myTaxiService

Software Engineering 2 -Project

Alberto Maria Metelli   Riccardo Mologni

Politecnico di Milano
M. Sc. in Computer Science and Engineering

Final Presentation, 22nd February 2016

Outline

1. Requirement analysis and specification
    - Actors, goals and requirements
    - Use cases
2. Design
    - Pattern + Style = Architecture
    - Architectural views
    - Algorithm design
3. Integration testing plan
    - Integration testing strategy
    - Example of integration testing procedure
4. Project plan
    - Cost estimation
    - Project planning and risk management

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Actors

- Passenger
  - Registered passenger
  - Unregistered passenger
- Taxi driver
- Call center operator

Goals

Jackson Zave approach - 1

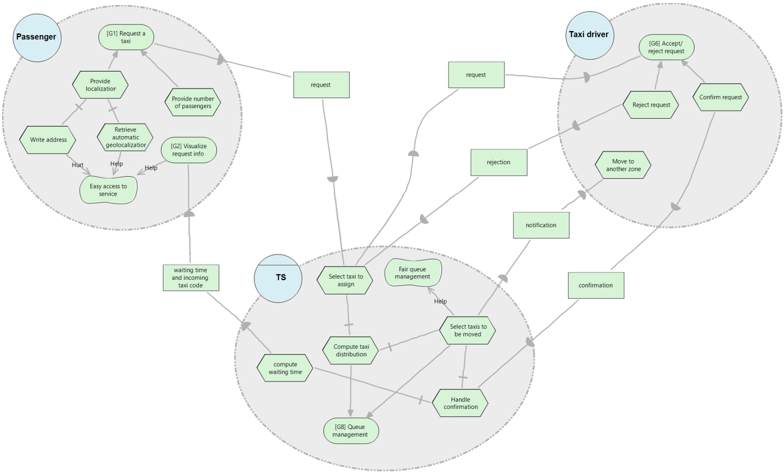
- [G1] Allow a passenger to request a taxi for his/her current position without registration.
- [G2] Allow the passenger to visualize the waiting time and the code of the incoming taxi for confirmed requests.
- [G3] Allow a registered passenger to have a personal area.
- [G4] Allow a registered passenger to reserve a taxi.

Goals

Jackson Zave approach - 2

- [G5] Allow a registered passenger to cancel or modify a previous reservation.
- [G6] Allow a taxi driver to either accept or reject a request coming from the system.
- [G7] Allow a taxi driver to inform the system about his/her availability.
- [G8] Ensure that available taxi queues enjoy “fair properties”.

i* model


Passenger
[G1] Request a taxi
Provide localization
Provide number of passengers
Write address
Retrieve automatic geolocalizatior
[G2] Visualize request info
Hurt
Help
Help
Easy access to service
request
request
Taxi driver
[G6] Accept/ reject request
Reject request
Confirm request
rejection
Move to another zone
notification
waiting time and incoming taxi code
TS
Select taxi to assign
Fair queue management
Help
Select taxis to be moved
Compute taxi distribution
compute waiting time
Handle confirmation
[G8] Queue management
confirmation

An example of functional requirement

Reservation

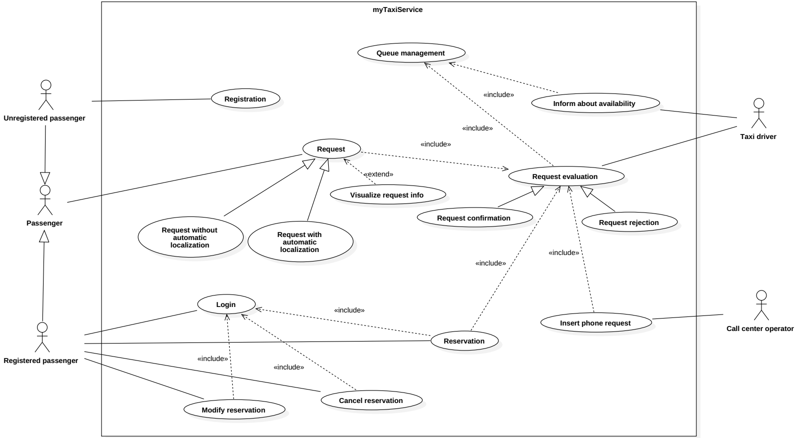
- **[G4] Allow a registered passenger to reserve a taxi.**
- [R4.1] TS shall provide the registered passenger with a form in which he/she has to insert the total number of passengers, the origin and the destination of the ride, the date and time of the meeting.
- [R4.2] TS shall accept only reservations made at least two hours in advance.
- [R4.3] TS shall store the reservation, allocate a request 10 minutes before the meeting time.

