POLITECNICO
MILANO 1863

Politecnico di Milano

COMPUTER SCIENCE AND ENGINEERING

Software Engineering 2

Project Plan

PowerEnJoy

Authors:
Francesco Fabiani
Jagadesh Manivannan
Niccolò Pozzolini

Professors:
Elisabetta Di Nitto
Luca Mottola

Academic Year 2016/2017

Contents

Chapter 1

Introduction

1.1 Revision history

Version	Date	Authors	Summary
1.0	22/01/2017	Fabiani, Manivannan, Pozzolini	Initial release

Table 1.1: Changelog of this document

1.2 Purpose and scope

The Project Plan (PP) document is intended to describe the best strategies for the management of PowerEnJoy with regards to all the aspects of the project, such as costs, schedule of the activities, resource allocation and effort estimation.

The product described is PowerEnJoy, a car-sharing service which offers to its users exclusively electric cars. It includes the common functionalities of its category: permitting to registered users to obtain the position of all the available cars, reserving one within a certain amount of time and continuously displaying the up-to-the-minute cost of the ride are just few of them. Moreover, PowerEnJoy stimulates users to behave virtuously towards the ecosystem by applying various types of discounts under specific conditions.

1.3 Definitions, acronyms, abbreviations

- *ACAP*: Analyst Capability

- *APEX*: Applications Experience
- *API*: Application Programming Interface
- *BCE*: Business Controller Entity
- *Car*: electric vehicle provided by the service
- *CPLX*: Product Complexity
- *DAG*: Directed Acyclic Graph
- *DB*: Database
- *DBMS*: Database Management System
- *DD*: Design Document
- *DOCU*: Documentation Match to Life-Cycle Needs
- *ER*: Entity-Relationship
- *GPS*: Global Positioning System
- *Guest* or *Guest user*: person not registered to the service
- *ITPD*: Integration Test Plan Document
- *LTEX*: Language and Tool Experience
- *MVC*: Model View Controller
- *OS*: Operating System, related both to desktop and mobile platforms
- *PCAP*: Programmer Capability
- *PCON*: Personnel Continuity
- *PIN*: Personal Identification Number
- *PLEX*: Platform Experience
- *PP*: Project Plan
- *PVOL*: Platform Volatility
- *RASD*: Requirements Analysis and Specification Document
- *Registered user*: see *User*

- *RELY*: Required Software Reliability
- *REST*: Representational State Transfer
- *RESTful*: that follows the REST principles
- *RUSE*: Developed for Reusability
- *Safe area*: set of parking spots where a user can leave a car without penalization
- *STOR*: Main Storage Constraint
- *User*: person with a valid driving license registered to the service
- *UX*: User eXperience
- *W3C*: World Wide Web Consortium

1.4 Reference documents

The PP document has been composed following the guidelines reported in the Requirements Analysis and Specification Document delivered for this project. Moreover, the part describing the cost estimation follows the indications described in the second revision of the procedural software cost estimation model named Constructive Cost Model (COCOMO II), developed by Barry W. Boehm.

With regards to the course named Software Engineering 2 and held by professors Luca Mottola and Elisabetta Di Nitto (Politecnico di Milano, a. y. 2016/17), the document conforms to the guidelines provided during the lectures and within the material of the course.

Chapter 2

Project size, cost and effort estimation

2.1 Size estimation

2.1.1 Internal logic files (ILFs)

The application includes a number of ILFs that will be used to store the information about users, payments, reservations, rides, cars, parking slots and discounts. Their weights will be calculated according to their structure's complexity. Most of them have a simple structure consisting in two or three fields, so we will assign simple weights to these entities.

Reservation has a more complex structure: it saves the user who creates the reservation, the car involved, the timestamp of the reservation and its status (RESERVED, RIDING, EXPIRED, COMPLETED, where COMPLETED means that the driving process has been successfully concluded), so we will assign a medium weight to this entity. The same weight will be applied to RIDE, that saves the references to the reservation and the user, the timpestamp of the ride's beginning, and its status (RIDING, COMPLETED).

The most complex entity we have is USER, that saves all the personal infos and contacts (used by the notification component), its balance, and the possibly empty references to the ongoing ride and relative reservation. We will then assign a complex weight to it.

