myTaxiService

Requirements Analysis and Specifications Document

Jacopo Strada, Luca Riva

November 11, 2015

Table of Contents

Purpose

This document is meant to describe a software solution for the *myTaxiService* problem. The document includes a description of the problem, functional and non-functional requirements and it is addressed to the IT department of the city administration.

Scope

The *myTaxiService* software focuses on helping the clients benefit from the service and ensures a fair management of taxi queues.

Goals I

G1: Allow clients to register an account in the system

G2: Allow taxi drivers to register an account in the system

G3: Allow clients to log in the mobile application or in the website

G4: Allow taxi drivers to log in the mobile application

G5: Allow clients to request a taxi

G6: Inform the passenger about the code of the incoming taxi

G7: Inform the passenger about the waiting time

G8: Inform the passenger about a possible call rejection

Goals II

G9: Retrieve the position of each taxi from their mobile application using GPS

G10: Allow taxi drivers to set their state as *available* or *unavailable*

G11: Notify the taxi drivers about a client's request

G12: Allow taxi drivers to accept or decline a certain client's request

G13: Guarantee a fair management of taxi queues

G14: Allow a user to reserve a taxi by specifying the origin and the destination of the ride at least two hours before

G15: Allow a driver to accept a client reservation

G16: Confirm the reservation to the user

G17: Supply APIs for further implementation of additional services in the future

Table of Contents

Product perspective

The software will consist in three different user interfaces: two different apps for the clients and the taxi drivers and one web application.
In order to make this working there will be a web server containing the application logic and a DBMS that stores all the data (such as users' credentials, taxi's positions, taxi's availability . . .). Mobile applications, web application and the server communicate through the Internet.
The system will also provide programmatic interfaces for other new services. At the moment the city has not any similar system, so the application will be completely independent.

User characteristics

This application is conceived for every taxi driver in the city area and for adult clients (18+) with a compatible device which satisfies the specification listed in the previous section. There are no specific levels of education or expertise needed apart from a decent knowledge of English language.

Domain properties

D1: If a client requests a ride they will wait the taxi's arrival

D2: If a client makes a reservation for a certain time they will be punctual waiting for the ride

D3: If a driver accepts a reservation or a request they will actually try to reach the client location

Assumptions I

A1: Possibility to gain access to the taxi driver's licences database of the city

A2: A client is able to make only one taxi request at once

A3: A client is not able to make two overlapped reservations

A4: If a request is rejected by all the taxi drivers in the queue the request will be discarded: it will not iterate the queue for a second time.

A5: As soon as the request is discarded the client will be notified and able to make a new call

A6: The system will not give the client any information regarding the cost of the ride

A7: If a client calls a taxi and the request is accepted the client will not be able to modify or erase the request.

Assumptions II

In order to assure an efficient service it is necessary a slight modification to the customer request:

A8: When a customer reserves a taxi the system allocates the request to all registered driver

A9: If a driver accepts the request the system confirms the reservation to the user

A10: If no driver confirms the reservation in at most an hour since the request the system deletes the reservation and notifies the client

A11: A taxi reservation's origin must be located inside one of the zones of the city

A12: A client is able to cancel a reservation at least ten minutes before the ride.

A13: The system notifies the driver about the ride they accepted ten minutes before the hour of the appointment with the client

Assumptions A8, A9, A10, A13 were made in order to avoid the risk of having an empty queue or no driver who would accept the reservation only ten minutes before the fixed hour of the ride.

Apportioning of requirements I

Reporting: A client may report the lack of service of a driver so that the city's government would be able to take action against the said driver. At the same time a driver may be able to report a client who misses fixed appointments, thus after few reclaims the client's account could be suspended or deactivated.

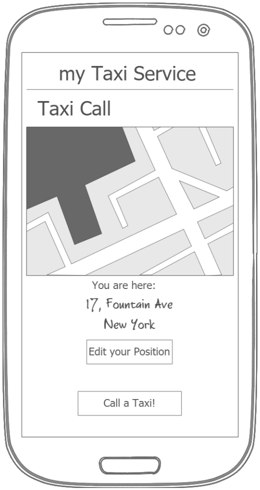
Payments: It would be interesting to give the client the possibility to pay the reservation service directly through the system, which could also provide an estimated price for the ride given the origin and the destination.