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Use cases

UML Use Case diagram



Reservation

Use case description - 1

Name	Reservation
Related goals	[G4]
Actors	Registered passenger
Entry condition	Passenger is logged in.
Flow of events	1. Passenger accesses to the reservation area. 2. Passenger inserts the required data (origin, destination, date, time, number of passengers). 3. Passenger confirms the reservation. 4. TS system whether data are valid. 5. QMA creates a new reservation and the related request is allocated.

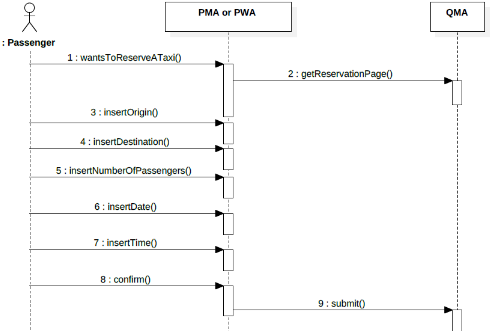
Reservation

Use case description - 2

Exit condition	The reservation is added to the TS system.
Exceptions	• If passenger does not confirm the operation is not performed. • If the data are not valid (origin, destination, number of passengers) an error message is shown to passenger and the operation is not performed. Passenger can repeat the process. • If the date and time are such that the reservation is not made at least two hour in advance an error message is shown to user and the operation is not performed. Passenger can repeat the process.

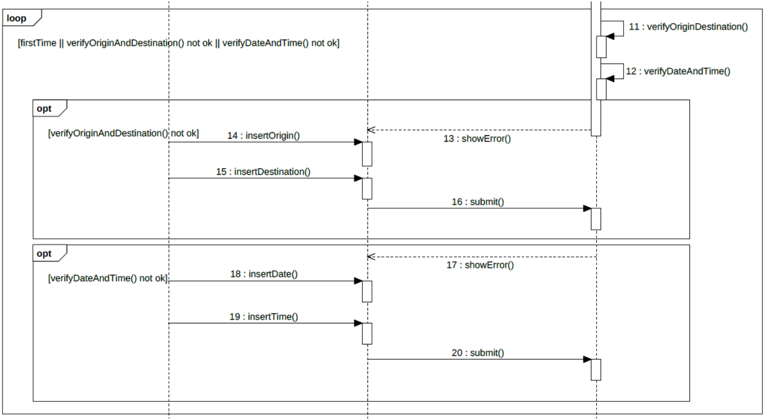
Reservation

Sequence diagram - 1



Reservation

Sequence diagram - 2


loop
[firstTime || verifyOriginAndDestination() not ok || verifyDateAndTime() not ok]
11 : verifyOriginDestination()
12 : verifyDateAndTime()
opt
[verifyOriginAndDestination() not ok]
13 : showError()
14 : insertOrigin()
15 : insertDestination()
16 : submit()
opt
[verifyDateAndTime() not ok]
17 : showError()
18 : insertDate()
19 : insertTime()
20 : submit()

Outline

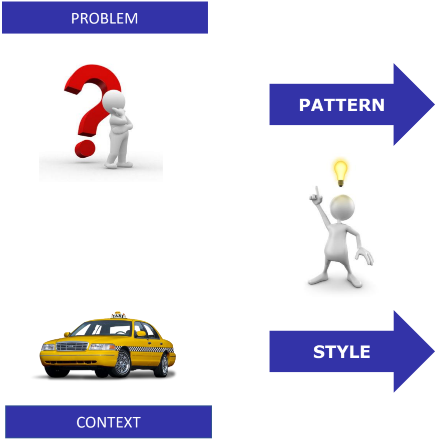
1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Pattern + Style = Architecture

