POLITECNICO
MILANO 1863

POWER ENJOY

Project Plan Document

Lorenzo Casalino - 877421

Tommaso Castagna

Document version 1.0

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.

1.2 Purpose

1.3 Scope

1.4 Definitions, Acronyms and Abbreviations

1.5 Reference Documents

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.

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.

Regarding the effort and cost estimation, the process is based on **algorithmic approach**, that is the use of an algorithmic model based on a simple equation which output depends on several factors regarding the project. The algorithmic model here used is based on *COCOMO II*.

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 category:

- **Internal Logical File.**
- **External Logical File.**
- **External Input.**
- **External Output.**
- **External Inquiry.**

For each of these type, 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*.

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

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

$$PFP_t = N_t * FP_t, t \in T = [ILF, ELG, EI, EO, EINQ]$$

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

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

$$UFP = \sum_t FPF_t, t \in T$$

2.1.1 Internal Logic File

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

- **User account informations**: this file contains the informations that any user is asked to type 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.

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	

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 a meaningful estimations, the following assumptions about the external services are taken in consideration.

	Data elements		
RET	$1 \rightarrow 4$	$5 \rightarrow 15$	15+
¡ 2	Low	Low	Average
2	Low	Average	High
¿ 2	Average	High	High

- **Driver licenses validator web service files**: we assume that the web service needs only to use informations regarding stored driver licenses and drivers informations, hence use of two record structure 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 plotting and so on involves a complex external file.

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	

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.
- **Rule policy activation**: whenever the driver performs a particular action described into the policy rules, the software system reacts applying the related effects to the reservation. Reservation informations and policy internal logic files are 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.

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
Rule policy activation	Average	4
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 Point	72	

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.
- **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 involvedo.
- **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.

IQ	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	

2.1.5 External Output

- **Notify reservation expired**.
- **Notify policy rule(s) application**.
- **Notify registration result**.

- **Notify login result**.
- **Notify logout result**.
- **Notify insertion result**.
- **Notify deletion result**.
- **Notify update result**.
- **Notify reservation request result**.
- **Notify unlock request result**.

EO	Complexity	Weigth
Notify reservation expired	Low	3
Notify policy rule(s) application	Low	3
Notify registration result	Low	3
Notify login result	Low	3
Notify logout result	Low	3
Notify insertion result	Low	3
Notify deletion result	Low	3
Notify update result	Low	3
Notify reservation request result	Low	3
Notify unlock request result	Low	3
Partial Function Point	30	

Applying the equation above shown, the unadjasted function point value is equal to 221 points. To give much flexibility as possible to the project, the gearing factors used to estimate the lower bound and the upper bound are calculated as the mathematical average between the gearing factors of J2EE and .NET, the two most widespread technologies.

Lower bound : 52 * 221 = 11.492 kilo-lines of code Upper bound : 64 * 221 14.144 kilo-lines of code

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.

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

EO	**Complexity**	**Weigth**
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	

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

$$E = B + 0.01 * \sum_{f}(SF_f) = 1,0838$$

with $B = 0.91$ and f representing one of the scale drivers.

3 Tasks schedule

This project in composed of 5 main parts:

1. Requirements Analysis and Specification Document
2. Design Document
3. Integration Testing Plan Document
4. Project Plan Document
5. Development

The first four parts of the project had a deadline already established, instead we assumed that four months are needed in order to complete the implementation part. 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 following images show the tasks related to this project.

