MyTaxiService
Final Presentation

Giorgio Pea - Andrea Sessa

Summary

- Project overview
- Requirements
- Design
- Testing
- Planning
- Inspection

Project overview

What is MyTaxiService?

MyTaxiService is an application that simplifies and speeds up the process of taking a taxi in the city

↓

A taxi within a tap

Less waiting time for the users

Less traffic for the city

Requirements

Some important terms

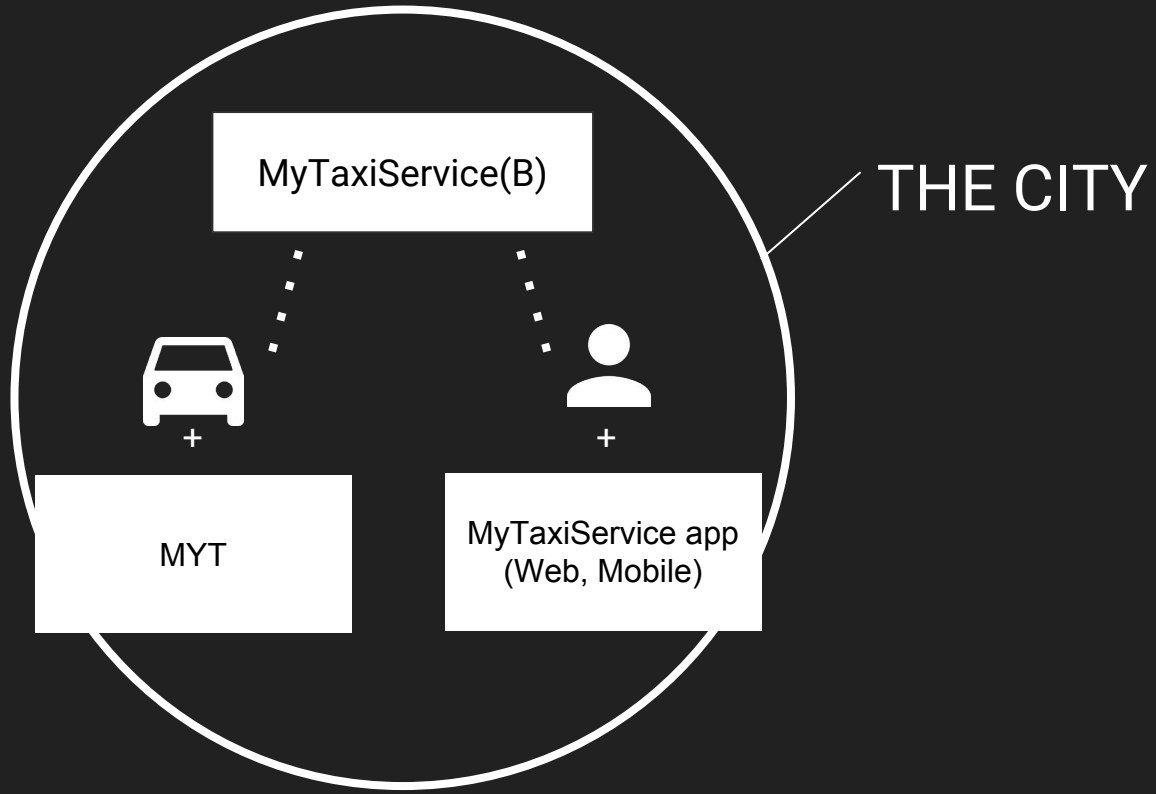
MTAXI : *a taxi that has joined MyTaxiService*

MYTAXISERVICE(B) : *the back-end of MyTaxiService*

RIDE REQUEST: *I want to be picked up by a mtaxi as soon as possible*

BOOKING REQUEST: *I want to be picked up by a mtaxi at a given date and time*

Some important Assumptions



Actors and what they can do

USER

Register as a user

REGISTERED USER

Login from the web or mobile app

TAXI DRIVER

Register as a driver

ADMINISTRATOR

- See accident/bad behavior reports
- suspend/restore mtaxies

Actors and what they can do - 1

LOGGED REGISTERED USER

- Request a ride
- Request a booking
- See the status of ride/booking requests
- Undo a booking request
- Logout