Apportioning of requirements II

Taxi sharing: An user may be ready to share a taxi, so the application could mark their requests in order to allow other clients to participate in the ride if they have a similar starting point and a similar destination. Obviously the advantage wold be the possibility to split the cost of the ride, which should be calculated by the system. The system should also compute the route for the taxi driver.

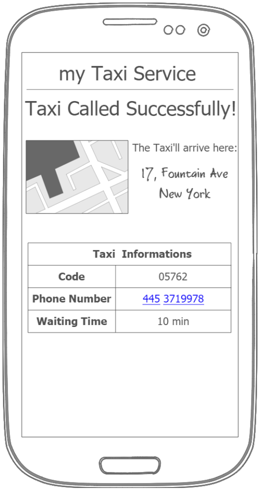
Table of Contents

Clients' User Interfaces - Taxi Call


my Taxi Service

Taxi Call

You are here:
17, Fountain Ave
New York

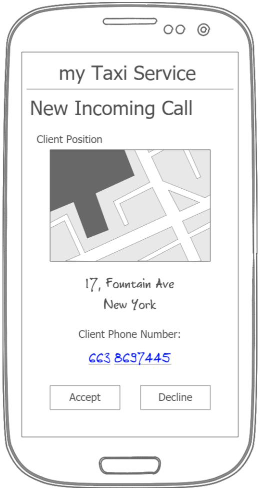

my Taxi Service

Taxi Called Successfully!

The Taxi'll arrive here:
17, Fountain Ave
New York

Taxi Informations	
Code	05762
Phone Number	445 3719978
Waiting Time	10 min

Taxi Drivers' User Interfaces - New Requests


my Taxi Service

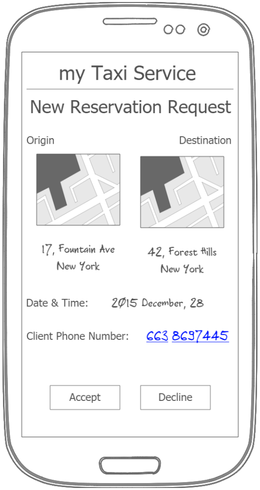
New Incoming Call

Client Position

17, Fountain Ave
New York

Client Phone Number:

663 8637445


my Taxi Service

New Reservation Request

Origin — Destination

17, Fountain Ave
New York

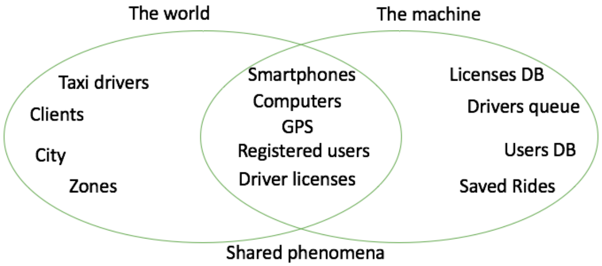
42, Forest Hills
New York

Date & Time: 2015 December, 28

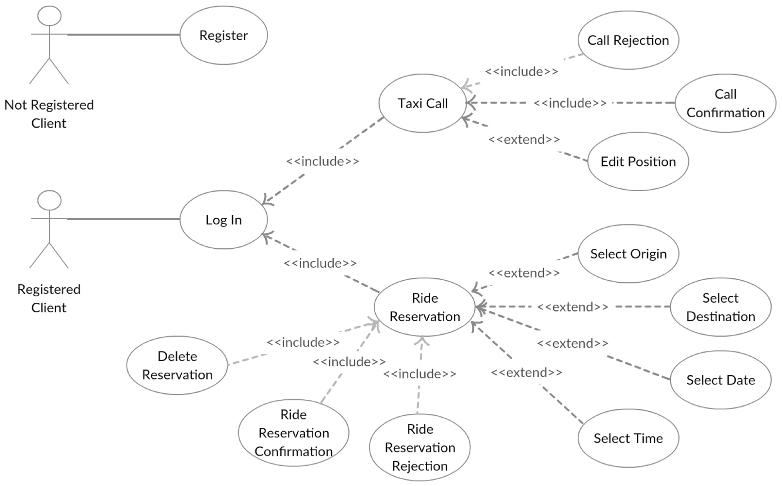
Client Phone Number: 663 8637445

The World and The Machine

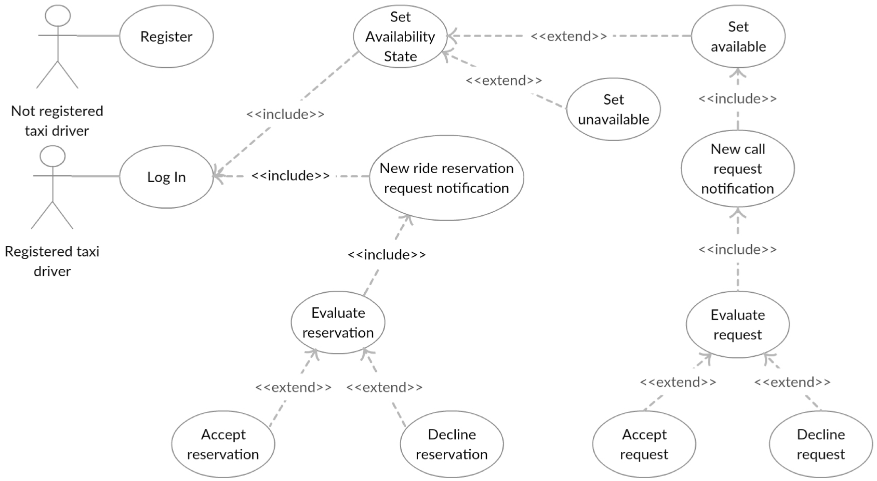
The model by M. Jackson & P. Zave is able to clearly schematize where the entities involved in the project are ideally located and which are the means of interaction between the world and the system at a first glance.



