Project plan

POLITECNICO
MILANO 1863

Software Engineering 2
PowerEnJoy

Authors:

Bianchi Alessandro (Mat. 875035)
Canciani Francesco (Mat. 807056)
Cannone Lorenzo (Mat. 875802)

Document version: 1.0

Academic year:
2016-2017
Release date:
22-01-2017

Contents

1.Introduction

1.1 Revision history

Version	Date	Authors	Summary
1.0	22-01-2017	Bianchi A., Canciani F., Cannone L.	First release

1.2 Purpose and scope

This document represents the Project Plan document for PowerEnJoy.

The main purposes of the project plan are to document planning assumptions and decisions, facilitate communication among project stakeholders, and document approved scope, cost, resources allocation and schedule of the activities.

In the first section of the document, function points are going to be applied to estimate the size of the project. Consequently, the COCOMO II approach will be used to estimate effort and cost.

In the second section, the estimation data that were just illustrated will be reused to provide a feasible schedule for the project's activities, that vary from requirements identification to integration testing.

The third section of the document is devoted to resource allocation. This section will detail how all the team members have been assigned to each of the various tasks that were previously identified. It is important to notice that we did so retrospectively, so with already a clear understanding of when and how the work took place.

Finally, in the last section, all the possible risks that could be faced during PowerEnJoy's project development are going to be listed. Each of those risks is also going to be followed by a reasonable counter approach that might be taken in case of event of that risk.

1.3 Definitions, Acronyms, Abbreviations

- **RASD**: Requirements Analysis and Specification Document
- **MSO**: Money Saving Option
- **J2EE**: Java Enterprise Edition
- **FP**: Function points
- **COCOMO**: Constructive COst Model
- **SLOC**: Source Lines of Code
- **UI**: User Interface
- **SF**: Scale Factor
- **MSO**: Money Saving Option

1.4 Reference documents

- Project description document: Assignments **AA 2016-2017.pdf**
- PowerEnJoy Requirements Analysis and Specification Document: **RASD.pdf**
- PowerEnJoy Design Document: **DD.pdf**
- PowerEnjoy Integration Testing Document: **ITPD.pdf**
- Project Plan Document examples:
  - Project planning example **document.pdf**
  - Example of usage of FP and **COCOMO.pdf**
- COCOMO II Model Definition Manual:
  - http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

2.Project size, cost and effort estimation

This section is specifically focused on providing some estimations of the expected size, cost and required effort to develop PowerEnJoy.

For the size estimation part, we will essentially use the Function Points approach, taking into account all the main functionalities of PowerEnJoy and estimating the correspondent amount of lines of code to be written in Java. This estimation will only take into account the parts of the project that concur to the implementation of the business logic and will disregard the aspects concerning the user interface.

For the cost and effort estimation, we will instead rely on the COCOMO II approach, using as initial guidance the amount of lines of code computed with the FP approach.

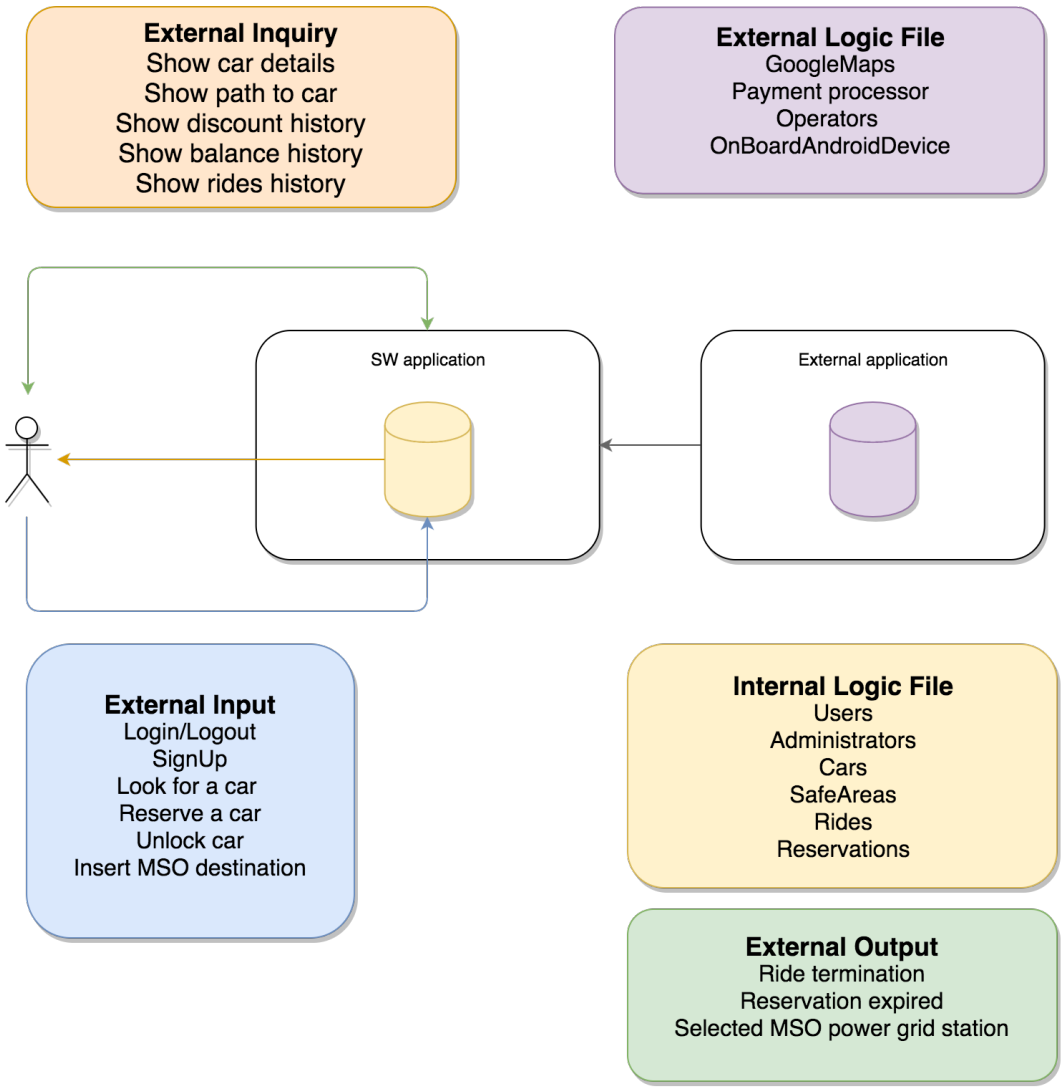
2.1 Size estimation: function points

The Function Points approach provides an estimation of the size of a project taking as inputs the amount of functionalities to be developed and their complexity.

The Function Points approach has been defined in 1975 by Allan Albrecht. The basic assumption is that the dimension of software can be characterised based on the functionalities that it has to offer. Albrecht has considered several applications, that eventually led him to define the number of FP as the sum of Function Types weighted in order to correlate them to the development effort. The weighting function points table that he produced and that we will use to evaluate our project is the following:

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

As we can see, he identified 5 major categories in which we can divide each of the system's functionalities. In the following diagram, we illustrate how we divided the system functionalities into the listed categories and, in the subsequent paragraphs, we will dig into further details to clarify how we reasoned on them to perform this categorisation.