PROBLEM

CONTEXT

Pattern + Style = Architecture



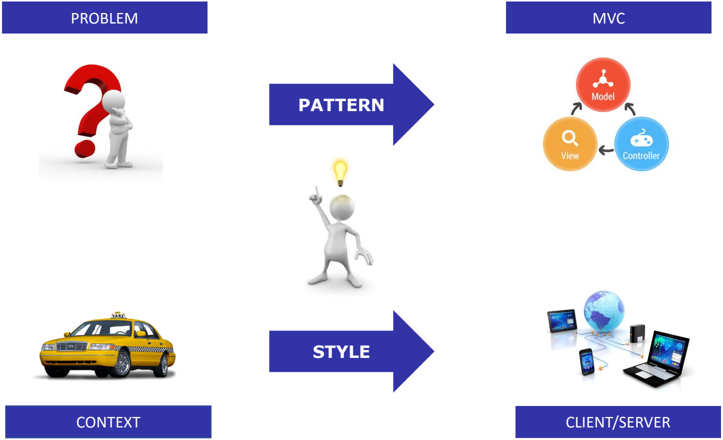
PROBLEM

PATTERN

STYLE

CONTEXT

Pattern + Style = Architecture



PROBLEM

PATTERN

MVC

CONTEXT

STYLE

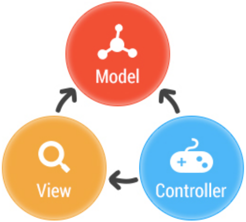
CLIENT/SERVER

MVC

Architectural pattern

Problem: applications with user interface

- Separation of concerns
- Design and conquer development
- Maintainability



Client/Server

Architectural style

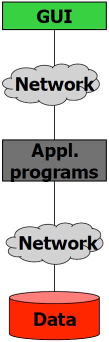
Context: distributed application

- Separation of roles
- Lot of source of information
- Just one elaboration point
- Maintainability

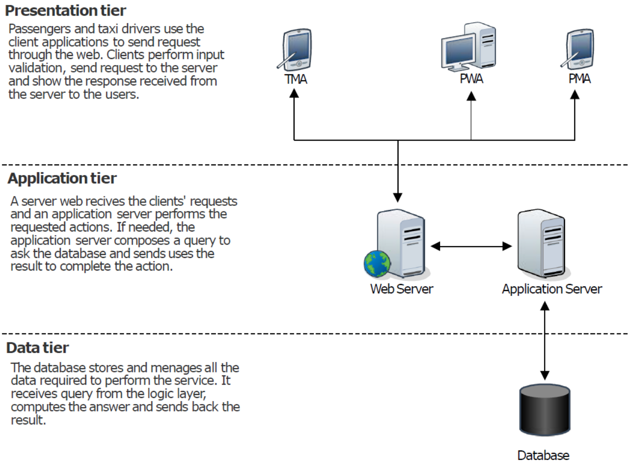
Three tier

Architectural style flavour

- Different computational and storage resource
- Presentation on clients (Tier 1)
- Separation between
  - data (Tier 3) and
  - business logic (Tier 2)
- Further separation within Tier 2:
  - visualization (web server)
  - processing (application server)



Overall architecture



Presentation tier

Passengers and taxi drivers use the client applications to send request through the web. Clients perform input validation, send request to the server and show the response received from the server to the users.

Application tier

A server web recives the clients' requests and an application server performs the requested actions. If needed, the application server composes a query to ask the database and sends uses the result to complete the action.

Data tier

The database stores and menages all the data required to perform the service. It receives query from the logic layer, computes the answer and sends back the result.

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Subsystems

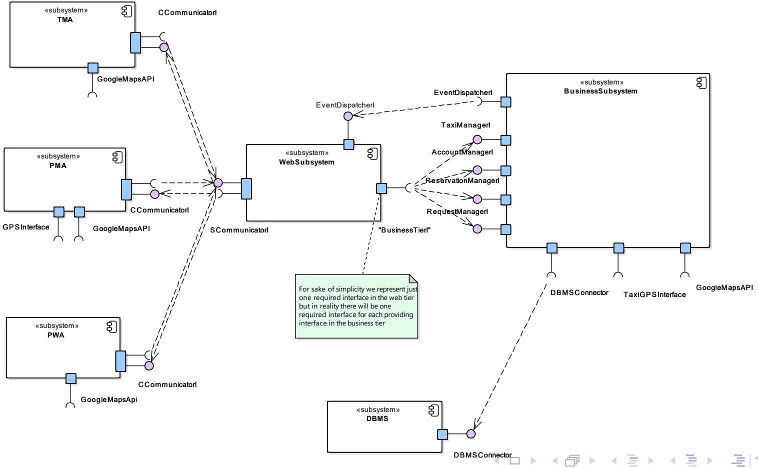
- TMA (Taxi Mobile Application)
- PMA (Passenger Mobile Application)
- PWA (Passenger Web Application)

- Web subsystem
- Business subsystem
- DBMS

Subsystems

High level component diagram

Link



For sake of simplicity we represent just one required interface in the web tier but in reality there will be one required interface for each providing interface in the business tier