ILF	Complexity	FPs
RESERVATION(user, car, time, status)	Medium	10
RIDE(user, resv, starttime, status)	Medium	10
USER(data, balance, resv, ride)	Complex	15
PAYMENT(user, amount)	Simple	7
CAR(id, loc, status)	Simple	7
PARKING(id, car)	Simple	7
DISCOUNT(id, spec)	Simple	7
Total		63

2.1.2 External Logic Files (ELFs)

The system interacts with the licence office system, but as explained in the RASD we designed it as an external inquiry: we upload the licence provided by the user retrieving a STRING response. We can then skip the licence office in this section. The system also interacts with the payment service, but this interaction works like an external inquiry, so it won't be calculated in this section.

2.1.3 External Inputs (EIs)

PowerEnJoy supports many kind of interactions, involving different entities, which are described below.

- *Register*: the registration involves just the user entity, but since it implies to interact with the licence office system, we will assign a medium weight to this operation.
- *Login, logout, update licence*: these operations just involve the user entity, so we will assign them simple weights.
- *Pay*: payment implies the interaction with the payment system through the payment gateway, plus updating the payment and user entities. We will then assign a medium weight to this operation.
- *Reserve car, abort reservation*: these operations involve three entities (reservation, car, user) and are two of the most dangerous operations concerning the database integrity. Complex weights will then be assigned.
- *Open car, end ride*: these operations involve four entities (reservation, car, user, ride) and as the previous two, these dangerous operations

concerning the database integrity. Complex weights will then be assigned.

- *Plug*: this operation is not trivial even if just two entities are involved, mid weight will be assigned.
- *Ban user, add/remove car*: the operations of adding/removing a car or banning a user (possible only for admin users) will imply other operations in the real world, but according to the system its still a basic operation involving just one entity. Simple weight will be assigned.

EI	Entities involved	Complexity	FPs
Register	User	Middle	4
Login	User	Simple	3
Logout	User	Simple	3
Update licence	User	Simple	3
Pay	Payment, User	Middle	4
Reserve car	Reservation, Car, User	Complex	6
Abort reservation	Reservation, Car, User	Complex	6
Open car	Reservation, Car, User, Ride	Complex	6
End ride	Reservation, Car, User, Ride	Complex	6
Plug	Car, Parking	Middle	4
Ban user	User	Simple	3
Add/remove car	Car	Simple	3
Total			51

2.1.4 External Inquiries (EQs)

As specified by the function points guidelines, an inquiry is essentially a data retrieval request performed by an user. PowerEnJoy supports a few interactions of this type that don't require complex computations:

- *Get available cars (zone)*: this operation retrieves the available cars given a zone (the visible area in the map). It just involves the car entity, simple weight will then be applied.
- *Get payment history*: this operation retrieves the history of user's payments. It just involves just the payment entity, simple weight will then be applied.

- *Get active reservation*: this operation retrieves, if there is one, the user's active reservation It involves reservation and user entities but its still a simple process, simple weight will then be applied.
- *Get available discounts*: this operation retrieves the list of available discounts. It involves just the discount entity, simple weight will be applied.

EQ	Entities involved	Complexity	FPs
Get available cars (zone)	Car	Simple	3
Get payment history	Payment	Simple	3
Get active reservation	Reservation, User	Simple	3
Get available discounts	Discount	Simple	3
Total			12

2.1.5 External Outputs (EOs)

As part of its normal behaviour, our system needs to communicate with the user outside the context of an inquiry in the following occasion:

- *Notify*: Despite how it may sound, notifying events is a quite complex operation in our system. There are different types of notification, different output flows (SMS, e-mail, push notification) and many events to be notified; a specific component (NotificationHelper) is designed to dispatch every notification according to its type and user settings.

EO	Entities involved	Complexity	FPs
Notify	User	Complex	7
Total			7

2.1.6 Overall estimation

The following table summarizes the results of our estimation activity:

Function type	Value
Internal Logic Files	63
External Logic Files	0
External Inputs	51
External Inquiries	12
External Outputs	7
Total	133

Considering Java Enterprise Edition as a development platform and disregarding the aspects concerning the implementation of the mobile applications (which can be thought as pure presentation with no business logic), we can estimate the total number of lines of code.

