POLITECNICO
MILANO 1863

POWER ENJOY

Project Plan Document

Lorenzo Casalino - 877421

Tommaso Castagna - 792326

Document version 1.1

Contents

1 Introduction

1.1 Revision History

The history of document revisions is here recorded in tabular format, mapping the document version with the changes brought to document itself.

The current version of the document is highlighted by the version number in bold format.

Version	Revision
1.0	First released version.
1.1	Added relevant definitions, acronyms and reference documents. Added some clarifications and corrected few typos.

1.2 Purpose

The hereby document has the purpose to support critical project decision, such as budget and human resources allocation, providing estimation for the project's size, the effort required, expected costs and duration.

1.3 Scope

The project planning document gives a first and broad estimation of the costs, time and effort required to develop the final product. All these aspects are faced and explored in details in each of the 4 main sections composing this document:

- **Size, effort and cost estimations**: estimated figures for size, costs, effort and time required to accomplish the final goal of the project are here presented, describing in details the estimation process and rationale followed, along with the mathematical tools exploited.
- **Schedule**: using as an input the outcomes derived by the estimation activities presented in the previous section, a first detailed scheduling of project's tasks is provided.
- **Resource allocation**: here the mapping between members of the team and the various task composing the whole project's life-cycle is shown.
- **Risk management**: any risk that may arise in any phase of the project is described and a rough possible solution, either to avoid the risk and either to recover the situation in case it occurs, is presented.

1.4 Definitions, Acronyms and Abbreviations

1.4.1 Definitions

- **Cost Driver**: team or project factor having a relevant impact on the estimated costs of the project.
- **External Input**: elementary process in which data crosses the boundary from outside to inside. The data contained in the external input may be used to maintain one or more internal logical files.
- **External Inquiry**: elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files, without their modification.
- **External Interface File**: a user identifiable group of logically related data used by the application to provide part of its functionalities. The data resides entirely outside the application boundary.
- **Internal Logical File**: a user identifiable group of logically related data that resides entirely within the application boundary, usually mantained by mean of external inputs.
- **External Output**: elementary process in which data generated by ILFs elaborations passes across the boundary from inside to outside.
- **Gearing Factor**: conversion factor used to convert in SLOC the size estimation expressed in function points.
- **Kilo-Source Lines Of Code**: metric identical to the SLOC but the number of code lines is expressed in thousands.
- **Partial Function Points**: function points estimation related to a given project's functionality.
- **Persons-Month**: metric used to measure the effort required for project's development.
- **Source Lines Of Code**: a metric used to estimated the size of a software project by the number of expected lines of source code.
- **Scale Driver**: project or team dependent factor having a relevant impact on the estimated size of the project itself.
- **Total Effort Multiplier**: multiplier encasing the contributions brought by each cost driver.
- **Total Scale Driver**: the total scale driver factor impacting on the project's size.
- **Unadjusted Function Points**: simple equation used to compute the total amount of function points assigned to the project under analysis.

1.4.2 Acronyms

- **EI**: External Input.
- **EIQ**: External Inquiry.
- **EIF**: External Interface File.
- **ILF**: Internal Logic File.
- **KSLOC**: Kilo-Source Lines Of Code.
- **EO**: External Output.
- **PFP**: Partial Function Points.
- **PM**: Persons-Month.
- **SLOC**: Source Lines Of Code.
- **TEM**: Total Effort Multiplier.
- **UFP**: Unadjusted Function Point.

1.5 Reference Documents

- PowerEnjoy: Requirements Analysis & Specification Document.
- PowerEnjoy: Design Document.
- Cocomo II: Model definition Manual, CSSE.
- MyTaxiService: Project plan document, Fabrizio Casati, Valerio Castelli.

2 Project Size, Effort and Cost Estimation

In this section the estimation process of the three key aspects for an effective project planning, namely **size**, **effort** and **costs** expected, is described in details, pointing out the rationale of each single step of the process itself.

«««< HEAD The **size estimation** process is led by a **functionality-provided** based approach, whereby the estimation is performed according to the functionalities that the software product is planned to provide. To support this strategy, **Function Points** technique is used. ======= The size estimation process is led by a **functionality-provided** based approach, whereby the estimation is made according to the functionalities that the software product is planned to provide. To support this strategy, *Function Points* technique is used. »»»> refs/remotes/origin/master

Regarding the **effort and cost** estimation, the evaluation is based on **COCOMO II**, an **algorithmic model** based on a simple equation which output depends on several factors regarding the project.

All the estimations described in the following sections are performed according to the **Early-design model**. We have chosen to adopt this model since we are asked to implement a new software service, with any legacy software/hardware to integrate, or pre-requisite regarding the software architecture.

2.1 Size Estimation

As explained in the introductory paragraph to this section, the size estimation effort is based on the estimation of the so-called *Function Points*.

Function points are a statistical method of estimate the size of a software project evaluating the different functionalities provided by the software product in exam.

According to this approach, functionalities are divided into 5 **function types**, or categories:

- **Internal Logical File**: a user identifiable group of logically related data that resides entirely within the application boundary, usually mantained by mean of external inputs.
- **External Interface File**: a user identifiable group of logically related data used by the application to provide part of its functionalities. The data resides entirely outside the application boundary.
- **External Input**: elementary process in which data crosses the boundary from outside to inside. The data contained in the external input may be used to maintain one or more internal logical files.
- **External Output**: elementary process in which data generated by ILFs elaborations passes across the boundary from inside to outside.

- **External Inquiry**: elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files, without their modification.

For each of these categories, a **weight** is associated. These weights are statistically determinated and vary according to the **complexity** of the function type.

The complexity of a function type can be derivided consulting the related **rating tables**.

In order to retrieve the number of function points assigned to the software product given a specific function type, a simple equation is applied:

$$PFP_t = N_t * FP_t \tag{1}$$

$$t \in T = \{ILF, EIG, EI, EO, EQ\}$$

This equation (1) returns the **partial function points** obtained by multiplying the number of functionalities of a certain catergory for the weight associated to that category.

$$UFP = \sum_t FPF_t, t \in T \tag{2}$$

The total number of function points assigned to the whole project is computed by the **Unadjusted Function Points** equation (2), which simply compute the sum of each PFP previously calculated.

2.1.1 Internal Logic File

- **User account informations**: this file contains the informations that any user is asked to provide at registration time to the service.

  The informations (Name, surname, password, email, telephone number, postal code, city, birthday, driving license, payment informations) are recorded inside one type of record.

- **Operator account informations**: the purpose is similar to the logic file storing the user account informations, but with the difference of not containing the driving license and the payment informations. Only one type of record is exploited.

- **Priviledges**: to avoid the execution of actions not allowed to some category of users, this file is used to map the priviledge level associated to each account category. Only one record type is used.