MTAXI DRIVER

- Accept a ride/booking request
- Complete a ride
- View the details of a request
- View zone change orders
- Report an accident

Functional requirements

Ride request specifying:

- Starting location
- Ending location
- Number of passenger

Booking request specifying:

- Starting location
- Ending location
- Number of passenger
- Date and time

Can be undone at most within 10 minutes from the indicated date/time

Functional requirements - 1

MyTaxiService must be aware of the distribution of mtaxies in the city's zones

MyTaxiService can change the distribution of mtaxies with Zone Change Orders

More balanced distribution of mtaxies, reduced waiting times, less traffic

Functional requirements - 2

For each city’s zone MyTaxiService has to manage a queue of available taxi

MyTaxiService has to be aware of the city’s traffic, zone per zone

MyTaxiService checks the traffic condition of a mtaxi’s zone before forward to that mtaxi any request

Important functional requirements - 3

MyTaxiService must use MYT devices to on mtaxies to:

Compute an aprox. waiting time for a passenger

Check the behavior of the mtaxi driver

Keep the organization of taxies in queues

Functional requirements - 4

A mtaxi can report an accident

MyTaxiService is aware of the work timetable of each mtaxi

Domain properties

- Traffic information is reliable
- GPS data is reliable
- Users are always present at the pick up locations

Constraints

- MyTaxiService requires Android 2.0 (or later) or IOS 6.0 (or later)
- MyTaxiService requires a browser that supports Javascript, CSS, HTML
- MyTaxiService requires an internet connection

Non functional requirements*

Availability : 24h/7d

Downtime(MTTR) : 1h per year

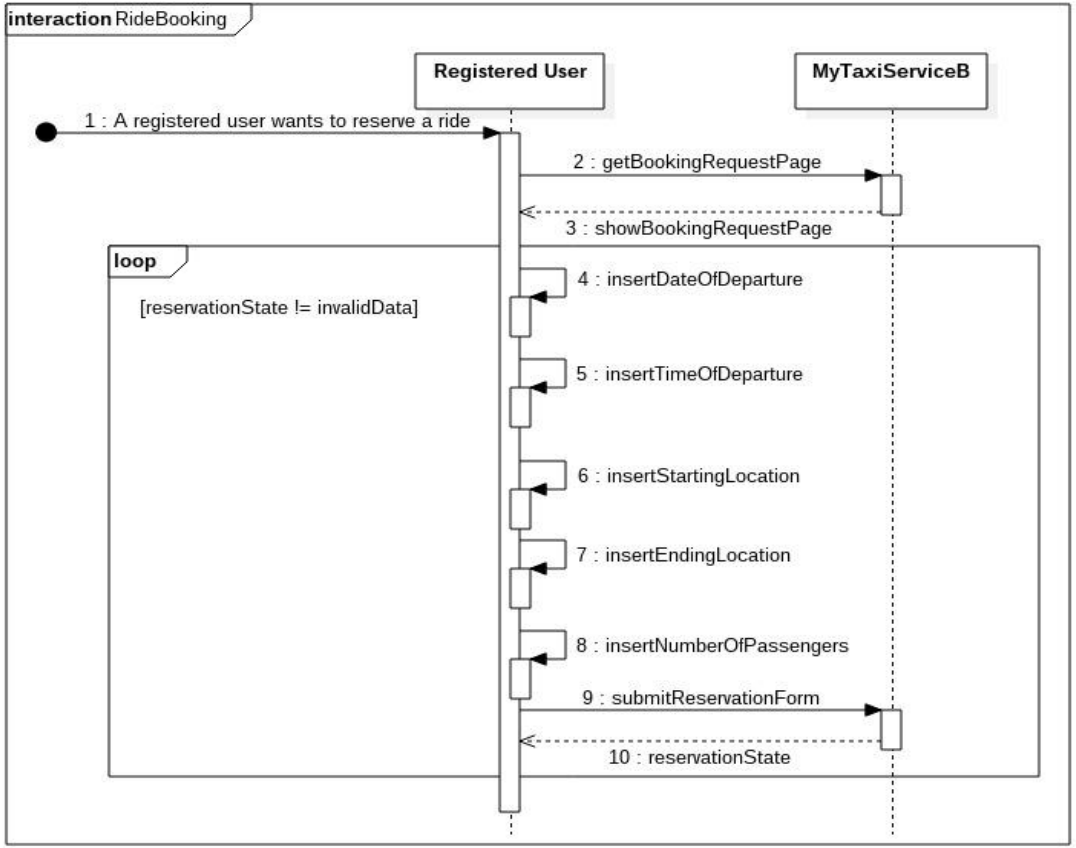
Maximum number of ride/booking request: 1000/h

