Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “MyTaxiService”

Requirements Analysis and Specifications Document

Version 0.5

Ivana Salerno

Alexis Rougnant

Daniel Vacca

February 2nd 2016

Table of contents

List of Tables 3
List of Figure 5
1.Introduction 6
1.1.Purpose 6
1.2.Scope 6
1.3.Definitions, acronyms, and abbreviations 6
1.4.References 7
1.5.Overview 8
2.Overall description 9
2.1.Product perspective 9
2.1.1.The world and the machine 9
2.1.2.External systems interfaces 10
2.1.3.User interfaces 11
2.1.4.Operations 15
2.2.Product functions – Goals 15
2.2.1.G1: Passenger can request a taxi either through a web application or a mobile application 15
2.2.2.G2: Taxi driver informs the system about his/her availability 15
2.2.3.G3: Taxi driver may confirm that he/she is taking care of a certain received request only through a mobile application 16
2.2.4.G4: Requests for taxi services are fairly managed 16
2.2.5.G5: Passengers can enable a taxi sharing option 16
2.2.6.G6: Passengers can reserve taxi service in advance 16
2.3.Stakeholders, users and actors 16
2.3.1.Stakeholders 17
2.3.2.Users and actors 17
2.4.Scenarios 18
2.4.1.Scenario 1: A passenger is successfully served 18
2.4.2.Scenario 2: A taxi driver declines the request 19
2.4.3.Scenario 3: A passenger makes a reservation 19
2.4.4.Scenario 4: A passenger makes a sharing request 20
2.4.5.Scenario 5: A taxi driver registers in MyTaxiService 20
2.5.Use Cases 21

2.5.1.Create account (passenger) 22
2.5.2.Authenticate (passenger) 22
2.5.3.Edit account (passenger) 22
2.5.4.Create account (taxi driver) 22
2.5.5.Authenticate (taxi driver) 24
2.5.6.Edit account (taxi driver) 24
2.5.7.Request taxi service 25
2.5.8.Cancel accepted request (by the passenger) 27
2.5.9.Cancel accepted request (by the taxi driver) 28
2.6.Constraints 29
2.7.Class model 29
2.8.Assumptions and dependencies 32
3.Specific requirements 34
3.1.G1: Passenger can request a taxi either through a web application or a mobile application 34
3.2.G2: Taxi driver informs the system about his/her availability 34
3.3.G3: Taxi driver may confirm that he/she is taking care of a certain received request only through a mobile application 35
3.4.G4: Requests for taxi services are fairly managed 35
3.5.G5: Passengers can enable a taxi sharing option 36
3.6.G6: Passengers can reserve taxi service in advance 36
3.7.Non-functional requirements 37
3.8.Non-requirements 37
4.Alloy Modeling 38
4.1.Entities 38
4.2.Facts 38
4.3.Predicates 38
4.4.Assertions 39
4.5.Generated world 40
5.Appendix 42
5.1.Used software 42
5.2.Worked hours 42
5.3.Revisions 42

List of Tables

Table 1: External system interfaces 11

Table 2: Stakeholders 17

Table 3: Users and actors 18

List of Figure

Figure 1: World and machine 10
Figure 2: Create account - Taxi driver 12
Figure 3: Create account – Passenger 13
Figure 4: Make a request for service 13
Figure 5: Passenger behavior 14
Figure 6: Taxi driver behavior 14
Figure 7: Use case model 21
Figure 8: Create Account Use Case 23
Figure 9: Authenticate Use Case 24
Figure 10: Edit Account Use Case 25
Figure 11: Request taxi service 27
Figure 12: Request taxi service 28
Figure 13: Cancel accepted request (taxi driver) Use Case 28
Figure 14: Class diagram 32
Figure 15: Alloy generated world 1 40
Figure 16: Alloy generated world 2 41

1. Introduction

Along this section we make on overview of the present document. We treat aspects like Purpose, Scope and Overview. Glossary and references are also included in this section.

1.1. Purpose

The present is the **R**equirement **A**nalysis and **S**pecification **D**ocument, RASD, concerning the project MyTaxiService for the Software Engineering 2 course at Politecnico di Milano.

The purpose of this document is to present a complete description of the product and the analysis of its domain (which includes exposal of stakeholders, scenarios, use cases, constraints and assumptions), in order to join them together to obtain the corresponding software requirements. Therefore, we also provide here a technical sheet for the further development, and an artifact which might even be used as a contract between the eventual developers and customer(s).

1.2. Scope

MyTaxiService is a Milano's government proposal for optimizing its taxi service, by simplifying the access of the passengers to the services and guaranteeing a fair management of taxi queues.

The passengers will be able to make requests for taxi services either through the MyTaxiServices's web site or its mobile application, by sending the corresponding *request for service information*. The system then replies to the passenger with the *accepted request information*, and he is successfully served. The passengers can also reserve taxi services in advance and share the taxi with other passengers.

The taxi drivers will be able to receive requests for taxi services in the mobile application only, whenever they have informed the system about their availability. A received request is accompanied by its corresponding *incoming request information*. In the moment that the request is accepted by the driver, the passenger is informed and receives the *accepted request information*.

The requests are managed and assigned to available drivers, according to the GPS position of the driver. The city is divided in zones and each one of these has an associated queue of available taxis. The request is assigned to the first driver in the corresponding queue.

1.3. Definitions, acronyms, and abbreviations

Accepted request information: corresponds to the following information: taxi's code, estimated arrival time, fee to be paid to the taxi driver, and possibly how many people the car will be shared with. It is received by the passenger when his request has been accepted.

Compatible request: Two requests are compatible if their origin is in the same zone and if the destination of one of them can be reached in the path that goes to the destination of the

other one. For deciding this, the sharing engine will ask the Maps server to calculate a shortest possible route from the origin to the destination. Using this, the sharing engine associates to the request all the zones that are reached within a radius of 1 km from any point of the estimated path.

Incoming request information: corresponds to the following information: the origin(s), destination(s), the eventually payed fee for the trip, and possibly the amount of passengers. It is received by the taxi driver together with a request for a service.

Passenger account information: corresponds to the following data: name, email address and password. It is complete if all the fields are filled in. It is correct if no other account has been registered with that email address and if this is a valid one.

Process request: the processing of a request means to create the *incoming request information*. Refer to the Request taxi service use case for more details.

Recognize request: the recognition of a request means to identify the *request for service information* just received in order to know the type of the request (normal, sharing, or reservation), and support the subsequent processing.

Request for service information: corresponds to the following information: the origin and destination of the trip (either as GPS positions or addresses), the amount of people, whether he wants to share or not the taxi, and the time he wants to be picked up (which can be *now* or a moment no before than 2 hours later). It is provided by the passenger when he makes a request for a service. This is considered to be correct if both origin and destination are valid places in Milano and if the amount of people does not exceed the maximum (which is four). This is considered to be complete if all fields have been filled in.

Taxi driver account information: corresponds to the following data: taxi code, name, username, password, email address, and taxi capacity. It is complete if all the fields have been filled in. It is correct if the data correspond to the one provided by the Milano Government Information System.

Zone: describes a portion of the city. They are have been already defined by the Government of Milano in the form of coordinates that correspond to the vertices of a polygon whose area is 2 km^2 approximately.

1.4. References

The following documents were used to develop the present document:

- MyTaxiService project AA 2015-2016 description
- RASD MeteoCal examples 1 and 2
- RASD SWIM example
- IEEE standard for requirement specification
- Lecture slides on Requirements engineering

1.5. Overview

This document is essentially structured in 5 parts:

- Part 1: Introduction, it gives a general description of the document and some primary information about the software.
- Part 2: Overall description, it gives more detailed information about the software, in particular it explains what the goals of the software are, who the stakeholders, users and actors are and how they interact with each other. This is achieved through the use of some cases and typical scenarios. At the end, a focus on constraints, assumptions and dependencies is made.
- Part 3: Specific requirements, it lists all the software requirements.
- Part 4: Alloy modelling,
- Part 5: Appendix.

2. Overall description

The intention of this section is to contextualize the developing software product and provide the background that justifies the subsequent definition of the requirements. We start by putting the MyTaxiService product in perspective with respect to its surrounding world, to proceed next to define its functionalities in terms of goals. After that, deeper analysis to relevant aspects are presented; those aspects include Stakeholders, users and actors, Scenarios, Constraints, and Assumptions and dependencies. The results are formalized through Use case models and a Class diagram.

2.1. Product perspective

In this subsection we present the MyTaxiService software from a context-oriented perspective. We start by applying "The world and the machine" paradigm to this particular case, and then we describe the interfaces that will allow the application satisfy the requirements.

MyTaxiService makes neither part of any already existing system nor replace an existing one, so there are no several constraints on the required interfaces.

2.1.1. The world and the machine

The phenomena we describe here (both of the world and machine) will be listed in the form of events.

This first list corresponds to world elements that are not observable by the machine but which are somehow relevant. Since they are not observable, they indicate the events that the system will not support at all:

- A taxi driver picks up a passenger
- A taxi driver takes a passenger to his destination
- A taxi driver gets paid for his service
- A taxi driver is stuck in traffic
- A street is in bad conditions
- A taxi is in bad conditions
- A passenger requests a taxi service though a phone call
- A ride cannot be completed because the taxi stops working

The list below shows the world phenomena that are observable by the machine:

- A passenger creates an account
- A passenger logs in
- A passenger edits his account
- A passenger requests a taxi service (normal, shared or reservation)
- A taxi driver creates an account

- A taxi driver logs in
- A taxi driver edits his account
- A taxi driver informs his availability
- A taxi driver accepts a passenger's request
- A taxi driver declines a passenger's request
- A passenger cancels a request
- A taxi driver cancels an accepted request

The third list exposes the events that occur inside the machine but are still observable by the world:

- A processed request is sent to a taxi driver
- A passenger is informed about his request's result
- A taxi driver is informed about the request cancelation
- A passenger is informed about the request cancelation

Finally we have the phenomena occurring inside the machine and unreachable by the world. We list only few of them since they refer to the internal work of the machine and it does not concern the requirements process:

- A request is processed (the address is assigned to a zone, available taxi driver is found, trip-mates are searched, etc.)
- A request is recognized

