Taxi Company ‘Ola Cab’

Object Oriented System Analysis

Opgesteld door:

Cedric Hermans	R0449493
Dries Hugaerts	R0629197
Nathan Olmanst	R0594509

Table of content

Contract

Employees and role description:

- Hugaerts Dries: Junior Project Manager
- Hermans Cedric: Senior Project Consultant
- Olmanst Nathan: Junior Project Consultant

Terms and conditions:

1. Attendance in both theoretical and practical courses is mandatory.
2. Theoretical and practical courses have to be followed thoroughly, same goes for any group work done from home.
3. Working from home will be done using Skype.
4. Group work at home has to be taken seriously.
5. All deadlines have to be respected.
6. Communication will be done using Facebook's messenger application.
7. Project documents will be shared through google drive.

In case of Illness and or absence:

8. If possible, when sick, group work has to be done from home using Skype
9. If not able to work from home with Skype, the absence will have to be justified with a medical certificate.
10. Absence or lateness due to traffic, bad weather conditions, strikes or any other inconveniences can be tolerated up to a maximum of 5 times.

Problem Description

Introduction

Jayesh Brahmbhatt is an employee of a taxi company, named ‘Ola Cabs’, located in Brussels. He contacted our consultancy to work out an idea for their new taxi management system. Momentarily they use the phone network to organize the work, but they want a software system to ease the process. We had a meeting where he explained his company’s needs and made a summary of it.

Project summary

The goal of this taxi management system is to monitor all taxis from the company in a central database. The available information in the database includes: the taxi’s current location, the driver’s details and the taxi’s current status (available, in route, picked up, etc.…). This is used to display a live image of any taxi available within the company.

The “live” taxi management system will be made possible due to a GPS/onboard computer that will be implemented in every taxi car. The taxi driver then has to login on this GPS/onboard computer and will be able to indicate his current status to the central server.

This centralized information can then easily be used by a booking agent to then deliver the best possible service for the customer. Once the booking agent picked out the nearest taxi driver, he will assign the route, and communicate any extra information to take into account, using the onboard computer. The driver can then decide to either accept or decline the assignment.

The clients will benefit from this taxi management system by receiving an alert/sms as soon as a taxi is on his way, along with the estimated time of arrival.

Example

As an example, a client calling for a taxi would go as follows:

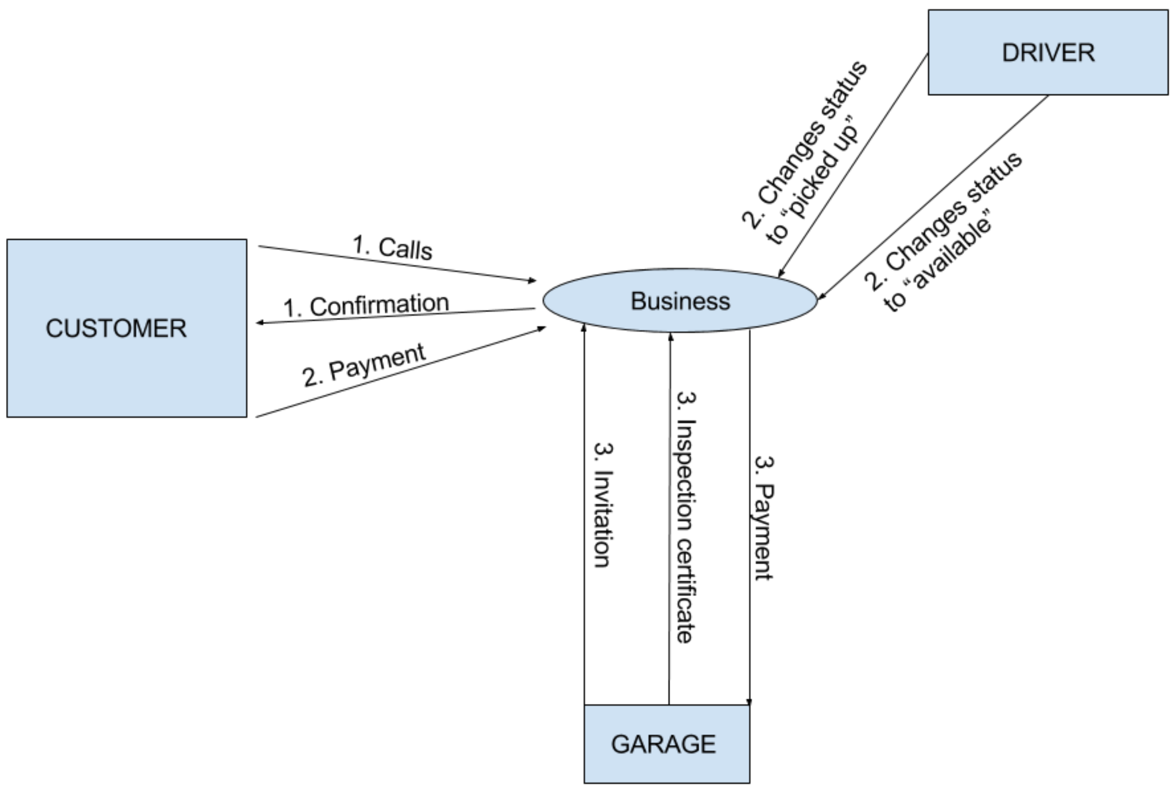
The caller calls the company and gets redirected to a booking agent. From this point onwards, the caller will be able to state his current location, his destination as well as any extra information such as luggage or amount of passengers involved. The booking agent can then easily pick out the nearest taxi car from the database and assign the task. As soon as this is done, an alert can be sent to the caller, informing him that a cab is on its way.

