Software engineering 2 project

PowerEnJoy

Project Plan Document
(PPD)

Authors:
Madaffari Federico (matriculation number: 792873)(Computer Science)
Mandrioli Claudio (matriculation number: 849973)(Automation and Control)

Table of Contents

Introduction

Revision history

This is the version 1.0 of this document.

Purpose and scope

The purpose of this document is to present an a priori analysis of the timing and costs for the production of the software management system for the PowerEnJoy service.
The project is analyzed from both the qualitative (for instance the expected complexity) and quantitative (i.e. the expected dimensions of the software) points of view. This is done also relying on automated procedures.
The results of this analysis are then used to provide a schedule for the actual realization of the product.

Document structure

This document is structured in five sections:

- The introduction, which is this section, that briefly presents the purpose and content of the document.
- The cost size and cost-effort estimation that shows how two automated tool are used for evaluating the named properties of the project.
- The Project Scheduling and the Resource Allocation that present the expected allocations of time and resources to the various steps of the development of the system.
- The Risk Management that tries to foresee the major problems that may occur during the development and proposes both some prevention strategies and some counteractions.

Acronyms

- ILF: internal logic files
- EIF: external input files
- EI: external inputs
- EO: external outputs
- EQ: external queries
- FP: function points
- JEE: java enterprise edition
- LOC: lines of code
- PM: person-month

Reference document

- Assignments AA 2016-2017
- COCOMO user manual (http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf)

Project size and cost-effort estimation

In this section algorithmic approaches are used to estimate the size of the project, and the related costs.

For the size estimation the Function Points methodology is used: this approach will provide an estimation in terms of a number that can be (according to some statistics) related to the overall number of lines of code of the final product.

Instead for the cost-effort estimation the COCOMO procedure is used. Again this is a statistical estimation: it should provide an estimation of the men-months required for the development of the project.

Size estimation: Function Points

The Function Points methodology uses the required functionalities of the system to be developed in order to estimate the future size of the product. The idea is to divide those features into categories (called Function Types) and assign different effort weights to those. Those weight are defined proportionally to the supposed level of complexity of the feature (low, average, high).

The function types are:

- Internal logical file
- External interface file
- External input
- External output
- External inquiry

The used weights for the Function Points procedure are presented in the following table

Function types	Weight		
	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

Now for each Function Type are listed the required functionalities and the number of related FP is retrieved.

Internal logical file

The ILF represent the homogenous set of data that the software system is supposed to manage when working.
We have evidenced the subsequent datasets:

- User: the system will be required to manage the profiles of the registered users. For each user will be stored and managed personal info, credential for access the service and other information. There will be plenty of those data therefore we consider the weight related to this data set HIGH.
- Ride: the system will store the information related to each ride of each user, also in relation to the related reservation. Due to the high connection with other data structures the complexity of this data set is considered HIGH.
- Reservation: users will be able to reserve cars and this fact will be tracked by the system by means of this dataset. Those information must be managed on real time and are interacting with other data structures: for those reasons the complexity is set to HIGH.
- Car: the system will use a dataset to manage the cars of the service. The number of those cars will be constant or at least will rarely change. Moreover we don't expect this to be a complex data structure: for those reasons the complexity level is set to LOW.
- Area: the system will have to recognize different areas of the city and those areas will be stored in this dataset. This dataset is not supposed to change in time and wont store much information more than coordinates. For those reasons the complexity level is set to LOW.
- Payment: for each user the system will store payment information. Those data are therefore in a relation one to one with the users that are numerous; anyhow this dataset is expected to be quite simpler. For those reasons the complexity is set to AVERAGE.

The following table resumes the level of complexity of each ILF and the subsequent related Function Points:

ILF	Complexity	Weight
User	High	15
Ride	High	15
Reservation	High	15
Car	Low	7
Area	Low	7
Payment	Average	10

The resulting amount of Function Points related to IFL is: 69.

External interface file

The EIF represent the homogeneous data sets that will be used by the application but generated and maintained by other application.
The system will interact with the subsequent external systems:

- Payment handler: the dataset used by the software system and managed by the payment handler will represent the data related to the payment of a ride. Those data will be simple even though there will plenty of them therefore the associated complexity is set to LOW.
- Maps service: this dataset will allow the software system to locate cars and users. It will be simple but with considerable large dimensions and for apparent reasons will be used frequently. According to those facts we set the related complexity to AVERAGE.