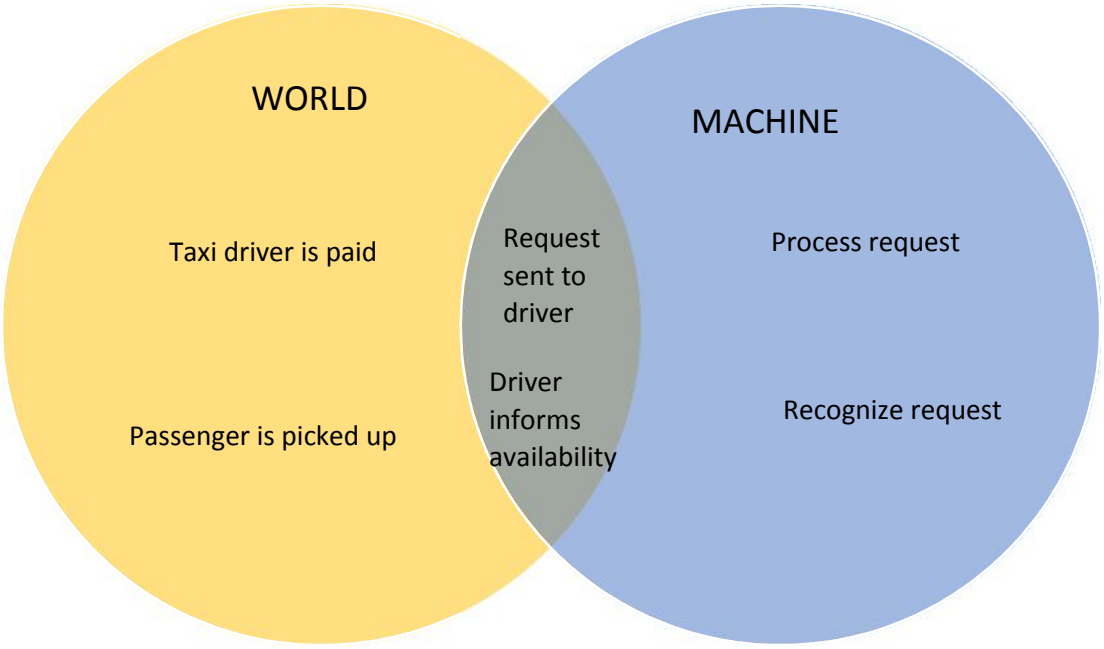

Figure 1: World and machine

2.1.2. External systems interfaces

The following table relates the external systems that will interact with MyTaxiService, the nature of the interface and the description of the interaction.

External system	Nature of interface	Description of interaction
Maps server	Provided	MyTaxiService will use the maps service to perform the following tasks: - Resolve the passenger provided address to actual coordinates - Validate the GPS positions given by the passenger and taxi driver - Display a map to both the driver and passenger with the driver's and the pick-up position - Calculate the estimated arrival time for the taxi driver
Milano government's information system	Provided	MyTaxiService will access to a database of the government of Milano to obtain the valid assigned codes and verify the driver's identity.
Email server	Provided	MyTaxiService will use an email server to send notifications to the passengers when they perform a reservation and to confirm the email address of the passengers. This will also be used to send confirmation messages.

Table 1: External system interfaces

2.1.3. User interfaces

MyTaxiService will interact with two types of users: passengers and taxi drivers.

A passenger can interact with the system through the web site or the mobile application, and the system must allow him to perform all the following logical actions regardless where he accesses from:

- Create account
- Log in
- Edit account
- Make a request for a taxi service (normal, shared or reservation)
- View the current state of his request
- View the response to his request
- View the information of the taxi driver
- View the driver's and the pick-up position
- Cancel a request

A taxi driver can interact with the system only through the mobile application, and the system must allow them to perform all the following logical actions:

- Create account
- Log in
- Edit account
- Inform his availability
- View received requests
- Accept or decline received requests
- View his own position and the pick-up point's position
- Cancel an accepted request

The following diagrams illustrate in a rough manner the aspect of those interfaces:

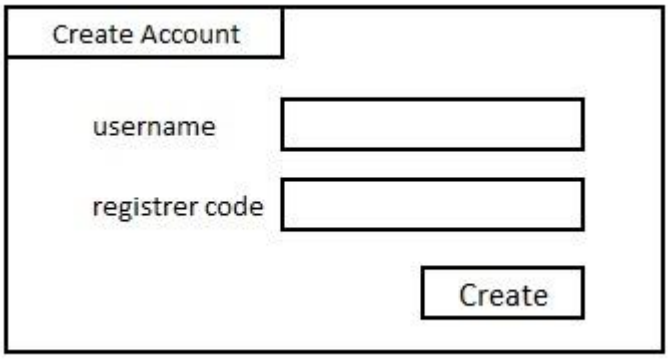

Figure 2: Create account - Taxi driver

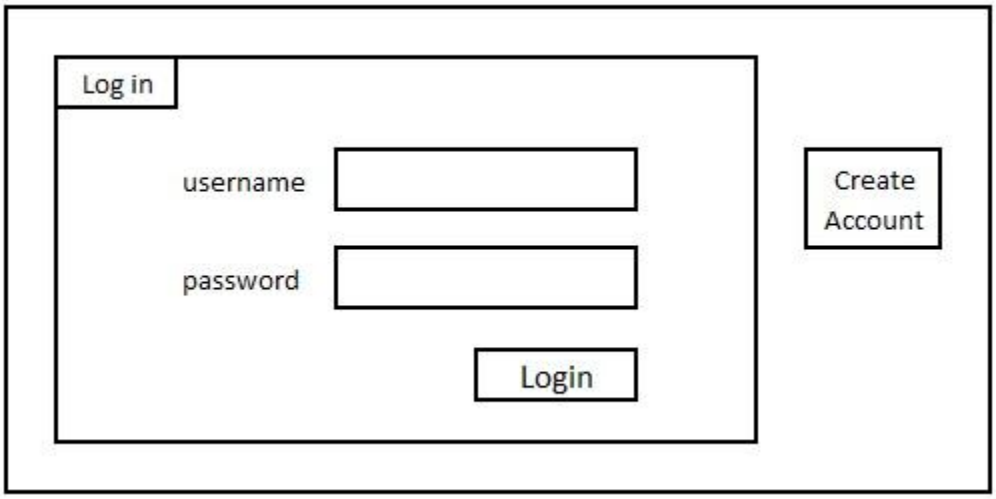

Figure 3: Create account – Passenger

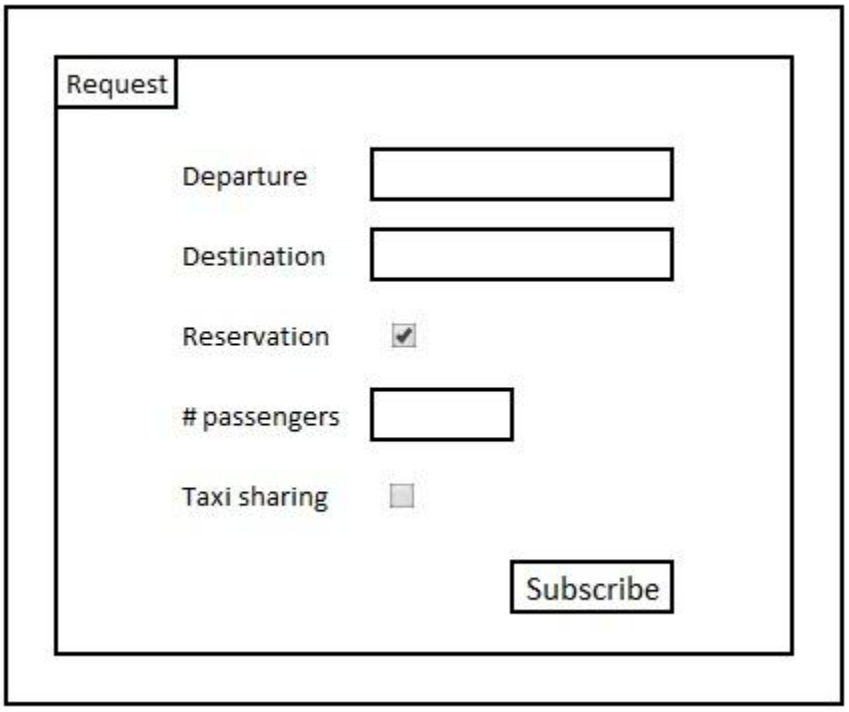

Figure 4: Make a request for service

The following state-chart diagrams model the overall behavior of a passenger while interacts with the system:

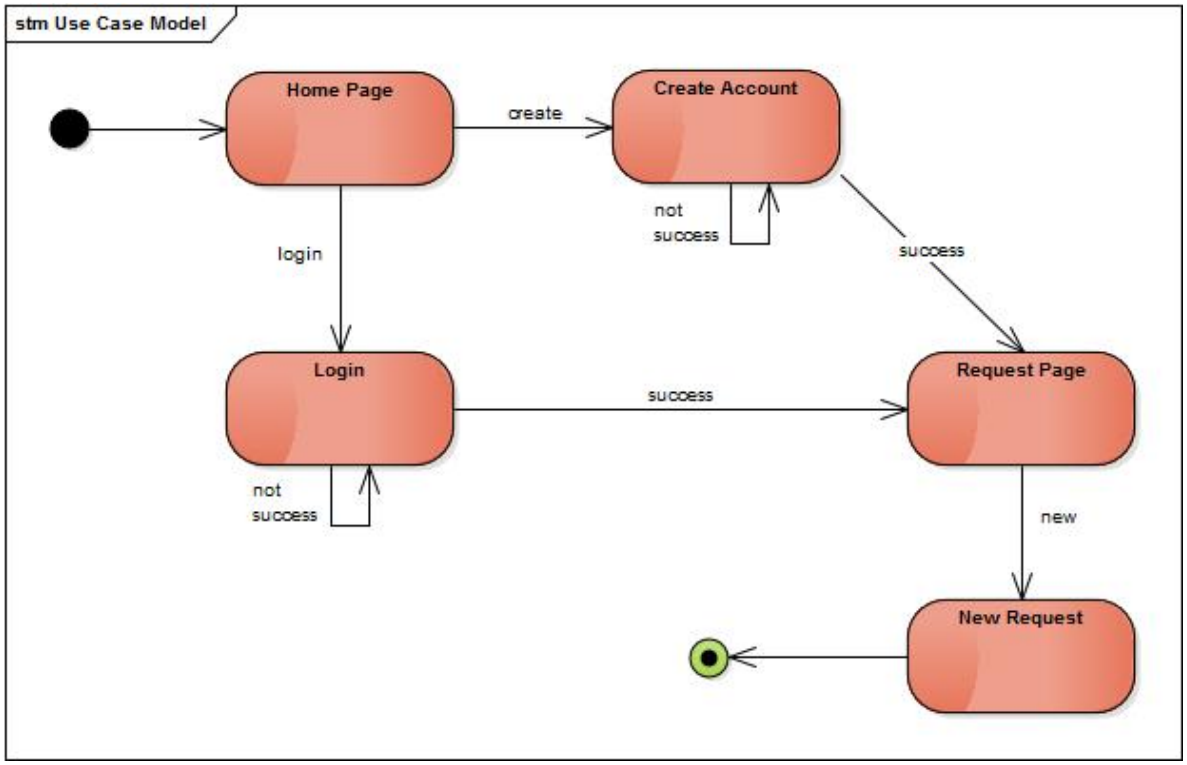

Figure 5: Passenger behavior

The following state-chart diagrams model the overall behavior of a taxi driver while interacts with the system:

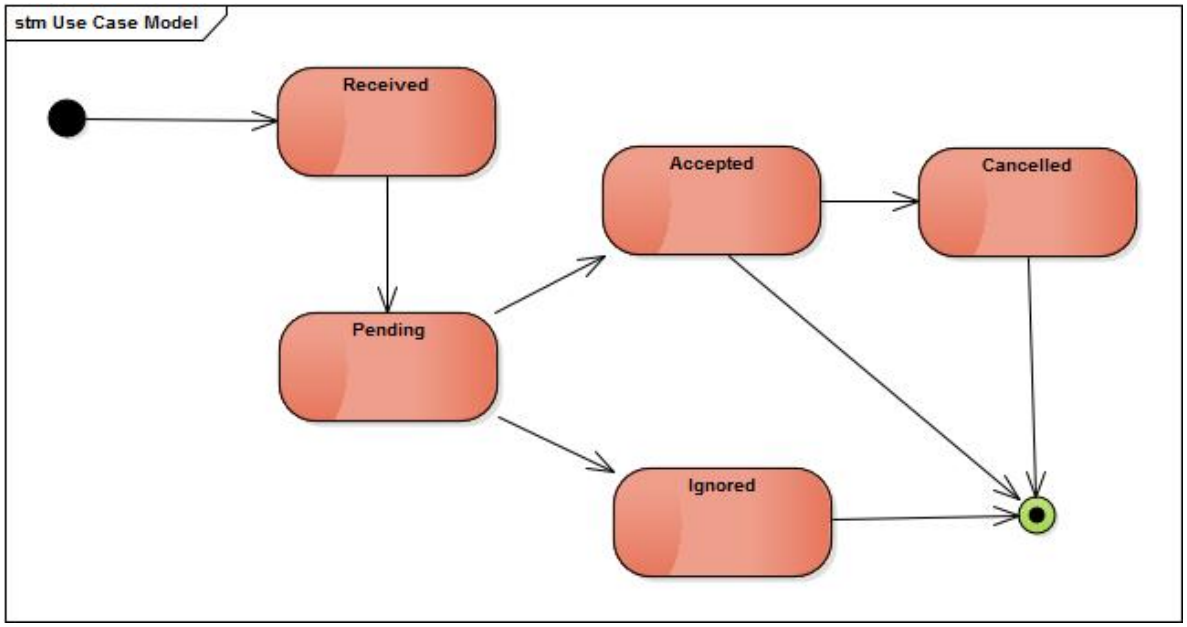

Figure 6: Taxi driver behavior

According to one of the project's constraints, the system will have to provide an interface to eventual developers to allow them to implement further functionalities. Only the following basic primitives will be exposed:

- Receive requests from passengers (only immediate and non-sharing ones)
- Search taxi driver in a given zone
- Send requests to taxi drivers
- Receive taxi drivers' response
- Send response to passengers
- Receive passenger and driver cancelations

2.1.4. Operations

According to the previous section, MyTaxiService will support access only to two type of users. The interaction that they will hold with the system shall only be under one mode of operation, so no additional ones will be implemented (e. g. administrator mode). The description of such operation mode has already been described.

2.2. Product functions – Goals

In this subsection we expose the product functions in the form of goals. Since we are following "*The world and the machine"* approach, such goals are listed in the form of events whose accomplishment shall be supported by the system.

2.2.1. G1: Passenger can request a taxi either through a web application or a mobile application

With this goal we want the system to support the passenger in making requests. This includes:

- Create account
- Log in
- Edit account
- Filling in the request information
- Sending the request
- Receiving a response
- Cancel a request

2.2.2. G2: Taxi driver informs the system about his/her availability

With this goal we want the system to support the taxi driver in informing his availability. This also includes the creation/editing of his account, the login process and the status setting.

2.2.3. G3: Taxi driver may confirm that he/she is taking care of a certain received request only through a mobile application

With this goal we want the system to support the taxi driver in receiving requests and the subsequent possible actions:

- Receive the request
- Accept the request
- Decline the request
- Cancel accepted request
- View current position, origin(s) and destination(s)

2.2.4. G4: Requests for taxi services are fairly managed

With this goal we want the system to manage and assign the received requests in a fair manner through the use of zone queues and specific policies:

- Requests are sent to the first driver in the queue (or adjacent queues if they that one is empty)
- If a driver accepts a request he is removed from the queue
- If a driver declines a request he is sent to the bottom of the queue
- If a driver cancels an already accepted request he is sent to the bottom of the queue
- If a taxi driver receives a cancelation he is sent to the top of the queue
- If a driver enters in a new zone he is sent to the bottom of the queue
- If a driver does not respond to an incoming request after 20 seconds he is sent to the bottom of the queue

2.2.5. G5: Passengers can enable a taxi sharing option

With this goal we want the system to support the passenger in making sharing requests. This includes the procedure that the system must follow to match compatible requests and the appropriate calculation of the fee.

2.2.6. G6: Passengers can reserve taxi service in advance

With this goal we want the system to support the passenger in making reservation requests.

2.3. Stakeholders, users and actors

This subsection begins with a briefly description of the people who are interested in the development of the MyTaxiService product. Further information about those which will interact closer with the system is provided afterwards.

2.3.1. Stakeholders

The following table presents the stakeholders concerning the MyTaxiService software creation.

Stakeholder	Description
Government of Milano	The project is initiated by the government of the city of Milano, which we assume is being represented by the course teacher.
Taxi drivers in Milano	They want to have the service requests fairly managed.
Taxi users in Milano	They want to easily access the taxi service.
Production engineers	The project team that will design the solution and plan its realization. Represented by the authors of this document.
Developers	It will be an external team, actually we will use the source code of an already existing project.
Testers	Our project team will test the solution, identify bugs and inspect the code as well
Communicators	Our project team will provide some reviews during the evolution of the project

Table 2: Stakeholders

2.3.2. Users and actors

The description of the product allows us to have almost the same entities as users and as actors. The following table presents those entities and their corresponding role.

Entity	Description	Role
Passenger	Person who uses MyTaxiService to make a request for a taxi service. He has to perform a log-in into the system to make requests. He can send requests either through the web site or the mobile application.	User, Actor.
Taxi driver	Person who makes use of the MyTaxiService to attend	User, Actor.

	requests for taxi services. He must have an account to log-in into the system, which includes a taxi code. He receives the requests in the mobile application.	
MyTaxiService Developer	Developer who uses the programmatic interfaces of the solution to add some features on top of it.	User.

Table 3: Users and actors

2.4. Scenarios

In this section we mean to give an initial description of the expected system behavior by presenting informal but concrete examples in the form of scenarios.

2.4.1. Scenario 1: A passenger is successfully served

Saria has a job interview the morning after and she is so excited that she can't sleep. During the night she receives a call from her cousin who tells her that she won't make it to go with her by car because her car is out of order. She also tells her that she can reach her job using public transports. In that moment Saria panics. She doesn't know what to do because she doesn't want to get to work by public transport as she is a little bit picky. In that moment she decides to open the mobile application MyTaxiService she downloaded a few days earlier and see how it works. What she sees is a form which she has to fill with all the information necessary to request a Taxi:

- The location where the Taxi has to pick her up
- The location where she has to go
- Whether she needs the Taxi in that moment or she wants to make a reservation
- How many passengers will join the ride
- Whether the option of "Taxi Sharing" can be taken into consideration or not

Saria sees that MyTaxiService is easy to use so she immediately calms down and feels better. As soon as the alarm clock goes off she decides to fill in the form to make her request in the following way:

- Via Sagivasa 21
- Via Job 8
- Now
- 1
- No

Then Saria taps “Subscribe” to send the request.

What she sees now is a report of her request indicating the price and the duration of the ride. After a while she receives a phone message revealing the code 8689, which is the code of the Taxi that will pick her up, and the waiting time which is 14 minutes.

2.4.2. Scenario 2: A taxi driver declines the request

Mario celebrated one of his friend's birthday in the downtown, and after that he wanted to go home and decided to call a taxi using the mobile application of MyTaxiService. He opened the application and filled the following data:

- The location from where he wants to be picked up: his current location
- The destination location: his home address
- When he wants this ride: now
- For how many people he orders the ride: 1
- If he wants to share the ride: no

The application showed a sum up of his request with the price and the duration of the ride. Mario validated his request.

The application asked the taxi driver 1256, who was the first in the queue related to the area from where the ride is supposed to start, if he wanted to take care of this ride. Unfortunately, the taxi driver didn't want to take care of this ride, he send a negative answer to the application. Then, the application put the taxi driver 1256 at the end of the queue and asked the taxi driver 230, who was the second in the queue, for the same request. This time, the taxi driver accepted the request. Thus, the application changed the status of this taxi driver to unavailable and put him out of the queue.

Finally, Mario received a positive answer from the application with the number of the taxi he will take. The taxi 230 came to pick him around 10 minutes later and he went home.

2.4.3. Scenario 3: A passenger makes a reservation

It’s 31st October. Evening. Sara, Giovanna and a few friends are getting ready for a Halloween party. They don’t want to get to the party venue by car because they had been robbed a few months earlier in that very same place. They are thinking about getting there by public transport also because they take into consideration the fact that they might be a bit drunk at the end of the night and therefore not being able to drive home. Since the day after they have to be home quite early because of a few obligations the morning after, they can’t leave the club too late as there are very few public transports at that time. So they think about their cousin talking them about a really useful application to request Taxis but unfortunately they can’t remember how the app is called. They try to google it and they find MyTaxiService. Sara immediately downloads

the application on her phone and she finds out that they can actually make a reservation. So she fills in the form as follows:

- Leaving from: Via Gino 34
- Destination: Via Fabrique 162
- Reservation: 1st November 5 am
- People: 6
- Taxi sharing: No

After tapping “Subscribe” she receives the message with the Taxi code.