- **Parking area informations**: area informations, such as area identifier and coordinates describing the geographical boundaries of the parking area, are stored in this internal file, by means of one record type.
- **Safe parking area informations**: extension of the previous internal logic file, which adds the presence of the total number of parking spots inside the area.
- **Vehicle informations**: this file contains both the usual informations characterizing a vehicle (Plate number, frame number, model, matriculation date, fuel type, shift type) and the informations regarding the actual status of the vehicle (Fuel percentage, availability, position). Two records are used to store, respectively, the type of informations described above.
- **Reservation informations**: the informations regarding each reservation is stored by the support of three records.

  The first record stores an identificator to the reservation, the begin and end date of the reservation, the identificator of the reserving user, the plate number of the reserved vehicle and the total charge.

  The second structure is used to keep track of events that trigger a policy rule. An identificator to the event, the identificator of the policy rule, event date, event condition and effect are stored.

  The third record is used to map a reservation to one or more event.
- **Policy rules**: to implement the detection of good or bad behaviours, according to the policy adopted by the car sharing society, the rules are encoded and stored in an appropriated format.

  This logic file serves this purpose, using one record structure to store the rule identificator, the encoded conditions and encoded effects.
- **Mantainance tasks**: mantainance tasks assigned to a certain operator are stored by this logic file, using two record type.

  The first used to store the identificator task and its description, while the second is used as a support, mapping tasks to a specific operator using his/her identificator.

	Data elements		
RET	$1 \rightarrow 19$	$20 \rightarrow 50$	51+
1	Low	Low	Average
$2 \rightarrow 5$	Low	Average	High
6+	Average	High	High

Table 2: ILF and EIF's rating table

Making reference to the ILF rating table, we can evaluate the partial function point value:

ILF	Complexity	Weigth
User account informations	Low	7
Operator account informations	Low	7
Priviledges	Low	7
Parking area informations	Low	7
Safe parking area informations	Low	7
Vehicle informations	Low	7
Reservation informations	Low	7
Policy rules	Low	7
Mantainance tasks	Low	7
Partial Function Point	63	

Table 3: ILF's PFP table

2.1.2 External Interface File

The external services used by our system are developed by third parts. Consequently, it's quite hard to understand how many record types and data elements of each external file are used.

In order to perform meaningful estimations, the following assumptions about the external services are taken in consideration.

- **Driver licenses validator web service files**: we assume that the web service needs only informations regarding stored driver licenses and drivers informations, hence use of two record element types.
- **Payment system web service files**: as for the previous web service, it's reasonable think that the payment web service relies only the payment informations and payment service's customers informations.
- **Google maps web service files**: the kind of informations needed by this service are unknown to us, but we assume that the operations performed for map and vehicle positions representation involves a complex external file.

The weigths associated to each of the above EIF can be retrieved in the table 2.

EIF	Complexity	Weight
Driver licenses validator web service	Low	5
Payment system web service	Low	5
Google maps web service	High	10
Partial Function Point	20	

Table 4: EIF's PFP table

2.1.3 External Input

- **Login**: elementary operation that involves the user account informations (or the operator account informations).
- **Logout**: elementary operation that does not require the involvement of any internal or external file to achieve the result.
- **Register account**: functionality that relies on the user account or the operator internal logic file.
- **Delete account**: operation relying only on user account or operator account internal logic file.
- **Update user account informations**: this operation is more complex with respect to the last two since modification of a user information can involves other internal or external files.

  Indeed, for instance, the payment informations editing needs the intervention of the payment web service in order to verify the validity of the updated informations, hence the involvement of the payment system external file.

  This external input relies on the user account internal file, the payment system web service and driver license web service external files.
- **Update operator account informations**: with respect to the update operation regarding the user account informations, this process is simpler since only the operator informations ILF is involved.
- **Create reservation**: the creation of a reservation from a user is a quite simple operation, involving only the reservation and the vehicle informations internal logic files.
- **Terminate reservation**: complex operation that need the support of the reservation informations, vehicle informations internal logic file and the payment web service external file.
- **Update reservation**: updating the information regarding an already terminated reservation is a quite simple operation that involves the reservation informations.
- **New vehicle insertion**: simple operation relying only on the vehicle informations internal logic file.
- **Vehicle deletion**: simple operation relying only on the vehicle informations internal logic file.
- **Vehicle informations update**: simple operation where vehicle informations internal file is used.
- **Vehicle unlock**: operation not involving any internal or external file.

- **Event comunication**: whenever the driver performs a particular action described into the policy rules, the car's central system react sending to software system the information about the event. The reservation ILF is involved.
- **Insert new policy rule**: operation through which the policy rules information file is modified.
- **Remove policy rule**: elementary operation involving only the policy rules information file.
- **Update policy rule**: simple operation involving the policy internal file.
- **Insert parking area**: the insertion of a new parking area, either a normal area or a safe area, is an operation which involves the parking area and the safe parking area internal files.
- **Remove parking area**: basic operation involving the parking area and the safe parking area internal files.
- **Update parking area informations**: same complexity and same internal files involved as for the previous two parking area-related functionalities.
- **Insert new task**: new tasks to be assigned to an operator is a simple operation where only the mantainance task informations file is involved.
- **Remove task**: simple functionality modifying the mantainance task information file.
- **Update task**: simple operation involving solely the mantainance task information file.

	Data elements		
FRT	$1 \rightarrow 4$	$5 \rightarrow 15$	16+
0-1	Low	Low	Average
2-3	Low	Average	High
4+	Average	High	High

Table 5: EI's rating table

2.1.4 External Inquiries

- **Retrieve reservation history**: the retrieve of the full reservation history (or part of it) is an operation that involves only two internal logic files, namely the reservation informations and the vehicle informations ILF.
- **Retrieve account informations**: operation involving either the user account informations ILF or the operator ILF.

EI	Complexity	Weigth
Login	Low	3
Logout	Low	3
Register account	Low	3
Delete account	Low	3
Update user account	Average	4
Update operator account	Low	3
Create reservation	Low	3
Terminate reservation	Average	4
Update reservation	Low	3
Vehicle insertion	Low	3
Vehicle deletion	Low	3
Vehicle update	Low	3
Vehicle unlock	Low	3
Event comunication	Low	3
Insert policy rule	Low	3
Remove policy rule	Low	3
Update policy rule	Low	3
Insert parking area	Low	3
Remove parking area	Low	3
Update parking area	Low	3
Insert task	Low	3
Remove task	Low	3
Update task	Low	3
Partial Function Points	71	

Table 6: EI's PFP table

- **Retrieve vehicle list**: simple operation involving the vehicle informations ILF.
- **Retrieve vehicle informations**: the characteristic informations regarding a vehicle are retrieved inquiring solely the vehicle informations ILF.
- **Retrieve vehicle status**: the informations regarding the current status of a specific vehicle are stored into the vehicle informations ILF.
- **Retrieve current reservation informations**: operation similar to the reservation history retrive operation. The same ILF are involved.
- **Retrieve area list**: simple inquire involving the area informations ILF.
- **Retrieve specific area informations**: as the previous operation, this one is simple as well and involves the same ILF.
- **Retrieve mantainance operator list**: only the operator ILF is involved in this operation.
- **Retrieve mantainance task list**: simple inquire operations regarding

the mantainance task informations ILF.