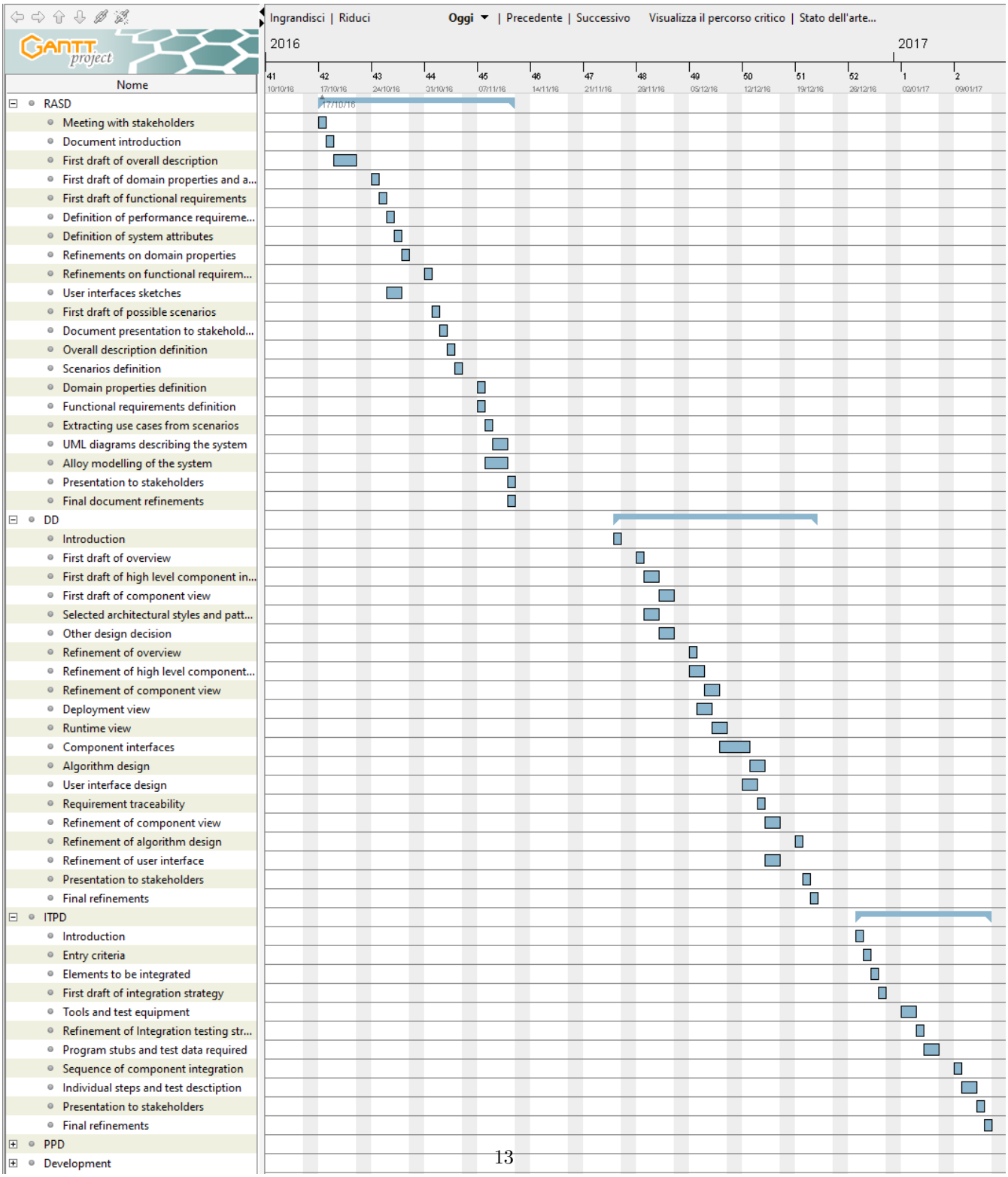

Figure 1: Project tasks 1

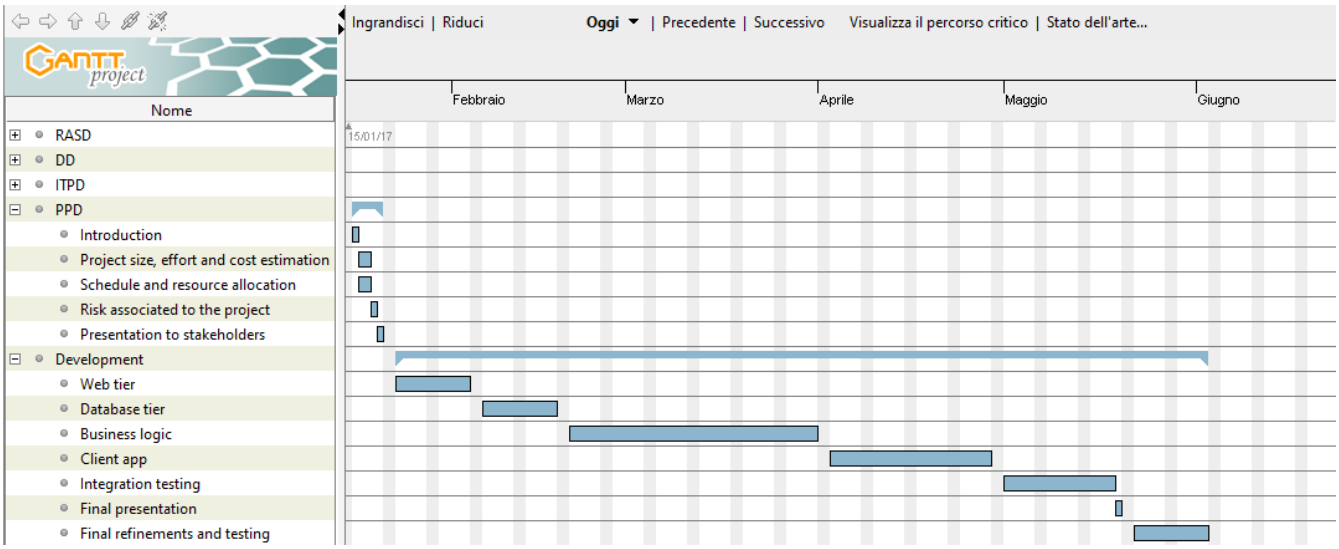

Figure 2: Project tasks 2

Following is a more accurate definition of each task deadline.

PowerEnJoy 21-gen-2017

Attività 2

Nome	Data di fine	Data d'inizio
RASD	11/11/16	17/10/16
Meeting with stakeholders	17/10/16	17/10/16
Document introduction	18/10/16	18/10/16
First draft of overall description	21/10/16	19/10/16
First draft of domain properties and assumptions	24/10/16	24/10/16
First draft of functional requirements	25/10/16	25/10/16
Definition of performance requirements	26/10/16	26/10/16
Definition of system attributes	27/10/16	27/10/16
Refinements on domain properties	28/10/16	28/10/16
Refinements on functional requirements	31/10/16	31/10/16
User interfaces sketches	27/10/16	26/10/16
First draft of possible scenarios	01/11/16	01/11/16
Document presentation to stakeholders	02/11/16	02/11/16
Overall description definition	03/11/16	03/11/16
Scenarios definition	04/11/16	04/11/16
Domain properties definition	07/11/16	07/11/16
Functional requirements definition	07/11/16	07/11/16