The following table resumes the level of complexity of each EIF and the subsequent related Function Points:

EIF	Complexity	Function Points
Payment handler	Low	5
Maps service	Average	7

The resulting amount of Function Points related to EIF is: 12.

External input

The EI represent the information that the system is supposed to receive and elaborate as input from external boundaries.
The system will receive the subsequent inputs:

- User registration: this set of inputs are related to the first access of a user to the service. The interaction will be characterized by the simple insertion of the personal data of the user. For this reason the expected complexity of this procedure is EI to LOW.
- User login: this set of input is related to the data that the user will have to provide to access the service. Those information are apparently few and simple: for this reason the expected level of complexity is LOW.
- User profile update: this set of input is related to the possibility for a user to change its personal info after the user registration. For this reason we cannot expect to have procedures more complex than those that we analyzed for the registration: therefore the expected complexity is EI to LOW.
- Car research: this set of input is related to the only request of the user of display cars positions and info. This input therefore provides a simple information (almost only a position around which look for the cars). For this reason the expected complexity of this EI is LOW.
- Car reservation: this set of input represent the procedure that a user must perform to reserve a car. The information exchanged are few and simple (user information and car information) therefore the expected complexity is LOW.
- Cancel a reservation: for this EI can be made considerations completely analogous to those made for the car reservation. For this reason the expected complexity of this EI is set to LOW.
- Car sensors: this set of input is related to all the information that will be provided by the sensors positioned on each car. Those information will be

used by the system to perform several and different operation (for instance ride management and car management): for this reason the expected complexity of those input is set to HIGH.

The following table resumes the level of complexity of each EI and the subsequent related Function Points:

EI	Complexity	Function Points
User registration	Low	3
User login	Low	3
User profile update	Low	3
Car research	Low	3
Car reservation	Low	3
Cancel a reservation	Low	3
Car sensors	High	6

The resulting amount of Function Points related to EI is: 24.

External output

The EO are the set of outputs that allow the system to show information to the external environment.

- User notification: this set of outputs regard all the information that the system may want to provide on real time to the user. Due to the timing requirement and the variety that those information may have the expected complexity of those EO is HIGH.
- Available cars display: this set of outputs is provided to the user when he il looking for available cars in a certain position. According to this definition the system will have to coordinate different data structures and also to interact with the external system of the Maps service: this clearly results in a HIGH level of complexity.
- Payment notification: this is the set of outputs that the system is supposed to provide to the user at the end of a ride. This involves the computation of the total amount that will be charged that requires the computation of eventual discounts and fines. In order to complete this operation different data will be collected and processed. For this reason the expected complexity of this set of output is HIGH.

The following table resumes the level of complexity of each EO and the subsequent related Function Points:

EO	Complexity	Function Points
User notification	High	7
Available car display	High	7
Payment notification	High	7

The resulting amount of Function Points related to EO is: 21.

External inquiry

The EI is a set of simple operations that involve both the receiving and displaying of some data form the system point of view. Those operations shouldn't involve any significant processing the required data.

- Profile visualization: this operation is the normal display of the personal info of a user logged in. Even though this data structure is estimated to be of relevant dimensions there are no reasons to foresee any complexity in the simple retrieval and display of it. The expected complexity is therefore LOW.
- Pending reservation info visualization: this operation is related to the request by the user to see the information related to its active reservation (if he has one). The data to be displayed are the ones of the reserved car and of the reservation timing. As the timing of the reservation is an information strictly related to the instant in which the information are required we can consider this set of operations at least a bit more complex than the ones related to the Profile visualization: for this reason the estimated complexity in this case is AVERAGE.

The following table resumes the level of complexity of each EO and the subsequent related Function Points:

EI	Complexity	Function Points
Profile visualization	Low	3
Pending reservation info visualization	Average	4

The resulting amount of Function Points related to EI is: 7.