2.1.1 Internal Logic Files (ILFs)

The first category that we will consider is Internal Logic Files. By definition, an internal logic file is a homogeneous set of data used and managed by the application. PowerEnJoy relies on a number of ILFs to store the information it needs to offer the required functionalities. A list of the various ILFs that we have identified is the following:

- Users: the entity has a simple structure as it is composed of a small number of fields, which include static information like user personal information, email, password, driving licence, etc.
- Administrators: as for the previous entity, there is not much complexity related to this logic file as there is a very limited amount of fields that compose this object, that are limited to the need of identifying a singular administrator among the others.
- Cars: total different approach for this entity. Cars have static informations, like the car's licence plate and everything that relates to the specification of the car model. However, because of its constant interaction with the remote system installed inside the car, to communicate availability, battery level, charging condition, and so on, we classified this ILF as high complexity.
- SafeAreas: holds the concept of PGSs and parking spots. It contains static information as the stations locations and the number of total parking spots and power plugs in a given PGS. However, it is fundamental to determine the availability of power plugs at a given station, which is the most relevant parameter in the calculation of the MSO power grid station. It is a dynamic information, that is not as crucial as the interaction with the cars but that still needs some caution. Because of this, we evaluated the complexity as medium.
- Rides: it contains information that are static and dynamic, like the starting position, the final position, the duration of the ride based on when the ride started and when it ended. With the help of information that are retrieved from the car (e.g. when the user has exited the car) to update dynamically several fields of this entity, and since rides are what matters the most for what concerns the money that our clients have to pay, we want to manage them correctly, and that is why we think that a medium complexity should be a good evaluation for this ILF.

- **Reservations**: same ideas come along for the reservation entity. Reservations are key in the PowerEnJoy system, just like the rides. The system has to keep updated information about reservation times and duration, in order to manage correctly the availability of cars and potential extra fees that have to be applied to the user. Due to the constant dependence on users and cars interaction, we will classify this ILF with a medium complexity.

ILF	Complexity	FPs
Users	Low	7
Administrators	Low	7
Cars	High	15
SafeAreas	Medium	10
Rides	Medium	10
Reservations	Medium	10
TOTAL		**59**

2.1.2 External Logic Files (ELFs)

The second category that we will consider is External Logic Files. By definition, an external logic file is a homogeneous set of data used by the application but generated and maintained by other applications. PowerEnJoy relies on several ELFs to perform its functionalities. A list of the various ELFs that we have identified is the following:

- **GoogleMaps**: the interaction with this external mapping service is key to development of the money saving option additional functionality. The reason for that is that this mapping service allows us to compute the distance between two locations and also provide the shortest path to reach a certain destination. The complexity however is not high, because all the computation of the finding the best path given two locations is up to the external component. It will be our issue to correctly interface with it and that is why we need a medium complexity for this ELF.

- PaymentProcessor: same complexity concerns hold for the payment processor. It is up to this external component to perform the transactions between PowerEnJoy's account and the user's account, also taking care of notifying the system in case something goes wrong with the transaction. However, interfacing with this external component will need particular cautious since the managed information are particularly sensitive and it would be bad not to handle them professionally.
- Operators: totally different approach is the interface with the operators. This component is deployed inside the old PowerEnJoy system and mainly serves as a facilitator for our administration to manage the correct assistance that cars need. The interaction between this component and the new system is very limited to an exchange of simple information like the list of cars that need assistance and their location. There is no point of overestimating the complexity of this ELF and that is the reason for the low evaluation.
- OnBoardAndroidDevice: the most relevant and complex external component that is taken in consideration is the device that sits inside each of the PowerEnJoy car. The OnBoardAndroidDevice is the component that mainly and constantly must interact with the PowerEnJoy internal system to manage the correct use of the cars. This includes commands sent to the device that must be applied through the car's actuators and also retrieving information from the sensors, like knowing whether the car has been started, if it is charging or not, how many passengers are still inside the car and if all the car doors have been closed to allow the remote locking of them. The complexity must be rated to high.

EI	Complexity	FPs
GoogleMaps	Medium	7
PaymentProcessor	Medium	7
Operators	Low	5
OnBoardAndroidDevice	High	10
TOTAL		**29**

2.1.3 External Inputs (EIs)

The third category that we will consider is External Inputs. By definition, an external input is an elementary operation needed to elaborate data coming from the external environment. PowerEnJoy supports different kind of interactions with the user, especially coming from the mobile application. In fact, we have taken in consideration the non functional requirements illustrated in the RASD to specify the following list of EIs:

Login/Logout: standard operations that all business application nowadays have. It is definitely nothing that we have to develop from scratch so low complexity is assigned.

Signup: follows the same reasoning approach of the login/logout.

Look for a car: being able to look for a car based on user's device GPS or by a given address are operations that do not involve particularly hard operations, mainly because they rely on the external mapping service that does all the computation. Low complexity is assigned.

Reserve a car: being able to reserve a car involves several interactions among system components. Before allowing the creation of a new reservation element, the system has to check whether the selected car is an available condition. If so, it has to make it unavailable for all the other users and start a timeout for the case of an expired reservation. The complexity cannot be simple but not rated too high also.

Unlock a car: this operation implies basically the same considerations that were made for the "Reserve a car" EI, but this time, what rises a little bit the basic complexity of the operation is that the system must be able to determine when the user is within a defined range of distance from the reserved car, in order to allow them to see the "Open car" button on their mobile application. This possibility involves the interaction with the car, the user and the mapping service so at least a medium complexity is required.

Insert MSO destination: this operation triggers the search for what is the best power grid station in which the user can leave the car in order to get a discount. In the RASD we defined a possible algorithm that could implement this specification and it did not involve too much effort to outline it, so we can evaluate this as a low complexity EI.

EI	Complexity	FPs
Login/Logout	Low	6 (3*2)
SignUp	Low	3
Look for a car	Low	3
Reserve a car	Medium	4
Unlock a car	Medium	4
Insert MSO destination	Low	3
TOTAL		**23**

2.1.4 External Inquiries (EQs)

The fourth category that we will consider is External Inquiries. By definition, an external inquiry is an elementary operation that involves input and output, usually without significant elaboration of data from logic files. PowerEnJoy supports a few operations of this kind, which do not require complex computations:

- Show car details
- Show path to car
- Show discount history
- Show balance history
- Show rides history

We do not go into detail for each of these inquiries because they all represent very straightforward operations that simply involve data retrieval from the database, except for the "Show path to car" that instead of looking into the database, just inputs the query to the mapping service that outputs to the user the path to reach the selected car.

EQ	Complexity	FPs
Show car details	Low	6 (3*2)
Show path to car	Low	3
Show discount history	Low	3
Show balance history	Medium	4
Show rides history	Medium	4
TOTAL		**23**

2.1.5 External Outputs (EOs)

The last and fifth category that we will consider is External Outputs. By definition, an external output is an elementary operation that generates data for the external environment, which usually includes the elaboration of data from logic files. As part of its regular behavior, PowerEnJoy needs to communicate with the user outside the context of an inquiry. These situations are:

Ride termination: this operation involves the termination of a ride that, as we stated in the previously released documents, coincides when the user exits the car and the car doors lock automatically. The system then has to store all the information regarding the ride into the database, not before having computed all the eventual discounts or extra fees that have to apply to the final cost for the ride. All these operations are pretty straightforward, but they involve several components, like the car, the payment processor and the database. Medium complexity is assigned.

Reservation expired: it is triggered whenever the user does not pick up the car in time after having reserved it. This operation sets the car as available again and send a 1€ fee to the user. This operation involves a couple of components but it is considered pretty straightforward for its nature so it is given a low complexity value.

Selected MSO power grid station: all the complexity involved with this specific functionality has been assigned to the "Insert MSO destination" EI so what remains is to simply output the result to the user. Low complexity is assigned.

