Politecnico di Milano

A.A. 2015-2016

Software Engineering 2 project

Project Plan

Alessandro Pozzi (mat. 852358), Marco Romani (mat. 852361)

2 February 2016

Version 1.0

Summary

Introduction

1.1 Revision history

February 2, 2016 – First Version (1.0) of this document.

1.2 Document Introduction

The aim of this document is to plan a subdivision of the project into tasks and create a schedule that can be followed during the implementation phase. However, in the MyTaxiService's project context, such planning will be referred retrospectively to the preparation and writing of MTS's documents: the Requirement Analysis and Specification Document, the Design Document, the Integration Test Plan Document and this very same document. In addition, the Code Inspection Document, even if not related with MTS, will be considered.

Furthermore, the Function Points approach and the COCOMO model will be used to evaluate the effort, the duration and other parameters related to MyTaxiService's application. Such analysis will be, of course, referred to an eventual real implementation of the system.

1.3 List of definitions and abbreviations

- MTS – MyTaxiService

1.4 List of references

1.4.1 MyTaxiService's Documents

- MyTaxiService's Requirements and Analysis and Specification Document (Alessandro Pozzi, Marco Romani)
- MyTaxiService's Design Document (Alessandro Pozzi, Marco Romani)
- MyTaxiService's Integration Test Plan Document (Alessandro Pozzi, Marco Romani)

1.4.2 Other references

- COCOMO manual:

http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf

- Table for the SLOC conversion (COCOMO):

  http://www.qsm.com/resources/function-point-languages-table

2. Cost estimation

2.1 Functions points

We will now evaluate the parameters that are used to calculate the Function Points.

2.1.1 Internal logical files

MTS has to store internal information about:

- Taxi
- Users
- Rides
- City zones
- Taxi queues

These can all be considered simple structures, except the Rides, which are composed by many crucial parameters, and the city zones, which are more complex due to the fact that they represent geographical locations that can be interconnected. These two structure can be considered to be of a medium complexity.

Thus, 3 $(simple\ strucuture) * 7\ (simple\ weight) + 2\ (medium\ structure) * 10\ (medium\ weight) = 41$

2.1.2 External interface files

The only data provided by the external interfaces is:

- Taxi location

The taxi location contains only information about the taxi id and its GPS position, thus it can be considered as a simple structure.

$$1\ (simple\ structure) * 5\ (simple\ weight) = 5$$

2.1.3 External inputs

The following are the functionalities offered by the MTS application which can be considered as inputs.

Simple complexity operations:

- Login

- Logout
- Change the taxi driver's availability (driver side)
- Accept a ride
- Refuse a ride

Medium complexity operations:

- Registration
- Request a ride
- Create a driver's account (admin side)
- Delete a ride
- Delete a driver's account (admin side)
- Change the status of a ride (admin side)

High complexity operations:

- Reserve a ride

$$5\ (simple\ operations) * 3(simple\ weight) + 6\ (medium\ operations) * 4\ (medium\ weight) + 1\ (complex\ operation) * 6\ (complex\ weight) = 45$$

Where we have considered as *simple* all the operations that involves basic functionalities and only little data processing of the *Internal Logical Files.* Medium complexity operations are considered to be involving a higher data processing and management, especially regarding the creation or deletion of data. "Reserve a ride" has been considered as the only highly complex operation because it's a more advanced version of the "Request a ride" functionality.

2.1.4 External Outputs

- Emails
- Mobile app notifications
- Web notifications
- Ride requests to taxi drivers

Here we consider all the operations as simple, except for the "Ride request to taxi drivers" because it contains data about an Internal Logical File: the ride.

$$3\ (simple\ outputs) * 4\ (simple\ weight) + 1\ (complex\ output) * 7\ (complex\ weight) = 19$$

2.1.5 External inquiries

- Visualize the users' account information
- Visualize the details of a single ride
- Visualize the history of rides of a user

We consider the first two operations as simple, while the "Visualize the history of rides of a user" is a medium complexity operation because it involves the retrieval of multiple data.

$$2\ (simple\ inquiries) * 3\ (simple\ weight) + 1(medium\ inquiry) * 4\ (medium\ weight) = 10$$

2.1.6 Total number of function points

The un-adjusted function points (UFP) results:

$$UFP = 120$$

This value can be further adjusted by applying a final correction which depends by other parameters that can be extracted from the project's design phase. However, we won't perform such correction because it usually doesn't improve the precision of the estimation (in some cases it may even get it worse). Instead, we will use the UFP estimation in combination with COCOMO in order to estimate the project effort.