Maximum training time for the use of the app: 10 min

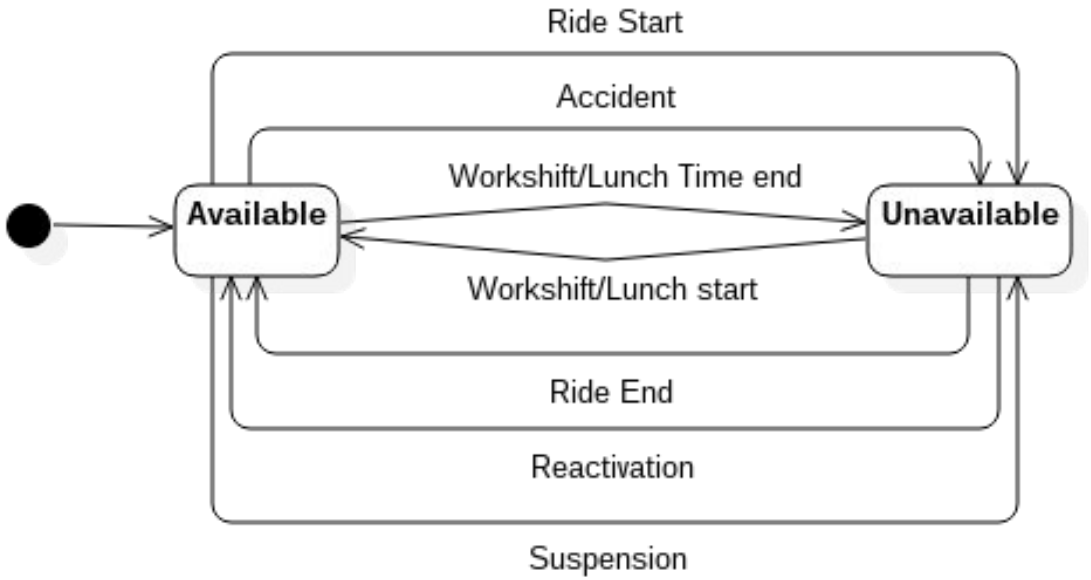
Maximum clicks to reach of all functionalities: 3 clicks

*Very ideal

SD - Booking request



State Diagram - Mtaxi



Design

Design Document - Recap

- Already presented in November
- Architectural Style: three tier Client-Server with polling
- Major Updates
  - The algorithm to redistribute mtaxies into zones has been improved
  - Corrected a small graphical error in the architecture diagram