EO	Complexity	FPs
Ride termination	Medium	5
Reservation expired	Low	4
Select MSO power grid station	Low	4
TOTAL		**13**

2.1.6 Overall estimation

The following table summarises the result of our estimation activity:

Function type	FPs
Internal Logic Files	59
External Logic Files	29
External Inputs	23
External Inquiries	15
External Outputs	13
TOTAL	**139**

Considering J2EE as a development platform and disregarding the aspects concerning the implementation of the mobile application (which can be thought as pure presentation with no business logic), we can estimate the total number of lines of code using, as a reference, the table illustrated at this link: http://www.qsm.com/resources/function-point-languages-table, from which we extract the relevant record:

	QSM SLOC/FP Data			
Language	**Avg**	**Median**	**Low**	**High**
J2EE	46	49	15	67

Using the average rate of conversion, we obtain the final estimation:

SLOC = 139 x 46 = **6394**

2.2 Cost and effort estimation: COCOMO II

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

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 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

A brief description for each scale driver:

- Precedentness: it reflects the previous experience of our team with the development of similar previously developed projects. Since we are not expert on the field, this value will be low.
- Development flexibility: it reflects the degree of flexibility in the development process with respect to the external specification and requirements. Our clients have pre established some clear and explicit goals that our system must achieve, but they also gave us space for several assumptions and interpretation so this value will be high.
- Risk resolution: reflects the level of awareness and reactiveness with respect to risks. The risk analysis we performed is pretty straightforward, but it does not go into a very detailed and algorithmic strategy for each of the possible risks, so we will set this value as nominal.
- Team cohesion: accounts for the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others. These difficulties may arise from differences in stakeholder objectives and cultures; difficulties in reconciling objectives; and stakeholders' lack of experience and familiarity in operating as a team. As far as we are concerned, all the stakeholders seem to share the same view about the project and are very applied to achieve the common result. The team members also, considering that this is not the first time in which they work together for a project, allows us to assign a high value for this factor.
- Process maturity: reflects the overall maturity of the team. We have previously worked on a project that involved the use of Java but not to develop an enterprise application so we will set this value to low.

The result of our evaluation is the following:

Scale Drivers	Factor	Value
Precedentedness (PREC)	Low	4.96
Development flexibility (FLEX)	High	2.03
Risk resolution (RESL)	Nominal	4.24
Team cohesion (TEAM)	High	3.29
Process maturity (PMAT)	Level 1 Upper	6.24
TOTAL		**20.76**

2.2.2 Cost Drivers

- **Required software reliability**: A failure of PowerEnJoy system would not pose a risk to human life but a defect in its functioning could definitely result in a significant financial loss, because user could not be able to book cars. This is the reason why the RELY driver is set to high.

RELY Cost Driver

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	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**: This measure considers the effective size of our database. We cannot estimate the size of it but considering that we might have to store several records inside the DB, and that we have obtained a reasonably low value of estimated SLOC, the DATA Cost Driver should be set to high.

DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	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**: Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. Using Table 19, of the referenced document "COCOMO II Model Definition Manual", we have evaluated the overall complexity of the problem with a nominal value.

CPLX Cost Driver

Rating Levels	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**: In our case, the reusability requirements are limited to the scope of the project itself, so the RUSE cost driver is set to nominal.

RUSE Cost Driver

RUSE Descriptors:		none	across project	across program	across product line	across multiple product
STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- **Documentation match to life-cycle needs**: This parameter describes the relationship between the documentation and the application requirements. In our case, every need of the product life-cycle is already foreseen in the documentation, so the DOCU cost driver 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 Levels	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 constraint**: This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource. Due to a sincere improvement in the efficiency of modern CPUs, we do not overestimate the use of available execution time required by our system, and because of this, we set the TIME Cost Driver to a nominal value.

TIME Cost Driver

TIME Descriptors:			≤ 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 Levels	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: This rating represents the degree of main storage constraint imposed on a software system or subsystem. Following the same considerations provided for the TIME Cost Driver, the remarkable increase in available processor execution time and main storage, allows us not to worry about storage capacity. Given the necessities of our software system, which do not rely on massive storage availability, we decided to set this STOR Cost Driver to nominal.

STOR Cost Driver

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	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: For what concerns the core system, we do not expect our fundamental platforms to change very often. However, the client applications may require at least a major release once every six months to keep up with the pace imposed by the improvements on the smartphones operating systems. For this reason, this parameter is set to nominal.

PVOL Cost Driver

PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	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**: We think the analysis of the problem has been conducted in a thorough and complete way with respect to a potential real world implementation, considering the documentation that we have produced so far (RASD, DD, ITPD). For this reason, this parameter is set to high.

ACAP Cost Driver

ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	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**: This Cost Driver is not rated based on the experience of the programmer. Instead, evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. We value this estimation as being high, given the capability of our team to collaborate in a very efficient and cooperative manner.

PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	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**: This parameter is quite relevant in our case, since the time we can spend on this project is limited. For this reason, this parameter is set to very low.

PCON Cost Driver

PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

- **Application experience**: The rating for this cost driver is dependent on the level of applications experience of the project team developing the software system. We have some experience in the development of Java applications, but we have never tackled a Java EE system before. Because of this, we set this parameter to low.

APEX Cost Driver

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	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**: We don't have any experience with the Java EE platform, and very few preceding practices concerning the development of distributed system and user interfaces. For this reason, we're going to set this parameter to low.

PLEX Cost Driver

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	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**: This is a measure of the level of programming language and software tool experience of the project team developing the software system. Our experience with the project's programming language (JEE) is limited, and we think that, even though we have used tools that perform requirements and design representation and analysis, we do not want to overestimate our abilities and so set a LTEX Cost Driver to low.

LTEX Cost Driver

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

- **Use of software tools**: We have conceived, especially in the integration testing plan document, several tools that will ease the development and testing of the system. However, the listed tools, with the addition for example of a common Java IDE, are nothing but basic life-cycle tools, moderately integrated. For this reason, TOOL Cost Driver is set to nominal.

TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend 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 Levels	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**: Determining this cost driver rating involves the assessment and judgement-based averaging of two factors: site collocation and communication support. For what concerns the site collocation, we all three live in separate cities. This results in a low value for this case. Instead, we need to set the value to very high for the communication support, because we severely relied on wideband electronic communication, which occasionally led to video conferences between the team members. These two values will be averaged to result in an effort multiplier of 0.975.

SITE Cost Driver

SITE: Collocation Descriptors:	Inter-national	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 communicat ion.	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating Levels	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**: Despite our efforts were well distributed over the available development time, the definition of all the required documentation took a consistent amount of time, especially for the requirement analysis and the design phases. For this reason, this parameter is set to high.