Components

Business Subsystem

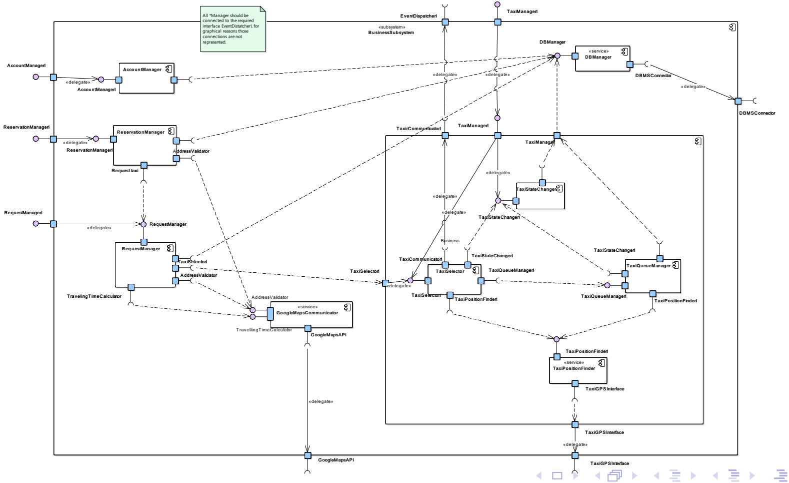
- AccountManager
- RequestManager
- ReservationManager
- GoogleMapsCommunicatior
- DBManager

- TaxiManager (*macro-component*)
  - TaxiSelector
  - TaxiPositionFineder
  - TaxiQueueManager
  - TaxiStateChanger

Components

Business Tier - Component diagram

Link



All *Manager should be connected to the required interface EventDistatcherI, for graphical reasons those connections are not represented.

Cloud deployment

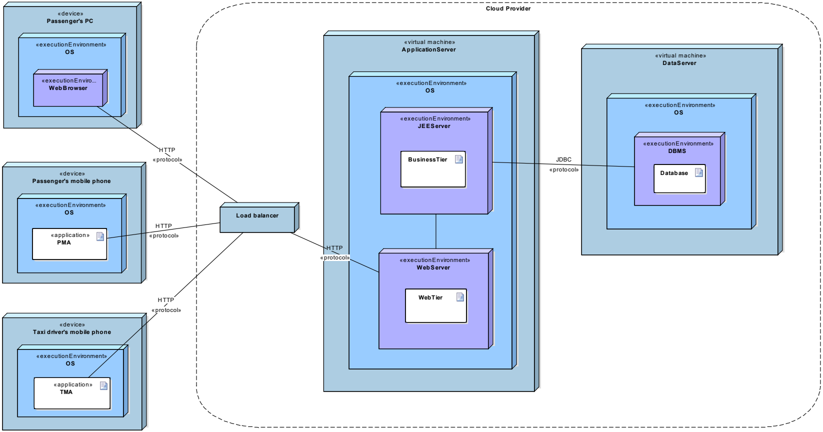
Motivations

IaaS deployment

- Different traffic loads: upward and downward *scalability*
- Costs: no acquisition costs, *pay-as-you-go*
- Security: physical security
- Availability: redundancy in hw and configurations, automatic backup

Cloud deployment

Deployment diagram

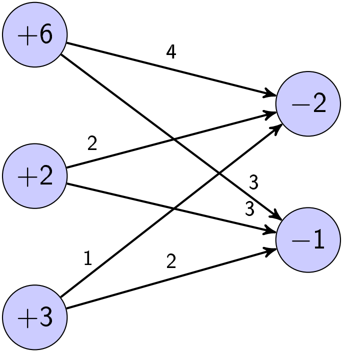


Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Taxi queue management

Example



Taxi queue management

Algorithm

1. Find a maximum flow
2. While there exists a negative cost cycle
    1. Build the incremental network
    2. Find the negative cost cycle
    3. Update the flow

The complexity is $O(|Z||A'|^2 k_{max} d_{max})$

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Levels of integration

- *component level*: each component will be integrated and tested against every dependent component in the contest of the subsystem to which it belongs
- *subsystems level*: once each subsystem is entirely integrated, all of them will be integrated and tested.