Integration Test Plan

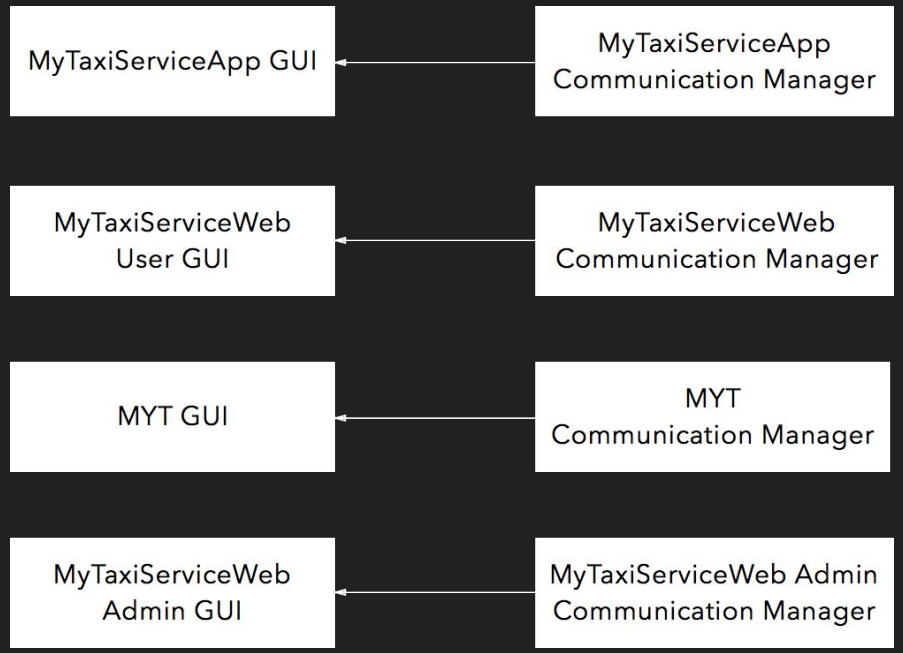
Integration Strategy

→ Grouped Bottom-Up Integration Strategy

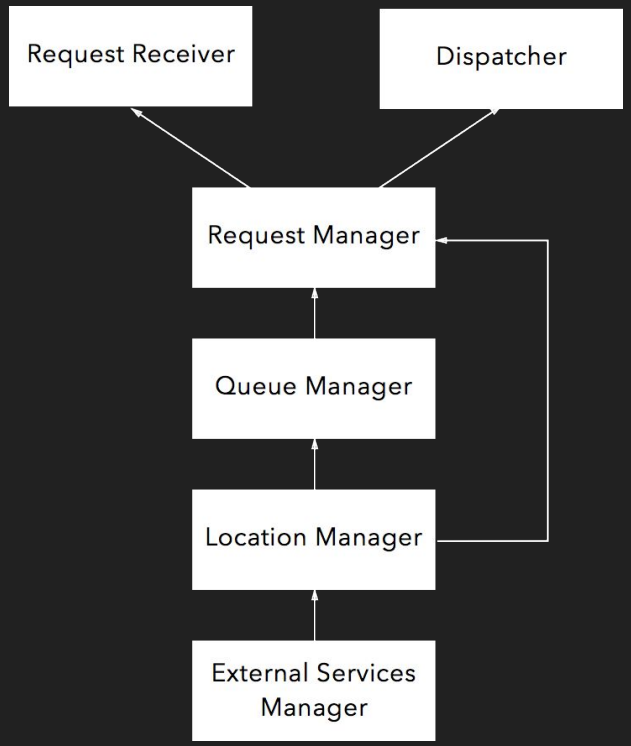
MOTIVATIONS:

- The strategy adapts to the hierachical structure of the software components
- No need to use a more complex integration strategy for this kind of project

View Subsystem integration plan



Controller Subsytem Integration Plan



Intersubsystems integration plan

Project Planning

Size and Complexity estimation of the project

↓

Function Points

Function Points - ILF

User:	*SIMPLE*	Zone:	*SIMPLE*
Administrator:	*SIMPLE*	Location:	*SIMPLE*
Mtaxi driver:	*SIMPLE*	Ride Request:	*SIMPLE*
Mtaxi:	*SIMPLE*	Booking Request:	*SIMPLE*
Work Time Table:	*SIMPLE*	Queue:	*SIMPLE*

$$Total = numberOf\ ILF * 7 = 7 * 7 = 49$$

Function Points - ELF

External Traffic Data: *MEDIUM*

$$Total = numberOf\ ELF * 7 = 1 * 7 = 7$$

Function Points - EO

AWT Notification: *COMPLEX*

Zone Change Notification: *COMPLEX*

$$Total = numberOf\ EO * 7 = 2 * 7 = 14$$

Function Points - EI

Ride Request Creation: *SIMPLE*
Booking Request Creation:*SIMPLE*
Booking Request Editing: *SIMPLE*
User Login/Logout: *SIMPLE*