2.4.4. Scenario 4: A passenger makes a sharing request

At 5 p. m. in the afternoon, Fabio is so tired of his bad luck day. The previous night he stayed up all night long studying for his Software Engineering 2 final exam, so he did not set the alarm and he missed the review lesson. Later, in the hurry for getting to the next class on time, he spilled his coffee and forgot the money for lunch. Expecting to get back home soon, Fabio makes a request for a taxi trough the MyTaxiService app, and after filling out his origin and destination he enables the trip-sharing option to save some money. The confirmation message informs that someone has joined his request, and the day totally changed for Fabio when he got into the car and realized that his trip-mate was his Software Engineering teacher. He got an excellent score on his exam, and MyTaxiService got a 5-star service evaluation.

2.4.5. Scenario 5: A taxi driver registers in MyTaxiService

Carlo is a taxi driver who has just become a dad. Unfortunately, he is not working a lot these days since he doesn’t receive many requests. While he is a the Taxi queue area, he hears some guys talking about an application which allows people to request a Taxi without making a call which sometimes takes quite a long time to receive an answer. He also hears that this service is offered by Milano city. He decides to go to the City Hall of Milano to get information about that service. An employee tells him about this new platform which helps people and taxi drivers to increase the efficiency of the taxi service. He then gives Carlo a sheet of paper with all the information and documentation needed to get a register code which he can use to register to the service. What he has to bring is:

- Copy of identity card
- Copy of fiscal code
- Copy of driving license
- Copy of taxi license
- Taxi register
- Telephone number
- Email

A few days later, Carlo goes back to the register office with all the documents and the employee tells him that he will receive an email with an username and the register code he will need to create a personal account on the app. In effect after two days he receives the following information:

- Username: carlosal140581
- Register code: d61ivn784u3

Once he connects to the platform through the official website or the app, he has to choose the option “Sign in” and insert username and register code. By doing so, the system can ask him to create and confirm a password. After that, Carlo will be a Taxi Driver of MyTaxiService.

2.5. Use Cases

The previous informal scenarios are now formalized by means of use cases. Initially, all the use cases are listed through the Use case model, and then they are described individually. The descriptions are in the Excel document *Use case specification*; here we only add the sequence diagrams.

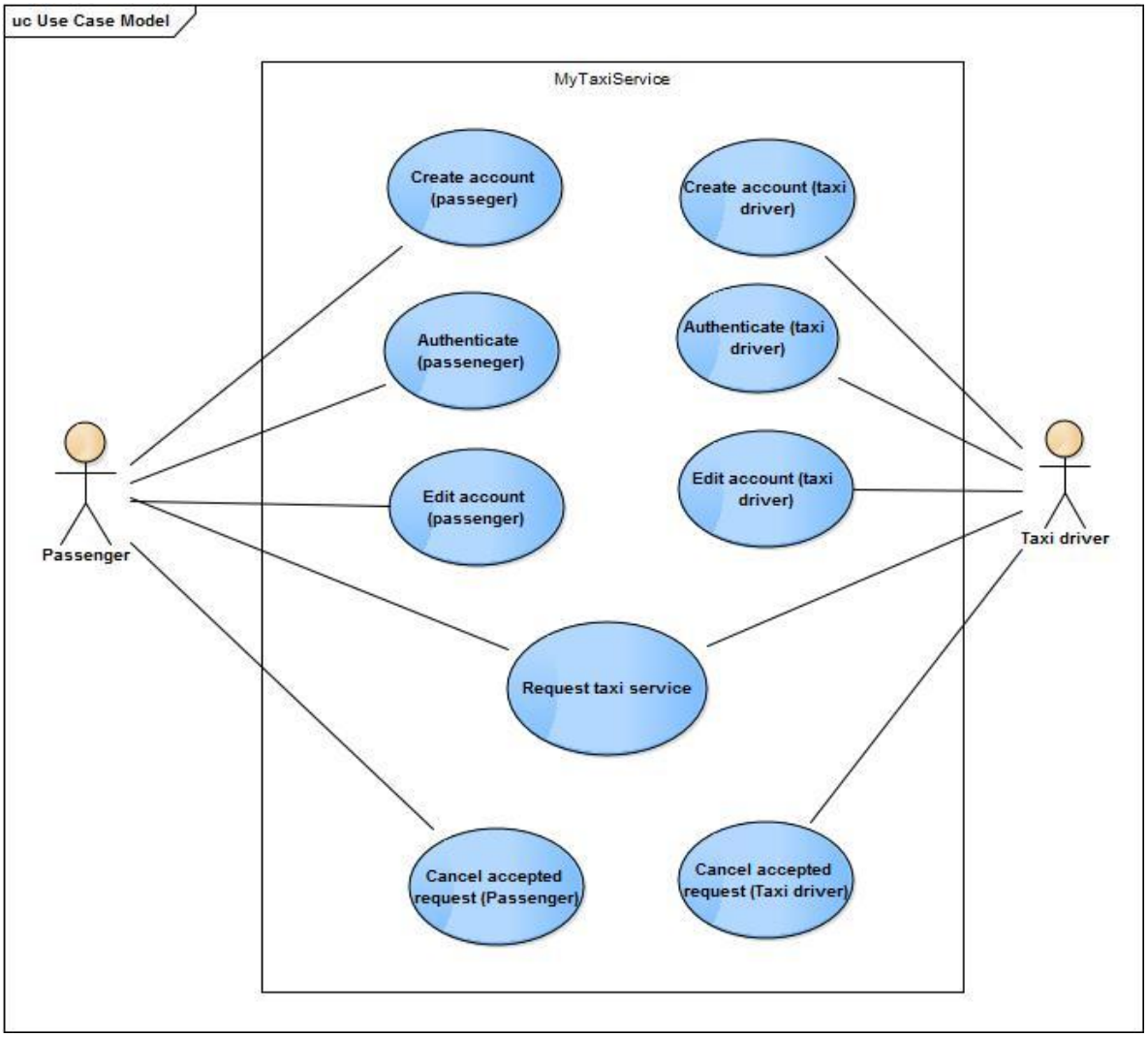

Figure 7: Use case model

2.5.1. Create account (passenger)

This use case illustrates the flow of events related to the creation of an account by a passenger. The *passenger account information* is defined in the section *Definitions, acronyms and abbreviations* of this document. The sequence diagram is similar to Create account (taxi driver), so it is not included.

2.5.2. Authenticate (passenger)

This use case illustrates the flow of events related to the authentication of a passenger. The sequence diagram is similar to Authenticate (taxi driver), so it is not included.

2.5.3. Edit account (passenger)

This use case illustrates the flow of events related to the modification of the *passenger account information* by a passenger. The sequence diagram is similar to Authenticate (taxi driver), so it is not included.

2.5.4. Create account (taxi driver)

This use case illustrates the flow of events related to the creation of an account by a taxi driver. The *taxi driver account information* is defined in the section *Definitions, acronyms and abbreviations* of this document. Note that the validation of the information is done with the data provided by the Milano Government Information System.

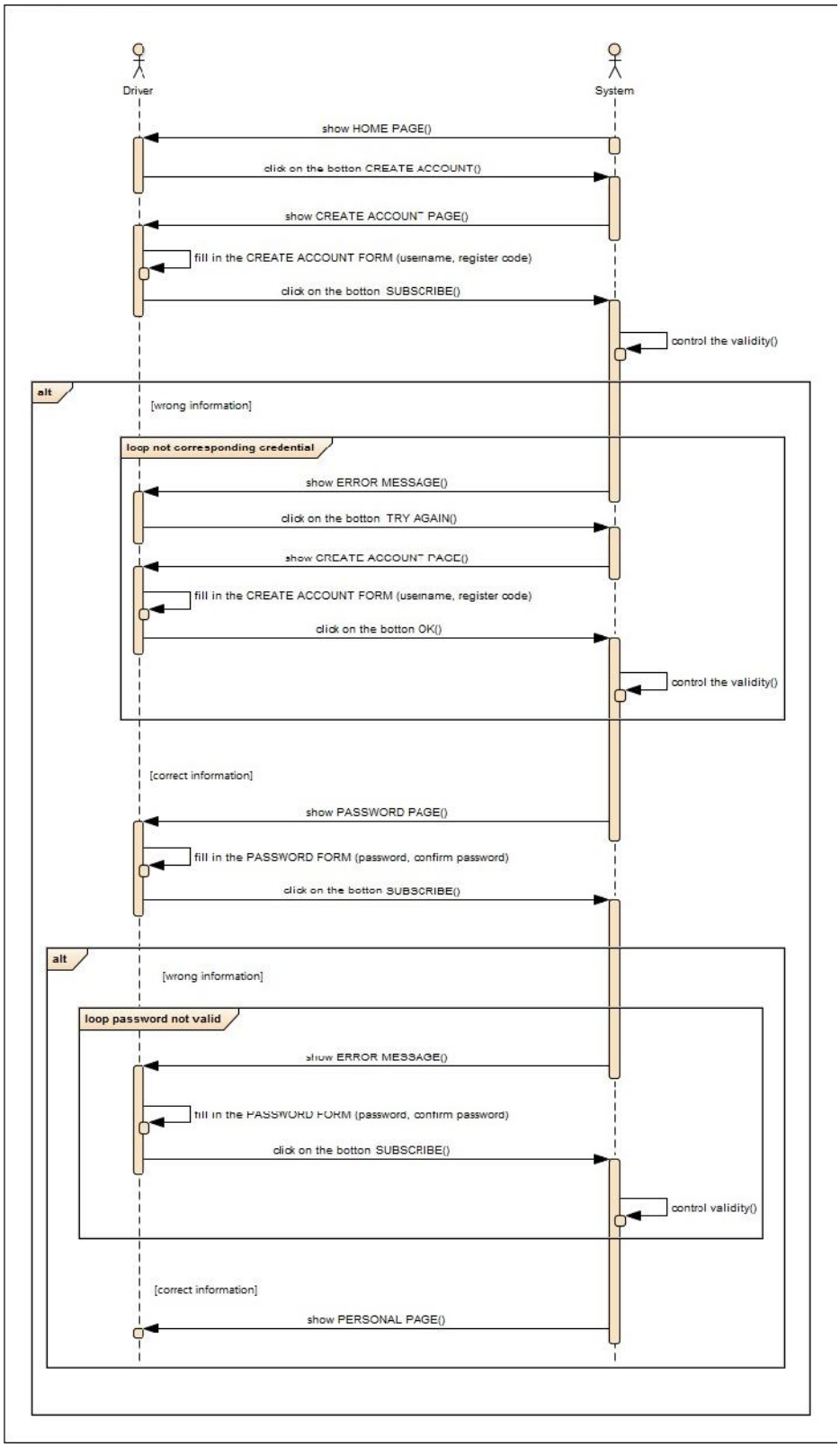

Figure 8: Create Account Use Case

2.5.5. Authenticate (taxi driver)

This use case illustrates the flow of events related to the authentication of a taxi driver.