Extracting use cases from scenarios	08/11/16	08/11/16
UML diagrams describing the system	10/11/16	09/11/16
Alloy modelling of the system	10/11/16	08/11/16
Presentation to stakeholders	11/11/16	11/11/16
Final document refinements	11/11/16	11/11/16
DD	21/12/16	25/11/16
Introduction	25/11/16	25/11/16
First draft of overview	28/11/16	28/11/16
First draft of high level component interaction	30/11/16	29/11/16
First draft of component view	02/12/16	01/12/16
Selected architectural styles and patterns	30/11/16	29/11/16
Other design decision	02/12/16	01/12/16
Refinement of overview	05/12/16	05/12/16
Refinement of high level component view	06/12/16	05/12/16
Refinement of component view	08/12/16	07/12/16
Deployment view	07/12/16	06/12/16
Runtime view	09/12/16	08/12/16
Component interfaces	12/12/16	09/12/16
Algorithm design	14/12/16	13/12/16
User interface design	13/12/16	12/12/16
Requirement traceability	14/12/16	14/12/16
Refinement of component view	16/12/16	15/12/16
Refinement of algorithm design	19/12/16	19/12/16
Refinement of user interface	16/12/16	15/12/16
Presentation to stakeholders	20/12/16	20/12/16
Final refinements	21/12/16	21/12/16