Depending on the conversion rate, we have a lower bound of

$$KSLOC = 133 \times 51 = 6783$$

and an upper bound of

$$KSLOC = 133 \times 73 = 9709$$

2.2 Cost and effort estimation

In this section we are going to use the COCOMO II approach to estimate the cost and effort needed to develop the PowerEnJoy application.

2.2.1 Scale drivers

In order to evaluate the values of the scale drivers, we refer to the following official COCOMO II table:

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	5.0	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	Level 1 lower	Level 1 upper	Level 2	Level 3	Level 4	Level 5
SF_j	7.80	6.24	4.68	3.12	1.56	0.00

Table 2.1: Scale Factor values (SF_j) for COCOMO II Models

A brief description for each scale driver:

- *Precedentedness*: this factor determines or reveals the level of exposure or experience in development of large scale projects or similar kind of projects that out team has done before. Since we have developed few projects like this, we can set this value to be Nominal.
- *Development flexibility*: it determines the degree of flexibility in the development process with respect to the external specification and requirements. In our project, the functionalities and requirements are clear and well defined with no specific mention about the technology. Hence this value would be low.
- *Architecture/Risk resolution*: it determines the level of awareness and reactivity with respect to risks. Since we have an extremely good risk management plan, we consider this value to be very high.
- *Team cohesion*: it determines if all the Stakeholders are able to work in a team and share same vision and commitment. Since our team is highly co-operative, the value is very high.
- *Process maturity*: we have a done an extremely fair work to meet our goals successfully in this project. Since we had prior experience in successfully dealing these kind of projects, the value is set to Level 4.

The results of our evaluation is the following:

Scale Driver	Factor	Value
Precedentedness (PREC)	Nominal	3.72
Development flexibility (FLEX)	Low	4.05
Risk resolution (RESL)	Very high	1.41
Team cohesion (TEAM)	Very high	1.10
Process maturity (PMAT)	Level 4	1.56
Total		11.84

2.2.2 Cost drivers

Product factors

- *Required Software Reliability (RELY)*:

  The software application is developed in such a way that the main aim is to reserve and take a ride in the Cars in the city. Any malfunctioning could lead to important financial loss. Considering this, the RELY cost driver is set to high.

RELY cost drivers						
RELY descriptors	slightly inconvenience	easily recoverable losses	moderate recoverable losses	high financial loss	risk to human life	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- *Database size (DATA)*:

  This factor considers the effective size of our database. We do'nt know this value exactly. But based on the lower and upper bound values of the SLOC, which is 10.000-15.000 SLOC, we can estimate roughly that our system can reach a 3GB database size. Since it is distributed over 10.000-15.000 SLOC, the ratio D/P (measured as testing DB bytes/program SLOC) is between 209 and 314, resulting in the DATA cost driver being high.

DATA cost drivers						
DATA descriptors		$\frac{D}{P} < 10$	$10 \leq \frac{D}{P} < 100$	$100 \leq \frac{D}{P} < 1000$	$\frac{D}{P} \geq 1000$	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- *Product complexity (CPLX)*:

  This factor is related to the complex logics involved in implementing the product as a whole. Hence, we set it to very high according to the CPLX cost driver table.

CPLX cost drivers						
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	0.73	0.87	1.00	1.17	1.34	1.74

- *Developed for Reusability (RUSE)*:

  In our project, we use many individual piece of codes that can be made reusable for other services or functions. Hence the RUSE cost driver is set to nominal.

RUSE cost drivers						
RUSE descriptors		None	Across project	Across program	Across product line	Across multiple product
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- *Documentation Match to Life-Cycle Needs (DOCU)*:

This factor describes the relationship between the documentation and the application requirements. The product life-cycle needs are explicitly mentioned clearly in the documentation. Hence the DOCU cost driver is set to nominal.

DOCU cost drivers						
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 level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	0.81	0.91	1.00	1.11	1.23	n/a

Platform factors

- *Execution Time Constraint (TIME)*:

  This factor describes the approximated value of CPU usage with respect to the hardware specifications. Our PowerEnJoy application has vast functionalities as a software and hence the TIME cost driver is set to be very high.

TIME cost drivers						
TIME descriptors			$\leq$ 50% use of available execution time	70% use of available execution time	85% use of available execution time	90% use of available execution time
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- *Main Storage Constraint (STOR)*:

  This factor describes the approximated storage space with respect to the hardware specifications. Our PowerEnJoy application has

vast functionalities as a software. Keeping this in mind, the disk drives can store up to enough terabytes and hence the STOR cost driver is set to be high.