Total Function Points and overall estimation

Here below is reported a table summarizing the retrieved function points for each Function Type:

Function types	Function points
ILF	69
EIF	12
EI	24
EO	21
EQ	7

The total amount of Function Points is the simple sum of those values:

$$\text{TOTAL FUNCTION POINTS} = 69 + 12 + 24 + 21 + 7 = 133$$

This value can now be used to retrieve an estimate of the lines of code that will be required to implement the software management system.
The formula is the sequent:

$$LOC = FP \cdot AVC$$

where AVC is an empirically estimated constant that relates the function points to the lines of code of a project according to the used language. In our case the language is already chosen and is JEE.
According to the formula lower bound for the estimation of the LOC is:

$$LOC_{lower\ bound} = 133 \cdot 46 = 6118$$

Instead a upper bound for the estimation is:

$$LOC_{upper\ bound} = 133 \cdot 67 = 8911$$

Cost and effort estimation: COCOMO

The COCOMO procedure is based on an equation that relates the estimated size of the project to the required person-months to develop the full product. The equation moreover involves other parameters that should model the characteristics of the working group, also in relation to the effective contents of the project.

The equation was retrieved as a regression from data of existing projects and therefore it is only a statistic estimation; anyhow there are evidences that the results may provide meaningful information on the costs and time required for the actual development of the product.

The named equation is:

$$PM = A \cdot size^{E} \cdot \prod_{1 \le i \le n} EM_i$$

Where:

- PM is the acronym for Person-Month
- A is a constant value (equal to 2.94) that relates the person month to the kilo source line of code
- Size is the estimated dimension of the project in terms of the number of kilo source line of code (which we will deduce from the FP)
- E is a coefficient that represent the aggregation of five SCALE FACTORS that model some aspects of the team and of the project
- EM is the Effort Multiplier that is related to the COST DRIVERS that are again parameters modeling the developing team and requirements of the project.

First of all the procedure requires to distinguish whether we are in the Early-Design or Post-Architecture phase according to the fact that at least sketch of the architecture of the software to be developed is already available or not. As this document was started at the day zero of the project we can clearly locate us in the Early Design Phase.
This choice, according to the procedure, defines the Cost Drivers to use for the computation of the Effort Multiplier.

The subsequent sections present in detail the computation of the scale factors and of the cost drivers. The last one instead presents the final computation of the Effort Equation.

Scale Factors

The following table lists all the Scale Factors that have to be considered according to the COCOMO procedure with the respective level to evaluate for each of them.

Scale Factor Values, SF_j, for COCOMO II Models

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

Now we all factors are listed and a value is assigned to them according to some evaluations of the project and team properties.

Precedentedness

The precedentedness captures the background of the development team according to the kind of project in question. In our case one of the components of the team has some experience in the development of web applications while the other comes from completely another sector. As the team is composed by only two elements the fact that one of them is highly under-skilled is critical: for this reason the value or this scale factor is VERY LOW.

Development flexibility

The development flexibility accounts for the degree of freedom related to the requirements available for the developers. We considered the sequent factors to evaluate this scale factor:

- The requirement provided by the assignment are informal and allow for a certain degree of flexibility
- There are several product already on the market that provide the same service (a car sharing) for this reason it is important to develop a competitive product with respect to the standards of the market
- Non particular constraint are present from the technological point of view
- The software will have to interface with a couple of external pre-existent systems

As we have quite relevant considerations on both directions the chosen level of flexibility is NOMINAL.

Architecture / risk resolution

This scale factor accounts for the risk management plan and definition of the schedule of the developing process. Both of these factors are analyzed in the subsequent parts of this document, for this reason the level chosen for this scale factor is HIGH.

Team cohesion

This scale factor accounts for the soft skills of the involved team .The members of this team known each other from a long time and have attended the same middle school, high school and university. Those factors allow to select a VERY HIGH level of Team cohesion.

Process maturity

This last scale factor describes the maturity of the software development organization in the sense of how well the production process is able to produce the expected outcomes. In our case according to the project plan there will be a strong commitment in the production of documentation related to the production process (at least will be produced: requirement analysis and specification document, design document, integration testing document) for this reason the selected CMM level is 3 that corresponds to a HIGH scale factor.