- **Retrieve policy list**: the retrieval of the total list of policy rules present into the system or part of them is a simple operation involving the policy ILF.
- **Retrieve policy rule informations**: the informations regarding a specific policy rule involves the querying of the policy ILF.
- **Retrieve vehicles in specific area**: the achievement of this operation is reached through the querying of the vehicle informations ILF.

	Data elements		
FRT	$1 \rightarrow 5$	$6 \rightarrow 19$	20+
0-1	Low	Low	Average
2-3	Low	Average	High
4+	Average	High	High

Table 7: EQ and EO's rating table

EQ	**Complexity**	**Weigth**
Retrieve reservation history	Low	3
Retrieve account informations	Low	3
Retrieve vehicle list	Low	3
Retrieve vehicle status	Low	3
Retrieve current reservation info	Low	3
Retrieve area list	Low	3
Retrieve specific area info	Low	3
Retrieve mantainance operator list	Low	3
Retrieve mantainance task list	Low	3
Retrieve policy rule list	Low	3
Retrieve policy rule info	Low	3
Retrieve vehicle in specific area	Low	3
Partial Function Point	36	

Table 8: EQ's PFP table

2.1.5 External Output

In certain circumstances, the software system is required to comunicate to the final user outside the context of either an inquire or an external input.

These situations are very few in our projects and are the following:

- Notify the user of reservation expiration.
- Notify the user of policy rule(s) application.

Both of these external outputs refer to a single ILF, respectively the reservation ILF and policy rules ILF, while the data elements carried are one for the former (a simple message) and three for the latter (policy rule, event and effect).

According to the rating table 7, both these EOs are classifiable as *Low*.

EO	**Complexity**	**Weight**
Reservation expiration notification	Low	3
Policy rule(s) application notification	Low	3
Partial Function Points	6	

2.1.6 Size evaluation

In order to evaluate the approximated size through the function point method, the equation 2 is applied to the factors analyzed in the previous sections:

Function type	**FPF**
ILF	63
EIF	20
EI	71
EQ	36
EO	6
Total	196

Table 9: UFP table

Considering instead the **SLOC** as a project's size estimator, we can exploit the **gearing factors** to translate the UFP value in the corresponding SLOC estimation.

For each gearing factor exist different values, corresponding the minimum, maximum and average number of source code lines for a single function point.

In order to give a conservative estimation, we decide to assess the project's size using the average value and the maximum value to compute, respectively, its lower bound and the upper bound.

To give as much flexibility as possible to the project's implementation, without depending on specific technologies, languages or platforms, the gearing factors used to estimate the lower bound and the upper bound are calculated as the arithmetical average between the gearing factors of J2EE and .NET, the two most widespread technologies.

Lower bound in KSLOC

$$KSLOC^{-} = 52 * 196 = 10.192$$

Technology	Average	High
J2EE	46	67
.NET	57	60
Average	52	64

Table 10: UFP table

Upper bound in KSLOC

$$KSLOC^{+} = 64 * 196 = 12.544$$

2.2 Effort and Cost Estimation

2.2.1 Scale Drivers

The scale drivers are five distinct factors used to translate directly in the effort estimation process specific characteristics that have a huge relevance and impact on the project development.

To each scale driver can be assigned a value raging from *very low* to *very high*. The traslation in the effort estimation computation is performed associating to each scale driver's value a decimal value obtained through statistical analysis.

These five scale drivers are:

- **Precedentedness**: this factor measures the degree of familiarity that the development team has regarding the project under analysis.

  Our experience with this kind of projects is really limited, therefore a *Low* value is assigned.

- **Development flexibility**: this factor reflects the degree of flexibility allowed for the development of the project. This factor is computed taking in consideration the presence of pre-enstablished requirements and software conformance with external interface specification.

  Our projects involves some pre-defined requirements but no external interface specifications are required, hence the value of this scale driver is set to *nominal*.

- **Architecture/Risk resolution**: scale factor measuring the quality of the risk management plan and scheduling/budget compatibility with the latter.

  The risk management plan defined for this project embrace and define a recovery plan for almost every predictable risks, respecting the scheduling and the budget defined. *High* value is assigned.

- **Team cohesion**: scale factor measuring reflecting the degree of cooperation among the team members. Our team had divergence on project choices several time that brought a slow down in the project development. A *Nominal* value is assigned.
- **Process maturity**: factor measuring the degree of maturity reached by the organization's development processes. Since this is our first project, is quite hard to assess the maturity of our approach. Hence, even if the applied processes to the development could be typical of a level-3 organization, we set the factor's value to *level 2*, typical of processes designed for specific projects.

Scale Factor weigth SF_ss 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	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

Refering to the scale drivers table, the following table is produced

The **total scale factor** E has a value computable by mean of the following equation

$$E = B + 0.01 * \sum_{s \in S} (SF_s) = 1,0838 \tag{3}$$

with $B = 0.91$ and S representing the set of scale drivers.

Scale Driver	Complexity	SF
Precedentedness	Low	4.96
Development flexibility	Nominal	3.04
Architecture/Risk resolution	High	1.41
Team cohesion	Nominal	3.29
Process maturity	Level 2	4.68
Total value	17.38	

Table 12: Scale drivers summary

2.3 Cost Drivers

Similar to the scale factors, the cost drivers are a set of factors describing in terms of cost the impact that they have on the whole project. Each cost driver has a impact level, raging from *very low* to *very high*, and each impact level has a statistical decimal value associated.

The set of cost drivers vary according to the architectural state of the project, which may be in a*Post-architectural* state or *Early design* state.

Post-architectural's cost drivers may rely on more precise and detailed informations about the project, producing more precise estimates.

The early design model, instead, due to the lack of informations typical of early project provides less accurate figures regarding the effort estimation.

As specified in the introductory paragraph to this section, our project follow a **early design model**. The set of cost drivers used for the early design model is derived by the combinations of different cost drivers defined for the post-architectural model.

In the following, the cost drivers for the post-architectural model are described along with the rationale for the assigned value.

At the end of this analysis, the early design cost drivers values are computed and shown.

Early design cost drivers	Post-architecture cost drivers
Personell Capability [PERS]	ACAP PCAP PCOPN
Product Reliability and Complexity [RCPX]	RELY DATA CPLX DOCU
Developed for Reusability [RUSE]	RUSE
Platform Difficulty [PDIF]	TIME STOR PVOL
Personnel Experience [PREX]	APEX PLEX LTEX
Facilities [FCIL]	TOOL SITE
Required Development Schedule	SCED

Table 13: Early Design - Post-Architectural mapping

- **Required Software Reliability [RELY]**: this cost driver is a measure of the reliability required for the software to be developed. The value is assigned according to the level of danger and/or damage the interruption of the service may cause.

