Politecnico di Milano
A.A. 2015-2016
Software Engineering 2: "TAXInseconds"
Requirements **A**nalysis
and
Specifications **D**ocument

Roberto Clapis (841859), Erica Stella (854443)

October 31, 2015

Contents

1 Introduction

1.1 Purpose

This document represent the Requirement Analysis and Specification Document (RASD). It aims at explaining the domain of the system to be developed and the system itself in terms of functional requirements, nonfunctional requirements and constraints. It also provides several models of the system and typical use cases. It is intended for all the developers who will have to implement the system, the testers who will have to determine if the requirements have been met and the system analysts who will have to write specifications for other system that will relate to this one. It is also intended as a contractual basis thus being legally binding.

1.2 Actual System

The government of the city wants to optimize its taxi service with a completely new application. Therefore, we assume there are no previous systems to take into account.

1.3 Scope

The aim of the project TAXInseconds is to provide a new application to optimize the taxi service of the city that will be accessible via browser, mobile or public APIs.
The application will be available to the users in web and mobile forms and will have public APIs in order to expand and improve the service with additional features.
The city managed by TAXInseconds is divided in zones of 2 km^2 each and every zone has its own queue of taxis. The queues are automatically computed by the system with the information it receives from the GPS of the phones of the taxi drivers.
Taxi drivers can be available or not. Only available taxi drivers can be in a queue. When a taxi driver changes her state from not available to available the system automatically adds her to the queue of the zone she is currently in, based on the information of the GPS of her phone.
Users that are not registered can only see the estimated time of arrival of the nearest taxi with TAXInseconds.
Registered users can also request a taxi or make a reservation for a taxi. Reservations can only be made at least two hours before the ride and must be done specifying the starting location, the destination and the meeting time. Requests, instead, only need the starting location and the destination.
When a request is made, the first taxi driver of the queue of the starting location's zone is prompted to accept or reject it. If the taxi driver rejects it her state is automatically put on unavailable by the system. If a taxi driver doesn't accept or reject the request within 1 minute, it will be passed on to the next taxi

driver in the queue and the first one will be moved to the end of the queue. If there are no available taxis in the zone of the request the system will propagate the request to the closest available taxi.
When a request is accepted, the user that has made the request receives a notification from the system informing her of the code of the incoming taxi and the estimated time of arrival.
When a reservation is made, the system confirms it to the user and allocates a taxi 10 minutes before the meeting time. If a taxi for that zone is not available the closest available taxi will be notified. When a taxi driver accepts the reservation, the user receives from the system the code of the incoming taxi. If a taxi driver doesn't accept or reject the reservation within 1 minute, it will be passed on to the next taxi driver in the queue and the first one will be moved to the end of the queue.
Requests can be cancelled before they have been accepted by a taxi driver while reservation can be cancelled until 10 minutes before the meeting time.

1.4 Goals

- Provide an easy way to request a taxi.
- Provide an easy way to reserve a taxi.
- Guarantee a fair management of the taxi queues.
- Create an extensible system that allows expansion and interactions with other services.

1.5 Definition and Acronyms

1.5.1 Definitions

- *Guest:* a person that has to sign up or log in the system.
- *Secure Channel:* a communication channel to ensure privacy and autenticity for both the server and the clients
- *Logged in user:* a person that has already signed up and logged in the system.
- *Administrator:* a person authorized to modify the list of taxi drivers stored by the system.
- *Request:* a call from a registered user who needs a taxi immediately.
- *Meeting time:* the date and time in which the registered user needs the taxi in case of reservation.
- *Reservation:* a booking of a taxi at a certain meeting time.

- *State of a taxi driver:* the state the taxi driver is currently in. It can be available or not available. Taxi drivers are in a queue if and only if they're available.
- *Closest available taxi:* if there are no taxis in the zone of the request or of the reservation, the system automatically finds the closest available taxi choosing the one with the smallest estimated time of arrival from the taxi queues of the other zones.

1.5.2 Acronyms

- ETA: estimated time of arrival: the time, estimated by the system, that the closest available taxi will take to get to the starting location of the ride.
- CAT: closest available taxi (see definition in the previous period).
- API: application programming interface; it is a set of routines, protocols, and tools for building software applications on top of this one.
- MAD: maximum allowed delay; the maximum time, calculated by the system according to its information about distance and traffic, that a taxi driver has to get to the starting location of a request.