Strategies of integration

- component level: **Bottom-up**
- subsystems level: **Sandwich**

The overall methogology is known as *modified sandwich* strategy

Component level strategy

Bottom up

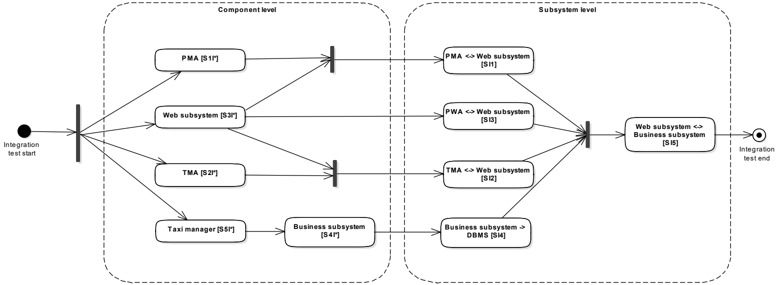
- only test drivers are used, no stubs
- suitable for object oriented design metodologies
- favours the evaluation of the performance requirements

Subsystem level strategy

Sandwich

- top and bottom layer tests done in parallel
- suitable for large projects having several subprojects

Sequence of integration



Component level

Subsystem level

Integration test start

PMA [S1I*]

Web subsystem [S3I*]

TMA [S2I*]

Taxi manager [S5I*]

Business subsystem [S4I*]

PMA <-> Web subsystem [SI1]

PWA <-> Web subsystem [SI3]

TMA <-> Web subsystem [SI2]

Business subsystem -> DBMS [SI4]

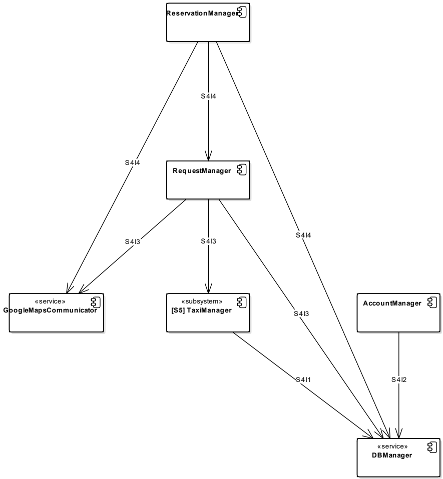
Web subsystem <-> Business subsystem [SI5]

Integration test end

Outline

1. Requirement analysis and specification
    - Actors, goals and requirements
    - Use cases
2. Design
    - Pattern + Style = Architecture
    - Architectural views
    - Algorithm design
3. Integration testing plan
    - Integration testing strategy
    - Example of integration testing procedure
4. Project plan
    - Cost estimation
    - Project planning and risk management

Business subsystem - 1



S4I3 RequestManager → TaxiManager

Test case identifier	**S4I3-T3**
Test items	RequestManager TaxiManager
Input specification	Create typical RequestManager input
Output specification	Check if the correct methods are called in the TaxiManager
Environmental needs	I1 succeeded
Tested funcional requirements	1. Check that when sendRequest is called on RequestManager, selectTaxi is called on TaxiManager.
Tested non funcional requirements	-
Testing technique	Automated

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

Function points count

Function type	FPs
ILF (Internal Logical Files)	52
ELF (External Logical Files)	25
EI (External Inputs)	48
EO (External Outputs)	9
EQ (External Inquiries)	12

$$UFP = ILF + ELF + EI + EO + EQ = 146 FPs$$

$$SLOC = SLOC/FP \cdot UFP = 46 \cdot 146 = 6716$$

COCOMO II

$$Effort = PM = 2.94 \cdot (6.716)^{1.0536} \cdot 1.149 = 25.1 \text{ person} - \text{month}$$

$$Duration = TDEV = 3.67 \cdot (25.1)^{0.30872} = 13.8 \text{ months}$$

$$Cost = 25.1 \cdot 2500\$ = 62832 \ \$$$

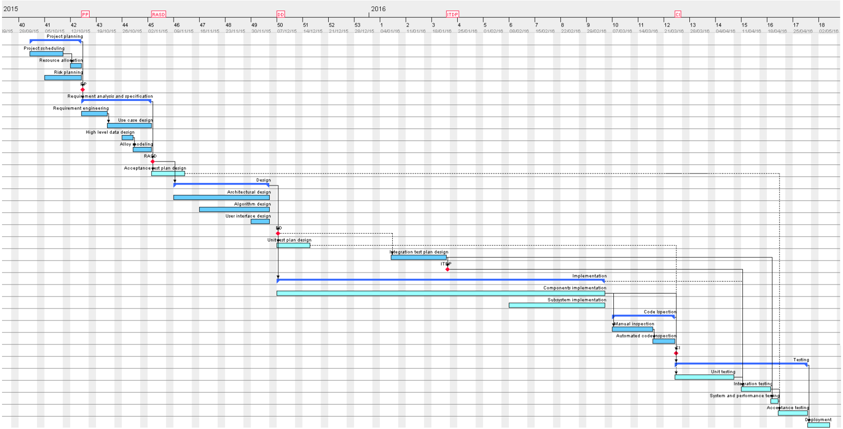
$$N = \frac{25.1}{13.8} = 1.82 \text{ person}$$