Business analysis

Business events

- Customer checks availability.
- Customer calls cab company.
- Booking agent dispatches a taxi.
- Every month the booking agent creates an invoice for customers.
- Every day the booking agents creates an overview of the scheduled drivers.
- Every morning, the driver consults his daily schedule.
- Driver picks up customer(s).
- Driver arrives at the destination.
- Technician inspects the taxis.

Context diagram


DRIVER
2. Changes status to "picked up"
2. Changes status to "available"
CUSTOMER
1. Calls
1. Confirmation
2. Payment
Business
3. Invitation
3. Inspection certificate
3. Payment
GARAGE

Business use-cases (+short description)

- Customer checks availability:
  When a customer checks the availability of taxi's, he gets an overview of the available taxi's.

- Customer calls cab company:
  When a customer calls the company, he/she is redirected to a booking agent.

- Booking agent dispatches a taxi:
  The booking agent selects a vehicle and assigns a task to it.

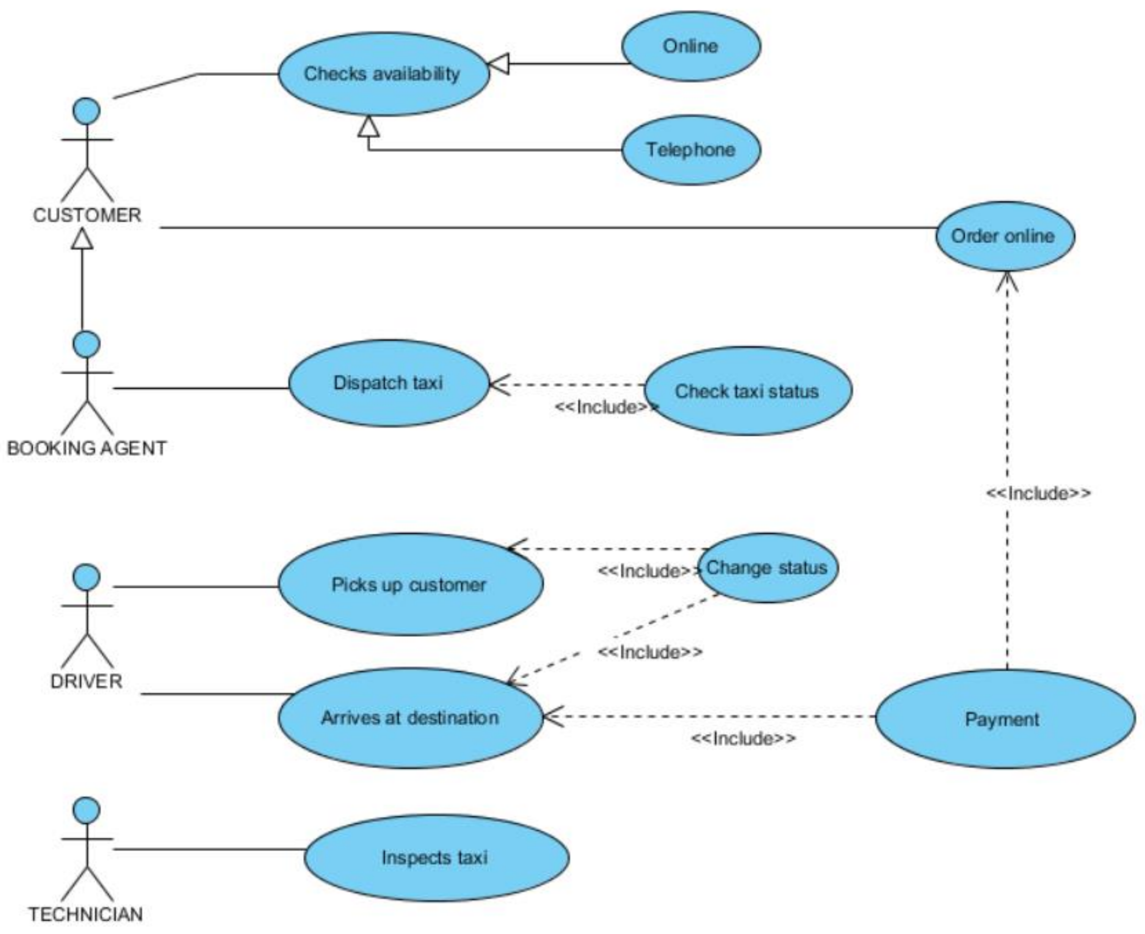
- Driver picks up a customer:
  The driver picks up a customer and changes his/her status to unavailable.

- Driver arrives at destination:
  The driver arrives at the destination requested by the customer.

- Technician inspects vehicles:
  The technician in the company inspects all the vehicles in preparation for the 6 monthly inspection.

Requirements analysis

Use-case diagram



Use-case descriptions

NAME:	**Check availability**
SUMMARY	Customer checks availability of taxi's.
ACTORS	Customer
PRECONDITION	N/A
SCENARIO	browse to website, Go to overview page, Checks available taxis.
EXCEPIONS	N/A
POSTCONDITION	Customer gets an overview of the available taxi's.

NAME:	**Call cab company**
SUMMARY	Customer calls the cab company for a taxi
ACTORS	Customer, booking agent
PRECONDITION	N/A
SCENARIO	Customer calls company. Customer gets redirected to a booking agent. Booking agent asks for info to determine most suited taxi to dispatch.
EXCEPIONS	N/A
POSTCONDITION	Booking agent is dispatching (see use-case 'Dispatch')