  Since the software system is designed to support a car-sharing service, a malfunction can lead to important finantial losses for the society. Hence the assigned value 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]**: the dimension of test data required for the software testing has an impact on the cost of development for the test.

  In particular, the database size required to store these test data is taken in consideration, evaluating the derived impact on the costs as the ration between the database size D and the LOSC P.

  Being in an early stage phase of the development, it's not easy to estimate the dimension of the database. We can assume a minimal dimension of 1 gigabytes. Therefore, the value of this cost driver is set to *Very High*.

DATA Cost Drivers						
DATA Descriptors		Testing DB bytes/pgm SLOC $<$ 10	$10 \leq D/P \leq 100$	$100 \leq /P \leq 000$	DP $>$ 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 cost driver is divided into five area, shown in the table below, and its value is the average of the five area's values. The CPLX's value is set to *High*.

CPLX Cost Driver						
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]**: this driver reflect the degree of reusability adopted for the implementation of system's components.

  No reusability feature with other projects is expected to be adopted in the design and implementation of the components, hence the cost driver's values is equal to *Nominal*.

RUSE Cost Driver						
RUSE Descriptors		None	Across project	Across program	Across product line	Across multiple product lines
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 cost driver measure the degree of suitability of the required documentation to the project's life-cycle.

  Since we're following a precise model of development which requires the appropriate documentation for each step, the documentation is right-sized to the project's life-cycle. The value is set to *Nominal*.

DOCU Cost Driver						
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

- **Execution Time Constraints [TIME]**: this factor estimates the expected amount of execution time used by the software system.

  The software system is expected to be deployed in a highly concurrent environment, with several requests. Therefore, the value of this driver is set to *Very High*.

TIME Cost Driver

TIME Descriptors			$\leq$ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% 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 Constraints [STOR]**: this rating represent the percentual amount of main storage expected to be used by the system.

  We expect that the high number of generated and stored data won't take more than the 50% of the total storage available in the system. The cost driver's value is set to *Nominal*.

STOR Cost Driver						
STOR Descriptors			$\leq$ 50% use of available storage	70% use of available storage	85% use of available storage	95% 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 driver measures the average time of hardware/software replacement and update expected.

  The software system is supposed to be deployed on an external cloud computing service. Major changes are expected to be applied to the software part or to the hardware living in the vehicles and performed quite rarely. Thus, the value of this driver is set to *Low*.

PVOL Cost Driver						
PVOL Descriptors		Major change every 12 mo., minor change every 1 mo.	Major: 6mo; minor: 2wk.	Major: 2mo, minor: 1wk	Major: 2wk; minor: 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]**: cost driver that measure the analysis and

design ability of the team regarding the project in analysis.

It's the first time we are facing with a project of this size and complexity. Therefore, the assigned value to this factor is *Nominal*.

ACAP Cost Driver						
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 Cability [PCAP]**: driver analyzing the efficiency and ability of the programmers involved in the project development.

  Since we are in a early stage phase, the project is not developed yet. Hence, we can only estimate this parameter with respect to the feedbacks we received in our past projects. The value of the driver is set to *High*.

PCAP Cost Driver						
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

- **Personnel Continuity [PCON]**: this driver measure the percentage of time that team members are unavailable for the project development.

  In our case, the driver is set to a *Very Low* value due to limited time we can spent for the project.

PCON Cost Driver						
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

- **Application Experience [APEX]**: rating describing the experience of whole team with this kind of projects.

  Our experience with this category of projects is quite limited, hence the value for this driver is *Low*.

APEX Cost Driver

APEX Descriptors	≤ 2 months	6 months	1 year	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]**: cost driver assessing the experience of the team with the platform used for the development.

  The project is in a early stage development, therefore the development platform is not decided yet. However, in our past projects we dealt with some important software applications that are part of the most important and used development platform, hence we can assign a value of *Nominal* to this driver.

PLEX Cost Driver						
PLEX Descriptors	≤ 2 months	6 months	1 year	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]**: experience and familiarity of the development team with he technology used, both languages and software tools, are describe by this driver.

  Some of the technologies and frameworks suppose to be used to achieve the goals of each step of the project's life cycle are quite unknown. Therefore, the set value for this driver is *Nominal*.

LTEX Cost Driver						
LTEX Descriptors	≤ 2 months	6 months	1 year	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

- **Use of Software Tools [TOOL]**: this driver attempts to assess the kind of tools used and their integration. The tools we expect to use are quite integrated and support quite well the project's life-cycle. The driver's value is set to *High*.

TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle 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]**: the characteristics evaluated by this driver are the collocation of the site development and the communication channels used to manage the development activity.

  Our team is collocated in two different cities (Multi-city collocation) and uses several communication channels (VoIP technologies, text messages, emails etc...). Hence, the rating of this driver is set to *Very High*.

SITE Cost Driver						
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 elect. comm., occasional video conf.	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

- **Required Development Schedule [SCED]**: this cost driver aims to rate the impact that the time constraints imposed to the project's team has on the whole project.

  The schedule assigned for this project is felt slightly compressed by the project's team, therefore the rate of this driver is set to *Low*.

SCED Cost Driver						
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

The total effort multipler derived from each cost driver is computed as follows:

$$TEM = \prod_{C} W_c \tag{4}$$

with C representing the set of all cost drivers, and W_c the weigth associated to the cost driver c.

Early design cost drivers	**Rating level**	**Effort multiplier**
PERS	Nomial	1.00
RCPX	Very High	1.91
RUSE	Nominal	1.00
PDIF	High	1.29
PREX	Low	1.22
FCIL	High	0.73
SCED	Low	1.14
Total Effort Multiplier		2.5015

Table 31: Total effort mulitplier table

2.3.1 Effort evaluation

The evaluation of the estimated effort for the development of this project is computed by mean of the following equation:

$$PM = A * Size^E * TEM \tag{5}$$

- A: statistical coefficient. The COCOMO II manual sets its current value to *2.94*.
- Size: the estimated size of the project in KSLOC computed in the *Size Estimation* section.
- E: the total scale factor computed in the *Scale Drivers* section.
- TEM: the total effort multipler.

Using the lower bound and upper bound values computed for the project's size, we get, respectively:

$$PM^- = 2.94 * 10.192^{1.0838} * 2.5015 \approx 92$$

$$PM^{+} = 2.94 * 12.544^{1.0838} * 2.5015 \approx 114$$

2.3.2 Project's duration evaluation

The estimated duration of the project can be derived by the following pair of equations, provided by the COCOMO II manual:

$$Duration = 3.67 * PM^{F} \tag{6}$$

$$F = 0.28 + 0.2 * (E - B) \tag{7}$$

- PM: the estimated effort required for the project development in person per month.
- E: the total scale factor computed in the *Scale Drivers* section.
- B: statistical coeffiecient. The COCOMO II manual sets its value to *0.91*.