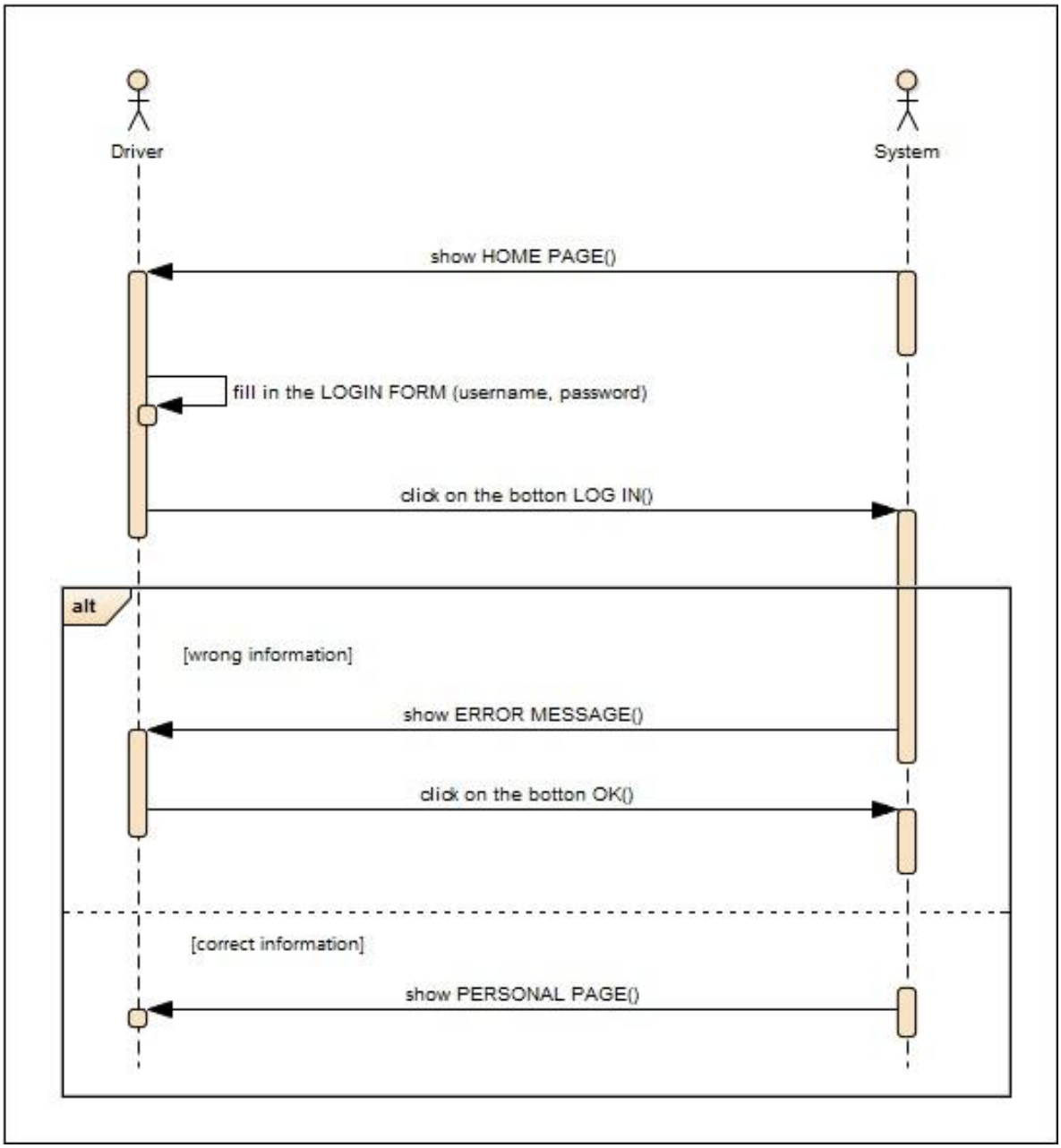

Figure 9: Authenticate Use Case

2.5.6. Edit account (taxi driver)

This use case illustrates the flow of events related to the modification of the *taxi account information* by a taxi driver. Note that the validation of the information is done with the data provided by the Milano Government Information System.

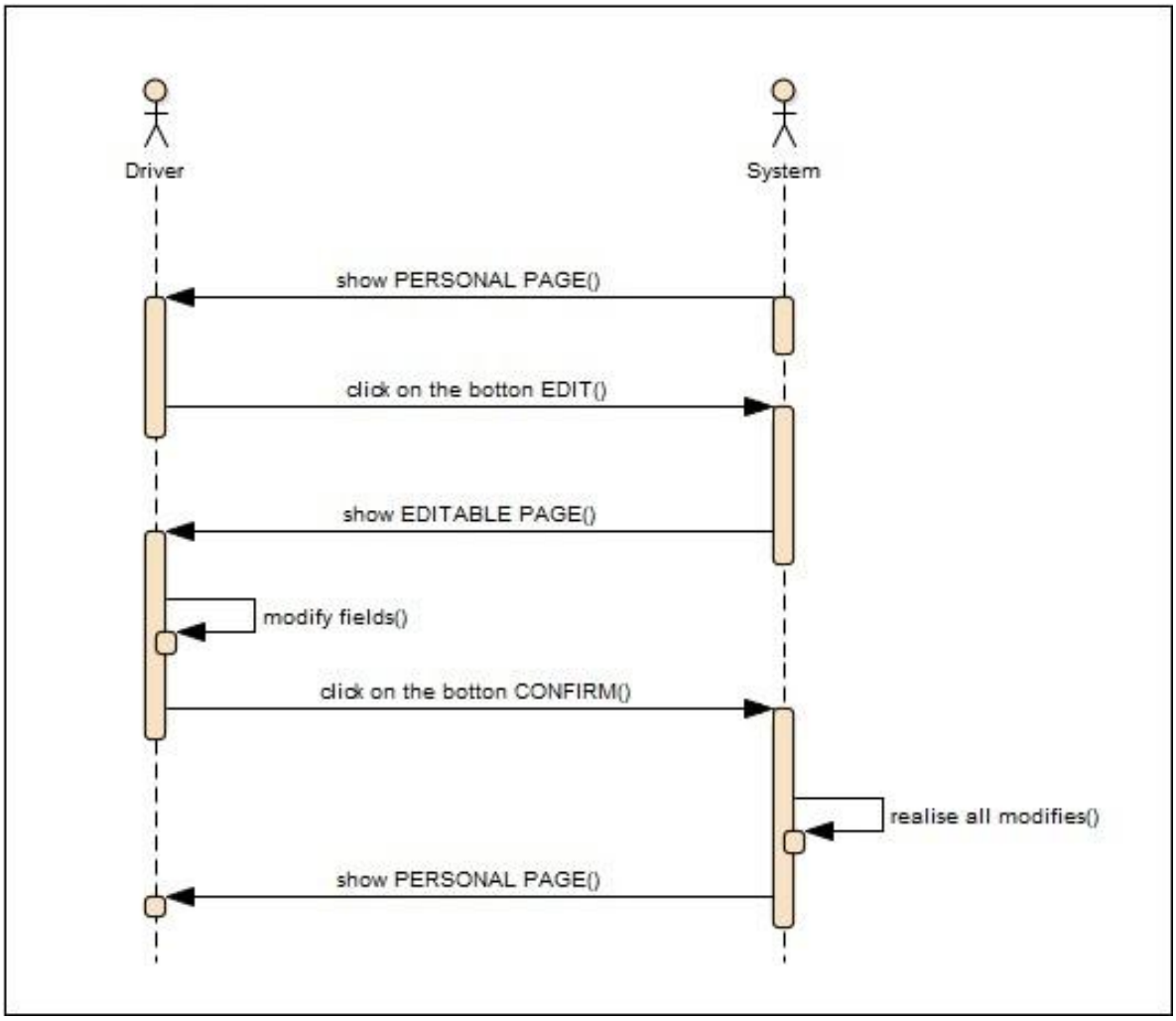

Figure 10: Edit Account Use Case

2.5.7. Request taxi service

This use case represents the flow of events associated to the request of a taxi service. Several comments must be done about this use case.

Note that the validation of the *request for service information* is done through the Maps server. In the case that the origin or destination are addresses, the Maps server resolves them to GPS position.

Here we refer to a general request because the main differences between the request types (immediate/reservation, sharing/not sharing) have to do with the processing times or steps but not with the actions performed by the actors.

To reach a general flow of events which is valid for any request type, the processing of a request in the sixth step is intended to encapsulate the differences. At this point, the

system has valid *request for service information* and wants to produce the *incoming request information* to be sent to taxi drivers. The procedure performed by the system in this moment can happen in either two ways, depending on the *request for service information* received:

- If the passenger does not want to share the taxi, the origin, destination and amount of people are set according to the information provided by the user, and the fee is calculated according to this positions.
- If the passenger wants to share the taxi, the system proceeds as follows:
  1) The system tries to find an already received sharing request which is compatible with the just received one. If it succeeds, the two requests are merged together and, if the total amount of people is not over the maximum (which is four) yet, then the system maintains the requests in expecting to find additional compatible one.
  2) If no additional compatible requests are found within at most three minutes after the first request has been received, then the system creates the corresponding *incoming request information*.
  3) If the total amount has been reached, then the system creates the corresponding *incoming request information*.
  4) If there are not any compatible received requests in the system, the system will keep the request and will try to find a compatible.
  5) If no compatible requests are found within three minutes, the system creates the corresponding *incoming request information.* This means that there might be sharing requests which are sent to taxi driver with no additional compatible requests.

Two requests are compatible if their origin is in the same zone and if the destination of one of them can be reached in the path that goes to the destination of the other one. For deciding this, the sharing engine will ask the Maps server to calculate a shortest possible route from the origin to the destination. Using this, the sharing engine associates to the request all the zones that are reached within a radius of 1 km from any point of the estimated path.

Note that the estimated arrival time is calculated through the Maps server.

The moment of this processing depends on the specified pick-up time. If it is *now*, it takes place immediately; else, it takes place ten minutes before the specified time (this concerns the reservation requests). This means that there is a time gap between steps five and six for the reservation requests.

Once the *incoming request information* is ready, the system tries to find a taxi driver who takes it.

The notification to the passenger is sent to the mobile application or web site if the request is not a reservation. For reservation, the response is sent via email.