STOR cost drivers						
STOR descriptors			$\leq$ 50% use of available storage	70% use of available storage	85% use of available storage	90% use of available storage
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- *Platform Volatility (PVOL)*:

  This factor describes the change in the basic or fundamental platform in which the system is designed. We do'nt change the platform often except for very few major releases or updated requested by the client. This will be done approximately for every 5months to be in sync with the latest evolving or trending technologies. Hence, the PVOL cost driver is set to nominal.

PVOL cost drivers						
PVOL descriptors		Major change every 12 months; minor change every 1 month	Major change every 6 months; minor change every 2 weeks	Major change every 2 months; minor change every 1 week	Major change every 2 weeks; minor change every 2 days	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	n/a	0.87	1.00	1.15	1.30	n/a

- *Analyst Capability (ACAP)*:

This factor describes the potential analysis that has been done with respect to the potential implementation in real world. Since we have done a regressive analysis, the ACAP cost driver is set to be high.

ACAP cost drivers						
ACAP descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- *Programmer Capability (PCAP)*:

  This factor describes the ability of the programmer to do a work without much difficulty. Our project has not been implemented yet our programmers have executed several projects like this successfully and hence the PCAP cost driver is set to be high.

PCAP cost drivers						
PCAP descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- *Applications Experience (APEX)*:

  Our team members are quite experienced with this kind of project development and hence the APEX cost driver is set to be high.

APEX cost drivers						
APEX descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- *Platform Experience (PLEX)*:

  Our team has a good and stable experience in Java EE platform and also a good knowledge about the integration with UI, Database and other tiers. Hence the PLEX cost driver is set to be high.

PLEX cost drivers						
PLEX descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- *Language and Tool Experience (LTEX)*:

  As we have mentioned before, since the knowledge of our programmers are good enough on this kind of project and Java EE platform, they possess a good standard of using tools in the development environment, server side and client side integration,etc. Hence the LTEX cost driver is set to be high.

LTEX cost drivers						
LTEX descriptors	≤ 2 months	6 months	1 years	3 years	6 years	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.20	1.09	1.00	0.91	0.84	n/a

- *Personnel Continuity (PCON)*:

  This factor describes the personnel turnover annually. Since our project is a short term project, the PCON cost driver is set to be very low.

PCON cost drivers						
PCON descriptors	48%/year	24%/year	12%/year	6%/year	3%/year	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.29	1.12	1.00	0.90	0.81	n/a

Project factors

- *Use of Software Tools (TOOL)*:

  Our application environment is complete and well integrated, so we will set this parameter as high.

TOOL cost drivers						
TOOL descriptors	edit, code, debug	simple, front-end, back-end CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- *Multisite Development (SITE)*:

Our application is designed in such a way that it relies on wide-band electronic communication at extremely good speeds (e.g. 3G, 4G) for connection. Hence the SITE cost driver is set to be very high.

SITE cost drivers						
SITE collocation descriptors	international	multi-city and multi-company	multi-city or multi-company	same city or metro area	same building or complex	fully collocated
SITE communications descriptors	some phone, mail	individual phone, FAX	narrow band email	wideband electronic communication	wideband electronic communication, occasional video conference	interactive multimedia
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.22	1.09	1.00	0.93	0.86	0.80

General factor

- *Required Development Schedule (SCED)*:

  The efforts was distributed or split equally in our project for all the documentation, yet there were certain time consuming process in analysing and development of the RASD and the DD documents for precision. Hence, the SCED cost driver is set to be high.