Attività

Nome	Data di fine	Data d'inizio
ITPD	13/01/17	27/12/16
Introduction	27/12/16	27/12/16
Entry criteria	28/12/16	28/12/16
Elements to be integrated	29/12/16	29/12/16
First draft of integration strategy	30/12/16	30/12/16
Tools and test equipment	03/01/17	02/01/17
Refinement of Integration testing strategy	04/01/17	04/01/17
Program stubs and test data required	06/01/17	05/01/17
Sequence of component integration	09/01/17	09/01/17
Individual steps and test desctiption	11/01/17	10/01/17
Presentation to stakeholders	12/01/17	12/01/17
Final refinements	13/01/17	13/01/17
PPD	20/01/17	16/01/17
Introduction	16/01/17	16/01/17
Project size, effort and cost estimation	18/01/17	17/01/17
Schedule and resource allocation	18/01/17	17/01/17
Risk associated to the project	19/01/17	19/01/17
Presentation to stakeholders	20/01/17	20/01/17
Development	02/06/17	23/01/17
Web tier	03/02/17	23/01/17
Database tier	17/02/17	06/02/17
Business logic	31/03/17	20/02/17
Client app	28/04/17	03/04/17
Integration testing	18/05/17	01/05/17
Final presentation	19/05/17	19/05/17
Final refinements and testing	02/06/17	22/05/17

4 Reseurces allocation

This section shows how the 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.