2.2 COCOMO

In the documents describing the MyTaxiService's project we have always tried to be as detached from the project's language as possible. In such way we managed to describe a high level view of the application without bounding the developers to the limitations (or, possibly, advantages) of a specific language. However, here we will evaluate the COCOMO parameters as if the project was to be developed in the J2EE language, which however seems to be the most obvious choice for this kind of application.

The average Source Lines of Code (**SLOC**) are calculated using a conversion factor of 46, as described in the table at http://www.qsm.com/resources/function-point-languages-table.

$$120\ FP * 46 = 5520\ lines\ of\ code$$

To calculate the project's **Effort**, we first evaluate the **Scale Drivers** according to the COCOMO manual (see the section 1.4 for references). The following parameters were estimated:

Scale Driver	**Factor**	**Value**
Precedentedness	Low	4.96
Development Flexibility	High	2.03
Risk Resolution	High	2.83
Team Cohesion	Very High	1.10
Process Maturity	Normal	4.68
Total		15.6

In the following table we have estimated the **Cost Drivers**:

Cost Driver	**Factor**	**Value**
Required Software Reliability	Low	0.92
Data Base Size	Nominal	1.00
Product Complexity	Nominal	1.00
Required Reusability	High	1.07
Documentation match to life-cycle needs	Nominal	1.00
Execution Time Constraint	Low	n/a
Main Storage Constraint	Low	n/a
Platform Volatility	Nominal	1.00
Analyst Capability	Very High	0.71
Programmer Capability	Nominal	1.00
Application Experience	Low	1.10
Platform Experience	Nominal	1.00
Language and Tool Experience	Low	1.09

Personnel continuity	Low	1.12
Usage of Software Tools	Nominal	1.00
Multisite development	Nominal	1.00
Required development schedule	Nominal	1.00
Product		0.93

From the Cost Drivers we can obtain the **exponent E**, which will be used in the Effort equation.

$$E = B + 0.01 * \sum ScaleDrivers\ Factor$$

With B = 0.91 (for COCOMO II.2000) we obtain:

$$E = 0.91 + 0.01 * 15.6 = 1.066$$

It is now possible to calculate the **Effort**:

$$Effort = A * EAF * KSLOC^{E}$$

With A=2.94 (for COCOMO II.2000) we obtain:

$$Effort = 2.94 * 0.93 * 5.520^{1.066} = 16{,}89\ Person/Month$$

An interesting value is also the **duration** of the project:

$$Duration = 3.67 * Effort^{F} = 8.840 = 9\ Months$$

Where $F = 0.28 + 0.2 * (E - B) = 0.311$

The **Number of people** required is:

$$N_{people} = \frac{effort}{duration} = 2.11 = 2\ people$$

An overview of the parameters seems to suggest a quite correct estimation. A group of two people should be able to develop properly the MTS application in a timespan of eight months. However, in our opinion, the final number of lines of code will probably be greater than the estimated.

3. Task identification

In this section we identify the main tasks of each assignment (i.e. of each document to be written) of the project. We have selected the tasks with a granularity that almost corresponds to the sections or paragraphs of each document.

Furthermore, for certain set of tasks some precedence constraints will be declared. However, strict constraints will be rare: virtually, every activity can be started and finished in any order, as long as the main points on which it lays have been discussed before between the group members. Note that this is also one the main risk of the project, as discussed in section 5.

In the diagram, we have used a green color to identify the tasks that were completed by a single person, while different colors have been used for the ones shared among the team members. This subdivision will be clear in the section 4.