Outline

1. Requirement analysis and specification
   - Actors, goals and requirements
   - Use cases
2. Design
   - Pattern + Style = Architecture
   - Architectural views
   - Algorithm design
3. Integration testing plan
   - Integration testing strategy
   - Example of integration testing procedure
4. Project plan
   - Cost estimation
   - Project planning and risk management

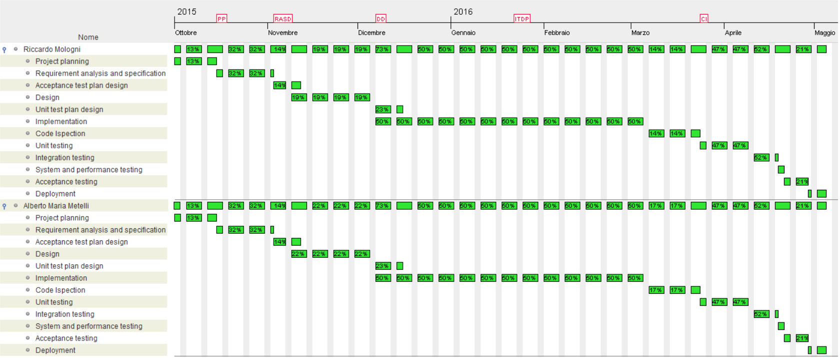
Tasks

Link


2015
2016
PP
RASD
DD
ITDP
CI
40 28/09/15
41 05/10/15
42 12/10/15
43 19/10/15
44 26/10/15
45 02/11/15
46 09/11/15
47 16/11/15
48 23/11/15
49 30/11/15
50 07/12/15
51 14/12/15
52 21/12/15
53 28/12/15
1 04/01/16
2 11/01/16
3 18/01/16
4 25/01/16
5 01/02/16
6 08/02/16
7 15/02/16
8 22/02/16
9 29/02/16
10 07/03/16
11 14/03/16
12 21/03/16
13 28/03/16
14 04/04/16
15 11/04/16
16 18/04/16
17 25/04/16
18 02/05/16
Project planning
Project scheduling
Resource allocation
Risk planning
PP
Requirement analysis and specification
Requirement engineering
Use case design
High level data design
Alloy modeling
RASD
Acceptance test plan design
Design
Architectural design
Algorithm design
User interface design
DD
Unit test plan design
Integration test plan design
ITDP
Implementation
Components implementation
Subsystem implementation
Code Ispection
Manual inspection
Automated code inspection
CI
Testing
Unit testing
Integration testing
System and performance testing
Acceptance testing
Deployment

Resources

Link



Nome

- Riccardo Mologni
  - Project planning
  - Requirement analysis and specification
  - Acceptance test plan design
  - Design
  - Unit test plan design
  - Implementation
  - Code Ispection
  - Unit testing
  - Integration testing
  - System and performance testing
  - Acceptance testing
  - Deployment
- Alberto Maria Metelli
  - Project planning
  - Requirement analysis and specification
  - Acceptance test plan design
  - Design
  - Unit test plan design
  - Implementation
  - Code Ispection
  - Unit testing
  - Integration testing
  - System and performance testing
  - Acceptance testing
  - Deployment

Some examples of risks

Name	*Description*	*Probability*	*Impact*
Requirement problem - 2	Customers changes the requirement in the late phases of the software development.	Possible	Catastrophic
Personnel problem - 1	Project manager is absent at critical times in the project.	Unlikely	Critical
Design problem - 2	The algorithm for taxi management proposed in the design phase are not correct.	Rare	Serious
Implementation problems - 2	Code does not follow quality guidelines.	Unlikely	Critical

Cumulative data

- *Total duration*: 67 days (536 h)
- *Total working hours*
  - Riccardo Mologni: 100 h
  - Alberto Maria Metelli: 108 h
- *Working hours per day*
  - Riccardo Mologni: 0,57 h/day
  - Alberto Maria Metelli: 0,61 h/day

Questions