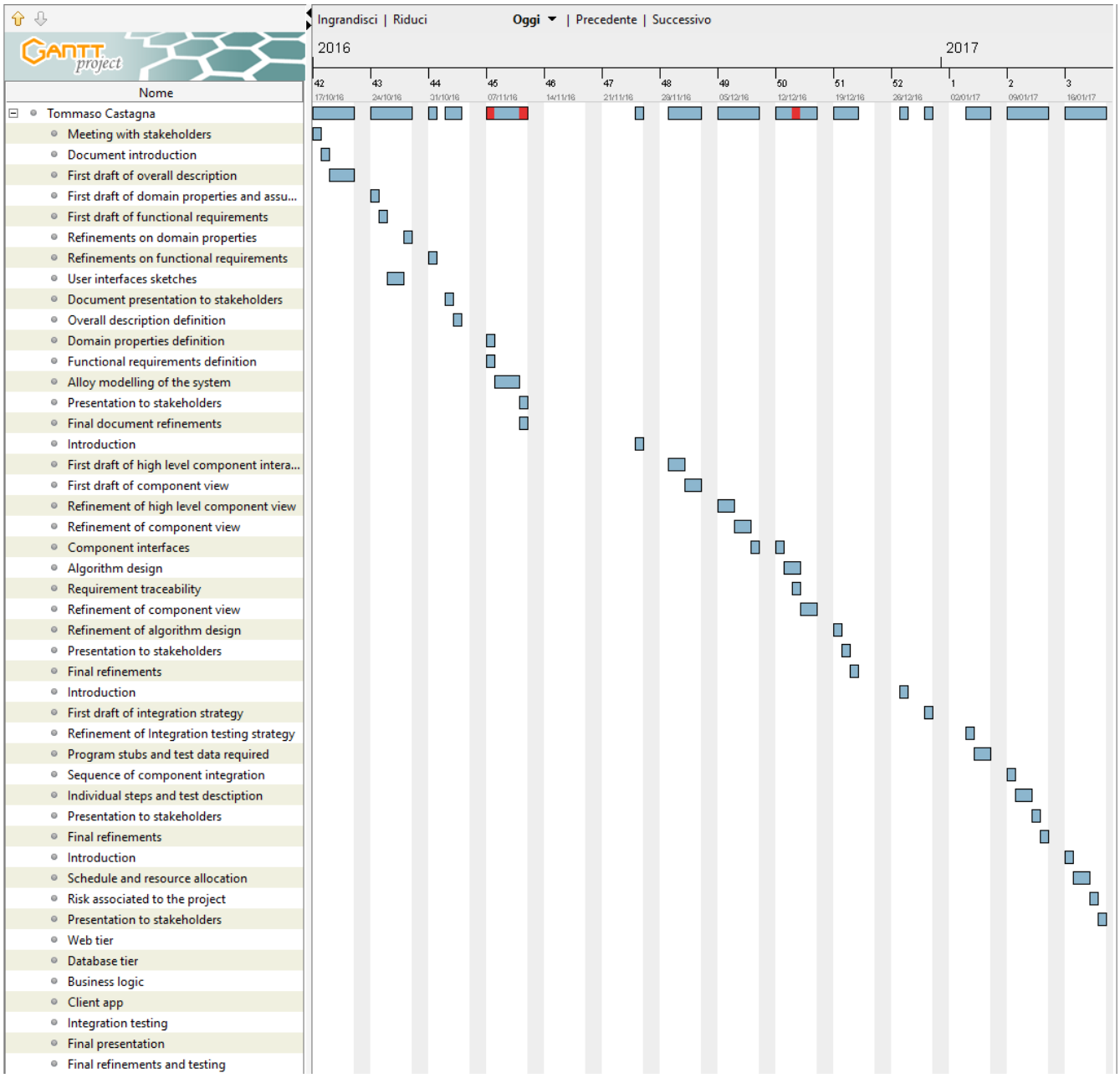

Figure 3: Tommaso allocation 1

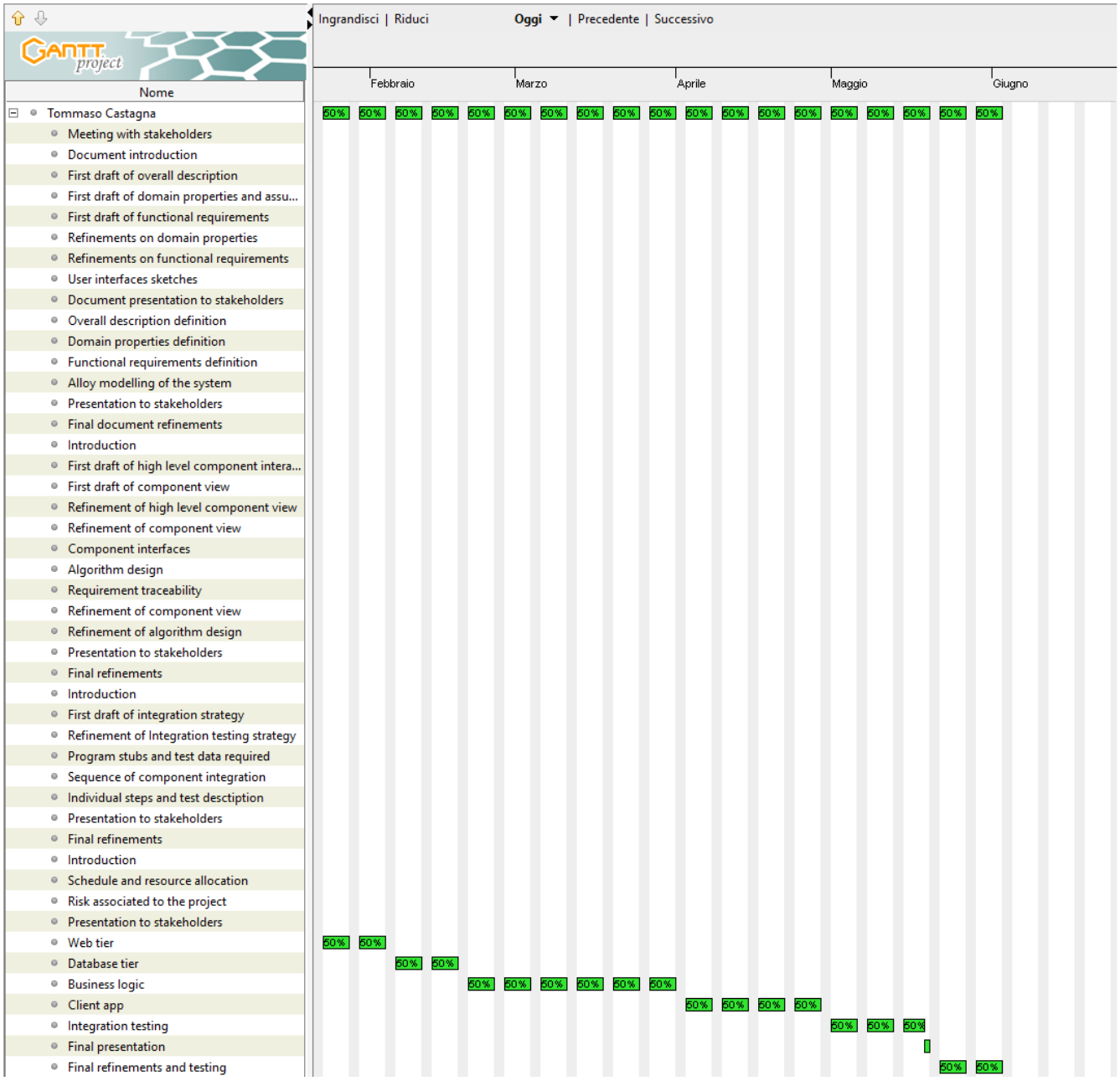