With the lower bound and upper bound values of the project's size we get as lower bound and upper bound of the project's duration in months:

$$F = 0.28 + 0.2 * (1.0838 - 0.91) = 0.31476$$

$$Duration^{-} = 3.67 * 92^{0.31476} = 15.23$$

$$Duration^{+} = 3.67 * 115^{0.31476} = 16.34$$

3 Tasks schedule

The project's life-cycle in composed by five main processes

1. Requirements analysis and specifications definition
2. Architecture design
3. Integration testing
4. Project plan development
5. Project implementation

«««< HEAD In the following of this section, for each of the above mentioned process the scheduling of each composing task is presented, pointing out the forecasted starting point, dead-lines and eventual task dependencies. The scheduling activity is performed basing on the project's duration estimated during the effort and cost estimation analysis.

It's worth noting that for the first four activities the dead-lines has been already defined.

Even though the implementation tasks are perfectly ordered one after the other, during the implementation it is likely that this order will not be followed due to possible errors or changes in the system structure. ======= The first four parts of the project had a deadline already established, instead we assumed that twelve months are needed in order to complete the implementation part. Details on this timing can be found in the *Project Size, Effort and Cost Estimation* section, considering we're still in an early design stage these numbers may vary greatly. Even though the implementation tasks are perfectly ordered one after the other, during the implementation it is likely that this order will not be followed due to possible errors or changes in the system structure. »»»> refs/remotes/origin/master

The following images show the tasks related to this project.

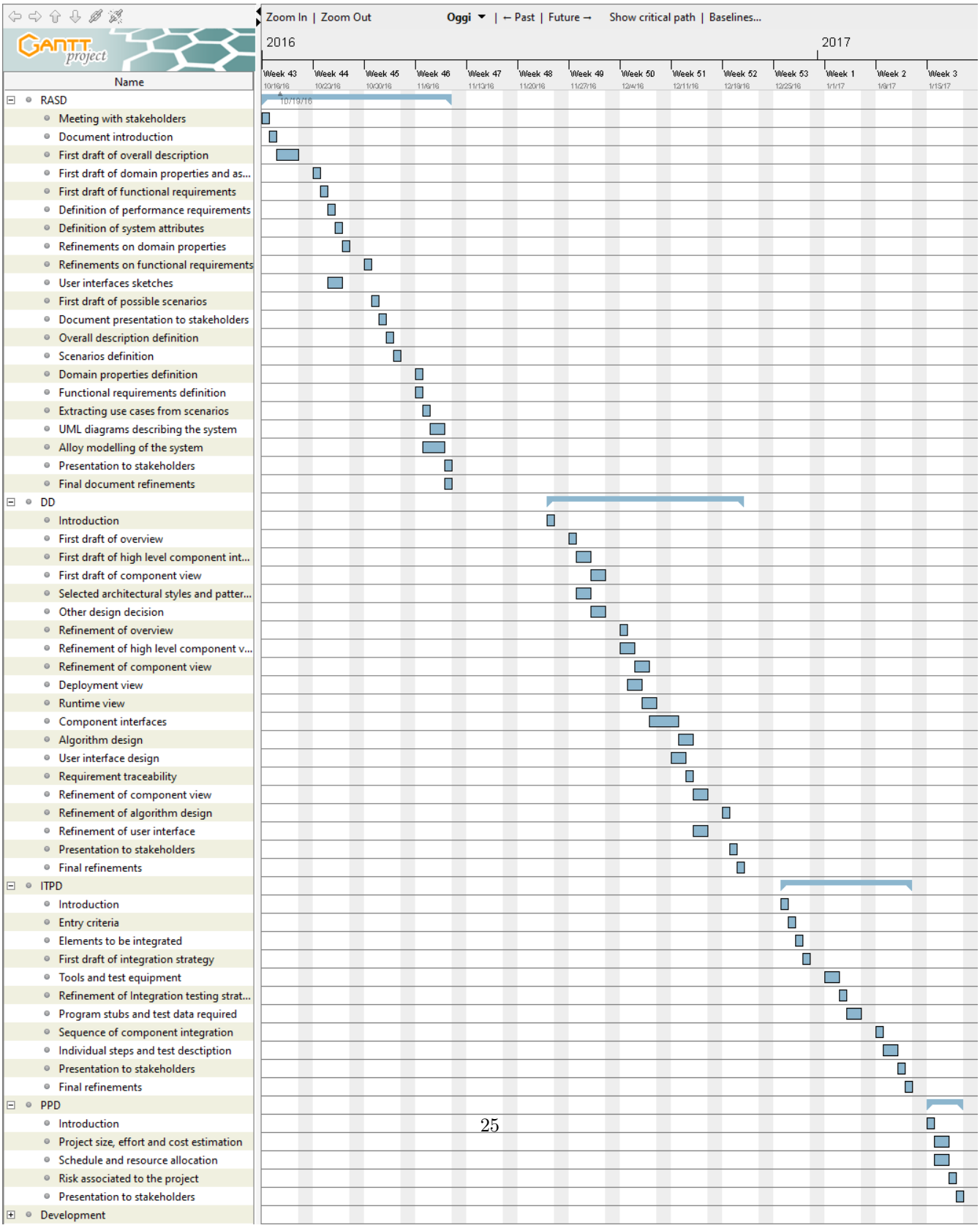

Figure 1: Project tasks 1

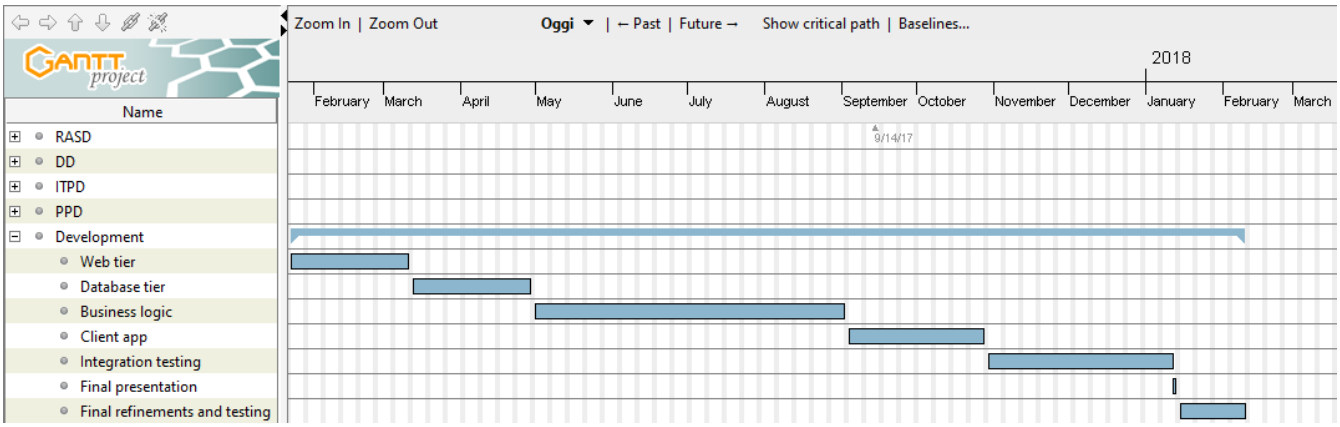

Figure 2: Project tasks 2

Following is a more accurate definition of each task deadline.