Computation of the E coefficient

Now that we have defined the scale factors we can compute the E coefficient according to the formula:

$$E = 0.91 + 0.01 \cdot \sum_i SF_i$$

Scale Factor	Level	Value
Precedentedness	Very low	6.20
Development flexibility	Nominal	3.04
Risk resolution	High	2.83
Team cohesion	Very high	1.10
Process maturity	High	3.12

The resulting value is:

E = 1.0729

Cost drivers

In this section we present the choices made for the Cost Driver coefficients that allow us to compute the effort multipliers. Those are multiplicative coefficient to apply directly to the overall calculus of the effort estimation.
As we are in the early design phase most of our coefficients will represent the aggregation of different cost drivers, in the sequent sections we analyzed each of them.
We will assign to each of the cost driver a value from VERY LOW(=1) to VERY HIGH (=5), then sum those values to get an overall rating for each Early Design Cost Driver. From these rating according to the tables from the COCOMO Model Definition Manual each Effort Multiplier is retrieved.

Personnel capability (PERS)

- Analyst capability (ACAP): this cost driver models the skills f the team members for what concerns the requirement and design phases. Thanks to its heterogeneity the team can guarantee a NOMINAL(3) level of analysis capability.
- Programmer capability (PCAP): this cost driver represents the capabilities of programming of the whole team. In our case although there is not a individual long experience in programming by all the elements of the team the cooperation aspects are well developed. For this reason the NOMINAL(3) value is assigned to this parameter.
- Personnel continuity (PCON): this cost driver represents the turnover that will eventually take place in the team. As no turnover is planned the selected value is VERY HIGH(5).

The sum of these cost driver values is 11, therefore the Early Design Cost Driver PERS is set to HIGH and the Effort Multiplier is 0.83.

Product reliability and complexity (RCPX)

- Software reliability (RELY): this cost driver accounts for the expected level of reliability that is required to the software to satisfy and the kind of damage that a failure may cause. A system failure in this case may cause in the worst case a moderate financial loss: for this reason the selected value for this cost driver is NOMINAL(3).
- Database size (DATA): this cost driver models the impact that large test data may have on the testing phase of the development. In our case the software will have to manage a huge amount of data and it this feature should be tested. For this reason the selected value for this cost driver is HIGH(4).
- Product complexity (CPLX): this cost driver accounts for the complexity that will characterize the product from the algorithmic and computational point of view. The software product is not expected to perform complex operations (the most critical can be the ones related to the managing of car positions and of payments that however are not expected to be very complex). For this reason the chosen value for this cost driver is LOW(1).
- Documentation match to life-cycle needs (DOCU): this cost driver models the expected time that will be dedicated to the production and management of the documentation for the project. The documentation will play an important role in the development of this project, for this reason the value of this cost driver is set to HIGH(4).

The sum of these cost driver values is 12, therefore the Early Design Cost Driver RCPX is set to NOMINAL and the Effort Multiplier is 1.

Reusability (RUSE)

This cost driver accounts for the level of reusability that is required to the components of the product. High reusability would obviously require effort in generating more general and multi-purpose software. In our case no reusability is required if not inside the system itself: according to this reasoning the NOMINAL(3) value is chosen.

The corresponding Effort Multiplier is NOMINAL and it is 1.

Platform difficulty (PDIF)

- Execution time constraint (TIME): This cost driver models the measure of the execution time constraint imposed upon a software system. Since the available execution time expected to be used by the system-to-be is going to be huge, TIME is set to HIGH(4).
- Main storage constraint (STOR): This cost driver models the degree of main storage constraint imposed on a software system or subsystem. The system-to-be is going to be frequently used by a large customer base, however a

situation of storage saturation is highly improbable because of the modern technology storage resource. Therefore the STORE is set to HIGH(4).
- Platform volatility (PVOL): This cost driver represents the frequency of major updates that the system must required. Since our system-to-be will not require frequent major updates this is set to LOW(2).

The sum of these cost driver values is 10, therefore the Early Design Cost Driver PDIF is set to HIGH and the Effort Multiplier is 1.29.