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 Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

Overall, the result of the evaluation is expressed in the following table:

Cost Driver	Cost Driver	Value
Required software reliability (RELY)	High	1,10
Database size (DATA)	High	1,14
Product complexity (CPLX)	Nominal	1,00
Developed for reusability (RUSE)	Nominal	1,00
Documentation match to life-cycle needs (DOCU)	Nominal	1,00
Execution time constraint (TIME)	Nominal	1,00
Main storage constraint (STOR)	Nominal	1,00
Platform volatility (PVOL)	Nominal	0,88
Analyst capability (ACAP)	High	0,85
Programmer capability (PCAP)	High	0,88
Personnel continuity (PCON)	Very Low	1,29
Applications experience (APEX)	Low	1,10
Platform experience (PLEX)	Low	1,09
Language and tool experience (LTEX)	Low	1,09
Use of software tools (TOOL)	Nominal	1,00
Multisite development (SITE)	(Low) and (Very High)	0,975
Required development schedule (SCED)	High	1,00
TOTAL		**1,54184**

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** = 1.54184 [product of all cost drivers]
- **KSLOC** = 6.394 [Kilo Lines of Code, as previously estimated]
- **E** = 1.1176 [$B + 0.01 \times \sum_j SF_j$; B = 0.91 for COCOMO II ; $\sum_j SF_j = 20.76$]

With these parameters we can compute the final effort estimation value:

$$Effort = A \times EAF \times KSLOC^{E} = 2.94 \times 1.54184 \times 6.394^{1.1176} = \mathbf{36.05\ PM}$$

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** = 0.32152 [0.28 + 0.2 x (E - B) ; E = 1.1176 ; B = 0.91]
- **Effort** = 36.05 PM

With these parameters we can compute the final duration estimation value:

$$Duration = 3.67 \times EffortF = 3.67 \times (36.05)^{0.32152} = 11.62 \text{ months}$$

3.Schedule

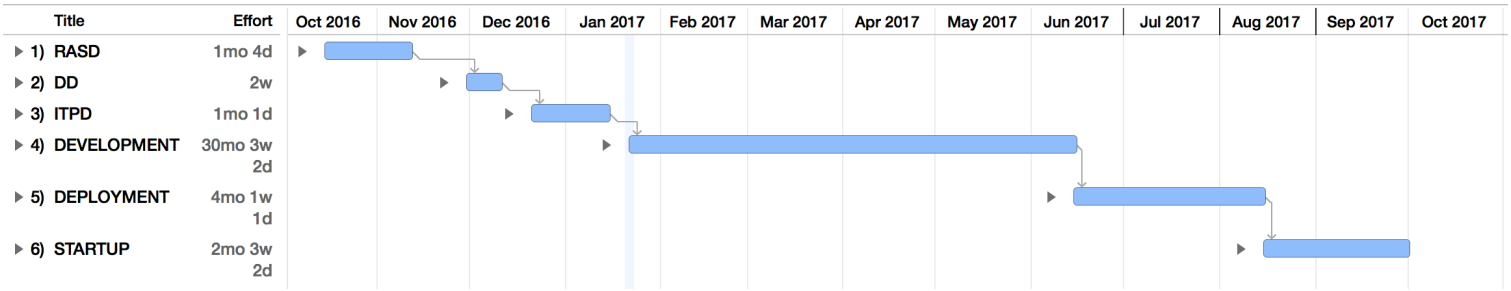
In this chapter, we are going to provide a general high-level project schedule.

The granularity of all the identified tasks is self explanatory and understandable from the following diagrams so we will not dig into further details about this topic.

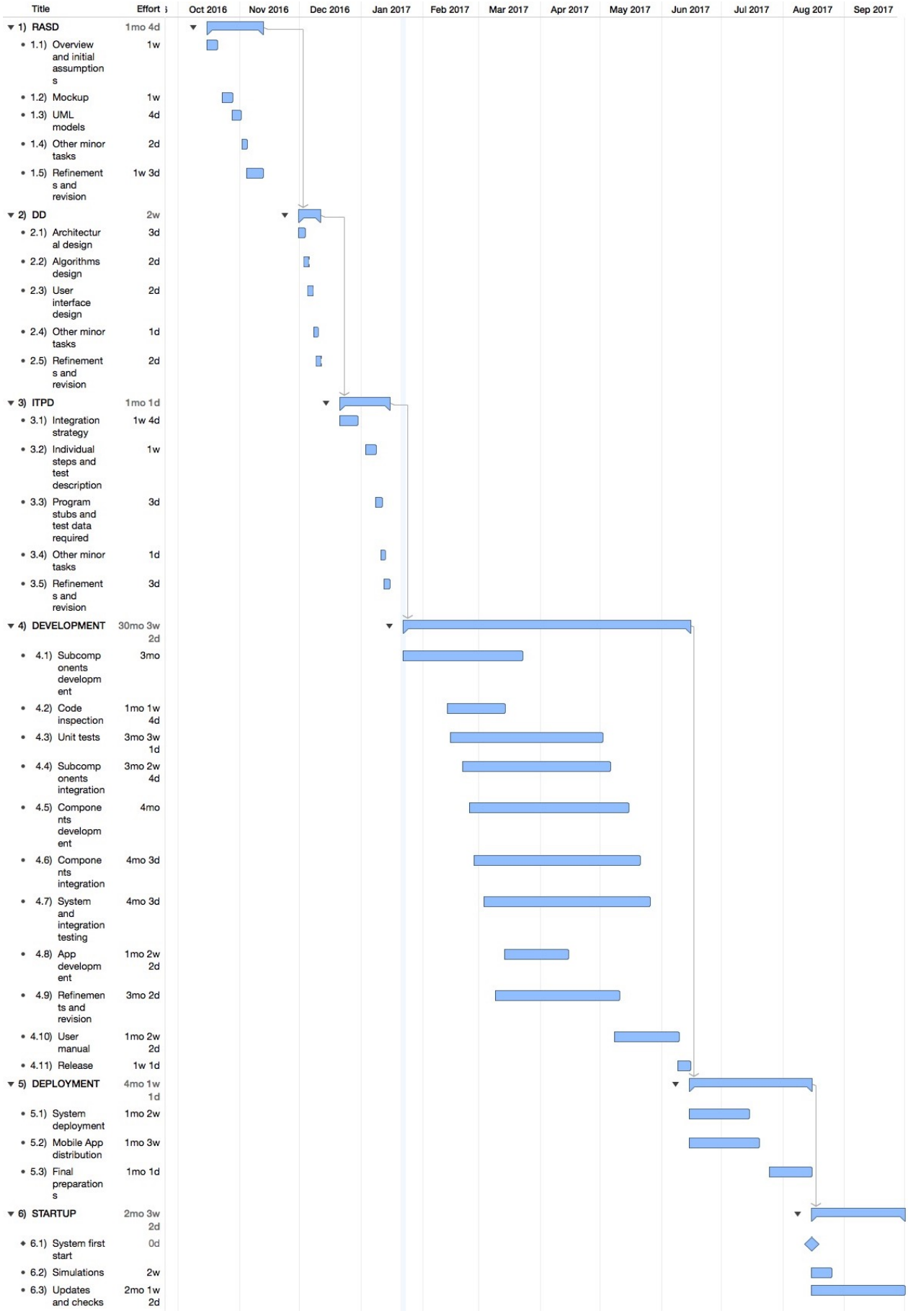
However, it is important to highlight a few things:

- From the beginning, October 16th 2016, up to the day of the ITPD deadline, which was on January 15th 2017, we have composed the schedule retrospectively, that means that we already know how all the actual work took place, in order to meet the imposed deadlines. From that point forward, we used the COCOMO evaluation that we estimated in the previous chapter, to predict a possible schedule for the remaining major tasks that would take place in an actual implementation of the project.
- We decided not to include in this schedule the Project Plan Document because, by trying to take into account a real implementation of the project, the project planning task should have taken place at the very beginning of the project, so it would make no sense to include it here like a task that takes place halfway through the schedule.
- We used the Gantt chart, which is a type of bar chart, devised by Henry Gantt in the 1910s, to illustrate the project schedule.