SCED cost drivers						
SCED descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating level	Very low	Low	Nominal	High	Very high	Extra high
Effort multipliers	1.43	1.14	1.00	1.00	1.00	n/a

Results

Overall our results are expressed in the following table:

Cost Driver	Factor	Value
Required software reliability (RELY)	High	1.10
Database size (DATA)	High	1.14
Product complexity (CPLX)	Very high	1.34
Required reusability (RUSE)	Nominal	1.00
Documentation match to life-cycle needs (DOCU)	Nominal	1.00
Execution time constraint (TIME)	Very high	1.29
Main storage constraint (STOR)	High	1.11
Platform volatility (PVOL)	Nominal	1.00
Analyst capability (ACAP)	High	0.85
Programmer capability (PCAP)	High	0.88
Application experience (APEX)	High	0.88
Platform Experience (PLEX)	High	0.91
Language and Tool Experience (LTEX)	High	0.91
Personnel continuity (PCON)	Very low	1.12
Usage of Software Tools (TOOL)	High	0.90
Multisite development (SITE)	Very high	0.86
Required development schedule (SCED)	High	1.00
Total		1.13694

2.2.3 Effort equation

This final equation gives us the effort estimation measured in Person-Months (PM):

$$Effort = A \times EAF \times KSLOC^{E}$$

where:

$A = 2.94$ (for COCOMO II)

$EAF = 1.13694$ (product of all cost drivers)

$E = B + 0.01 \times \sum_{i} SF[i] = B + 0.01 \times 11.84 = 0.91 + 0.1184 = 1.0284$

(exponent derived from the scale drivers, with B = 0.91 for COCOMO II)

With this parameters we can compute the effort value, which has a lower bound of:

$$\begin{aligned} Effort = A \times EAF \times KSLOC^E &= 2.94 \times 1.13694 \times 6.783^{1.0284} \\ &= 23.94 \text{ PM} \approx 24 \text{ PM} \end{aligned}$$

and an upper bound of:

$$\begin{aligned} Effort = A \times EAF \times KSLOC^E &= 2.94 \times 1.13694 \times 9.709^{1.0284} \\ &= 34.61 \text{ PM} \approx 35 \text{ PM} \end{aligned}$$

2.2.4 Schedule estimation

Regarding the final schedule, we are going to use the following formula:

$$Duration = 3.67 \times Effort^F$$

where:

$$F = 0.28 + 0.2 \times (E - B) = 0.28 + 0.2 \times 0.1184 = 0.31368$$

As a lower bound, we consider

$$\begin{aligned} &Effort = 23.94 \text{ PM} \\ &Duration = 3.67 \times 23.94^{0.31368} = 9.94 \text{ months} \end{aligned}$$

while as an upper bound, we consider

$$\begin{aligned} &Effort = 34.61 \text{ PM} \\ &Duration = 3.67 \times 34.61^{0.31368} = 11.15 \text{ months} \end{aligned}$$

Chapter 3

Schedule

The main tasks involving this project are:

- Delivering the Requirement Analysis and Specification Document, containing the goals, the domain assumptions, and the functional and non-functional requirements of the software system.
- Delivering the Design Document, containing the architecture and the design of the software system.
- Delivering the Integration Testing Plan Document, containing the strategy used to perform integration testing on the system.
- Delivering the Project Plan, which is this document.
- Preparing a brief presentation about the delivered documents, with slides.
- Implementing the software system and write unit tests.
- Performing integration testing on the system.

Please note that, as new requirements can emerge, new choices are made and the development goes on, the process can be iterated multiple times. In particular, unit and integration testing will be continuously performed throughout the development process.

However, some tasks need to be concluded before some other can begin: the dependency graph for the activities is shown in figure 3.1.

The first five tasks for the project are already defined by the document about describing the assignment, together with the deadlines for the delivery of the RASD, the Design Document and the ITPD. The date for the presentation is also fixed. So, those activities are already scheduled.

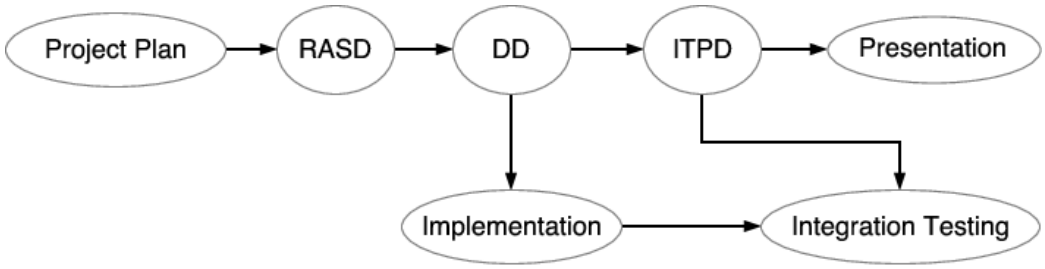

Figure 3.1: DAG for the dependencies among tasks

There are no fixed deadlines, instead, for the development of the software. Based on the COCOMO estimation performed in chapter 2, we expect the entire project to last about 10 months, so it will be presumably finished by August 2017. The schedule for our project is outlined in table 3.1, while figure 3.2 shows the Gantt chart for PowerEnJoy.