Figure 4: Tommaso allocation 2

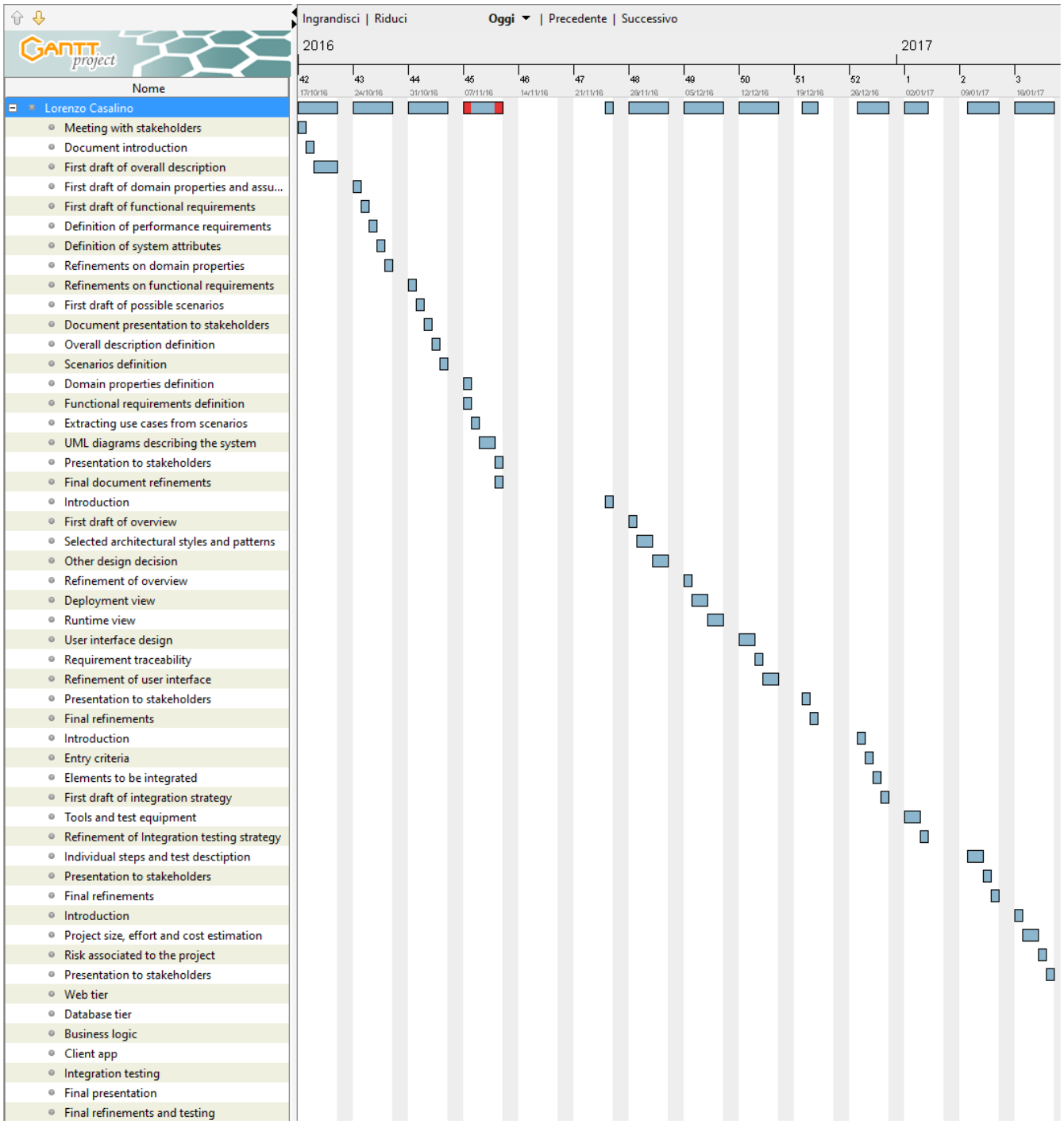