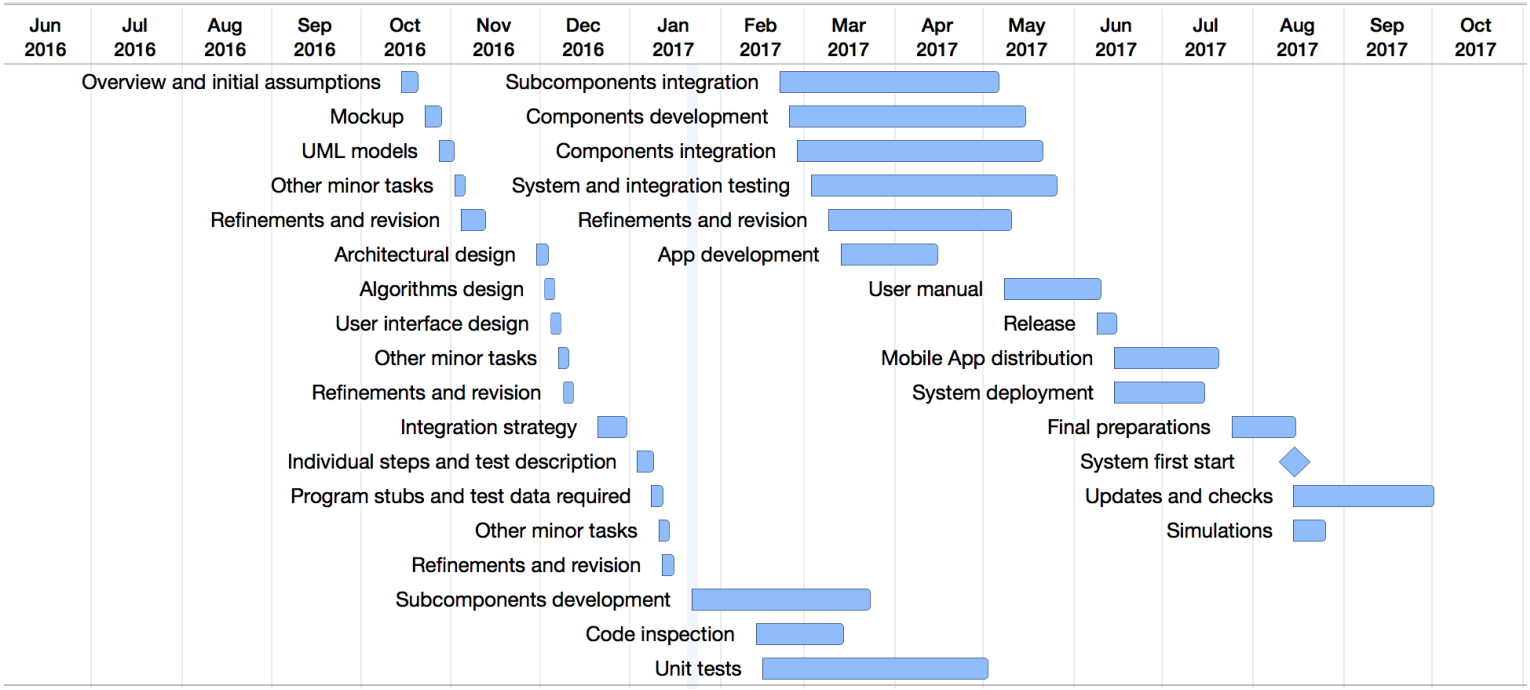
3.1 Overview


Title
Effort
Oct 2016
Nov 2016
Dec 2016
Jan 2017
Feb 2017
Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
1) RASD
1mo 4d
2) DD
2w
3) ITPD
1mo 1d
4) DEVELOPMENT
30mo 3w 2d
5) DEPLOYMENT
4mo 1w 1d
6) STARTUP
2mo 3w 2d