Untitled Gantt Project

Jan 22, 2017

2

Tasks

Name	Begin date	End date
RASD	10/17/16	11/11/16
Meeting with stakeholders	10/17/16	10/17/16
Document introduction	10/18/16	10/18/16
First draft of overall description	10/19/16	10/21/16
First draft of domain properties and assumptions	10/24/16	10/24/16
First draft of functional requirements	10/25/16	10/25/16
Definition of performance requirements	10/26/16	10/26/16
Definition of system attributes	10/27/16	10/27/16
Refinements on domain properties	10/28/16	10/28/16
Refinements on functional requirements	10/31/16	10/31/16
User interfaces sketches	10/26/16	10/27/16
First draft of possible scenarios	11/1/16	11/1/16
Document presentation to stakeholders	11/2/16	11/2/16
Overall description definition	11/3/16	11/3/16
Scenarios definition	11/4/16	11/4/16
Domain properties definition	11/7/16	11/7/16
Functional requirements definition	11/7/16	11/7/16
Extracting use cases from scenarios	11/8/16	11/8/16
UML diagrams describing the system	11/9/16	11/10/16
Alloy modelling of the system	11/8/16	11/10/16
Presentation to stakeholders	11/11/16	11/11/16
Final document refinements	11/11/16	11/11/16
DD	11/25/16	12/21/16
Introduction	11/25/16	11/25/16
First draft of overview	11/28/16	11/28/16
First draft of high level component interaction	11/29/16	11/30/16
First draft of component view	12/1/16	12/2/16
Selected architectural styles and patterns	11/29/16	11/30/16
Other design decision	12/1/16	12/2/16
Refinement of overview	12/5/16	12/5/16
Refinement of high level component view	12/5/16	12/6/16
Refinement of component view	12/7/16	12/8/16
Deployment view	12/6/16	12/7/16
Runtime view	12/8/16	12/9/16
Component interfaces	12/9/16	12/12/16
Algorithm design	12/13/16	12/14/16
User interface design	12/12/16	12/13/16
Requirement traceability	12/14/16	12/14/16
Refinement of component view	12/15/16	12/16/16
Refinement of algorithm design	12/19/16	12/19/16
Refinement of user interface	12/15/16	12/16/16
Presentation to stakeholders	12/20/16	12/20/16
Final refinements	12/21/16	12/21/16

Tasks

Name	Begin date	End date
ITPD	12/27/16	1/13/17
Introduction	12/27/16	12/27/16
Entry criteria	12/28/16	12/28/16
Elements to be integrated	12/29/16	12/29/16
First draft of integration strategy	12/30/16	12/30/16
Tools and test equipment	1/2/17	1/3/17
Refinement of Integration testing strategy	1/4/17	1/4/17
Program stubs and test data required	1/5/17	1/6/17
Sequence of component integration	1/9/17	1/9/17
Individual steps and test desctiption	1/10/17	1/11/17
Presentation to stakeholders	1/12/17	1/12/17
Final refinements	1/13/17	1/13/17
PPD	1/16/17	1/20/17
Introduction	1/16/17	1/16/17
Project size, effort and cost estimation	1/17/17	1/18/17
Schedule and resource allocation	1/17/17	1/18/17
Risk associated to the project	1/19/17	1/19/17
Presentation to stakeholders	1/20/17	1/20/17
Development	1/23/17	2/9/18
Web tier	1/23/17	3/10/17
Database tier	3/13/17	4/28/17
Business logic	5/1/17	9/1/17
Client app	9/4/17	10/27/17
Integration testing	10/30/17	1/11/18
Final presentation	1/12/18	1/12/18
Final refinements and testing	1/15/18	2/9/18

4 Resources allocation

This section shows how the human resources are allocated to the various tasks of the project. Even though almost all the tasks have a specific resource allocated to them we tried to involve all the team in each step.

This approach result in a longer project, but enables a grater controll of the work of each member with a consequent smaller number of misunderstandings.

The following images show the allocation of each member of the team.