User Registration: *SIMPLE*
Mtaxi driver Registration: *SIMPLE*
Driver Notification: *MEDIUM*
Administrator Oprations: *SIMPLE*

$$Total = numberOfSimpleEI * 3 + numberOf\ MediumEI * 4 =$$
$$7 * 3 + 1 * 4 = 25$$

Function Points - EIQ

User Profile Visualization: *SIMPLE*

User Ride Request Visualization: *SIMPLE*

User Booking Request Visualization: *SIMPLE*

Mtaxi Notification Visualization: *SIMPLE*

Mtaxi Accident Reports Visualization: *SIMPLE*

Mtaxi Bad Behavior Reports Visualization: *SIMPLE*

$$Total = numberOf\ EIQ * 3 = 6 * 3 = 18$$

UFP and SLOCS

UFP = 49 + 7 + 25 + 14 + 18 = 113

Supposed programming langauge:

Conversion Factor(SLOC/UFP): 46

SLOCS = 46 x 113 =

5198

Cost and Time estimation for the project

↓

COCOMO II

Cocomo II - Summary

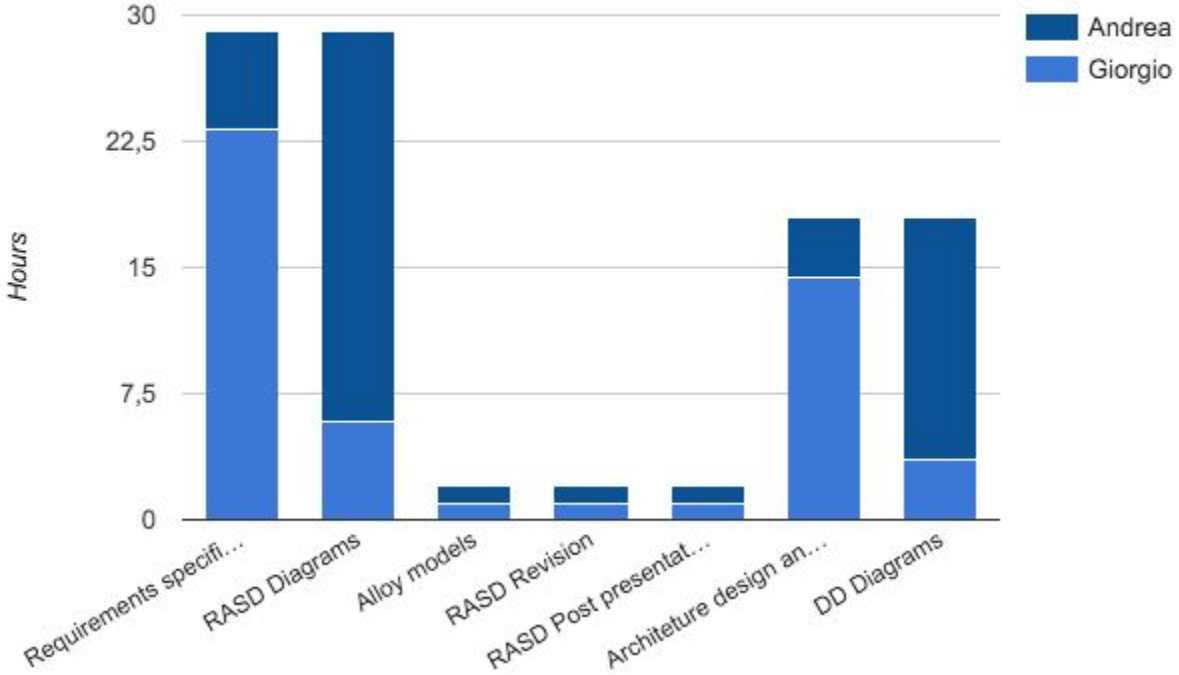
Given SLOCS : *5198*

Effort = 22.3 Person/Months

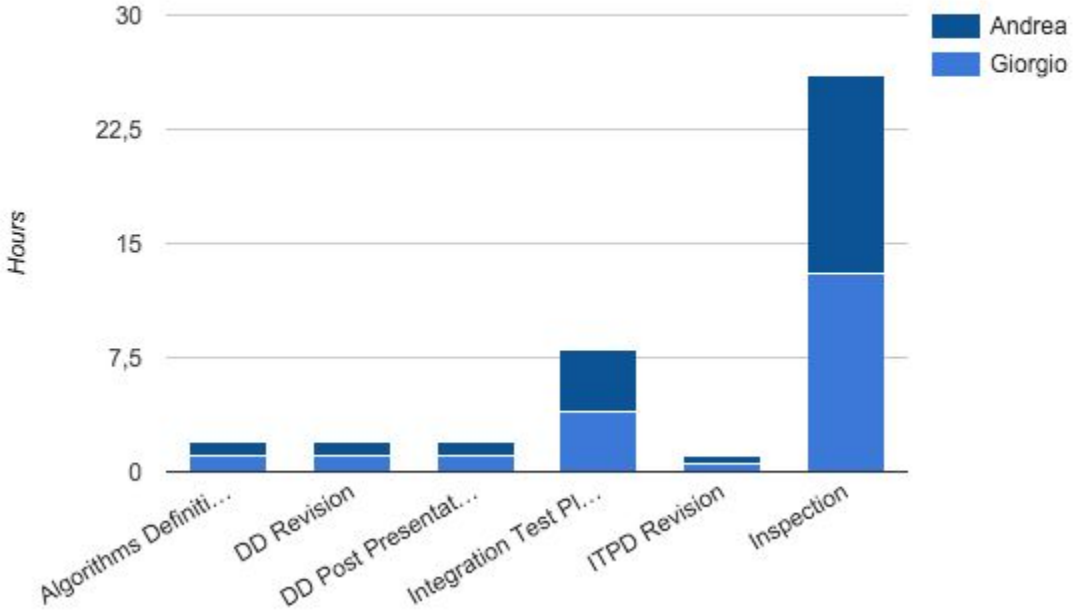
Duration = 10.2 months

Cost = 44576$ (2000 $/person)

Tasks


30
22,5
15
7,5
0
Hours
Andrea
Giorgio
Requirements specifi...
RASD Diagrams
Alloy models
RASD Revision
RASD Post presentat...
Architeture design an...
DD Diagrams

Tasks - 1


30
22,5
15
7,5
0
Hours
Andrea
Giorgio
Algorithms Definiti...
DD Revision
DD Post Presentat...
Integration Test Pl...
ITPD Revision
Inspection

Risks Management

Possible risks:

- Key development figures may be ill/busy at critical times
- Important changes to the project’s requirements and design may occur