Personnel experience (PREX)

- Application experience (APEX): This cost driver represents the level of applications experience of the project team developing the software system or subsystem. Since only one of the members has previous experience of coding but does not have experience with JEE, APEX is set to LOW(2).
- Platform experience (PLEX): This cost driver represents the level of experience of the team members with the use of powerful platforms, including user interface, database and networking. As stated before regarding the APEX cost driver, only one of the member has some kind of experience with software developing, therefore the PLEX is set to LOW(2).
- Language and tool experience (LTEX): This cost driver represents the level of experience with tools aimed to support requirements modelling and analysis. For the same reasons before, LTEX is set to LOW(2).

The sum of these cost driver values is 6, therefore the Early Design Cost Driver PREX is set to VERY LOW and the Effort Multiplier is 1.33.

Facilities (FCIL)

- Use of software tools (TOOL): This cost driver represents the complexity of the tools used and how much the system-to-be is influenced by them. Since for the system-to-be strong, mature lifecycle tools, moderately integrated are provided, TOOL is set to HIGH(3).
- Multisite development (SITE): This cost driver represents how the site collocation and the communication support of the system-to-be is handled. We think that the best value for our situation is HIGH(3).

The sum of these cost driver values is 12, therefore the Early Design Cost Driver FCIL is set to NOMINAL and the Effort Multiplier is 1.

Required development schedule (SCED)

This cost driver represents the schedule constraint imposed on the project team developing the software. It is set to Nominal since the project must satisfy imposed deadlines and therefore the effort of the team members must be equally scheduled over the development time.

The corresponding Effort Multiplier is NOMINAL and it is 1.

the following table resumes the retrieved Effort Multiplers:

Cost Driver	Factor	Value
PERS	High	0.83
RCPX	Nominal	1
RUSE	Nominal	1
PDIF	High	1.29
PREX	Very low	1.33
FCIL	Nominal	1
SCED	Nominal	1

Effort Equation computation

Now that we have retrieved all the necessary coefficient we report the expected person-moths that we can compute from the Effort Equation.
The equation is:

$$PM = A \cdot size^{E} \cdot \prod_{1 \leq i \leq n} EM_i$$

Using the lower bound for the estimation of the lines of code

$$PM = 2.94 \cdot 11.551 \cdot 1.4240 = 35.81$$

instead using the lower bound

$$PM = 2.94 \cdot 17.293 \cdot 1.4240 = 72.55$$

Those values obviously must be distributed between the two components of the team that will actually develop the product.

Project Scheduling

Schedule and milestones

The following table depicts the way milestones are coupled to various project phases and when they are scheduled:

- The team budget of 2 people x 1470 hours = 2940 hours;
- The project deadline of July 23th 2018
- The final presentation of July 14th 2018
- The intermediate presentation of January 11th 2017
- The peer evaluation deadline of November 26th 2016
- A holiday from December 26th till January 6th
- A holiday from February 27th till March 3th

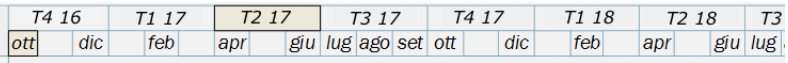
This table provides also indicative dates in which deliverables are validated and distributed.

Phase	Milestone	Description	Deliverables	Planned Date
Start project	M1	Project kick-off		07/10/2016
User Requirements		Analysis Document Approved	Planning document	19/10/2016
	M2	User requirements document approved	RASD (Requirements document)	02/11/2016
Software Requirements		Prototype Approved		26/11/2016
	M3	Software requirements document approved		16/12/2016
Architectural Design Document	M4	Architectural Design Document approved	Document Design – Test Plan document	11/10/2017
Detailed Design	M5	Coding complete	Codes and tests	18/04/2018
Transfer phase	M6	Acceptance test successful		14/06/2018
	M7	Product delivery complete	Finished product.	23/6/2018

Gant chart

The following is a Gantt chart to describe the chosen schedule for the project:

Activity	Start	End	Duration
Planning	**07/10/2016**	**02/11/2016**	**19g**
Problems analysis	07/10/2016	08/10/2016	2g
Domain analysis	09/10/2016	19/10/2016	7g
RASD and project planning	20/10/2016	02/11/2016	10g
Requirements Analysis	**03/11/2016**	**26/11/2016**	**18g**
Functional and non-functional requirements documentation	03/11/2016	09/11/2016	5g
Use cases analysis	10/11/2016	12/11/2016	3g
Activity diagram, sequence diagram, UX	13/11/2016	26/11/2016	10g
Detailed analysis	**27/11/2016**	**10/12/2016**	**10g**
Definition of functionalities and system limits	27/11/2016	07/12/2016	7g
Software modules analysis	08/12/2016	10/12/2016	3g
Logic layer analysis	11/12/2016	16/12/2016	4g
Design	**11/12/2016**	**11/01/2017**	**22g**
Project activities analysis	17/12/2016	17/12/2016	1g
Presentation design and service design	18/12/2016	31/12/2016	10g
Physical components analysis	01/01/2017	11/01/2017	7g
Development	**12/01/2017**	**14/06/2018**	**372g**
Coding	12/01/2017	18/04/2018	331g
Integration & system test	18/04/2018	12/06/2018	40g
Software installation	12/06/2018	14/06/2018	3g
Deployment	**15/06/2018**	**23/06/2018**	**7g**
System activation	15/06/2018	16/06/2018	2g
User training	15/06/2018	23/06/2018	7g
Maintenance	19/06/2018	08/09/2018	60g



Resource allocation

In this small group of two people, it is not necessary the presence of a Project Manager. This is due to the fact that the group is very limited and communication problem does not exist. Project decisions can be taken efficiently and in short time, simply meeting and discussing in group.
Since the team is composed by only two people, we have not decided to split the tasks and all of them are accomplished simultaneously. The team will work always together in order to improve project's efficiency and to have a continuous communication between the members. This kind of teamwork should reduce the misunderstandings in the group and the errors that may occur during the project. Moreover, the efficiency should increment because, in case a risk arises, the team can always face it together.

ID	Name and Surname, Activities List	Begin	End	Duration
1	**Federico Madaffari**	**07/10/2016**	**08/09/2018**	503g
2	Planning	07/10/2016	02/11/2016	19g
3	Requirements Analysis	03/11/2016	26/11/2016	18g
4	Detailed Analysis	27/11/2016	10/12/2016	10g
5	Design	11/12/2016	11/01/2017	22g
6	Development	12/01/2017	14/06/2018	372g
7	Deployment	15/06/2018	23/06/2018	7g
8	Maintenance	19/06/2018	08/09/2018	60g
9	**Claudio Mandrioli**	**07/10/2016**	**08/09/2018**	503g
10	Planning	07/10/2016	02/11/2016	19g
11	Requirements Analysis	03/11/2016	26/11/2016	18g
12	Detailed Analysis	27/11/2016	10/12/2016	10g
13	Design	11/12/2016	11/01/2017	22g
14	Development	12/01/2017	14/06/2018	372g
15	Deployment	15/06/2018	23/06/2018	7g
16	Maintenance	19/06/2018	08/09/2018	60g

Risk management

This section describes the possible risks from which the project might be threatened.
We have identified the following categories of risks:

1) Project Risks
2) Technical Risks
3) Business Risks

For each risk, a detailed description, a probability to occur, the action associated to prevent the risk and the impact of the risk are given.

Project Risks

We have reported only the important project risks that pose a threat to the project plan and could delay the project schedule or increase the overall costs.

Problem of communication

Description: Team members usually work separately. This can cause misunderstandings and conflicts about the division of different tasks.
Probability: Medium
Prevention: After a meeting, one person creates a report. Every participant and every person who should have been a participant of the meeting should get a copy of this report. Team members should not hesitate to ask and re-ask questions if things are unclear.
Correction: When there is a problem of communication, a new meeting is arranged in order to clarify.
Impact: Medium

Lack of time

Description: It can occur that the project requires more time than expected to be carried out.
Probability: High
Prevention: The time schedule is planned with care.
Correction: When tasks fail to be finished in time or when they are finished earlier than planned the project planning is adjusted. If lack of time becomes severe, the application is released as a beta version, after a consultation with the costumer, with only some functionalities offered while less essential features will be developed later.
Impact: High

Illness or absence of team members

Description: One or more of the team members can't work because of personal problems.
Probability: High
Prevention: Team members should warn timely a planned period of absence.
Correction: Since the team is composed by only two members and the illness period can't be forecast, when the time schedule is planned, it is needed to extend the delivery forecast.
Impact: High

Lack of knowledge of team members

Description: The team is formed by people who do not know each-other.
Probability: Low
Prevention: It must be considered the technical knowledge needed for developing the project.