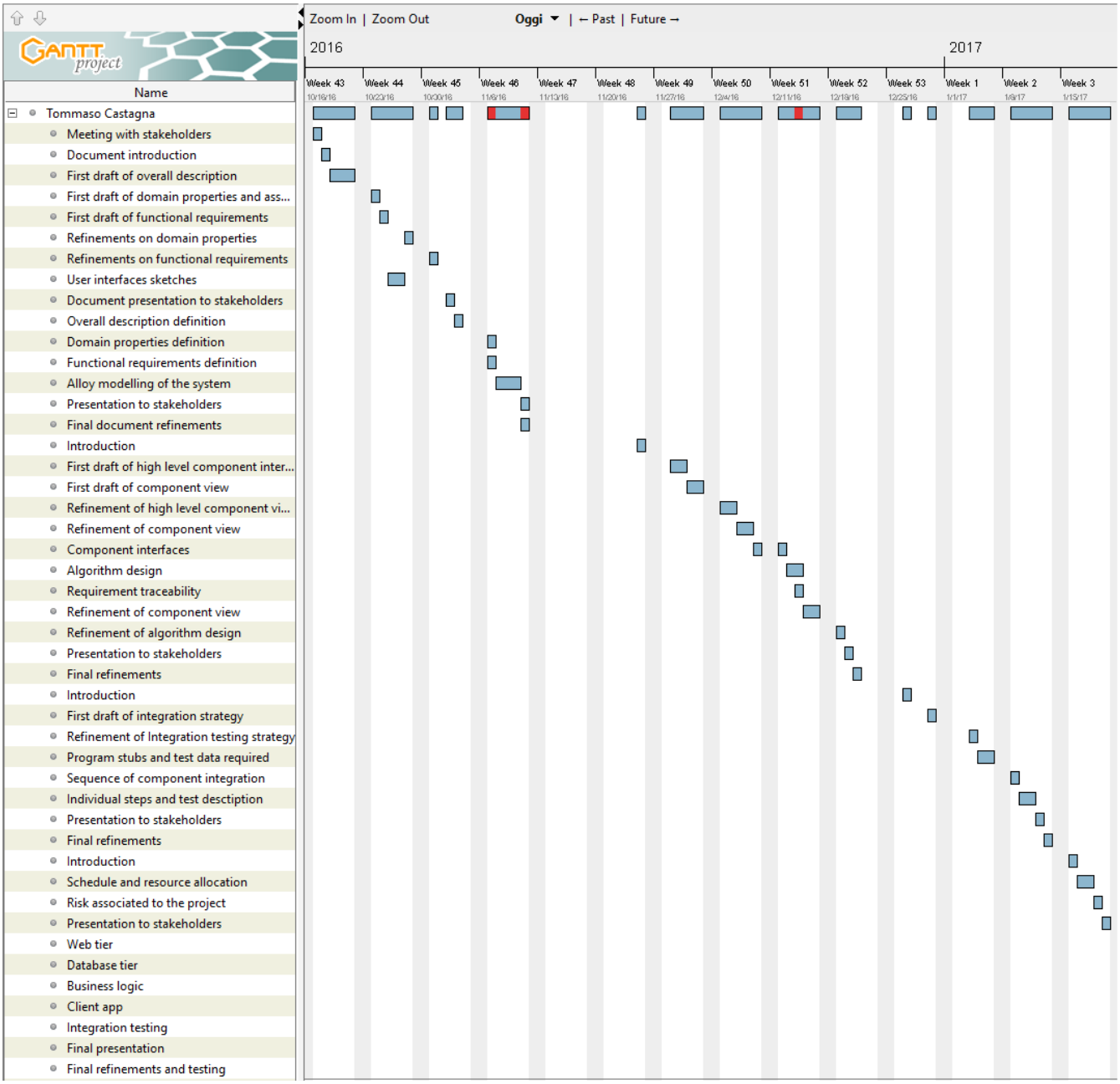

Figure 3: Tommaso allocation 1

Figure 4: Tommaso allocation 2

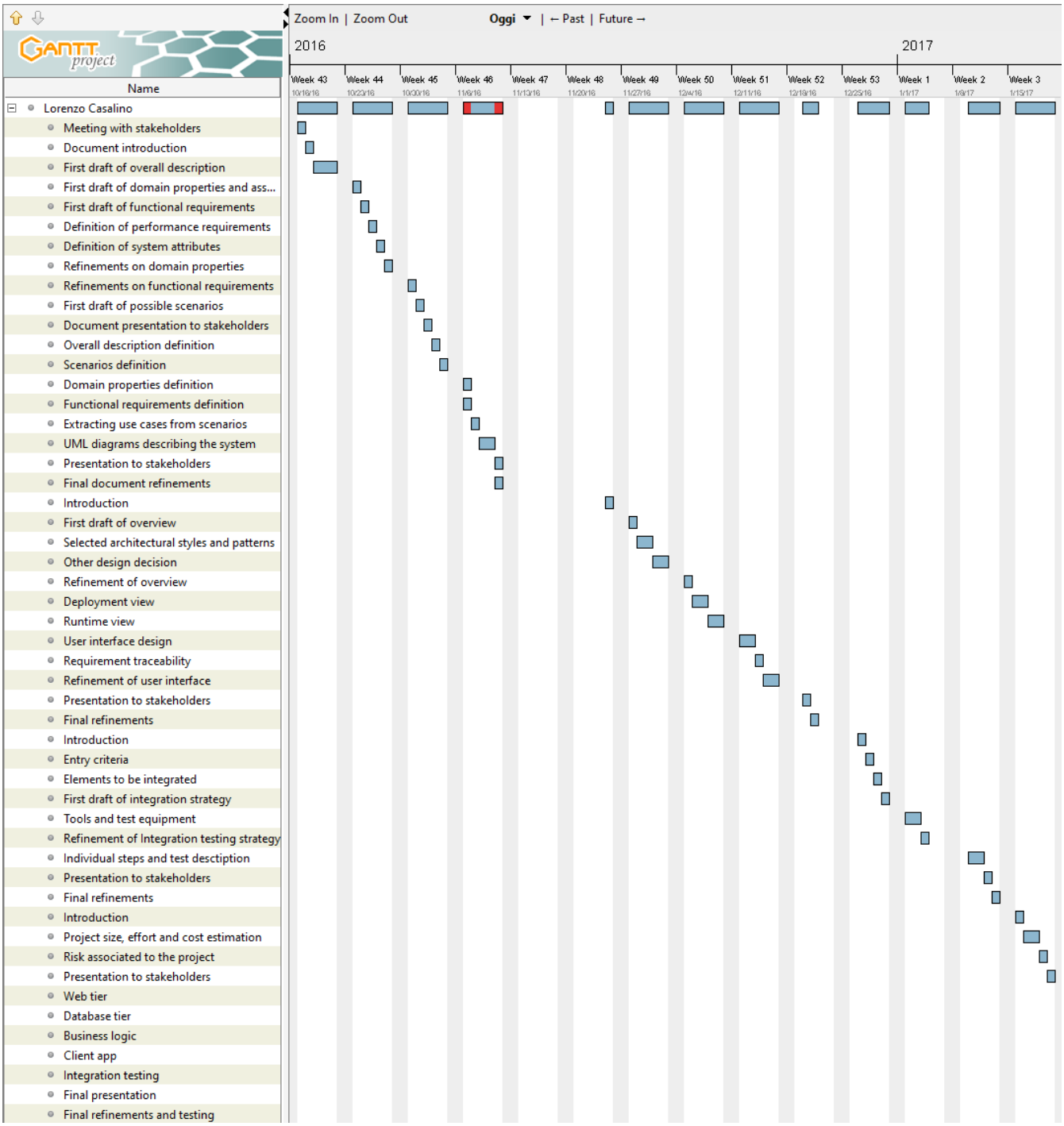

Figure 5: Lorenzo allocation 1

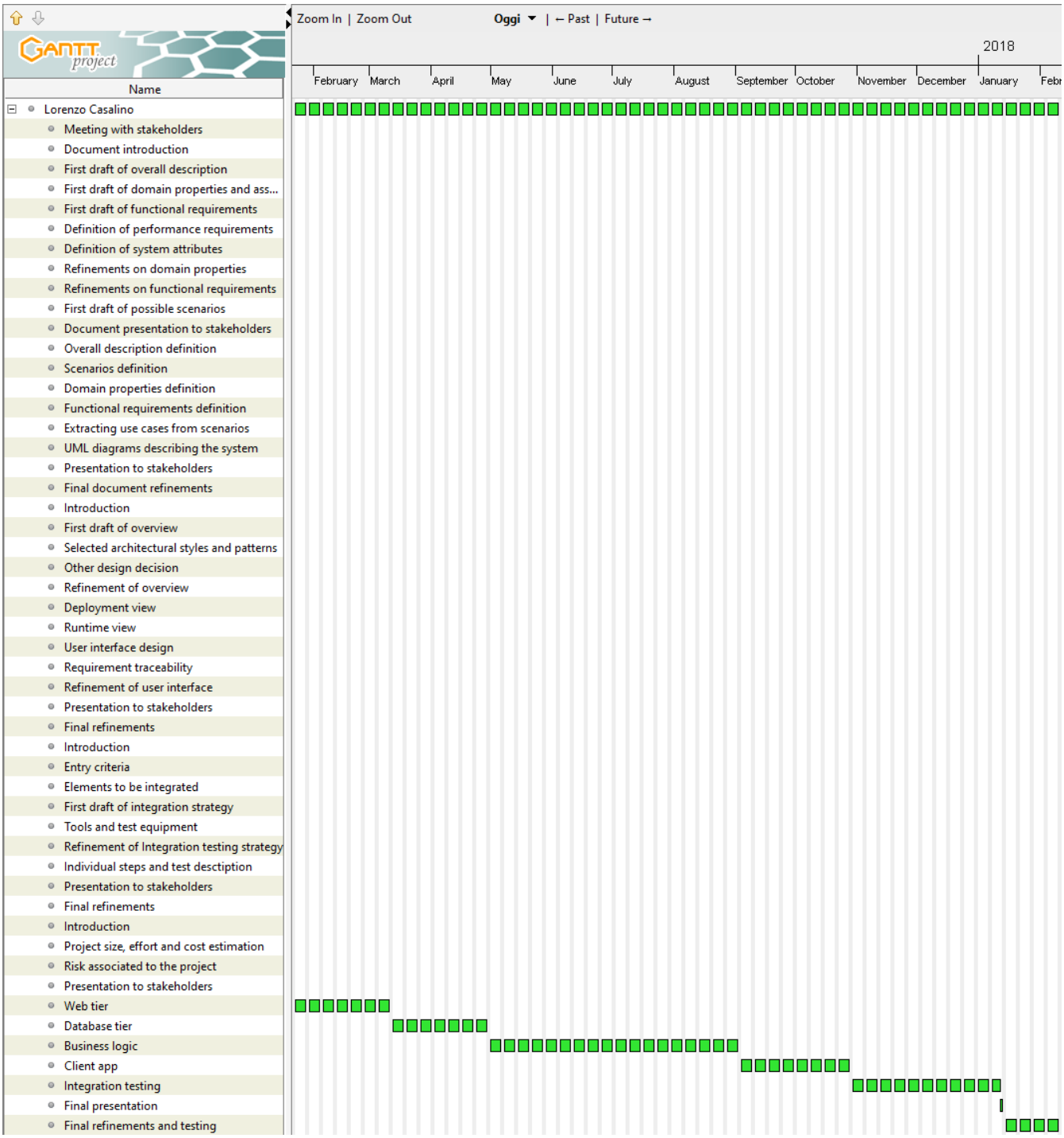

Figure 6: Lorenzo allocation 2

5 Risk Management

In this section of the document we will be focusing on the possible risks on which we may incur. We can devide those risks into three different categories:

- Project risks
- Technical risks
- Business risks

In order to minimize the chances of possible risks to happen and damages deriving from them we decided to use a proactive risk approach. This approach consists of identifying possible risks, analyzing them and to develop a contingency plan in order to manage the consequences deriving from those problems.

5.1 Project Risks

This is a list of possible project risks in which we may incur during the realization of the project.

Changes in project requirements: This may happen for various reasons, and cannot be prevented. We can only mitigate the consequences of this problem by writing code that is easily maintanable and extensible.

Lack of familiarity with the used technologies: This problem will probably occur due to the fact that the team has never programmed before using Java EE. This could have been originally avoided by hiring programmers that alredy had some experience with the selected frameworks.

Unrealistic deadlines: This is one of the most occurring problems in this kind of projects. A possible solution to it could be a first release of the system that is working but doesn't include all the side functionalities, that will be implemented later in time.

Staff ill or quitting the company: This is another unpredictable risk in the project. In order to mitigate the consequences we can split the responsibilities among multiple people so that no one is essential to the development of the project.

5.2 Technical Risks

Following is a list of possible technical risks in which we may incur during the realization of the project.

Losing source code: This problem is easily resolved due to the large amount and high reliability of versioning systems available on the market.

Infrastructure downtime: The availability of the system is strictly related to the company we rely on for the cloud infrastructure, an in depth search of the different proposals on the market can avoid this problem.

Failure on component integration: The failure of an integration test on a component can force the team to rewrite a large amount of code with a consequent loss of time. In order to avoid this kind of problems we need to precisely specify the interface of each component and we should also run integration test in an early stage of the creation of the component using stubs and drivers.

Code unreadability: This problem may often occur in a young and unexperienced team. To avoid this to happen is sufficent to apply the rules of good coding and to regularly inspect your work.