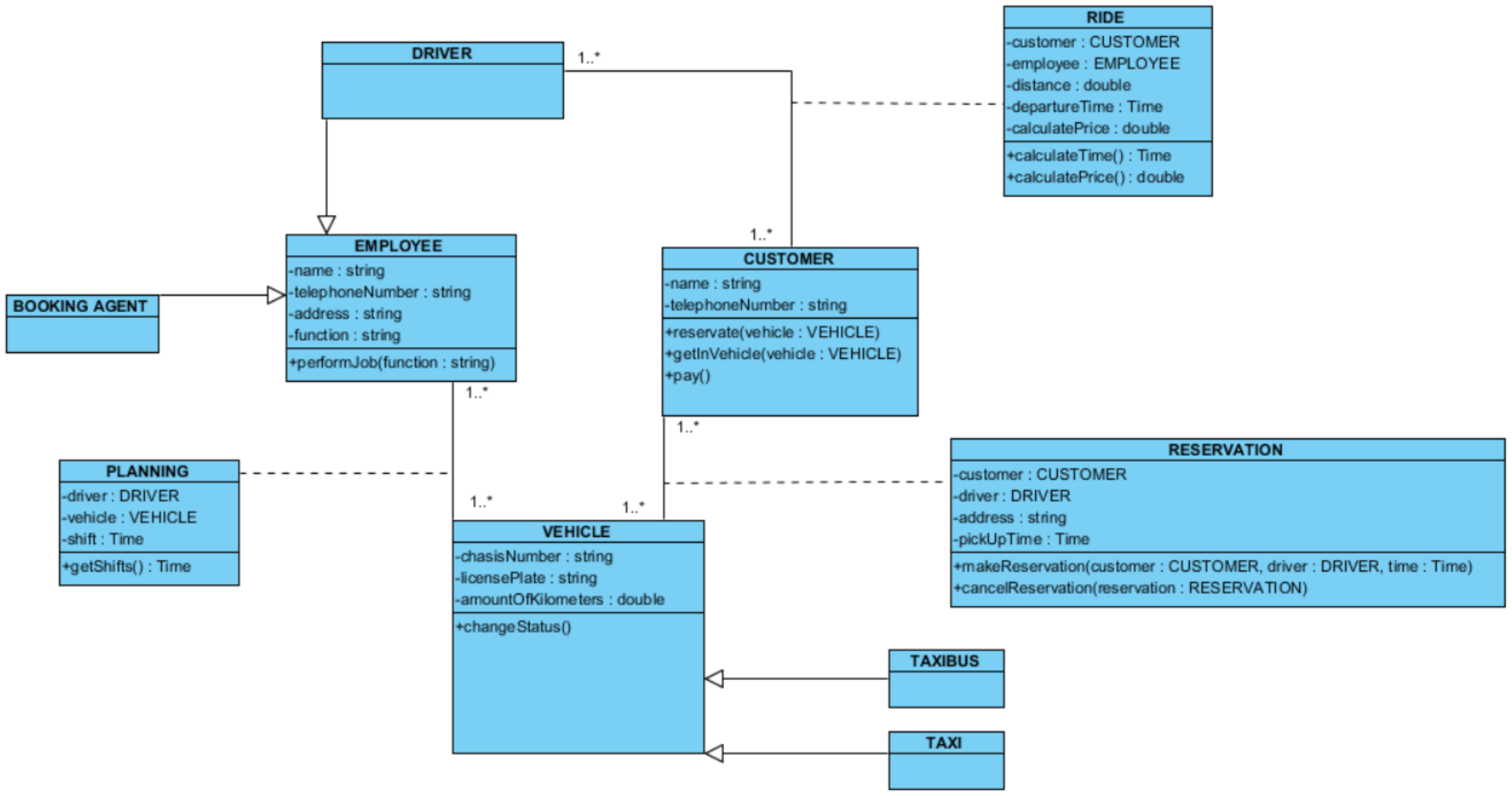
NAME:	**Dispatch**
SUMMARY	Booking Agent selects a cab and assigns a task.
ACTORS	Booking agent
PRECONDITION	N/A
SCENARIO	Booking agent looks for the taxi best suited for the route - if no available taxi -> exception 1 Booking agent contacts the driver and assigns the task - if driver does not respond -> exception 2
EXCEPIONS	Exception 1: - propose to call back later - If customer declines, cancel reservation. Exception 2: - select other available taxi
POSTCONDITION	A taxi will be in route to pick up the customer.

NAME:	**Pick up customer**
SUMMARY	Customer gets into the taxi, driver sets status to unavailable
ACTORS	Driver
PRECONDITION	Customer gets in taxi
SCENARIO	Driver set's status to unavailable
EXCEPIONS	N/A
POSTCONDITION	Customer gets an overview of the available taxi's.

NAME:	**Arrives at destination**
SUMMARY	Driver arrives at the customer's requested end location.
ACTORS	Driver
PRECONDITION	Customer in car
SCENARIO	Customer pay's the requested amount - Customer doesn't want to pay → exception 1 Customer leaves the car - Customer doesn't want to leave → exception 1
EXCEPIONS	Exception 1: - Call cops
POSTCONDITION	Customer paid and Driver is available for a new job

NAME:	**Inspects**
SUMMARY	Technician inspects the car for the 6 monthly inspection.
ACTORS	Technician
PRECONDITION	Invitation for car inspection
SCENARIO	Cars get inspected - Car isn't fit for service → exception 1
EXCEPIONS	Exception 1: - Car goes to garage for maintenance - if not fixable, sell.
POSTCONDITION	Cars 6 monthly inspection is passed

Domain model



Model dictionary

- Employee
  Class defining an employee that works for the company.