Correction: If there is a lack of knowledge of a team member, it is necessary to train him/her for learning the knowledge necessary.
Impact: Low

The customer changes his mind about the requirements

Description: During the development of the project the requirements can change unexpectedly.
Probability: High
Prevention: It is obviously explained to the customer, that after he has accepted a version of the RASD, the RASD cannot be changed by the customer's wish only.
Correction: If the customer changes his mind during the development phase, his new requirements can be incorporated in the RASD, using a style of programming that takes advantage of reusable and extensible code.
Impact: Medium

Technical Risks

We have reported only the important technical risks that pose a threat to the quality of the software product complicating the implementation.

Data Loss

Description: A big portion od the application data gets lost
Probability: Low
Prevention: Store data in multiple locations and have one or more backup database.
Correction: Team members must operate timely to adjust the inconvenience. Also all the work, that depends on the loss data, should be halted until the error is corrected.
Impact: High

Server crash

Description: Upon being fully functional and operative, the system may experience brief or long periods of downtime
Probability: Low
Prevention: The risk could be made less likely by decoupling the clients' activities and making some clients independent from other components of the system.
Correction: Team members must operate timely to adjust the inconvenience.
Impact: Medium

User Interface mismatch

Description:
Probability: Low
Prevention: All User Interface are designed in advance in order to avoid any kind of mismatching.
Correction: When there is a user interface that is not suitable for the end-user, it is designed again.
Impact: Low

Component is complex to implement

Description:
Probability: Low
Prevention: A deep analysis of the product to develop must be done before the beginning of the development phase.
Correction: If a component is too complex to implement, maybe it can be split in other subcomponents in order to make the components easier to implement.
Impact: Medium

Business Risks

We have reported only the important business risks that pose a threat to the whole software product and its profitability.

Wrong budget estimation

Description: The budget is not enough to furtherly develop the software.
Probability: Low
Prevention: The project budget is estimated with care.
Correction: When the real budget needed is greater than the budget previously estimated, a new meeting with the costumer is planned in order to make the decision of make available a new amount of budget or not.
Impact: Medium

Bankruptcy

Description: The income from the usage of the application may not be sufficient to support maintenance and development of the system.
Probability: Medium
Prevention: A good feasibility study should help avoiding this critical situation.
Correction: None.
Impact: High