Activity	**Start date**	**Deadline**
RASD	16/10/2016	13/11/2016
DD	14/11/2016	11/12/2016
IPTD	12/12/2016	15/01/2017
Project Plan	05/01/2017	22/01/2017
Presentation	23/01/2017	20/02/2017
Implementation	21/02/2017	26/07/2017
Integration testing	27/07/2017	20/08/2017

Table 3.1: Schedule for the project tasks

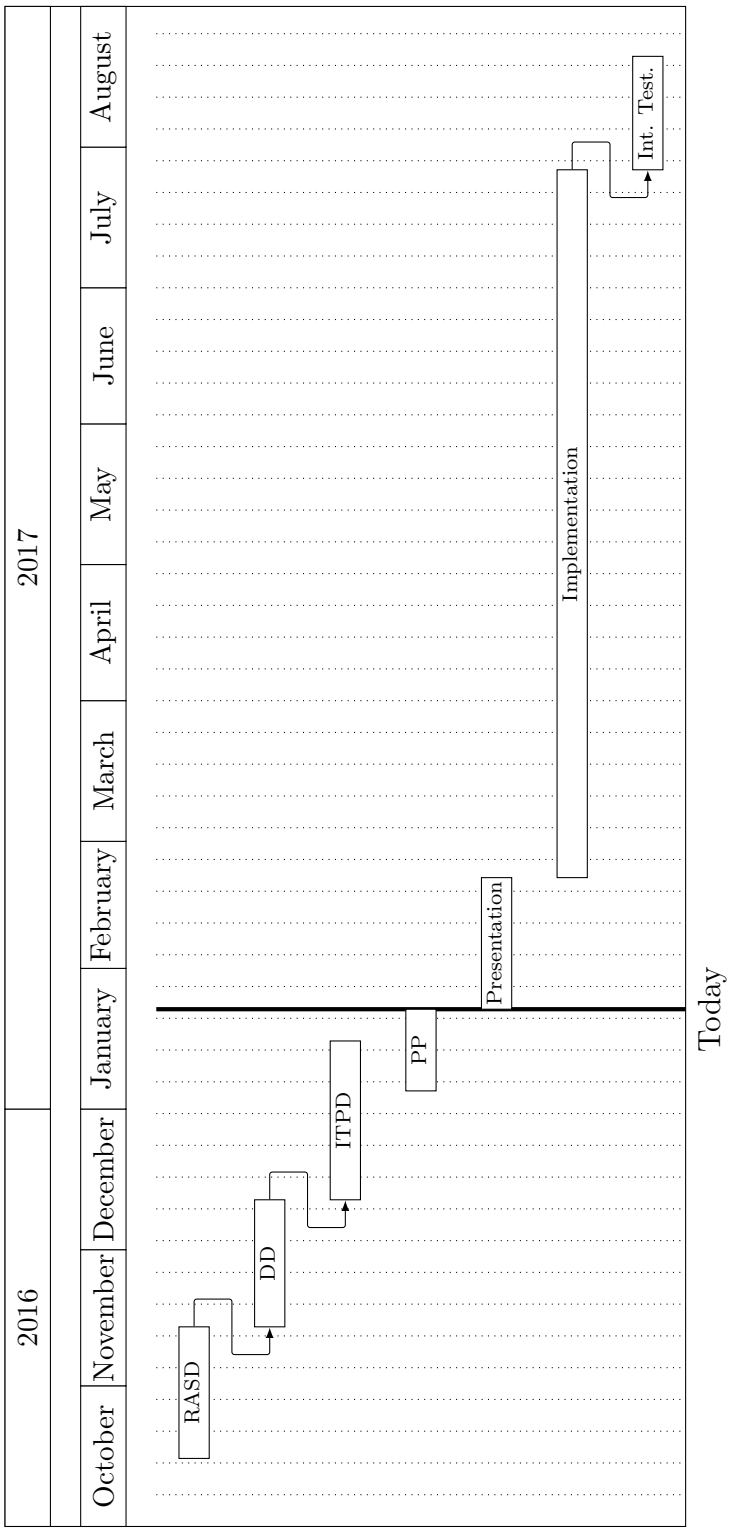

Figure 3.2: Gantt chart of the project

Chapter 4

Resource allocation

This chapter has the purpose to show how the available resources are allocated to the project. It does not aim to be a very specific schedule, in fact the tasks are grouped with a high level of abstraction and resources are assigned to these high-level tasks. This is because micromanaging the work of people on a large project is of little use.