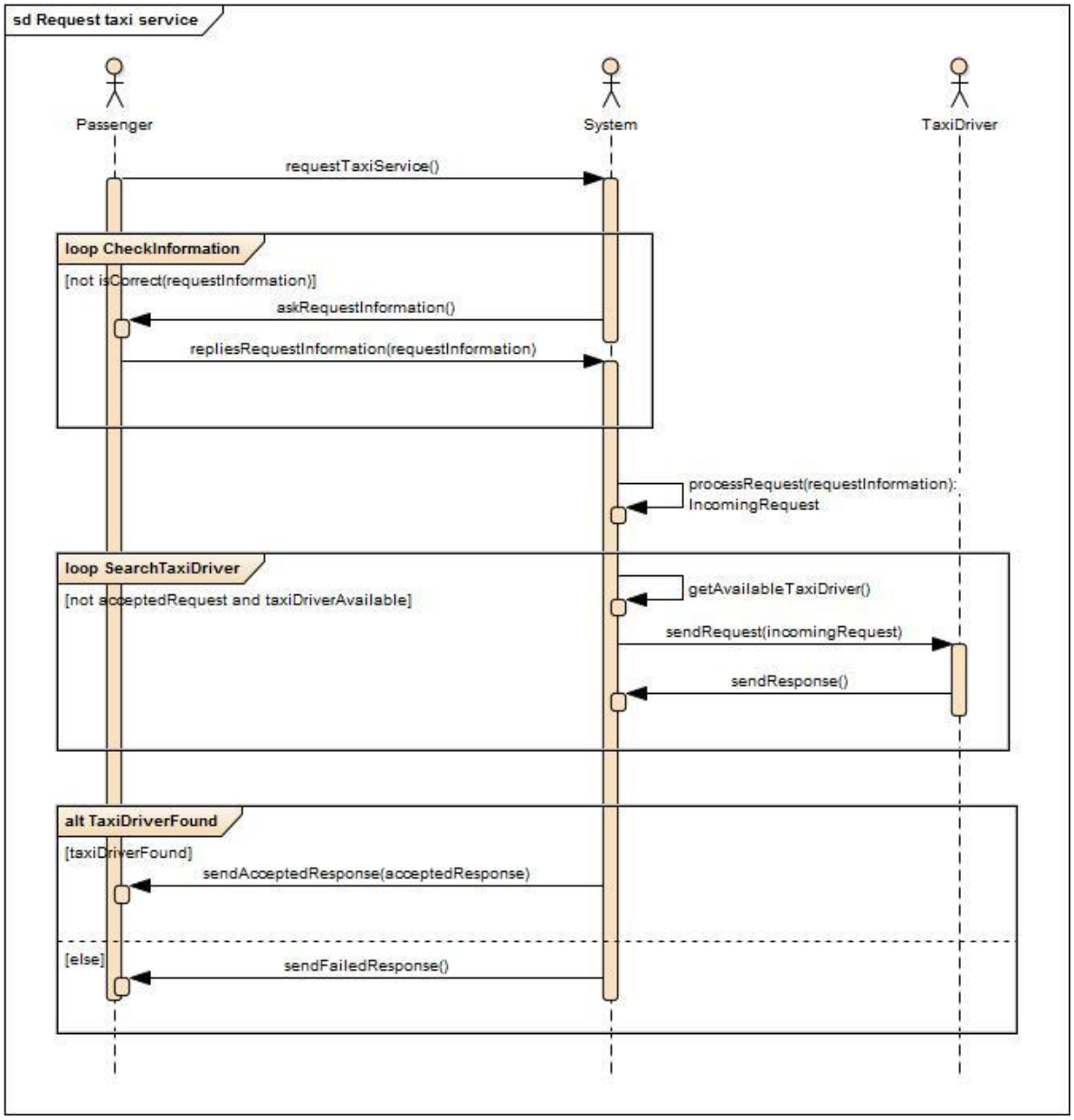

Figure 11: Request taxi service

2.5.8. Cancel accepted request (by the passenger)

This use case illustrates the cancelling of a request by a passenger.

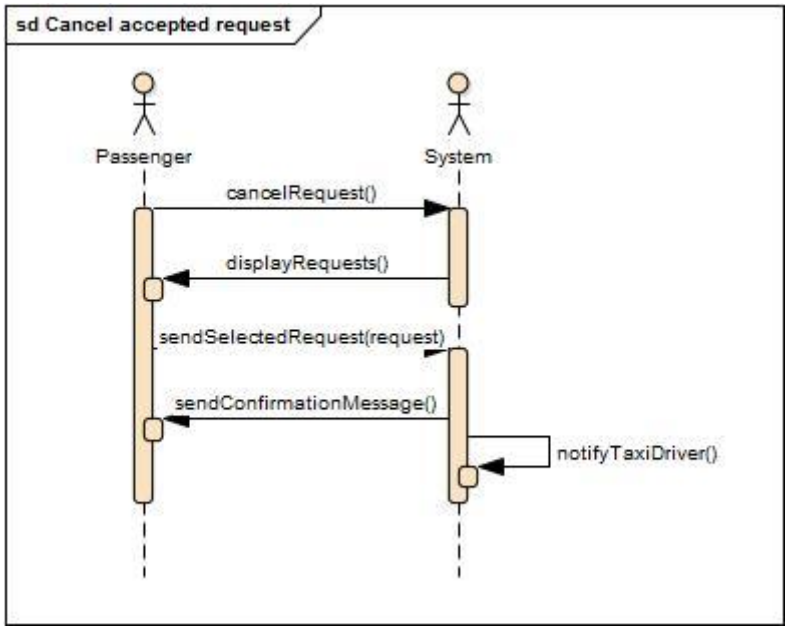

Figure 12: Request taxi service

2.5.9. Cancel accepted request (by the taxi driver)

This use case illustrates the cancelling of a request by a passenger.

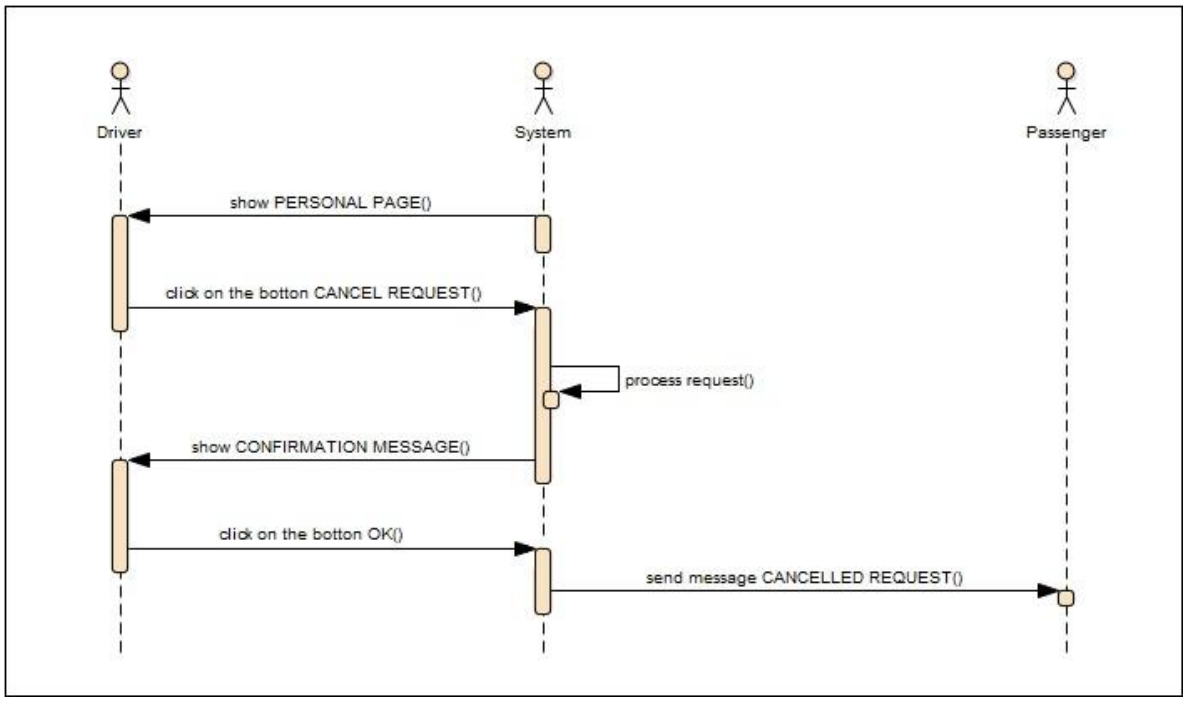

Figure 13: Cancel accepted request (taxi driver) Use Case

2.6. Constraints

In this subsection we describe the constraints that the software product must be developed under.

- **Interfaces to other applications:** MyTaxiService needs to deal with a geolocation application to locate both taxi drivers and users, and it needs to deal with a map application as well in order to determine itineraries.
- **Regulatory policies:** MyTaxiService needs to respect the taxi economical and legal frame, such as ride prices for day, night, week-end or even legal duration taxi drivers are allowed to work consecutively.
- **Taxi fee:** In order to precise the previous point, here is a description of the fee system. The fee calculation depends on two different parameters: the distance and the duration of the ride. Moreover those parameters are depending on the date and the time, for instance the price per kilometer and hour is not the same during the day and the night and neither for a day during school holidays and a regular day.
  Those fees are given by the Milano government and thus have to be respected. Concerning the shared ride, the functioning has to be discussed with the stakeholders but might be based on the regular fees and be split between the passengers.

2.7. Class model

The class model presented below is intended to provide a static view of the interactions that the real-world entities will hold between them, once the MyTaxiService is implemented and deployed. Brief descriptions of such entities and relationships are also given.

- Passenger: It is a user of the service, it might represent one or more future passengers
- Taxi driver: It is a taxi driver that is allowed to use the service by the government
- Government: It is the entity that own the service and propose it to the passengers. It is in charge of deciding which taxi driver is allowed to use the service (it give them a taxi code in a previous registration)
- MyTaxiService: It represents the processing unit of the system
- Milano government information system: It is the data storage unit, providing data bases as well such as the set of allowing taxi drivers
- Request for service: It is the first state of a request, the one send by a passenger
- Incoming request: It is the second state of a request that comes from the processing of one or more requests for service by MyTaxiService and send to a taxi driver
- Accepted request: It is the third state of a request that appears after a taxi driver accept an incoming request

Description of the process: A passenger is able to send a request for service to the MytaxiSercice which will be processed by the system and associated to others in the case of a sharing request. After this process, MytaxiSercice send to a taxi driver an incoming request which corresponds to those requests for service. Then the taxi driver send an answer to

MyTaxiService. Finally, in the case of a positive answer, MyTaxiService send an accepted request to the passengers concerning by the previous requests for service.