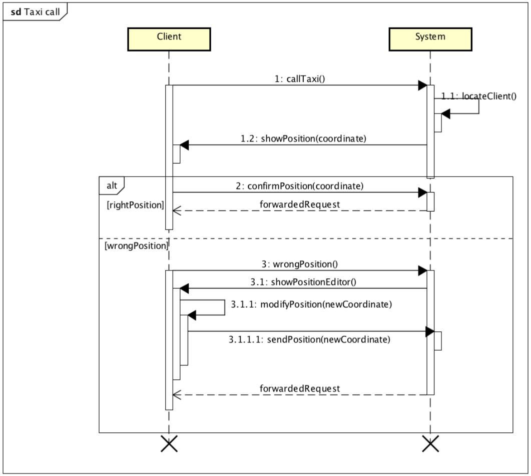
Clients UML use case diagrams



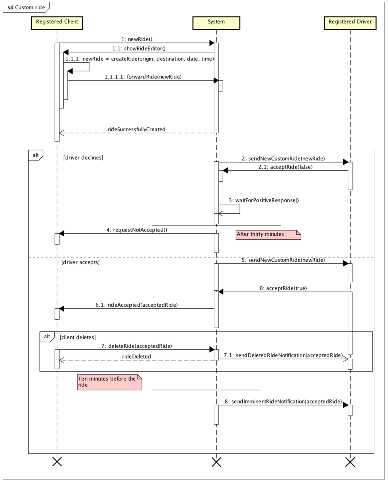
Taxi Drivers UML use case diagrams



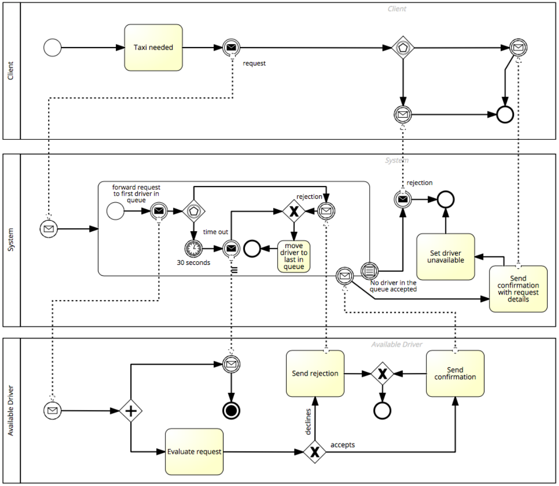
Taxi call UML sequence diagram


sd Taxi call
Client
System
1: callTaxi()
1.1: locateClient()
1.2: showPosition(coordinate)
alt
[rightPosition]
2: confirmPosition(coordinate)
forwardedRequest
[wrongPosition]
3: wrongPosition()
3.1: showPositionEditor()
3.1.1: modifyPosition(newCoordinate)
3.1.1.1: sendPosition(newCoordinate)
forwardedRequest

Taxi reservation UML sequence diagram


sd Custom ride
Registered Client
System
Registered Driver
1: newRide()
1.1: showRideEditor()
1.1.1: newRide = createRide(origin, destination, date, time)
1.1.1.1: forwardRide(newRide)
rideSuccessfullyCreated
alt
[driver declines]
2: sendNewCustomRide(newRide)
2.1: acceptRide(false)
3: waitForPositiveResponse()
4: requestNotAccepted()
After thirty minutes
[driver accepts]
5: sendNewCustomRide(newRide)
6: acceptRide(true)
6.1: rideAccepted(acceptedRide)
alt
[client deletes]
7: deleteRide(acceptedRide)
rideDeleted
7.1: sendDeletedRideNotification(acceptedRide)
Ten minutes before the ride
8: sendImminentRideNotification(acceptedRide)

BPMN taxi call



BPMN ride reservation

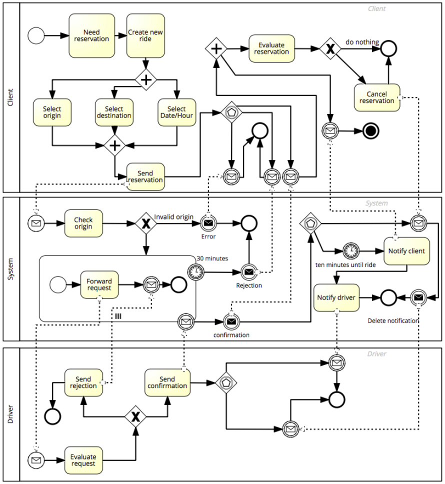


Table of Contents

Signatures I

```
// Import boolean utility
open util/boolean
// Primitive String Type
sig Strings{}

// General user of the service
abstract sig User {
  eMail: one Strings
}
// Taxi Driver
sig Driver extends User {
  isAvailable: one Bool
}
// Client
sig Client extends User {}

// Taxi Drivers Queue
sig Queue {
  root: lone Node
}
// Queue Element
sig Node {
  ndriver: one Driver,
```

Signatures II

```
  next: lone Node
}

// City Zone
sig Zone {
  queue: one Queue,
}

// Taxi Call made by a Client
sig TaxiCall {
  client: one Client
}
// Call Confirmation made by a Taxi Driver and related to a Taxi
    Call
sig CallConfirmation {
  call: one TaxiCall,
  cdriver: one Driver
}

// Ride Request made by a Client
sig RideRequest {
  client: one Client
}
```

Signatures III

```
// Ride Confirmation made by a Taxi Driver and related to a Taxi
    Call
sig RideReservationConfirmation {
  request: one RideRequest,
  rdriver: one Driver
}
```