- Driver
  Subtype of class Employee delegated to driving cars

- Booking Agent

- Subtype of class Employee delegated to answering the phone and dispatching a taxi

- Customer
  The Person who makes the order for a taxi, by phone or online.

- Vehicle
  Are the cars used by the company as taxi, driven by the “drivers”

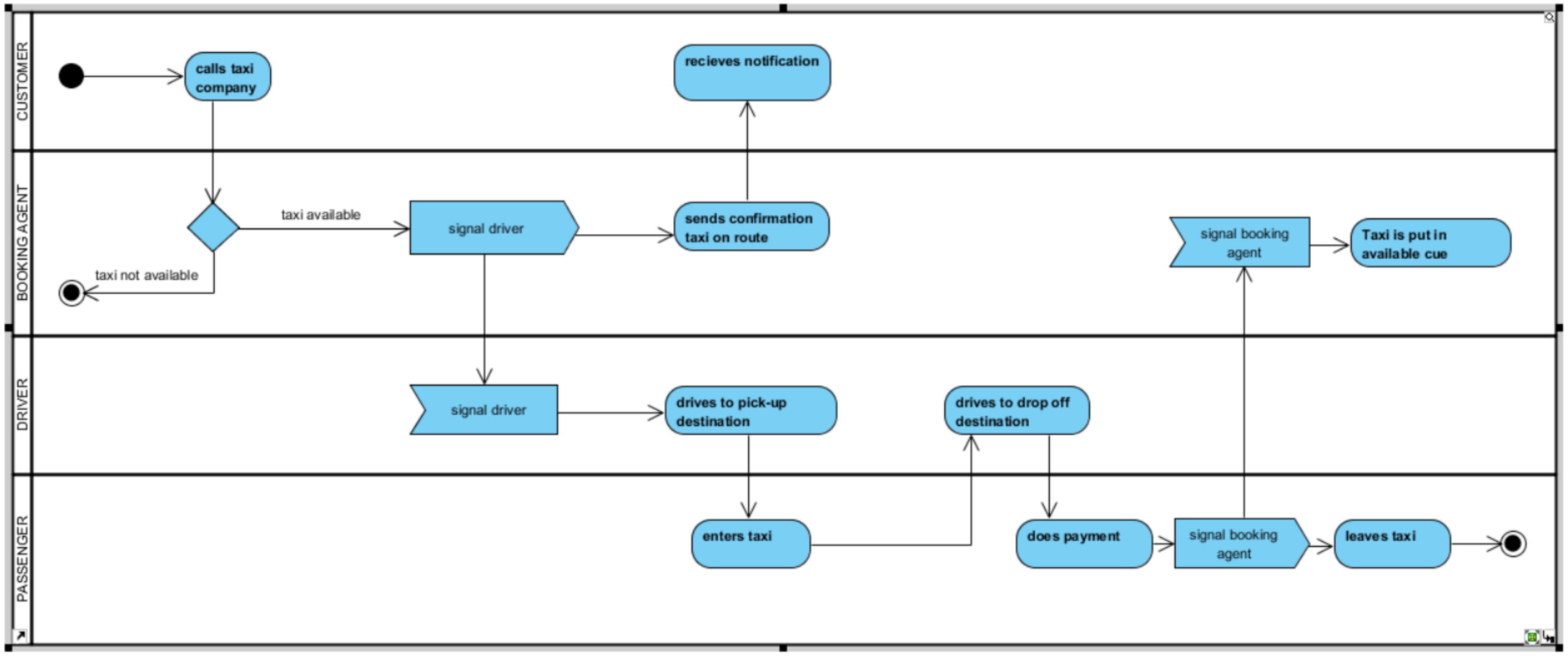
- Ride
  Gives detailed information on the ride from the moment the customer boards the taxi.