We can notice the links between the three request classes.

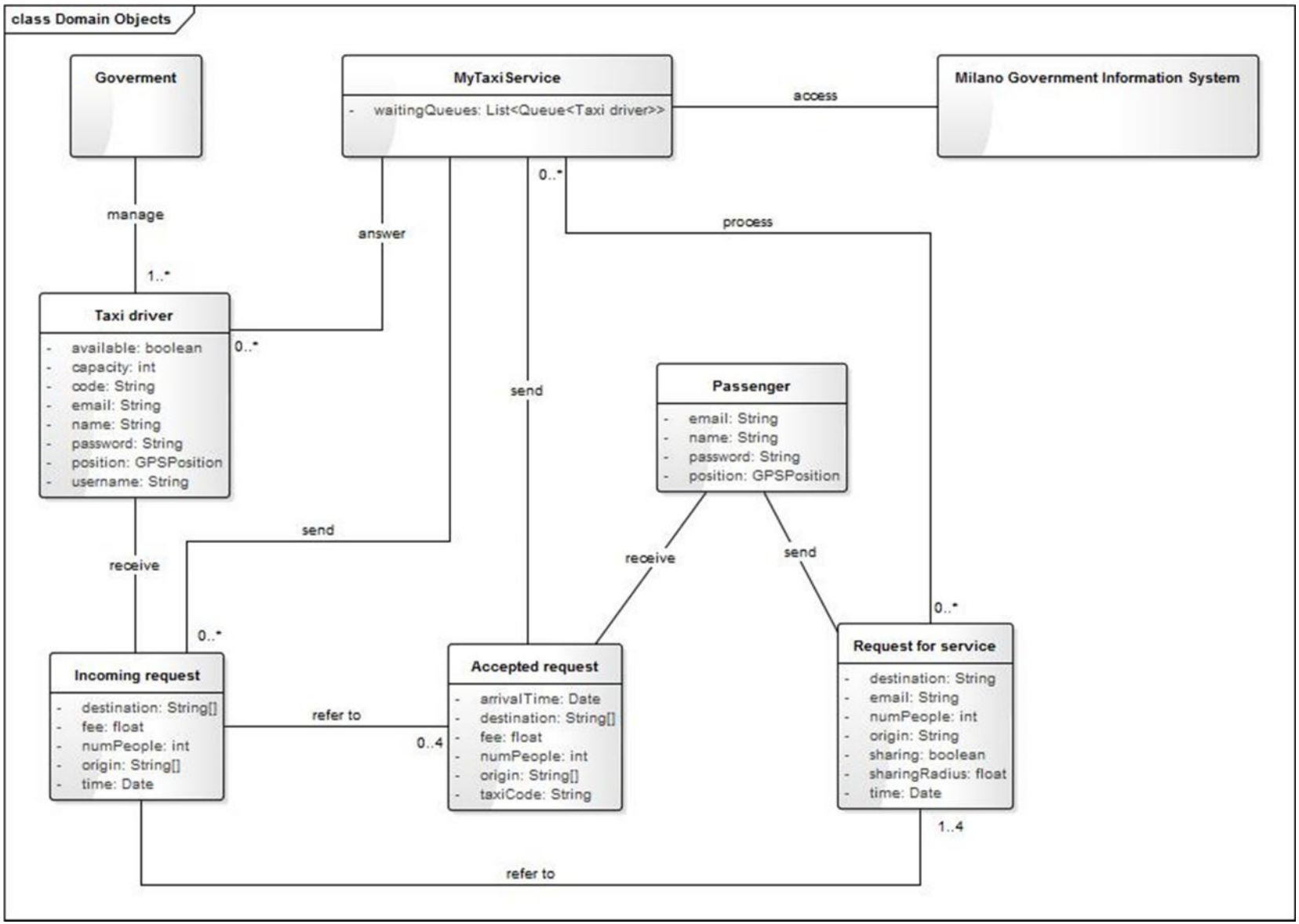
class Domain Objects
Goverment
MyTaxiService
- waitingQueues: List<Queue<Taxi driver>>
access
Milano Government Information System
manage
1..*
answer
0..*
process
Taxi driver
- available: boolean
- capacity: int
- code: String
- email: String
- name: String
- password: String
- position: GPSPosition
- username: String
0..*
send
Passenger
- email: String
- name: String
- password: String
- position: GPSPosition
receive
send
receive
send
0..*
0..*
Incoming request
- destination: String[]
- fee: float
- numPeople: int
- origin: String[]
- time: Date
refer to
0..4
Accepted request
- arrivalTime: Date
- destination: String[]
- fee: float
- numPeople: int
- origin: String[]
- taxiCode: String
Request for service
- destination: String
- email: String
- numPeople: int
- origin: String
- sharing: boolean
- sharingRadius: float
- time: Date
1..4
refer to

Figure 14: Class diagram

2.8. Assumptions and dependencies

In this section we list the assumptions about the world's behavior that, together with the requirements, will lead to the achievements of the goals of the product:

- To verify the identity of passengers it suffices to use a confirmation email message.
- The passengers pay the taxi driver for the service in cash.
- The description provided by the Government about the zones is enough to cover all the valid points of the city, and only points in the city.
- The requests for services that passengers make by means of phone calls are not included as requests in the system.
- Both users (passenger and taxi driver) edit their data whenever it is necessary (change of email address, taxi driver buys a new taxi, etc.). Taxi driver update their information to the Milano Government by themselves.
- The Maps server is capable of recognize the valid and invalid GPS positions given by the users.
- In case that the taxi driver has picked-up a passenger and he is unable to finish the ride, the driver is responsible of getting another taxi driver to finish the ride.
- The following items have already been agreed by taxi drivers, passengers and the government:
  - Method of calculation of the fee
  - Timeouts for driver's response (thirty seconds) and searching (three minutes)
  - Taxi driver will try to take the shortest path to make the rides
  - Maximum amount of people in the taxi (four people)
  - Minimum time for making a reservation (two hours)
- The position of the taxi drivers is updated every 30 seconds.
- Not too many requests are received so that a canceled sharing request (by a passenger) will not introduce long waits for the other affected passengers.
- The taxi driver informs that a trip has ended by saying that he is available again.
- For every accepted request, either the taxi driver arrives at the estimated arrival time or at most five minutes later, or he cancels the request.
- The taxi driver and the passenger do not have to communicate to each other before the pick-up, besides for what concerns the request (sending, receiving, or canceling).
- Every taxi driver which wants to register to MyTaxiService shall previously have gone to the Milano City Hall with the following documents:
  - Copy of identity card
  - Copy of fiscal code
  - Copy of driving license
  - Copy of taxi license
  - Taxi register
  - Telephone number

- Email

After that, he should have received an email with the code and username that will allow him to create the account.

3. Specific requirements

This section presents the specific requirements for the MyTaxiSerive software product, which are grouped according to the goal that each one contributes to achieve. Later we list the non-functional requirements.

3.1. G1: Passenger can request a taxi either through a web application or a mobile application

R1: The system must be accessible by the passengers through the website and the mobile application

R2: The system must allow passengers to create an account (refer to section 1.3, *passenger account information*)

R3: The system must allow passengers who have confirmed their email address to log in

R4: The system must allow passengers to edit their account information (refer to section 1.3, *passenger account information*)

R5: The system must provide a form in order to allow passengers to make a request (refer to section 1.3, *request for service information*)

R6: The system must allow the passenger to check the current state of his request (phase of processing, driver's location, *accepted request information*)

R7: The system must allow passengers to cancel a request

R8: The system must inform the taxi driver (and the other passengers if necessary) when he receives a cancelation

R9: The system must delete the *request for service information* and *accepted request information,* ten minutes after the estimated arrival time

R10: The system must not allow any passenger to make more than one request

R11: The system must not allow anonymous passengers to make requests

3.2. G2: Taxi driver informs the system about his/her availability

R1: The system must be accessible by the taxi drivers through a mobile application only

R2: Taxi drivers must be able to register to the application (refer to section 1.3, *taxi driver account information*)

R3: The system must allow registered taxi drivers to login

R4: The system must allow taxi drivers to edit their account information (refer to section 1.3, *taxi driver account information*)

R5: The system must provide an availability section allowing drivers to change their status

R6: The system must not allow unregistered taxi drivers to receive requests

R7: The system must delete the *incoming request information* when the taxi driver says he is available

3.3. G3: Taxi driver may confirm that he/she is taking care of a certain received request only through a mobile application

R1: When a taxi driver receives an incoming request, the system must allow him to see the *incoming request information*

R2: The system must allow taxi drivers to accept an incoming request

R3: When a taxi driver accepts an incoming request, the system must change is status to not available

R4: When a taxi driver accepts an incoming request, the system must send to the passenger the *accepted request information*

R5: When a taxi driver has accepted an incoming request, the system must allow him to see in a map his current location and the passenger(s) origin(s) and destination(s)

R6: The system must allow taxi drivers to decline an incoming request

R7: When a taxi driver declines an incoming request, the system must resend it to the next available taxi driver

R8: When no available taxi drivers are found, the system must cancel the request and inform it to the passenger(s)

R9: The system must allow taxi drivers to cancel an already accepted request

R10: When a taxi driver cancels an already accepted request, the system must inform it to the passenger

R11: When a passenger cancels a sharing request, the system must inform it to the passengers

3.4. G4: Requests for taxi services are fairly managed

R1: The system must be able to recognize the requests with correct *request for service information*

R2: The system must be able to process the recognized requests

R3: The system must use zone queues to manage available taxi drivers

R4: The system must send a processed request only to the first available taxi driver in the zone queue that corresponds to the origin of the request

R5: The system must use the taxi's GPS to know the position of the driver

R6: The system must use the Map service to assign a zone queue to the driver according to its position

R7: When the system has identified the zone of a just available taxi driver, it must send him to the bottom of the zone queue

R8: The system must refresh the zone of every available taxi driver every 30 seconds

R9: When an available taxi driver changes his zone, the system must send him to the bottom of the zone queue

R10: The system must send to the bottom of the zone queue those taxi drivers who decline a received request

R11: The system must send to the bottom of the zone queue those taxi drivers who cancel an already accepted request

R12: The system must send to the top of the zone queue those taxi drivers that have received a cancelation for an already accepted request

R13: The system must remove from of the zone queue those taxi drivers that change their status to not available

R14: The system must send to the bottom of the zone those taxi driver that do not responds to an incoming request after 30 seconds of receiving it

3.5. G5: Passengers can enable a taxi sharing option

R1: The system must provide a form in order to allow passengers to make a request (refer to 1.3 section, request for service information)

R2: The option "Taxi sharing" will be effectively activated only if there will be other people who have requested it in the same area at the same time

R3: Passenger will receive a notification by the system in which there will also be specified if the taxi sharing option has been effectively accepted

3.6. G6: Passengers can reserve taxi service in advance

R1: The system must provide a form in order to allow passengers to make a request (refer to 1.3 section, request for service information)

R2: The reservation must be made by the passenger at least 2 hours before time of departure

R3: The system has to confirm the reservation to the passenger

R4: The system allocates a taxi 10 minutes before the departure time in the requested place

3.7. Non-functional requirements

The following are the non-functional requirements of MyTaxiService:

- The system must provide an open interface to allow independent developers to add further features on the top of the application
- The system must be able to process five hundred requests per minute
- The system must have an availability of 99%
- After a failure, the system must recover before five minutes
- The system must be designed so that the users are able to use it after no more than one hour of training

3.8. Non-requirements

In order to complete the description of the software provided by the previous requirements, the items below represent potential requirements that our system will not deal with (some of them have been already mentioned in previous parts of the document:

- The system will not verify the passenger's identity with something else than his email address
- The system will not support the payment of the rides.
- The system will not have additional types of users beyond the passengers and taxi drivers.
- The system will not deal with requests that have been done using a phone call.

