# Change Log

# Preface

# Table of Contents

# List of Tables

# List of Illustrations

1. Introduction
   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. 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*.

After the trip is over, the passenger is asked to evaluate the driver’s service. This information will be used by the Milano’s government to improve the services.

The requests are managed and assigned to available drivers, according to the position provided by their taxi’s GPS. 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. 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.

**Built of the *incoming request information***: the *incoming request information* is built based on the unattended requests. When the system process a request, it may face two cases:

* The passenger does not want to share the taxi, so 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.
* The passenger wants to share the taxi, so the system tries to find up to three compatible requests within the established *searching time*. If it succeeds, then the origins, destinations, fee and amount of passengers are set accordingly. Else,

Compatible request:

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.

Request for service information: corresponds to the following information: the origin and destination of the trip, **the amount of people**, whether he wants to share or not the taxi. 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 places in Milano. This is considered to be complete if all fields have been filled in.

Searching time:

* 1. References
  2. Overview

1. Overall description
   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.

* + 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, we only name a small group of them:

* 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

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

* A passenger requests a taxi service (includes if he wants to share it or not)
* 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 an accepted request
* A passenger evaluates the taxi driver’s service

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

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 service evaluation is stored

### 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 |
| Taxi’s GPS | Provided | MyTaxiService will be able to know the current position of each taxi through their installed GPS. This will be useful to detect the zone in which the driver is located and assign him the corresponding requests. |
| Maps server | Provided | MyTaxiService will use the maps service to perform three tasks:   * Resolve the passenger provided address to actual coordinates * Assign the driver to a zone based on his GPS position * Display a map to both the driver and passenger with the driver’s and the pick-up position |

### 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:

* Make a request for a taxi service
* 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
* Be notified about the driver’s arrival
* Cancel an accepted request
* Evaluate the taxi driver

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:

* Inform his availability
* View received requests
* Accept or decline received requests
* View his own position and the pick-up point’s position
* Notify his arrival

We do not include here any description of the actual graphic user interface since it does not concern the requirements process.

### Operations

According to the previous section, MyTaxiService will support access only to two type of users. They will interact with the system only under the condition of users, so no additional operation modes will be available (like administrator or “super-user”). Such user condition was already described in that section.

* 1. Product functions – Goals

In this section we expose the product functions by listing the desired goals:

* G1: Passenger can request a taxi either through a web application or a mobile app.
* G2: Passenger receives the confirmation after his/her request has been sent and accepted by some taxi driver.
* G3: Passenger is notified when the taxi arrives at his/her location.
* G4: Taxi driver informs the system about his/her availability.
* G5: Taxi driver receives requests for services.
* G6: Taxi driver may confirm that he/she is taking care of a certain received request.
* G7: Taxi driver receives the passenger’s information when he/she accepts the request.
* G8: Requests for taxi services are fairly managed.
* G9: Passengers can enable a taxi sharing option.
  1. Stakeholders, users and actors
     1. Stakeholders
     2. Users and actors
* Passenger: person who makes use of the MyTaxiService to make a request for a taxi service. He does not have to perform a log-in into the system to make requests. He can send requests either through the web site or the mobile application.
* Taxi driver: person who makes use of the MyTaxiService to attend 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.
  1. Scenarios
  2. Use Cases
  3. Constraints
  4. Class model
  5. Assumptions and dependencies

Not login.

Radius to pick up and destination.

1. Specific requirements
   1. Functional requirements
   2. Non-functional requirements
2. Alloy Modeling
   1. Entities
   2. Facts
   3. Predicates
   4. Functions
   5. Asserts
   6. Generated world
3. Appendix
   1. Used software
   2. Worked hours
   3. Revisions