3.1 RASD

Assignment date: 15 October 2015

Deadline: 6 November 2015

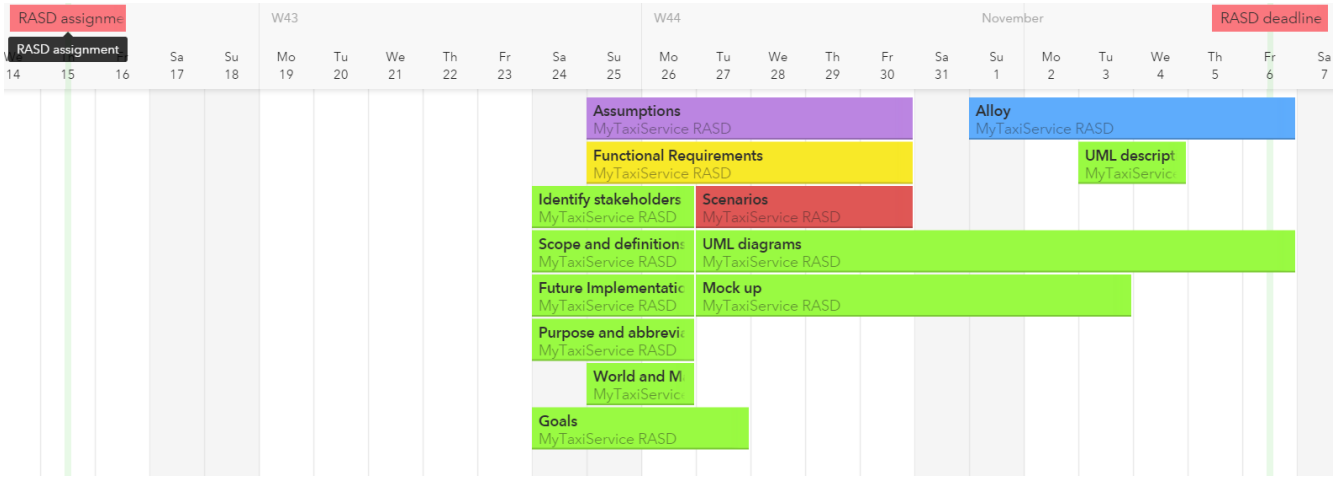
Tasks list:

- Assumptions
- Alloy
- Functional Requirements
- Identify StakeHolders
- Scenarios
- UML description
- Scope and Definitions
- UML Diagram
- Future Implementations
- Mockups
- Purpose and abbreviations
- World and the Machine
- Goals

Precedence:

- The *Goal* task must begin before the beginning of the *Functional Requirements* task.

- The *Functional Requirements* task must begin before the beginning of the *UML diagrams* task.
- The *Alloy* task must begin after the beginning of the *World and the Machine, Functional Requirements and Assumptions* tasks.



3.2 Design Document

Assignment date: 12 November 2015

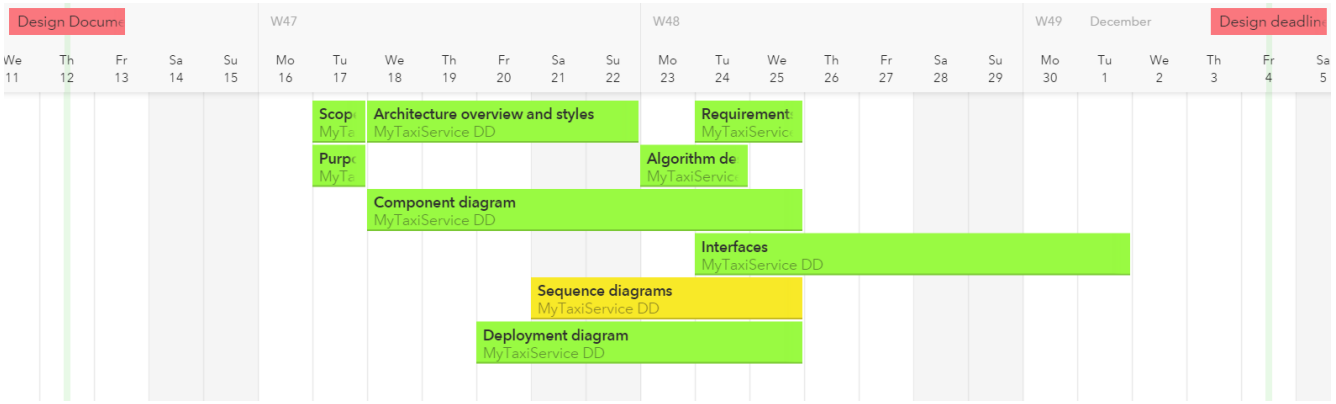
Deadline: 4 December 2015

Tasks list:

- Scope
- Purpose
- Architecture Overview and Styles
- Requirements traceability
- Algorithm description
- Component Diagram
- Interfaces
- Sequence Diagrams
- Deployment Diagram

Precedence:

- The *Interfaces* task must begin after the beginning of the *Component Diagram and Deployment Diagram* tasks.