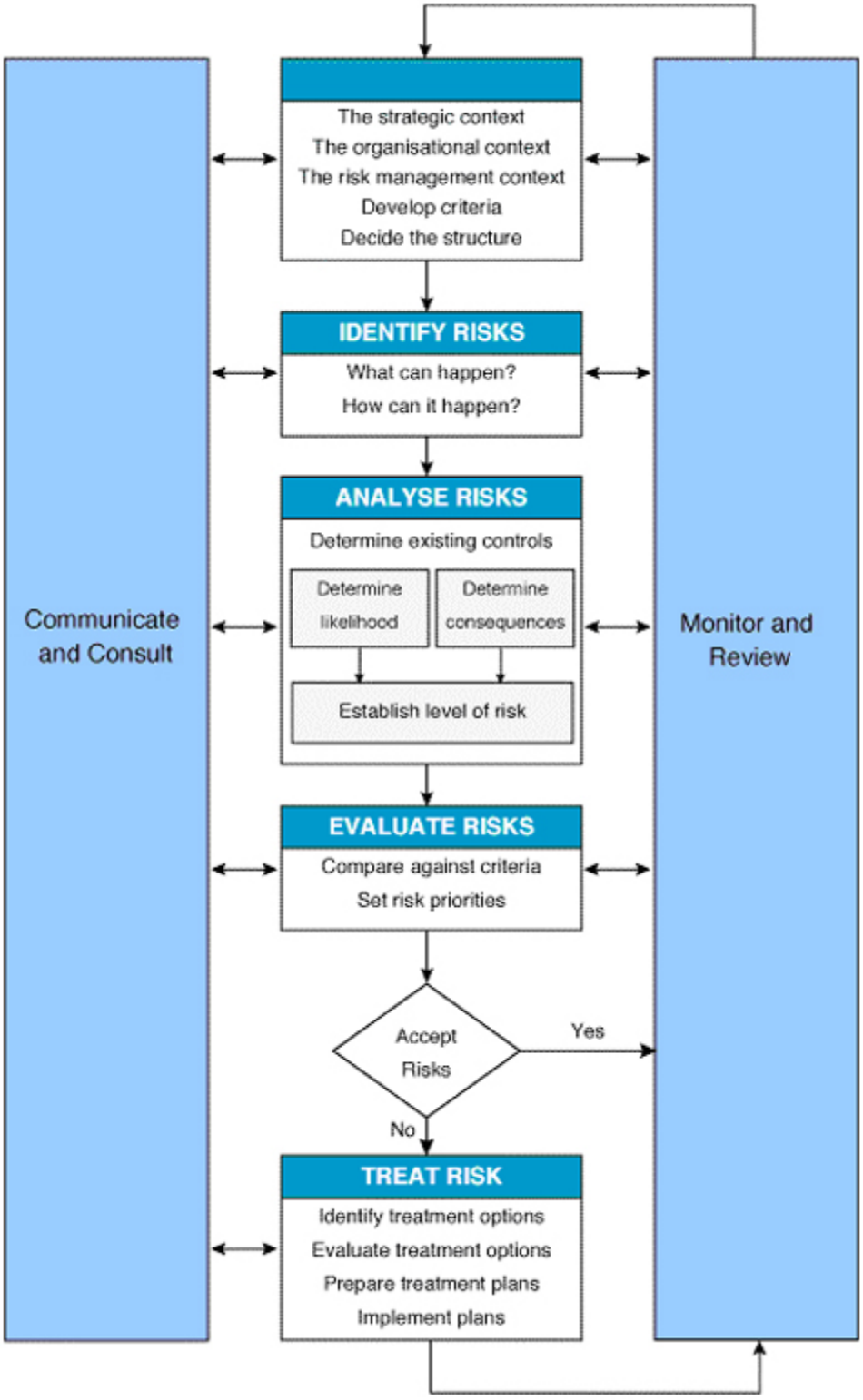
Competitors

Description: The same kind of service provided by the Power Enjoy system is offered by other competitors.
Probability: High
Prevention / Correction: Team members of PowerEnjoy must overcome the competitors by continuously providing updates and new functionalities to the application, based on customer feedback.
Impact: Medium

The risk Management Process

The risk management process is a set of procedures and practices for establishing, identifying, treating and monitoring all the risks about the project. The risk management contribute to identify the impact that could have the risks on the project.

The diagram below show how the risk management process works, explaining all the main phases that characterized it.



The risk management process is characterized by the following phases:

1. **Identify the risks:** this is the initial phase that can identify all the risks that can occur during the developing of the project.
2. **Identify the causes:** this phase aim to identify all the causes that can cause the risk.
3. **Analysis phase** that includes:
    a. **Likelihood estimation:** estimate the probability that a risk will occur. Generally, it is not a precise number but is enough generic (e.g. Low, Medium, High).
    b. **Consequences identification:** describe all the possible consequences that can happen when a risk occur, and how can influence the development of the project.
    c. **Risk level identification:** rate the level of the risk (e.g. Low, Medium or High).
4. **Risk Treatment:** the treatment of the risk includes all the possible solution that can be applied for resolving it.
5. **Monitor and Review:** monitoring of all risks and regular review of them.

Hours of work

Madaffari Federico

- 16/01/16 (3h) group work
- 18/01/16 (3h) group work
- 19/01/16 (2h)
- 22/01/17 (5h) group work

Mandrioli Claudio

- 16/01/16 (3h) group work
- 18/01/16 (3h) group work
- 19/01/16 (1h)
- 22/01/17 (5h) group work