3.2 Detailed Gantt



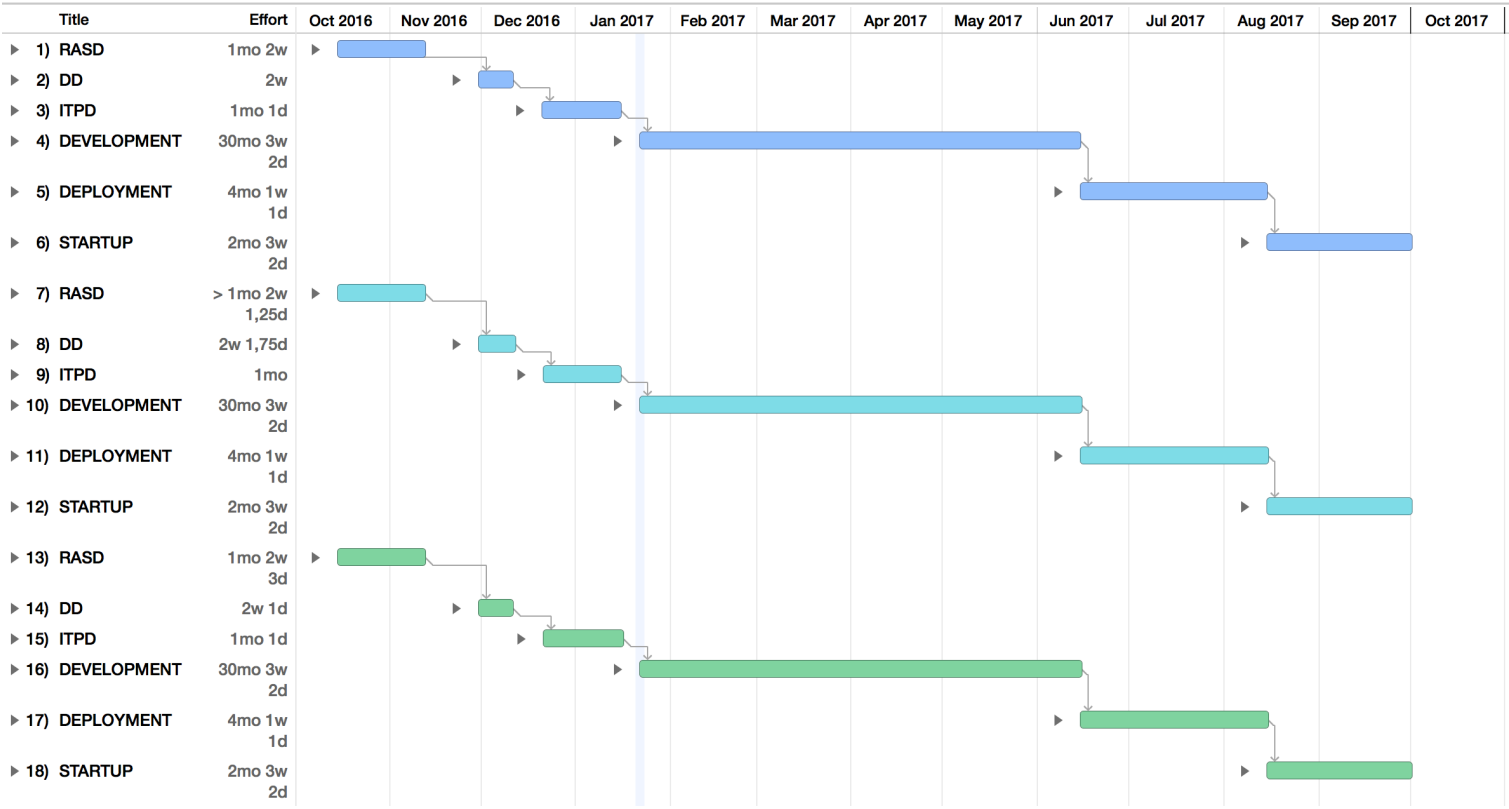
3.3 Detailed Timeline



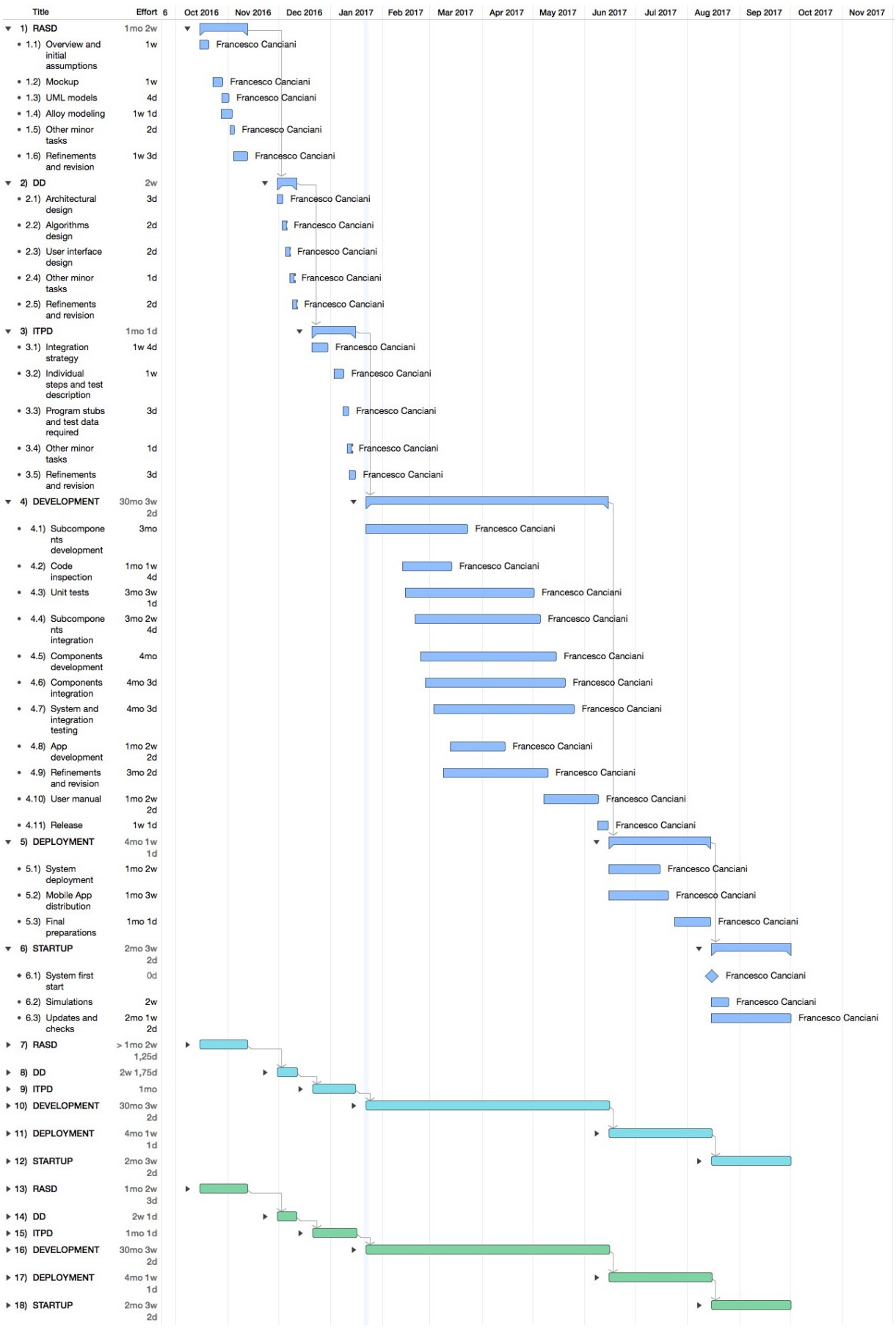
4.Resource allocation

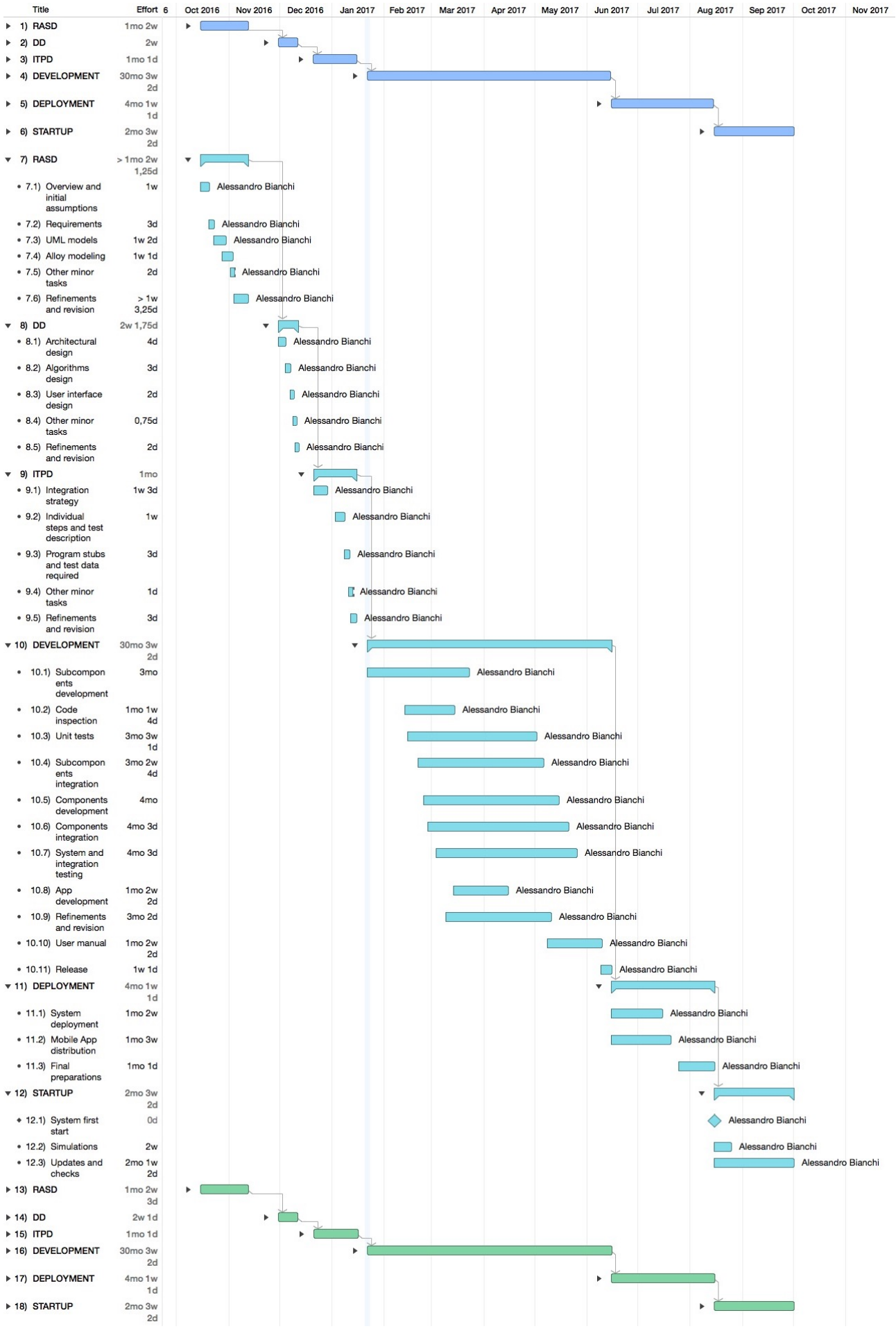
In this chapter we are going to provide a general overview of how the tasks defined in the previous schedule have been divided between the three team members. For this chapter also, the same observations that we made in the previous chapter hold.