3.3 Code Inspection

Assignment date: 9 December 2015

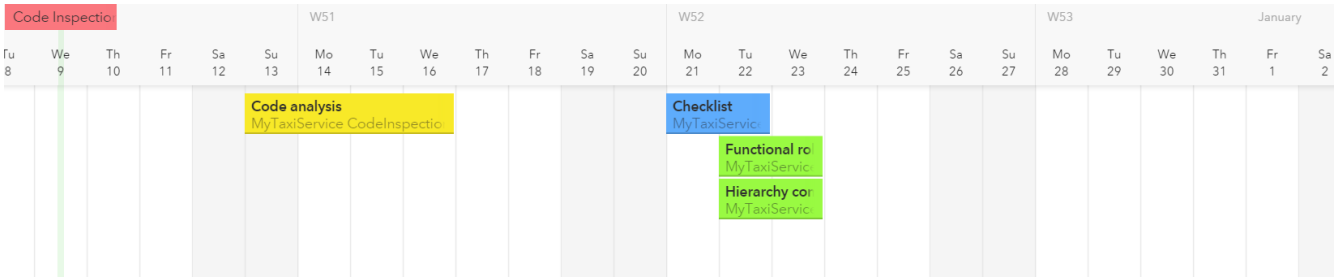
Deadline: 6 January 2016

Tasks List:

- Code analysis
- Cheklist
- Functional role
- Hierarchy comments

Precedence:

- The *Checklist, Functional Roles and Hierarchy Comments* tasks must begin after the completion of the *Code Analysis* tasks.



3.4 Integration Test Plan

Assignment date: 7 January 2016

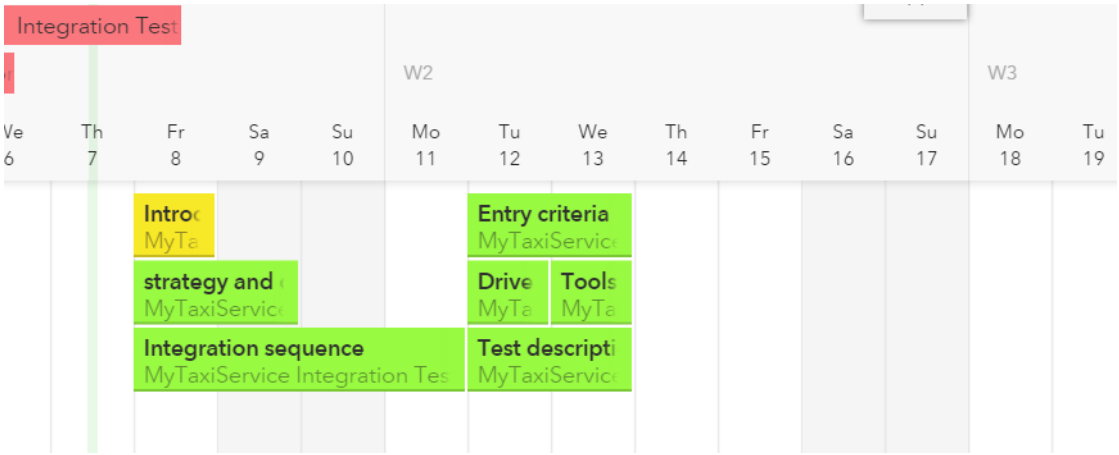
Deadline: 21 January 2016

Tasks list:

- Introduction
- Entry criteria
- Strategy and diagrams
- Drivers and Test Data
- Tools used
- Integration Sequence
- Test description

Precedence:

- The *Test Description* and *Driver and Test Data* tasks must begin after the completion of the *Strategy and Diagrams* and *Integration Sequence* tasks.