4. Alloy Modeling

This section presents the Alloy model of the current problem. The source coded is in the *MyTaxiService* allow file. We only list here what it is written in that file.

4.1. Entities

These are the entities used in the model (some attributes where omitted to simplify it):

- Zone: zone of the city
- TaxiDriver
- Passenger
- RqstForSrv: a request for service
- IncomingRqst: an incoming request
- AcceptedRqst: an accepted request
- MyTaxiService: the MyTaxiService system

4.2. Facts

The facts that determine the behavior of the world are:

- Every incoming or accepted request belongs to some MyTaxiService
- Every request for service belong to exactly one passenger and at most one incoming request
- Every accepted request belongs to exactly one incoming request and exactly one passenger
- Every incoming request belongs to at most one taxi driver
- If an incoming request is accepted, it generates as much accepted requests as requests for service, and has been taken by one taxi driver
- If a driver has an incoming request and has not accepted then he is available
- If a driver has accepted a request he is no longer available
- If a driver has an incoming request, he is in the same zone of one of its origins
- Every accepted requests belongs to the same MyTaxiServices of its incoming request
- Every incoming request belongs to the same MyTaxiServices of its requests for service
- Every accepted request is related to exactly one request for service through the passenger
- Not sharing incoming requests have only one request for service

4.3. Predicates

The following predicates model the actions that can be performed in the system:

- Creation of a requests
- Sending a request
- Processing a request

- Receiving an incoming request by the driver
- A driver accepts a request

4.4. Assertions

The following assertions verify the expected properties of the model:

- All sent requests were created
- All the incoming requests where processed
- All requests are sent to a driver in the same zone
- Not sharing incoming requests only have one request for service
- Requests are zone-coherent

4.5. Generated world

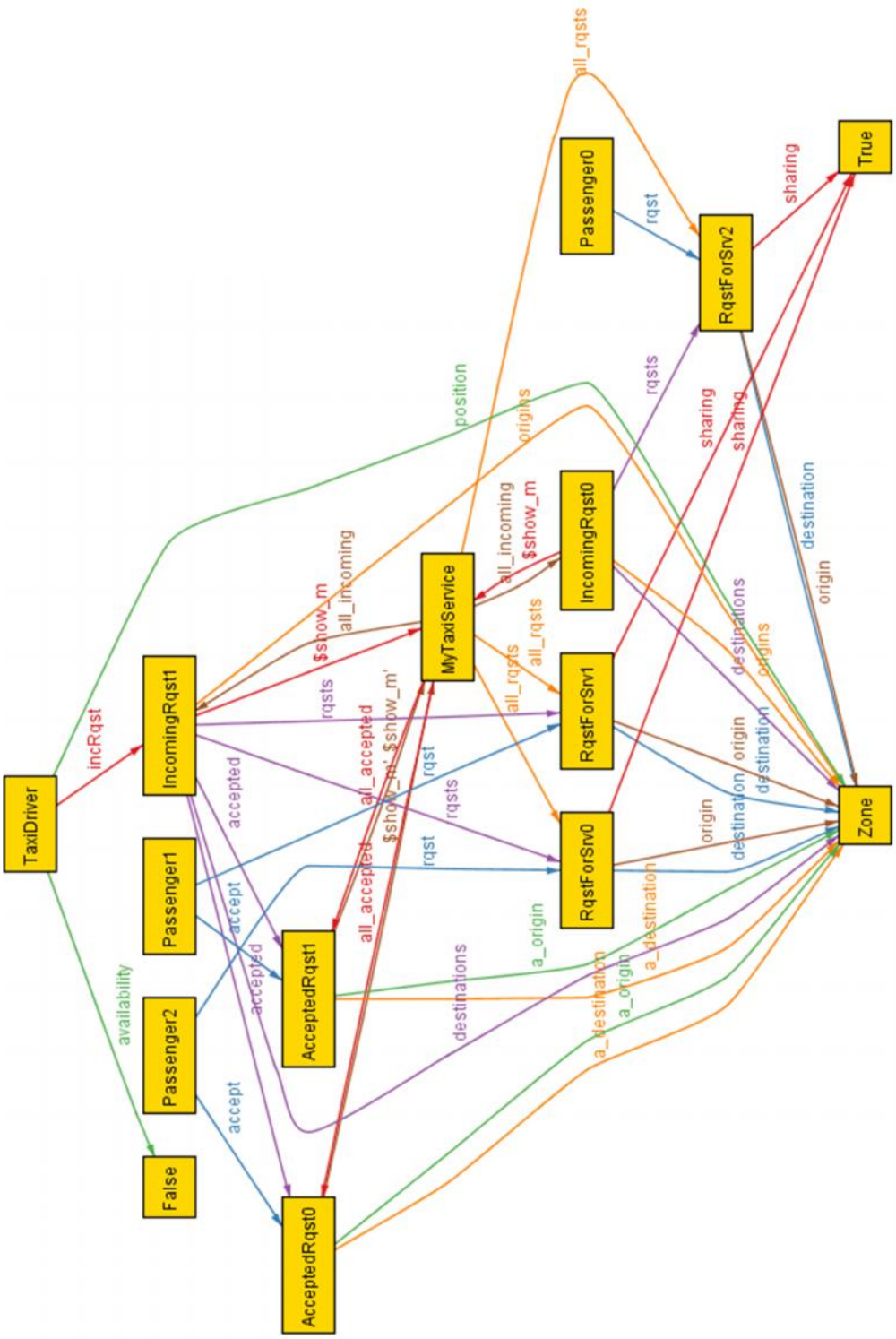

Figure 15: Alloy generated world 1

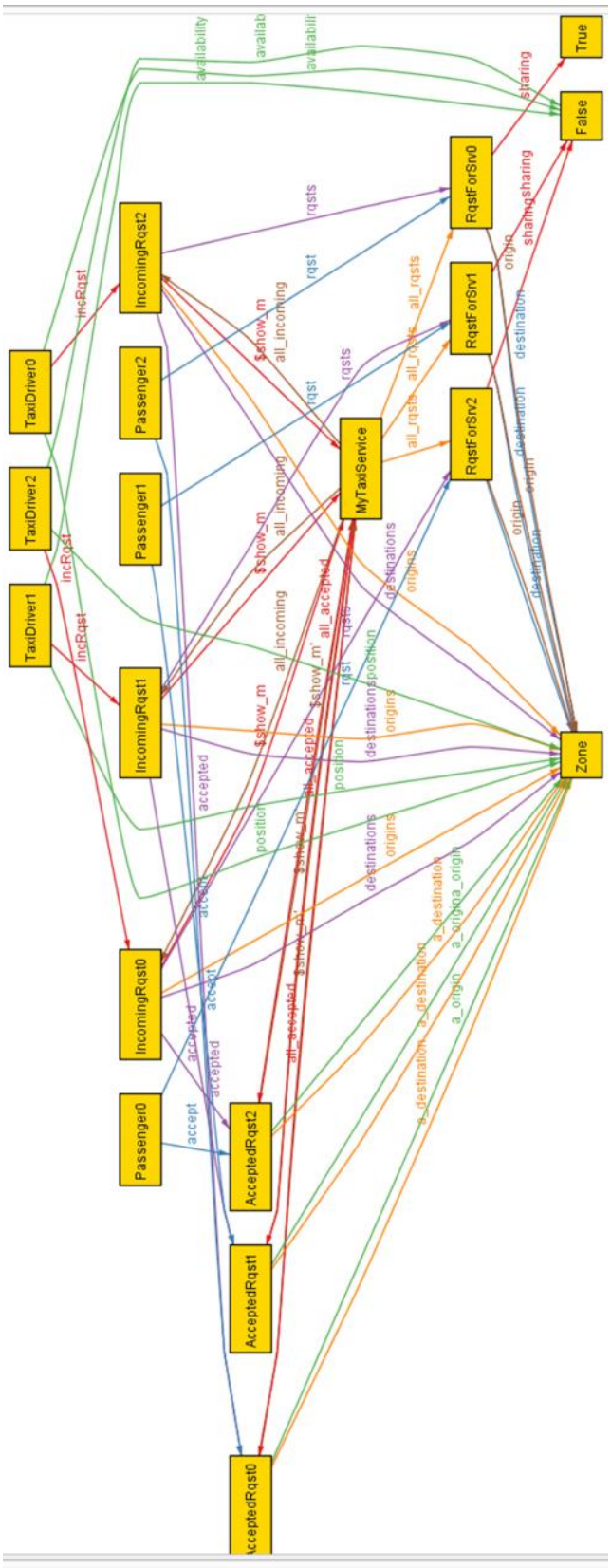

Figure 16: Alloy generated world 2

5. Appendix

5.1. Used software

For the developing of this documents, the following software tools were used:

- Microsoft Word: as the text editor
- Microsoft Excel: as the editor for some tables
- Enterprise architect: as the UML modeling tool
- Github: ad the repository and configuration manager tool

5.2. Worked hours

For each one of the authors of this document, the following is the approximated amount fo worked hours:

- Ivana Salerno: 27
- Alexis Rougnant: 31
- Daniel Vacca: 31

5.3. Revisions

In this section we list the most relevant changes that were done in the document after the first delivery (version 0.4 on November 4th):

- Further explanations on the use case Request taxi service, concerning the processing of the request. This includes the specification of the calculation of the fee.
- Several domain assumptions were added in order to clarify general aspects of the domain problem.
- Non-functional requirements were added.