Figure 5: Lorenzo allocation 1

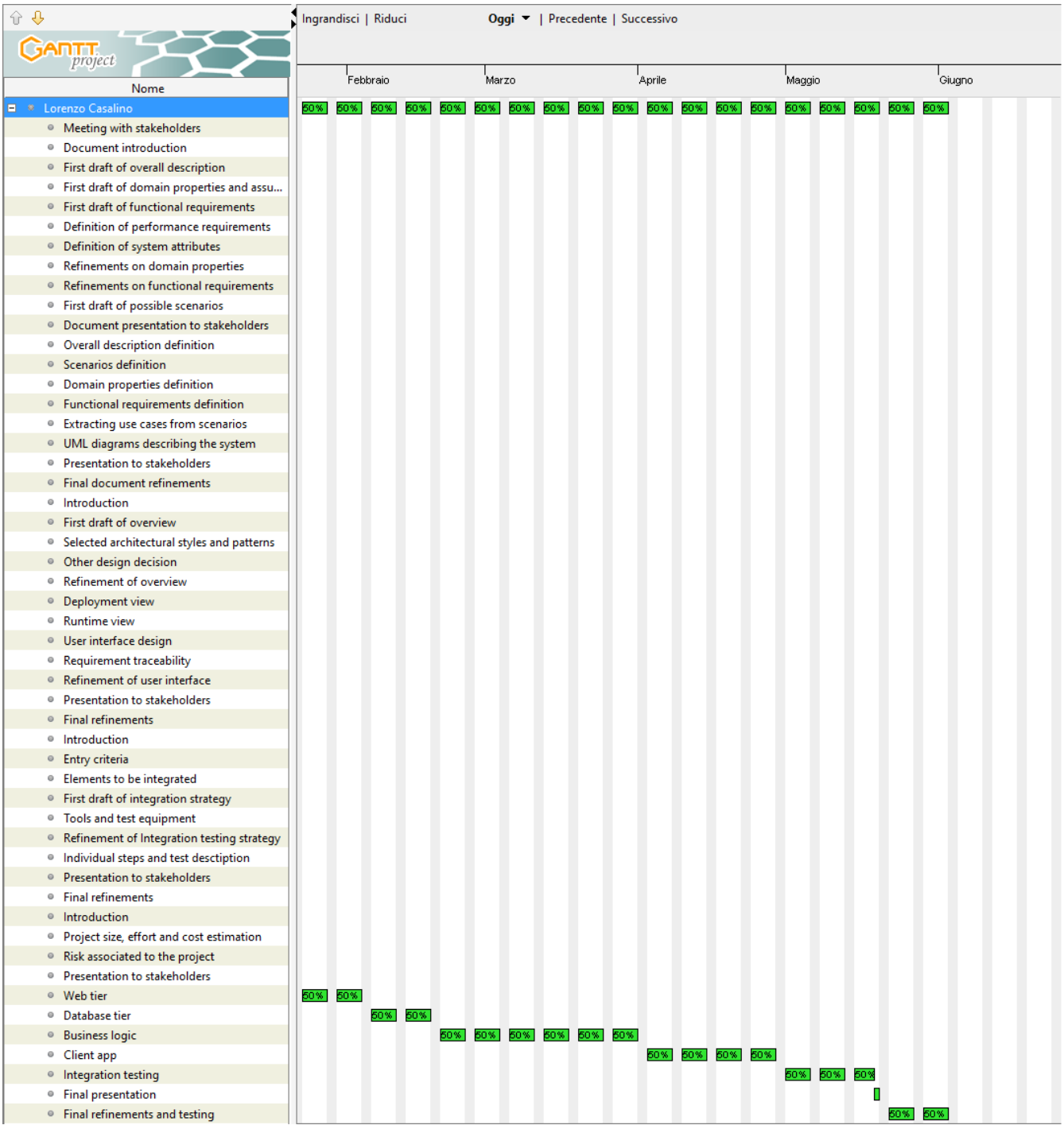

Figure 6: Lorenzo allocation 2