1.6 Actors

- *Guest*: guests are able to sign up, login or ask the system for an ETA.
- *Registered users*: after successfully logging in, registered users can request or reserve taxis or ask the system for an ETA.
- *Taxi drivers*: after successfully logging in, taxi drivers are able to set their current state as available or not and to accept or refuse requests.
- *Administrator*: after successfully logging in, the administrator will be the only user allowed to edit the taxi drivers list stored by the system.

1.7 References

- The document with the assignment for the project
- The IEEE Standard for SRS

1.8 Overview

This document is structured in three parts:

- Introduction: gives an high-level description of the software purposes and context.

- Overall Description: gives a general description of the application, focusing on the context of the system, going in details about domain assumptions and constraintS. The aim of this section is to provide a context to the whole project and show its integration with the real world.
- Specific Requirements: this section contains all of the software requirements to a level of detail aimed to be enough to design a system to satisfy said requirements, and testers to test that the system actually satisfies them. It also contains the detailed description of the possible interactions between the system and the world with a simulation and preview of the expected response of the system with given stimulation. (Details are given with Alloy specifications and UML diagrams)

2 Overall description

2.1 Product perspective

The TAXInseconds application will be released as a web application and as a mobile application. There are no existing systems to integrate it with.
It will provide a total of 4 main interfaces:

- For both type of users
  - Registered users
  - Guests
- For taxi drivers
- For administrators
- A non graphical interface for APIs

2.1.1 User Interfaces

2.1.2 System interfaces

2.1.2.1 Hardware Interfaces

- Owned by the users and taxi drivers:
  - Any device running Android 4.0+ or iOS 6+ (GPS capability will make more functionalities available)
  - Any computer able to run an HTML5-compatible browser
- Owned by the company:
  - The server on which the core of the application will run, and to which the applications, the web UI and the API-related clients will connect to.
  - Two machines optimized for a stateful firewall use
  - A machine with the DBMS

2.1.2.2 Software Interfaces

- Database Management System (DBMS):
  - Name: MySQL.
  - Version: 5.1.73
  - Source: https://www.mysql.com/
- HTTP/HTTPS server:
  - Name: Nginx
  - Version: 1.8.0
  - Source: http://nginx.org/
- PHP interpreter:
  - Name: PHP
  - Version: 5.3.3
  - Source: https://secure.php.net/
- Operating System:
  - Name: CentOS
  - Version: 7.1–1503
  - Source: https://www.centos.org/

2.1.2.3 Communication Interfaces

Protocol	Application layer Protocol	Port	Scope
TCP	HTTP	80	Upgrade to a secure connection over HTTPS
TCP	HTTPS	443	The web interface or the mobile apps
TCP	HTTPS/JSON	443	The APIs
TCP	HTTPS	443	The web interface or the mobile apps
TCP	DBMS over SSL	3306	Communication between the webserver and the DBMS

2.1.2.4 Memory constraints

- Primary memory:
  - for both taxi drivers' and clients's mobile devices at least 500MB
  - for the web application 1GB or more is suggested
  - for the server it is suggested to use a cloud service in order to resize memory according to traffic
- Secondary memory:

- mobile devices will need to have 50MB of free space on the device
- the web application requires no secondary memory
- for the server it is suggested to use a cloud service in order to resize memory according to traffic

2.1.2.5 Operations

2.2 Product functions

2.3 User characteristics

The TAXInseconds application is intended for all users who are at least 18 years old.

2.4 Assumptions and dependencies

2.4.1 Domain Assumptions

- All taxi drivers who intend to use the service will have a mobile phone with one of the supported mobile OSs
- All taxi drivers will have a phone with active GPS functionality
- The taxi drivers will grant the system the rights to handle their taxi codes
- Taxi drivers' phones will always have an internet connection while TAXInseconds is running
- Users will have access to the internet
- In order to help the client to select the current location a phone with GPS capability will be required
- Users will enter a valid email address during registration
- Users will enter valid credit card data during registration
- Users allow the app to access their credit in order to pay for the service
- The Owner of the app will have to build or rent an always-on DBMS and host for the Server-Side part of the app
- Users who have requested or reserved a taxi will always be present when the taxi arrives.
- There is always at least an available taxi to fulfill a request or a reservation in the whole city.

