies of the user interface in order to have a more meaningful and reliable estimation.

For the cost and effort estimation we will use the **COCOMO II** model, relying on the amount of lines of code estimated previously.

2.1 Size estimation

For the size estimation we will refer to the IFPUG 1994 standard [7] which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method.

In order to determine the complexity level, the IFPUG standard classifies each function count into Low, Average and High complexity levels depending on the number of data element types contained and the number of file types referenced. Tables 2.1a, 2.1b and 2.1c reassume the complexity levels for each type of file.

Once determined the complexity of each function, it's possible to define its weight, i.e., the relative effort required to implement it. Table 2.2 reassume the Unadjusted Function Points (UFP) for each complexity level.

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

(a) Complexity estimation for ILFs and EIFs.

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

(b) Complexity estimation for EOs and EQs.

	Data Elements		
Record Elements	1-4	5-15	16+
1	Low	Low	Avg.
2-3	Low	Avg.	High
3+	Avg.	High	High

(c) Complexity estimation for EIs.

Table 2.1: Estimation of complexities for different types of Function Points.

Function Type	**Complexity-Weight**		
	Low	Average	High
Internal Logic Files	7	10	15
External Interface Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Table 2.2: UFP complexity weights.

2.1.1 Internal Logic Files (ILFs)

The ILFs are those logical files generated, used or mantained by the software system. In our case they include all the information that the system has to store:

- Driver:
  - Driver information
  - CreditCard
- Rent:
  - Rental information
  - RentalEvents
- Car
- SafeArea
- CarAssistance

The Driver Record Element (RET) contains 15 Data Elements (DET's) and the CreditCard RET, which contains 2 DET's. Hence, according to the Table 2.1, the Driver ILF has a low complexity with an amount of 2 RET's and 17 DET's.

The Rent RET contains 6 DET's and the RentalEvent RET, which contains 2 DET's. Hence, the Rent ILF has a low complexity with an amount of 2 RET's and 8 DET's.

The Car, SafeArea and CarAssistance RET's contain respectively 11, 4 and 4 DET's, and their ILFs have a low complexity.

ILF	**Complexity**	**FPs**
Driver	Low	7
Rent	Low	7
Car	Low	7
SafeArea	Low	7
CarAssistance	Low	7
Total		**35**

Table 2.3: The ILFs complexity and the total Function Points.

2.1.2 External Interface Files (EIFs)

PowerEnJoy needs to interface with 3 services, described in the DD, in order to perform some operations. They are:

SMSGateway : for the SMS dispatching.

EmailSender : for the email dispatching.

PaymentGateway : for the payment execution.

Obviously, each of these services doesn't require too much data, so we can consider their complexity as low.

ELF	**Complexity**	**FPs**
SMSGateway	Low	5
EmailSender	Low	5
PaymentGateway	Low	5
Total		**15**

Table 2.4: The ELFs complexity and the total Function Points.

2.1.3 External Inputs (EIs) and External Inquiries (EQs)

The PowerEnJoy business logic provides a RESTful API for all the core functionalities of the system. All the API requests are completely described in the DD [2.5.3, p. 14] and summarized below. Thanks to this detailed description, the complexity estimation of the EIs and the EQs will be performed accurately.

- User creation: this request allows the creation of a new user. It requires all the user information as input and doesn't provide any output data. According to Table 2.1c, the EI of the request has an high complexity with an amount of 3 RET's and 12 DET's.
- User's salt bytes retrieval: this request allows to retrieve the user's salt bytes in order to use them for the password hashing. It requires only the username of the user as input and provides the user's salt bytes as output. The EI and the EQ of the request have both a low complexity.
- User deletion: this request allows to delete the user that performs the request. It requires only the credentials of the user (username and

hashed password) as input and doesn't provide any output data. The EI of the request has a low complexity.

- User's rental logbook retrieval: this request allows a user to retrieve his/her rental logbook. It requires only the credentials of the user as input and provides the logbook information as output. The EI and the EQ of the request have respectively a low and an average complexity. In particular we can observe that the EQ contains an amount of 2 RET's and 6 DET's which give the average complexity, according to Table 2.1b.

- Available cars' retrieval: this request allows a user to retrieve all the available cars in the nearby area. It requires the credentials of the user and his/her geographical position as input, and provides the data about the available cars as output. The EI and the EQ of the request have both a low complexity, with an amount of 3 RET's and 6 DET's for the second one.

- Car reservation: this request allows a user to reserve a choosen car. It just requires the credentials of the user and the license plate of the car as input, and doesn't provide any output data. The EI of the request has a low complexity.

- Current reservation retrieval

3 Tasks and schedule

4 Resource allocation

5 Risk management

A Appendix

A.1 Hours of work

This is the time spent redacting the ITPD

- Lorenzo Binosi - hours

References

[1] Software Engineering 2 Project, AA 2016/2017 - *Goal and approach, schedule and rules*

[2] Software Engineering 2 Project, AA 2016/2017 - *Assignments 4*

[3] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Requirements Analysis and Specification Document*

[4] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Design Document.*

[5] Lorenzo Binosi, *Software Engineering 2: PowerEnJoy - Integration Testing Plan Document.*

[6] 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_modelman2000.0.pdf

[7] IFPUG 1994 - *Function Point Counting Practices Manual*, Release 4.0, 1994.