The resulting schedule gives to all the team members a general overview on the whole project: every task involves the contribution of each member. This enlarges slightly the time needed for the completion of the project, but also induces a greater awareness of all the members of the team. It also enables a more accurate control on the work of each other in order to limit the number of bugs and misunderstandings.

The documents to be written related to the project are divided into arguments which mainly reflect the division in chapters; each argument will be assigned to two people and the third will have the duty to revision it.

The code will be divided into tiers which reflect the system architecture. Every tier will be assigned to two people; the third will write and execute unit tests.

The division of the work is elaborated according to the schedule (table 3.1 and figure 3.2).

Tables 4.1, 4.2, 4.3, 4.4, 4.5 show the division of work among the team members.

Resource	**16/10/2016 to 13/11/2016**			
	1^{st} week	2^{nd} week	3^{rd} week	4^{th} week
Fabiani	Introduction	Overall description	System features	Revision specification requirements
Manivannan	Overall description	Specification requirements	System features	Revision introduction
Pozzolini	Specification requirements	Introduction	Revision overall description	Revision System features

Table 4.1: Resource allocation for RASD

Resource	**14/11/2016 to 11/12/2016**			
	1^{st} week	2^{nd} week	3^{rd} week	4^{th} week
Fabiani	Architectural design	Algorithm design	UI design	Revision introduction
Manivannan	Algorithms	Introduction - Requirements traceability	Revision Architectural design	Revision UI design
Pozzolini	Requirements traceability	Architectural design	UI design	Revision algorithms

Table 4.2: Resource allocation for DD

Resource	**12/12/2016 to 15/01/2017**			
	1^{st} week	2^{nd} week	3^{rd} week	4^{th} week
Fabiani	Integration strategy		Individual steps	
Manivannan	Introduction - minor chapters		Individual steps	
Pozzolini	Individual steps		Revision individual steps	

Table 4.3: Resource allocation for ITPD

Resource	**05/01/2017 to 22/01/2017**	
	1_{st} week	2_{nd} week
Fabiani	Cocomo II	Risk
Manivannan	Functions points	Resource allocation
Pozzolini	Cocomo II	Task and schedule

Table 4.4: Resource allocation for PP document

Resource	**04/02/2017 to 20/02/2017**	
	1^{st} week	2^{nd} week
Fabiani	Slides	Revision slides
Manivannan	Slides	Revision slides
Pozzolini	Slides	Revision slides

Table 4.5: Resource allocation for presentation

Chapter 5

Risk management

Project scheduling could be an important source of problems. The schedule defined in the Project Plan can't of course forecast every possible issue that may occur during the following steps. To mitigate this kind of risk, some extra time has been allocated to the most critical activities to allow for adjustments. One of them concerns the payment platform.

The work concerning the integration of the payment system remains quite vague at the time of the planning. As a consequence, it is not possible to estimate, with a reasonable level of confidence, that the needed effort remains low. To mitigate this risk we decided to apply an higher weight to that task in the Function Points calculation.

Also changes in the environment, both inside and outside our company, may be sources of problems. The main threat coming from the outside are law changes in our sector; notwithstanding we estimate this risk to be very low, even if very dangerous if it would occur. The positive side is that the law - at least in this country - moves so slow that we will have much time to build up a strategy.

From the inside the main threat is the fluctuating IT job market. We must build up a team able to fulfil each task even if some member will be absent for a prolonged period of time. The point then is to avoid to have "key members", or at least to put beside them other people to learn the job. The only task where we can accept to have key members is the database interactions inspection. Our database in fact implements an important number of tables and and intense query execution, so the integrity plays an important role. Experts should then be involved if not present in the team.