Facts I

```
open signatures

// BEGIN: Definition of a generic Queue:
// A node cannot be connected to itself
fact nextNotReflexive {
  no n:Node | n = n.next
}
// Cycles must not be present
fact nextNotCyclic {
  no n:Node | n in n.^next
}
// Cannot exist a node not in a queue
fact allNodesBelongToAQueue {
  all n: Node | one q: Queue | n in q.root.*next
}
// END

// All the available taxi drivers must be in one and only one queue
fact availableDriverInQueue {
  all d: Driver | (one n: Node | d.isAvailable.isTrue implies d in n
     .ndriver) or d.isAvailable.isFalse
  all d: Driver, n: Node | d in n.ndriver implies d.isAvailable.
    isTrue
```

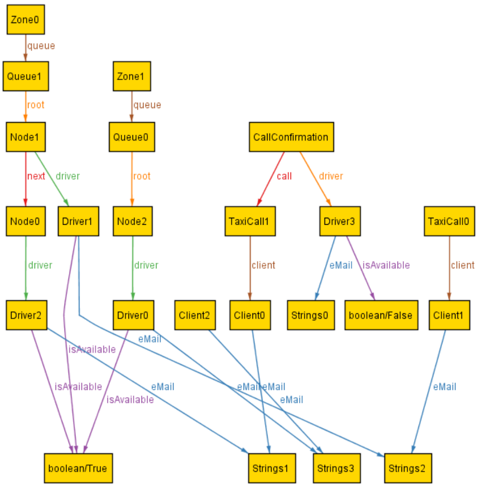
Facts II

```
}

// A queue must own to a zone
fact queueMustBeInAZone {
  all q:Queue | one z: Zone | q in z.queue
}

// A Taxi call can have at most a Call Confirmation
fact oneCallConfirmationPerTaxiCall {
  all c: TaxiCall, disj cc1,cc2: CallConfirmation | c in cc1.call
     implies not c in cc2.call
}

// A Client can make at most a Taxi Call at a time
fact oneTaxiCallPerClient {
  all c: Client, disj c1,c2: TaxiCall | c in c1.client implies not c
      in c2.client
}

// A Taxi Driver can accept at most a Call at a time
fact oneCallConfirmationPerDriver {
  all d: Driver, disj cc1,cc2: CallConfirmation | d in cc1.cdriver
     implies d not in cc2.cdriver
}
```

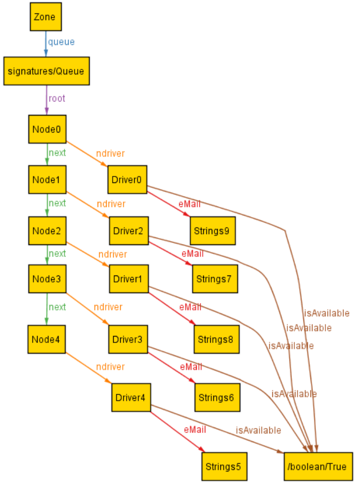
Facts III

```
// A Taxi Driver who accepts a Taxi Call must become "unavailable"
fact busyDriver{
  all d: Driver, cc: CallConfirmation | d in cc.cdriver implies d.
     isAvailable.isFalse
}

// A Ride Request can have at most a Ride Reservation Confirmation
fact oneRideReservationConfirmationPerRideRequest {
  all rr: RideRequest, disj rc1,rc2: RideReservationConfirmation |
     rr in rc1.request implies not rr in rc2.request
}

// Two different Users cannot have the same e-mail address but with
     the same e-mail a driver must be registered as a client too.
fact noSameEMail{
  all disj c1, c2: Client | not c1.eMail = c2.eMail
  all disj d1, d2: Driver | not d1.eMail = d2.eMail
}
```