3 Specific requirements

3.1 Functional Requirements

On the user side:

- For non logged in users (on both the WEB and Mobile interface):
  - The system will be able to calculate and show the ETA.
  - The system will provide a registration functionality
  - The system will provide a login functionality
- For logged in users (on both the WEB and Mobile interface):
  - The system will be able to calculate and show the ETA.
  - The system will provide a functionality to request a taxi at the given starting location
  - The system will provide a functionality to reserve a taxi at the given starting location and meeting time
  - The system will provide a functionality to modify the personal data of the user
  - The system will provide a logout functionality
  - The system will provide a functionality to notify users of the code of the incoming taxi
  - The system will provide a functionality to notify users of the ETA of the incoming taxi
  - The system will provide a logout functionality
- Through the API:
  - The system will be able to calculate and show the ETA.
  - Establish a Secure Channel
  - Using a Secure Channel and valid credentials:
    - Place a request for a given starting location
    - Place a reservation for given starting location and meeting time
    - Require the system to send a push message for updates about the status of a previous request

On the taxi driver side:

- If credentials have been invalidated a “reset password” procedure will be mandatory
- The system will provide a functionality to switch the state of the taxi driver from available to not available or vice versa

- The system will notify the taxi driver of an incoming request showing the starting location
- The system will notify the taxi driver of a reservation showing the starting location and the meeting time
- The system will provide a functionality to allow taxi drivers to accept or reject reservations and requests
- Handle timeout (if reaching the client takes too much time)
- The system will provide a logout functionality

Through the API:

- Establish a Secure Channel
- Through a Secure Channel and with valid credentials:
  - Toggle the driver state
  - Send a notification when a call is made for the driver
  - Accept the answer to the call, whether the answer is Acceptance or Refusal
  - Handle disconnection
  - Handle timeout (if no taxi reach the client in time)
-

On the admin side:

- Add a new taxi driver
- Remove a taxi driver
- Invalidate a taxi driver's credentials
- Set the area of the map handled by the system
- Modify registration data
- Log out

3.2 Non functional requirements

- *Cross platform*
  - There will be a UI for mobile and one for the Web platform
  - At least Android 4.0+ and iOS 6+ will be supported for the mobile version
  - At least Edge, Chrome and Firefox will have to be supported

  - The web application will have to use HTML5

- *Availability*
  - The application must always be available
  - If a query is taking some time a spinner will be shown, the app must never freeze
  - The system must store all of its data in an always-reachable database
  - Regular backups will be made in order to reduce or prevent data loss

- *Usability*
  - The UIs have to be user friendly and with a few, clear functionalities
  - The app will have few activities, mainly the login, the register and the call form.
  - When inserting a location for a reservation the user will be helped with autocompletion for locations.

- *Security*
  - In no situation sensible data will pass through an insecure channel
  - No one should be able to impersonate the taxi drivers, the clients or the admin without proper authentication

3.3 Scenarios

3.3.1 Scenario 1

Blair The Witch had to take her magical broom to the mechanic for the annual revision but she needs to go shopping to refill her stockpile of frog's tails. Her friend Mizune has told her about TAXInseconds, so she decides to give it a try. After downloading it on her smartphone, she signs up compiling the registration form with her username, password, email and credit card data. Now she can complain about how slow car-based transports are!

3.3.2 Scenario 2

Suzuka is having a date tonight but, unfortunately, her car doesn't want to start. After several failures, she decides to use TAXInseconds. After logging in, she requests a taxi specifying her home as the starting location and the restaurant's address as the destination. The system notifies Takeshi, the first taxi driver of the queue in Suzuka's zone, of the request. Takeshi accepts the request and the system sends Suzuka the code of Takeshi's taxi and the ETA so she can finally get to her date.