Delays in the delivery of external resources: This is another unpredictable problem that may occur, in our case, if the society providing the cars for our service cannot deliver the cars in time. A reactive action to this problem will be of no use, in fact the only solution is to preventively take into account this problem and schedule the delivery of the cars with enough room to compensate any problem.

Changes in the external services used in the project: Any change to the external services used inside our project can lead to a substantial review of the implementation of the whole system. This problem does not depend in any way from our company, and the only solution we have is to preventively search in the market of backup solutions for each of the external service used, and to write portable and reusable code.

5.3 Business Risks

A list of possible business risks in which we may incur during the realization of the project.

Internal bankruptcy: This problem has catastrophic consequences on the project and can only be avoided with a good feasibility study providing all the anticipated costs of the project.

External suppliers bankruptcy: This problem can be avoided in some measure by investigating in the financial condition of the society we want to rely on. Another solution can be to have a backup offer for a different supplier.

Changes in laws regulating key aspects of the project: This problem should not materialize since we are using electrical cars with no toxic emissions. However it may happen that parts of our car fleet will become forbidden by law (e.g. certain type of battery cells). In order to prevent this kind

of problems we must perform a research on the most recent technologies used in the fields in which our project is involved.

Following is the table containing all the previously presented problems along with their probability and estimated effects on the project

Risk	**Probability**	**Effects**
Changes in project requirements	Low	Moderate
Lack of familiarity with the used technologies	High	Moderate
Unrealistic deadlines	Medium	Moderate
Staff ill or quitting the company	Low	Critical

Table 32: Project Risks evaluation

Risk	**Probability**	**Effects**
Losing source code	Very low	Critical
Infrastructure downtime	Very low	Critical
Failure on component integration	Medium	Critical
Code unreadability	Medium	Moderate
Delays in the delivery of external resources	Low	Critical
Changes in the external services used in the project	Low	Critical

Table 33: Technical Risks evaluation

Risk	**Probability**	**Effects**
Internal bankruptcy	Very low	Catastrophic
External suppliers bankruptcy	Low	Critical
Changes in laws regulating key aspects of the project	Very low	Critical

Table 34: Business Risks evaluation

6 Effort Spent

Lorenzo Casalino

- 16 January 2017 - 1h 20m
- 17 January 2017 - 1h 20m
- 18 January 2017 - 2h 30m
- 19 January 2017 - 2h 20m
- 20 January 2017 - 4h
- 21 January 2017 - 6h 10m
- 22 January 2017 - 40m

Tommaso Castagna

- 19/01/16 - 30m
- 20/01/16 - 1h
- 21/01/16 - 2h30m
- 22/01/16 - 5h30m