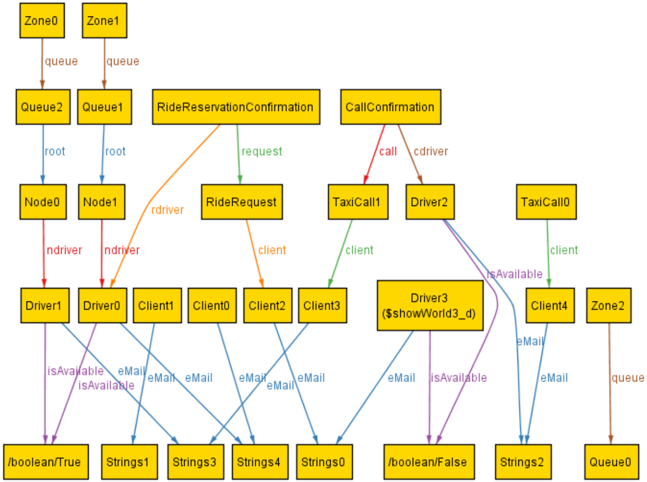
Worlds I


Zone0
queue
Queue1
root
Node1
next
driver
Node0
Driver1
Zone1
queue
Queue0
root
Node2
driver
Driver0
driver
Driver2
CallConfirmation
call
driver
TaxiCall1
Driver3
TaxiCall0
client
eMail
isAvailable
client
Client2
Client0
Strings0
boolean/False
Client1
isAvailable
isAvailable
isAvailable
eMail
eMail
eMail
eMail
eMail
eMail
boolean/True
Strings1
Strings3
Strings2

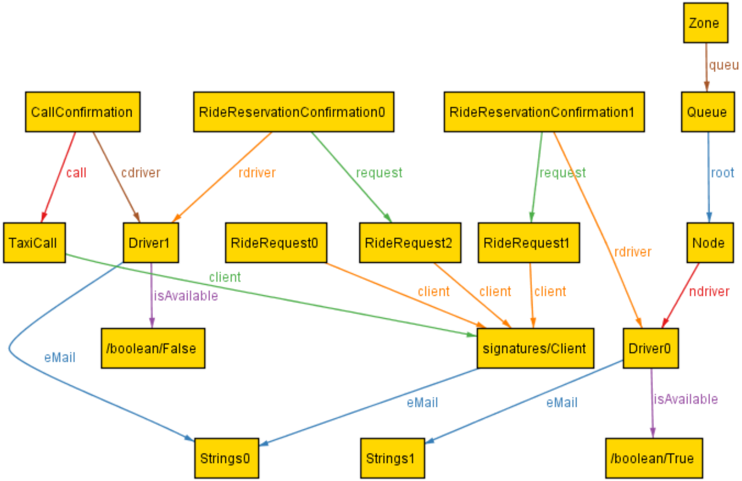
Worlds II


Zone
queue
signatures/Queue
root
Node0
next
ndriver
Node1
Driver0
next
ndriver
eMail
Node2
Driver2
Strings9
next
ndriver
eMail
Node3
Driver1
Strings7
next
ndriver
eMail
isAvailable
isAvailable
isAvailable
Node4
Driver3
Strings8
ndriver
eMail
isAvailable
Driver4
Strings6
eMail
isAvailable
Strings5
/boolean/True

Worlds III


Zone0
Zone1
queue
queue
Queue2
Queue1
RideReservationConfirmation
CallConfirmation
root
root
request
call
cdriver
Node0
Node1
rdriver
RideRequest
TaxiCall1
Driver2
TaxiCall0
ndriver
ndriver
client
client
client
isAvailable
Driver1
Driver0
Client1
Client0
Client2
Client3
Driver3 ($showWorld3_d)
Client4
Zone2
isAvailable
isAvailable
eMail
eMail
eMail
eMail
eMail
eMail
eMail
isAvailable
eMail
eMail
queue
/boolean/True
Strings1
Strings3
Strings4
Strings0
/boolean/False
Strings2
Queue0

Worlds IV


Zone
queu
CallConfirmation
RideReservationConfirmation0
RideReservationConfirmation1
Queue
call
cdriver
rdriver
request
request
root
TaxiCall
Driver1
RideRequest0
RideRequest2
RideRequest1
rdriver
Node
client
isAvailable
client
client
client
ndriver
eMail
/boolean/False
signatures/Client
Driver0
eMail
eMail
isAvailable
Strings0
Strings1
/boolean/True