3.3.3 Scenario 3

Ash Ketchum has to start his new adventure in Hoenn tomorrow but his mother is busy cleaning the house with Mr. Mime, so she can't take him to the airport to meet Prof. Oak. Ash decides to use once again TAXInseconds to reserve a taxi. After logging in, he makes a reservation for a taxi specifying the starting location as Pallet town, the destination and the meeting time. The morning after, 10 minutes before the meeting time, the system allocates Brock's taxi for the reservation and sends his code to Ash so that he knows who he'll meet to start his new adventure.

3.3.4 Scenario 4

Donald Duck is ready to start his first day as a taxi driver in Paperopoli. He jumps on his car, logs in TAXInseconds and changes his state to availables. Unfortunately he doesn't know that, in his zone, there are 15 taxis in the queue before him, so he'll have to be patient for some time.

3.3.5 Scenario 5

After quite some time, Daisy requests a taxi and it's finally Donald's turn! The system notifies him but... it's breakfast time and Donald's having a cappuccino in a very crowded bar. He doesn't hear the notification popping on his phone so, after 1 minute, the system forwards Daisy's request to the next taxi driver in the queue and changes Donald's state to not available.

3.3.6 Scenario 6

Mickey Mouse is

3.4 Use cases

3.4.1 Request a taxi

Actors: Registered user, taxi driver *Preconditions:* The user is on the homepage of the application *Flow of Events:*

- The user clicks the "Request a taxi" button
- The system shows the user a page where she can enter the starting location and destination of the request
- The user inserts the starting location and the destination and then clicks the "Request" button
- The system notifies the first taxi driver of the starting location's zone's queue. If there are no taxis in that zone, the system notifies the CAT. If a taxi driver rejects the request the system passes the request on to the next taxi driver in the queue and changes the first one's state to not

available. If a taxi driver doesn't accept or reject the request within 1 minute, the system passes the request on to the next taxi driver in the queue and moves the first one to the end of the queue

- The taxi driver accepts the request and goes to the starting location
- The system changes the state of the taxi driver to not available and notifies the user of the code of the incoming taxi and the ETA
- When the taxi driver arrives at the starting location, according to her GPS information, the system automatically makes the user pay the minimum fare

Postconditions: The taxi driver is not available anymore and has been removed from the queue. The user has been charged with the minimum fare. *Exceptions:*

- If there are no available taxis in the entire city an error message is shown and the request is not made
- If a taxi driver doesn't get to the starting location of the request within the MAD the system will pass on the request to the next taxi driver in the queue and will notify the user of the change with the new ETA and taxi code.

3.4.2 Reserve a taxi

Actors: Registered user, taxi driver *Preconditions:* The user is on the homepage of the application *Flow of Events:*

- The user clicks the "Reserve a taxi" button
- The system shows the user a page where she can enter the starting location, destination and the meeting time of the reservation.
- The user inserts the starting location, destination and meeting time and then clicks the "Reserve" button
- Ten minutes before the meeting time, the system notifies the first taxi driver of the starting location's zone's queue. If there are no taxis in that zone, the system notifies the CAT. If a taxi driver rejects the reservation the system passes it on to the next taxi driver in the queue and changes the first one's state to not available. If a taxi driver doesn't accept or reject the reservation within 1 minute, the system passes it on to the next taxi driver in the queue and moves the first one to the end of the queue
- The taxi driver accepts the reservation and goes to the starting location
- The system changes the state of the taxi driver to not available and notifies the user of the code of the incoming taxi and the ETA

- When the taxi driver arrives at the starting location, according to her GPS information, the system automatically makes the user pay the minimum fare

Postconditions: The taxi driver is not available anymore and has been removed from the queue. The user has been charged with the minimum fare. *Exceptions:*

- If there are no available taxis in the entire city an error message is shown and...
- If a taxi driver doesn't get to the starting location of the request within the MAD after the meeting time, the system will pass on the reservation to the next taxi driver in the queue and will notify the user of the change with the new ETA and taxi code.
- If the meeting time is not at least two hours after the reservation an error message will be displayed and the reservation won't be made.