- Database or other key components in the architecture do not perform as expected

Inspection

Artifacts under analysis

APPLICATION: *Glassfish*

CLASSES: *`SSIMediator.java, SSIServerlet.java`*

METHODS: *`init, requestHandler, processSSI, substituteVariables`*

`requestHandler` method

MISSION:

→ Takes care of process HTTP get or post requests managed by the servlet, indentifies the correct resource and invokes *processSSI()*

MAJOR ISSUES:

- Lack of comments
- Improper use of the “==” operator
- Unmanaged exceptions

substituteVariables method

MISSION

→ Parses the SSI commands contained in a given HTTP request via the SSIProcessor class and writes the result to the relevant output stream

MAJOR ISSUES

- Lack of comments
- Lack of javadoc
- Unmanaged exceptions

init method

MISSION:

→ Takes care of initialize the Java EE Servlet by retrieving and setting configuration parameters.

MAJOR ISSUES:

- Lack of comments
- Lack of javadoc
- Unmanaged exceptions

processSSI - method

MISSION

→ Parses the SSI commands contained in a given HTTP request via the SSIProcessor class and writes the result to the relevant output stream

MAJOR ISSUES

- Critical lack of comments
- Lack of javadoc
- Improper use of the “==” operation
- Possible BUG

```
URLConnection resourceInfo =
resource.openConnection();

InputStream resourceInputStream =
resourceInfo.getInputStream();
```

Thank You!
Any questions?