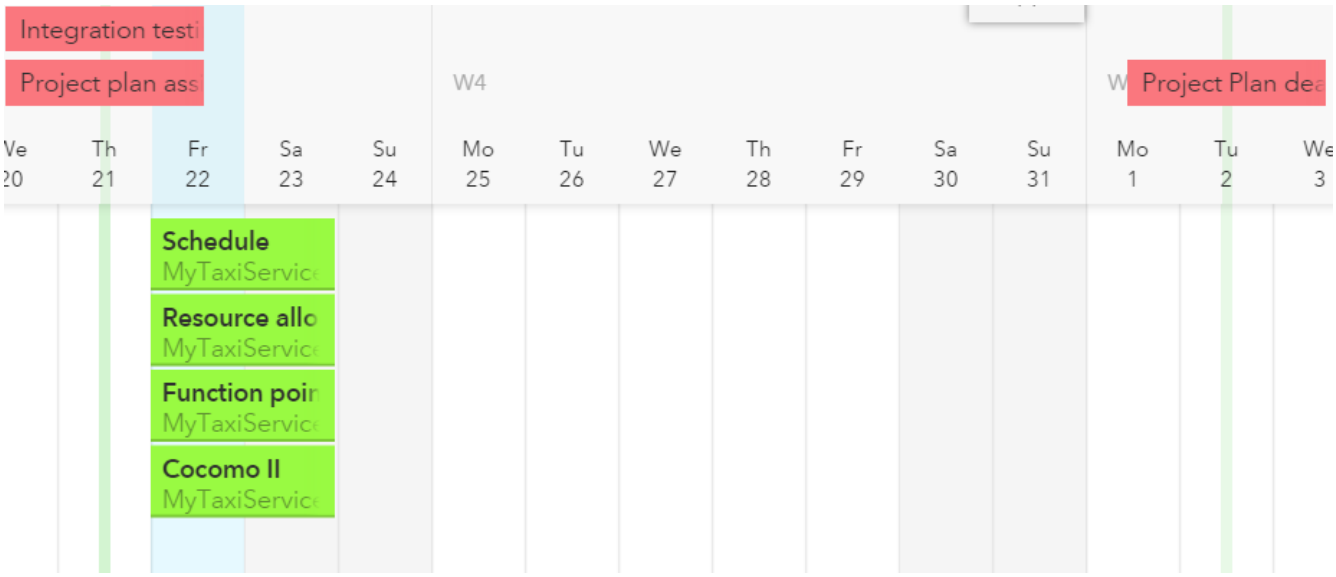
3.5 Project Plan

Assignment date: 22 January 2016

Deadline: 2 February 2016

Tasks list:

- Schedule
- Resource allocation
- Function points
- COCOMO II

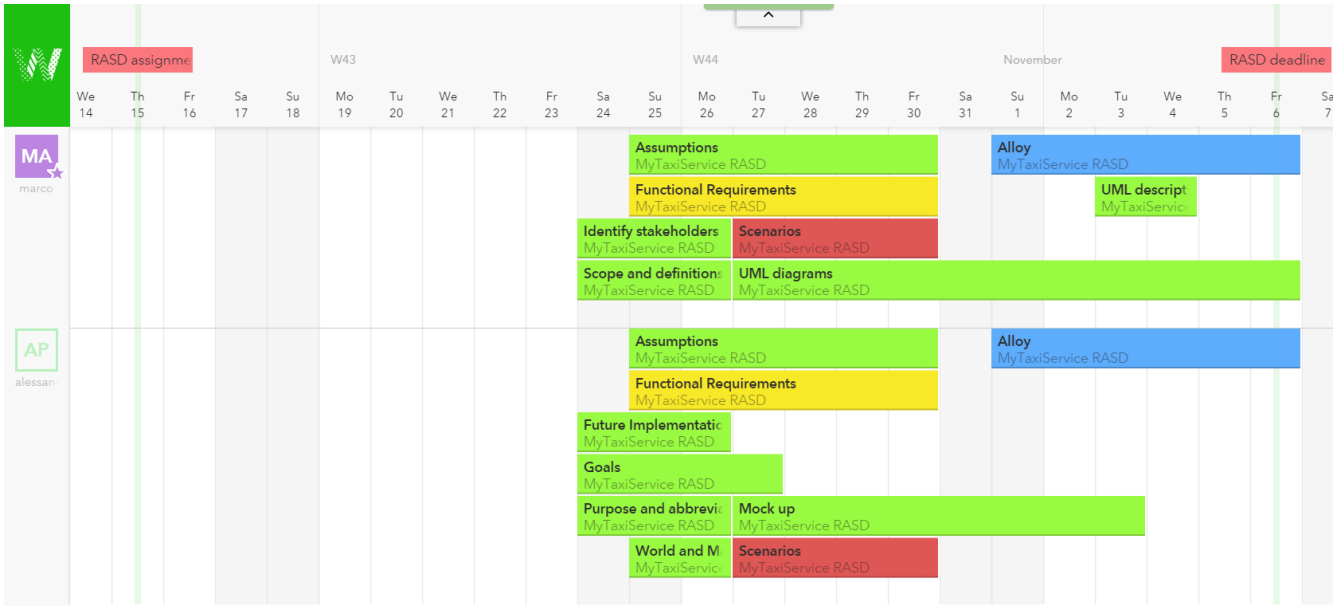


4. Resources allocation

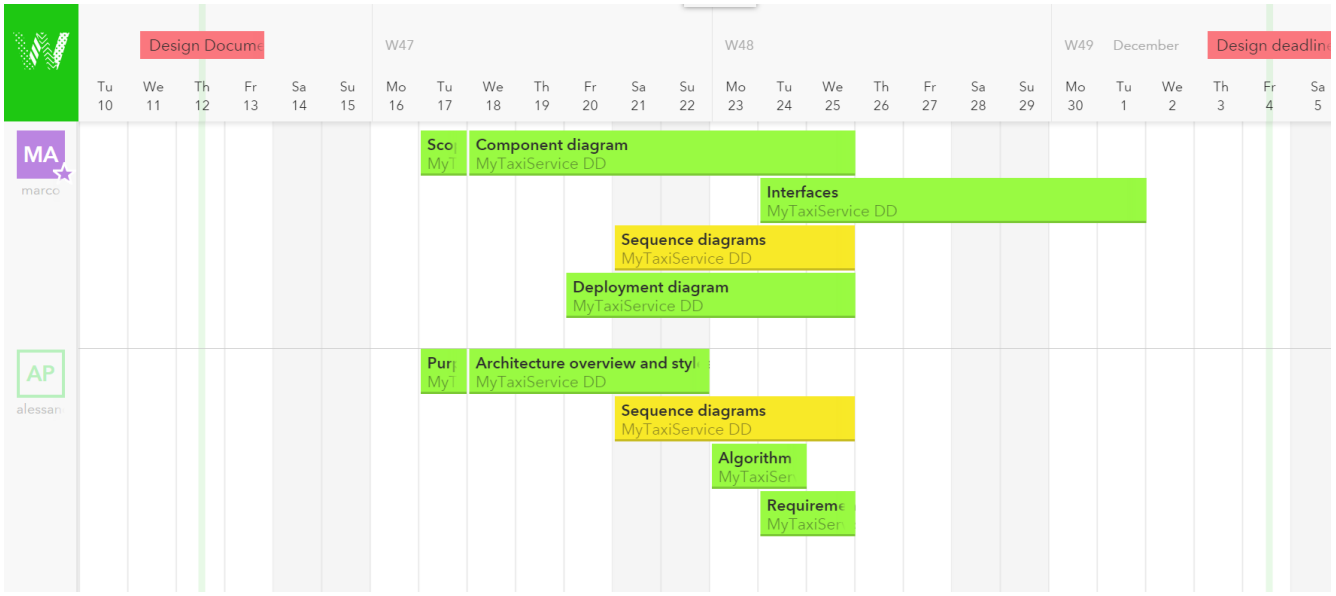
Abbreviations:

- **MA**: Marco Romani
- **AP**: Alessandro Pozzi

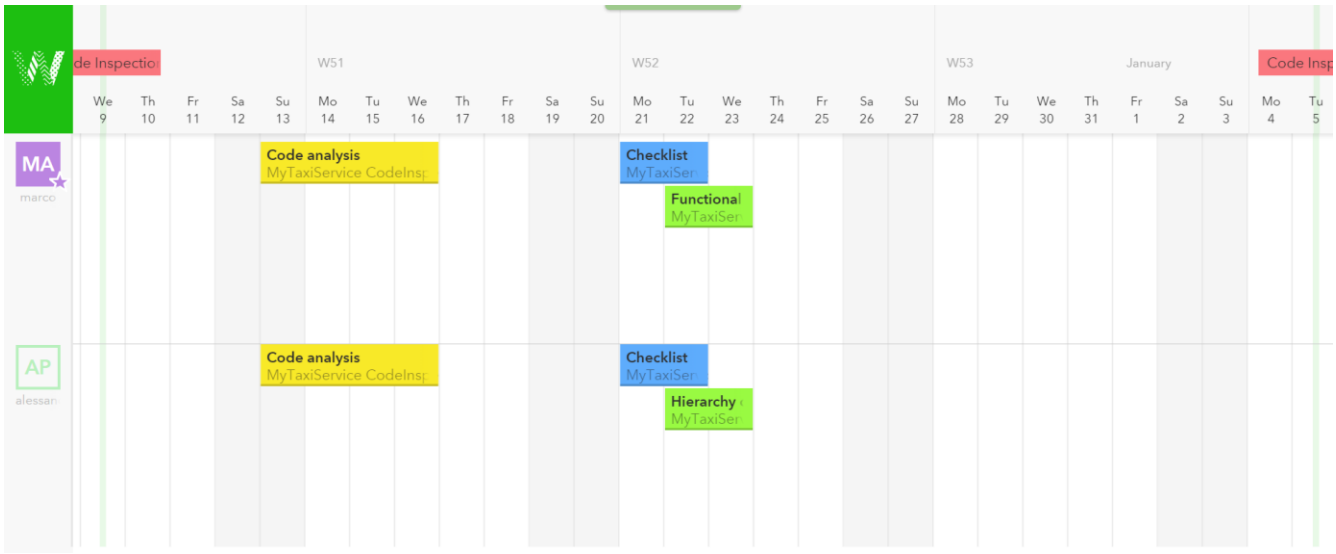
4.1 RASD



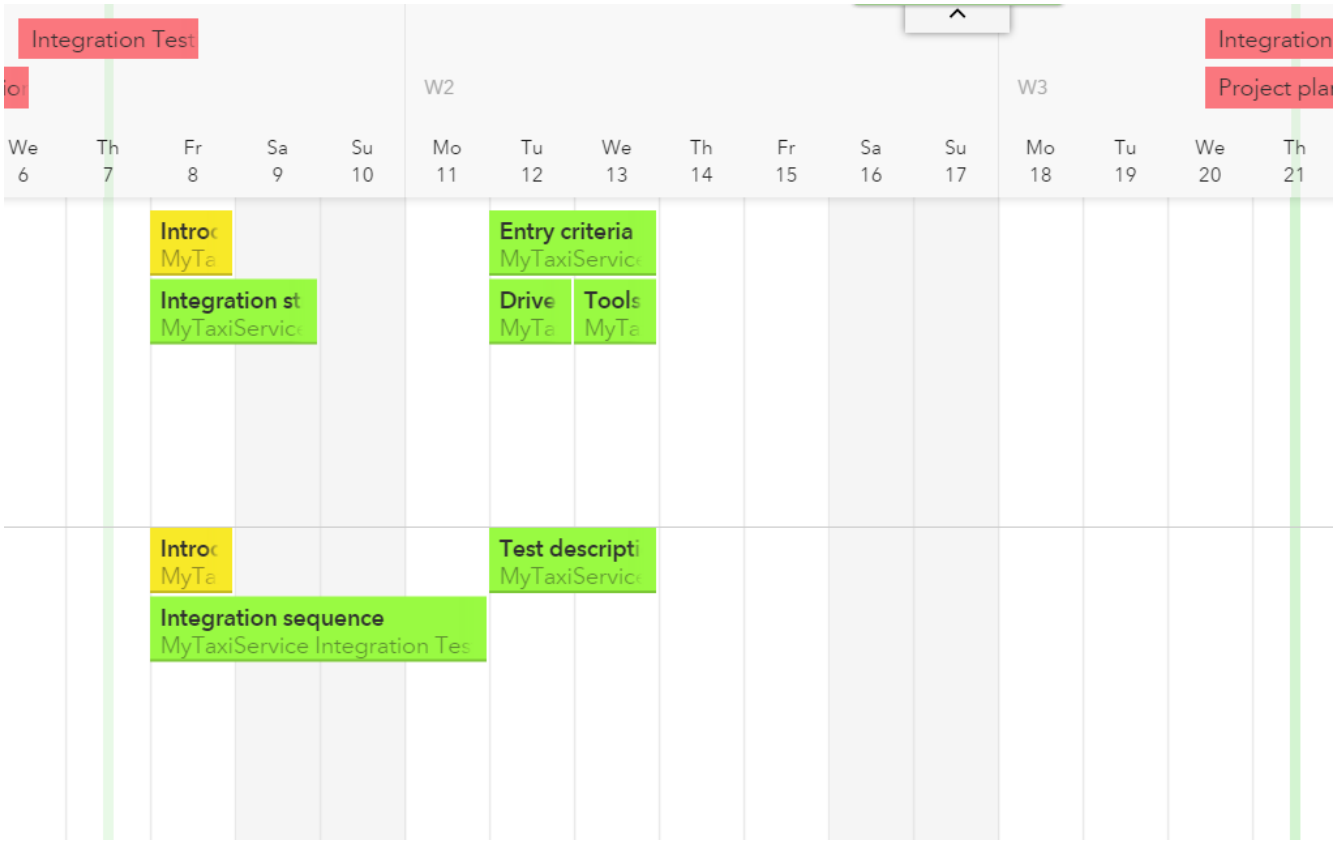
4.2 Design Document



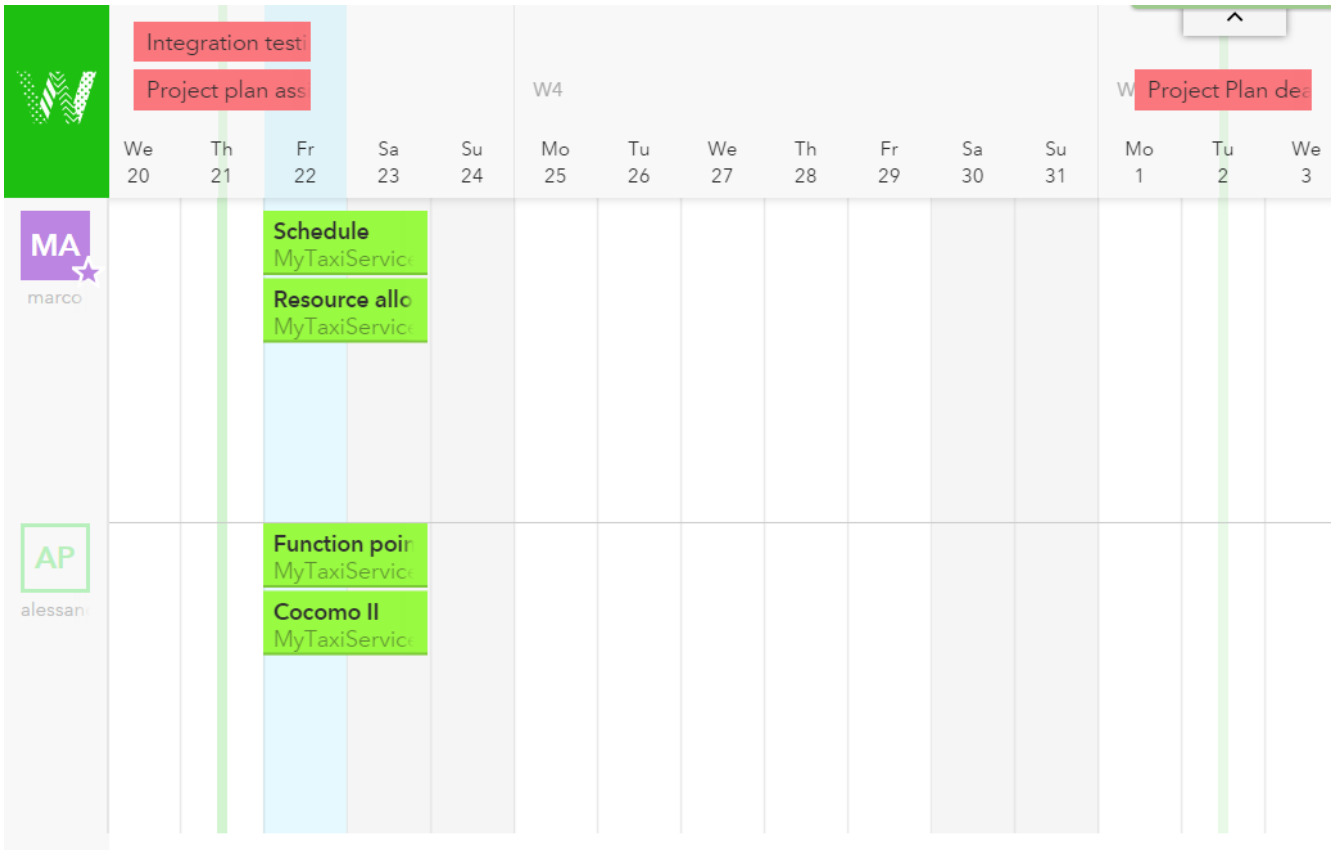
4.3 Code Inspection



4.4 Integration Test Plan



4.5 Project Plan



5. Risks of the project

The risks of the project, related to the writing of the documents and the general planning of the MyTaxiService application, can be considered the followings:

1) Misunderstanding between groups member and, consequently, incoherence in the description of the system's functionalities
2) Technical problems (e.g. applications crashing)
3) External tasks overlaps, shortage of time

The first problem is, of course, very well know and feared in any environment that expects a group of people working together. In this case, since the actual development of the project wasn't required, inconsistencies in the application's description can be solved with a modification of the involved documents. However, the cost of such modification increases with the number of the document written, so that, for example, an incoherence found in the Integration Test Plan Document implies dozens of updates in the RASD, Alloy and in the Design Document. The misunderstandings can arise because of various reasons: a too high-level view of the system itself, a poor description of the requirements and so on.

The second problem is mainly related to the instability or superficiality of the free software. Since this project is not founded, errors and limitations of such applications have actually caused some inconveniences.

The third problem is quite common in environments such universities, when the team members have to deal multiple projects and courses.