4 Alloy

4.1 Model

```
module TAXInseconds
//TAXIES
abstract sig Taxi{}
sig AvailableTaxi extends Taxi{
    //For the queues
    nextTaxi:lone AvailableTaxi,
    //For the service
    serve:lone ActiveClient
}
sig InactiveTaxi extends Taxi{}
sig TaxiQueue{root:AvailableTaxi}

//A Taxi can't be his next
fact nextTaxiNotReflexive {
    no t:AvailableTaxi| t = t.nextTaxi
}
//A Taxi can't be one of his followers in the queue
fact nextTaxiNotCyclic {
    no t:AvailableTaxi | t in t.^nextTaxi
}
//If a taxi is active he must be in exactly one queue
fact allAvailableTaxiesBelongToOneQueue {
    all t:AvailableTaxi | one q:TaxiQueue | t in q.root.*nextTaxi
}

//CLIENTS
abstract sig Client{}
sig ActiveClient extends Client{
    //For the queue
    nextClient:lone ActiveClient
}
sig nonActiveClient extends Client{}
sig ClientQueue{root:ActiveClient}

//A Client can't be his next
fact nextClientNotReflexive {
    no c:ActiveClient| c= c.nextClient
}
//A Client can't be one of his followers in the queue
fact nextClientNotCyclic {
    no c:ActiveClient| c in c.^nextClient
}
```

```
43      //If a client is Waiting for a Taxi, he must be in exactly
one queue
44      fact allActiveClientsBelongToOneQueue {
45          all c:ActiveClient| one q:ClientQueue| c in q.root.*nextClient
46      }
47
48      //AREAS
49      sig Area{
50          taxis:  one TaxiQueue,
51          clients:  one ClientQueue
52      }
53      fact oneQueueoneArea{
54          no c:ClientQueue | some disjoint a,a':Area |
55              c=a.clients and c=a'.clients
56          no t:TaxiQueue | some disjoint a,a':Area |
57              t=a.taxis and t=a'.taxis
58      }
59
60      //All queues must be connected to an Area
61      fact allQueuesInAreas{
62          all c:ClientQueue | some a:Area | c=a.clients
63          all t:TaxiQueue | some a:Area | t=a.taxis
64      }
65
66      //INTERACTIONS
67      //Clients are served only by one taxi
68      fact noClientsObiquity{
69          no disjoint t,t':AvailableTaxi | t'.serve=t.serve
70      }
71
72      //Clients are served in order
73      fact ClientsRespectQueuesEvenInItaly{
74          no c:ActiveClient | some t:AvailableTaxi |
75          c=t.serve and no t':AvailableTaxi | t'.serve=c. nextClient
76      }
77
78      //Taxies are serving in order
79      fact TaxisServeInOrder{
80          no t,t':AvailableTaxi | some c:ActiveClient |
81          t'=t. nextTaxi and c=t.serve and no c':ActiveClient| c'=t'.serve
82      }
83
84      //Taxies only serve clients in the same area
85      fact{
86          no t:AvailableTaxi | some a:Area | some c:ActiveClient
|
```

```
87          c=t.serve and t in a.taxis.root.*nextTaxi and
88          c not in a.clients.root.*nextClient
89      }
90
91      //FUNCTIONS
92      //Get who a taxi is serving
93      fun getTaxiClient[t:AvailableTaxi]:  lone ActiveClient{
94          t.serve
95      }
96
97      //Get who serves a client
98      fun getClientServer[c:ActiveClient]:  lone AvailableTaxi{
99          c. serve
100     }
101     //Get Queues for an area
102     fun getTaxisInArea[a:Area]:  set AvailableTaxi{
103         a.taxis.root.*nextTaxi
104     }
105     fun getActiveClientsInArea[a:Area]:  set ActiveClient{
106         a.clients.root.*nextClient
107     }
108
109     //ASSERTIONS
110     //hm, can't get what is the difference between facts and assertions
111
112     //PREDICATES
113     //Just show stuff in different ways
114     //Make a call
115
116     //TODO reservations (i can't understand how)
117     //TODO database (maybe)
118
119
120     pred show{}
121     run show{} for 5     but 2     Area, 2     TaxiQueue, 2     ClientQueue,
0     InactiveTaxi, 0     nonActiveClient, 6     AvailableTaxi, 6     ActiveClient
```

4.2 Result

4.3 Worlds Generated