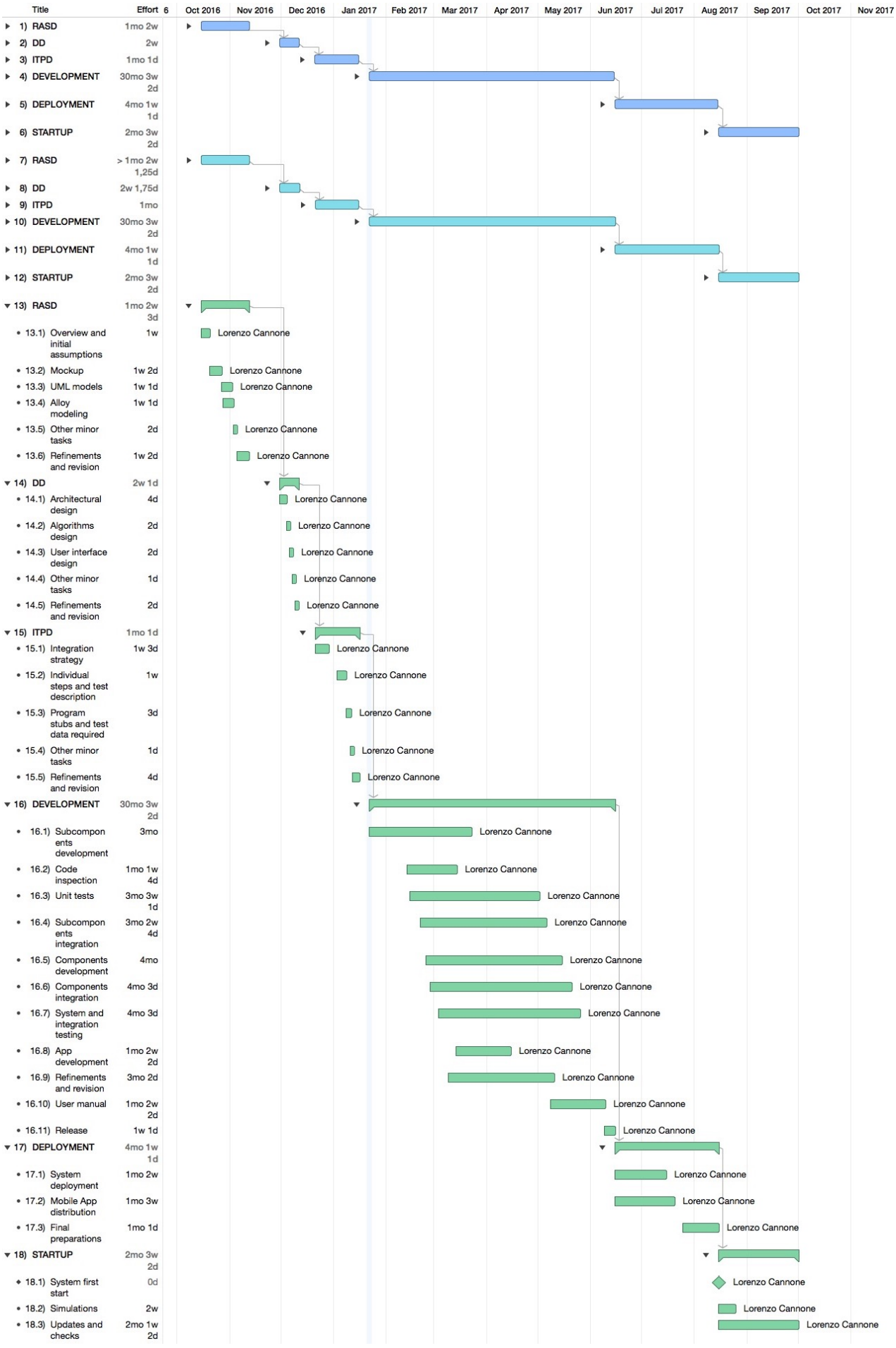
4.1 Overview Gantt

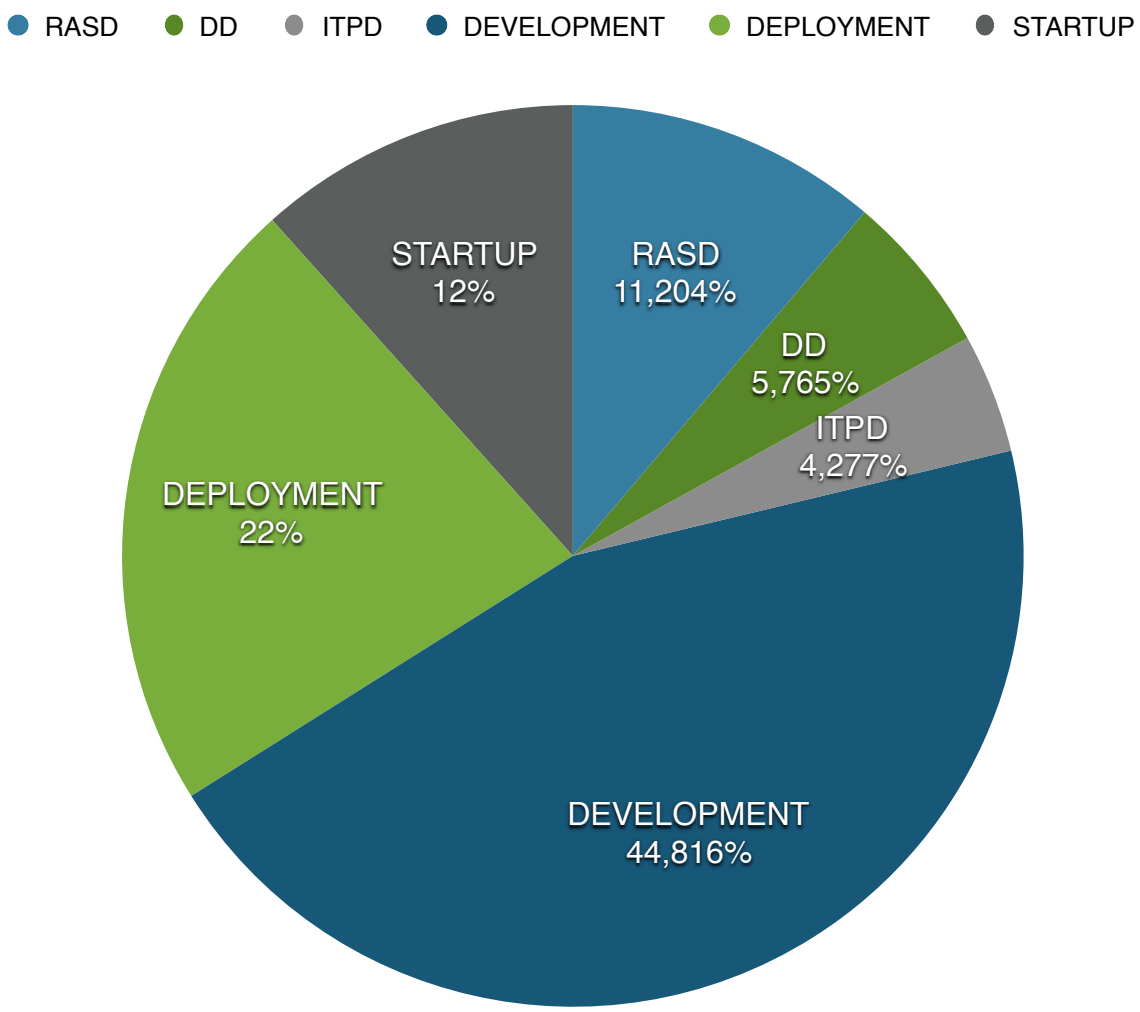


4.2 Detailed Gantt




Title
Effort 6
Oct 2016
Nov 2016
Dec 2016
Jan 2017
Feb 2017
Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
Nov 2017
1) RASD 1mo 2w
2) DD 2w
3) ITPD 1mo 1d
4) DEVELOPMENT 30mo 3w 2d
5) DEPLOYMENT 4mo 1w 1d
6) STARTUP 2mo 3w 2d
7) RASD > 1mo 2w 1,25d
• 7.1) Overview and initial assumptions 1w
• 7.2) Requirements 3d
• 7.3) UML models 1w 2d
• 7.4) Alloy modeling 1w 1d
• 7.5) Other minor tasks 2d
• 7.6) Refinements and revision > 1w 3,25d
8) DD 2w 1,75d
• 8.1) Architectural design 4d
• 8.2) Algorithms design 3d
• 8.3) User interface design 2d
• 8.4) Other minor tasks 0,75d
• 8.5) Refinements and revision 2d
9) ITPD 1mo
• 9.1) Integration strategy 1w 3d
• 9.2) Individual steps and test description 1w
• 9.3) Program stubs and test data required 3d
• 9.4) Other minor tasks 1d
• 9.5) Refinements and revision 3d
10) DEVELOPMENT 30mo 3w 2d
• 10.1) Subcomponents development 3mo
• 10.2) Code inspection 1mo 1w 4d
• 10.3) Unit tests 3mo 3w 1d
• 10.4) Subcomponents integration 3mo 2w 4d
• 10.5) Components development 4mo
• 10.6) Components integration 4mo 3d
• 10.7) System and integration testing 4mo 3d
• 10.8) App development 1mo 2w 2d
• 10.9) Refinements and revision 3mo 2d
• 10.10) User manual 1mo 2w 2d
• 10.11) Release 1w 1d
11) DEPLOYMENT 4mo 1w 1d
• 11.1) System deployment 1mo 2w
• 11.2) Mobile App distribution 1mo 3w
• 11.3) Final preparations 1mo 1d
12) STARTUP 2mo 3w 2d
• 12.1) System first start 0d
• 12.2) Simulations 2w
• 12.3) Updates and checks 2mo 1w 2d
13) RASD 1mo 2w 3d
14) DD 2w 1d
15) ITPD 1mo 1d
16) DEVELOPMENT 30mo 3w 2d
17) DEPLOYMENT 4mo 1w 1d
18) STARTUP 2mo 3w 2d
Alessandro Bianchi


Title
Effort
6
Oct 2016
Nov 2016
Dec 2016
Jan 2017
Feb 2017
Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
Nov 2017
1) RASD 1mo 2w
2) DD 2w
3) ITPD 1mo 1d
4) DEVELOPMENT 30mo 3w 2d
5) DEPLOYMENT 4mo 1w 1d
6) STARTUP 2mo 3w 2d
7) RASD > 1mo 2w 1,25d
8) DD 2w 1,75d
9) ITPD 1mo
10) DEVELOPMENT 30mo 3w 2d
11) DEPLOYMENT 4mo 1w 1d
12) STARTUP 2mo 3w 2d
13) RASD 1mo 2w 3d
• 13.1) Overview and initial assumptions 1w
• 13.2) Mockup 1w 2d
• 13.3) UML models 1w 1d
• 13.4) Alloy modeling 1w 1d
• 13.5) Other minor tasks 2d
• 13.6) Refinements and revision 1w 2d
14) DD 2w 1d
• 14.1) Architectural design 4d
• 14.2) Algorithms design 2d
• 14.3) User interface design 2d
• 14.4) Other minor tasks 1d
• 14.5) Refinements and revision 2d
15) ITPD 1mo 1d
• 15.1) Integration strategy 1w 3d
• 15.2) Individual steps and test description 1w
• 15.3) Program stubs and test data required 3d
• 15.4) Other minor tasks 1d
• 15.5) Refinements and revision 4d
16) DEVELOPMENT 30mo 3w 2d
• 16.1) Subcomponents development 3mo
• 16.2) Code inspection 1mo 1w 4d
• 16.3) Unit tests 3mo 3w 1d
• 16.4) Subcomponents integration 3mo 2w 4d
• 16.5) Components development 4mo
• 16.6) Components integration 4mo 3d
• 16.7) System and integration testing 4mo 3d
• 16.8) App development 1mo 2w 2d
• 16.9) Refinements and revision 3mo 2d
• 16.10) User manual 1mo 2w 2d
• 16.11) Release 1w 1d
17) DEPLOYMENT 4mo 1w 1d
• 17.1) System deployment 1mo 2w
• 17.2) Mobile App distribution 1mo 3w
• 17.3) Final preparations 1mo 1d
18) STARTUP 2mo 3w 2d
• 18.1) System first start 0d
• 18.2) Simulations 2w
• 18.3) Updates and checks 2mo 1w 2d
Lorenzo Cannone


RASD
DD
ITPD
DEVELOPMENT
DEPLOYMENT
STARTUP
STARTUP
12%
RASD
11,204%
DD
5,765%
ITPD
4,277%
DEPLOYMENT
22%
DEVELOPMENT
44,816%

5.Risk management

This section of the document is about risk management. Risk management is defined to be the identification, assessment, and prioritisation of risks, followed by a strategy or contingency plan that puts into words an action schema to manage them in a controlled and effective manner.

In the following table, some of the major risks that PowerEnJoy's development could face are assessed. For each of the identified risks, the probability that it will occur and the impact that it will do if it does occur are evaluated. In the last column of the table, we briefly provide a possible counteraction for each of the identified risks.

All the risks have been listed in a descending order of impact.

Risk	Probability	Impact	Strategy
Failure to achieve milestones and objectives.	Medium	Severe	New team meeting is required in order to recompose a schedule of the future activities to minimize additional costs.
One of the major stakeholders stops financiating the project or reduces dramatically its support.	Medium	Severe	Negotiations with the involved stakeholder to propose a more active role in the development of the project.
Complete loss of significant parts of the system's code implementation.	Low	Severe	Implement appropriate versioning systems and backup techniques distributed over multiple, redundant locations.
Number of customers does not keep up with the expected amount.	Medium	Significant	Sales team needs to come up with new ways to win the support of the customers (special offers, discounts for starters, incentives of any kind.

Risk	Probability	Impact	Strategy
Key staff member are ill at critical times in the project.	Medium	Significant	Team reorganization to identify a new resource allocation plan. Team members will have to deal with tasks that they were not assigned to at the beginning.
Underestimation of the correct performance of external components.	High	Significant	Design code to be as portable and platform independent as possible, in order to be easily adaptable in case of external components changes.
Application is too difficult to use for the average user or it is badly reviewed in the app stores.	Low	Moderate	Developers have to conceive a new UI that facilitates app's use, also by taking in consideration users' feedbacks.
Overestimation of developers' knowledge for a specific matter.	Low	Moderate	Put more attention in the hiring process beforehand to employ more knowledgeable people.
Car subsystem ends up being not very reliable.	Low	Moderate	Investigate the possibility of buying better performing devices to improve reliability.
Changes in the national legislation for what concerns car sharing.	Low	Minor	Always stay informed on the topic to be prepared in case of changes (typically laws take a long time to be approved).

6. Hours of work

To redact this document, we spent approximately 10 hours per person.