- Planning
  A detailed overview that gives the schedule of the driver. E.g. Which driver drives what car for that time.

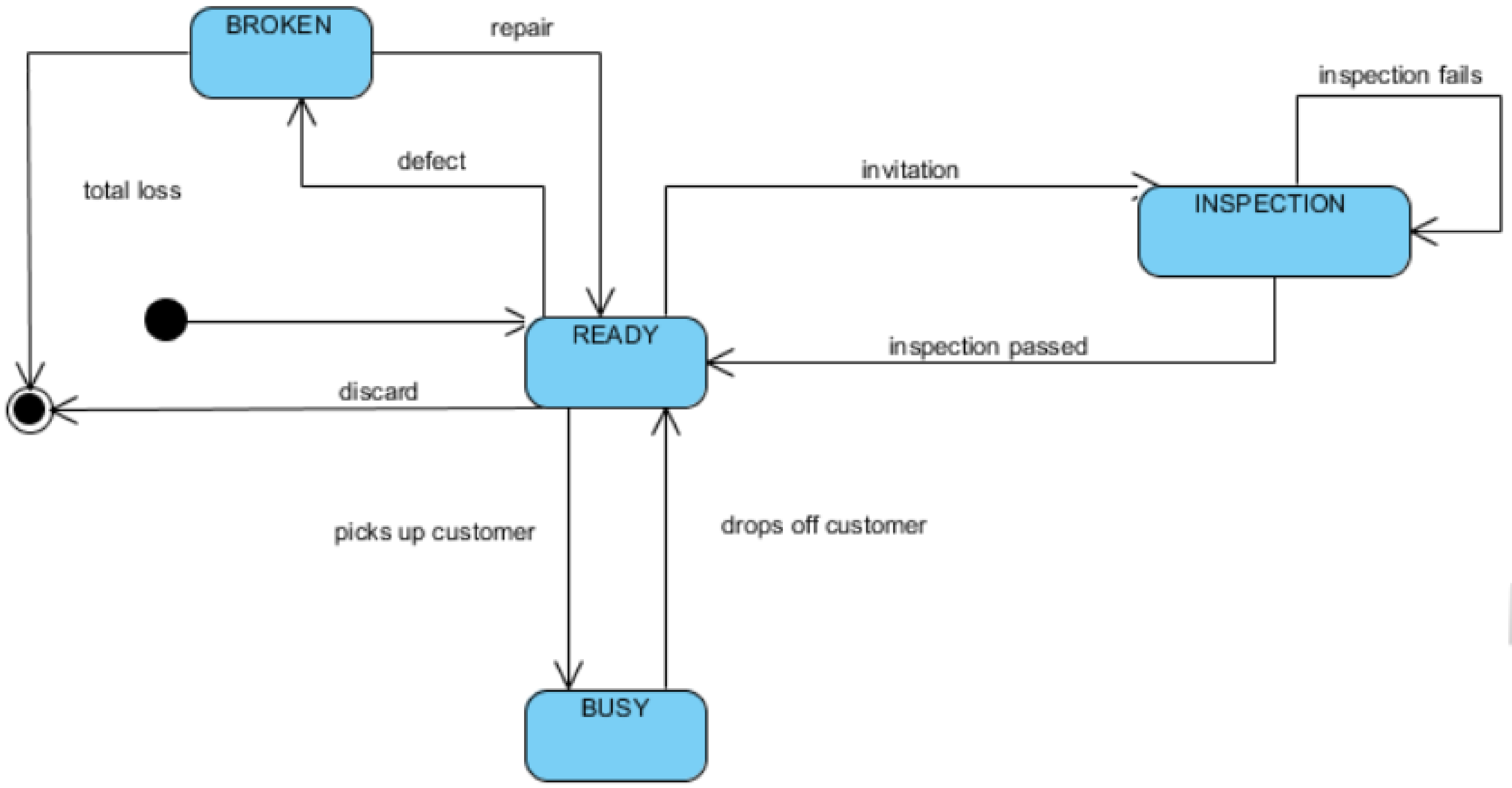
- Reservation
  A customer or booking agent can make a reservation for a taxi.

Dynamic layer- activity-and state diagrams

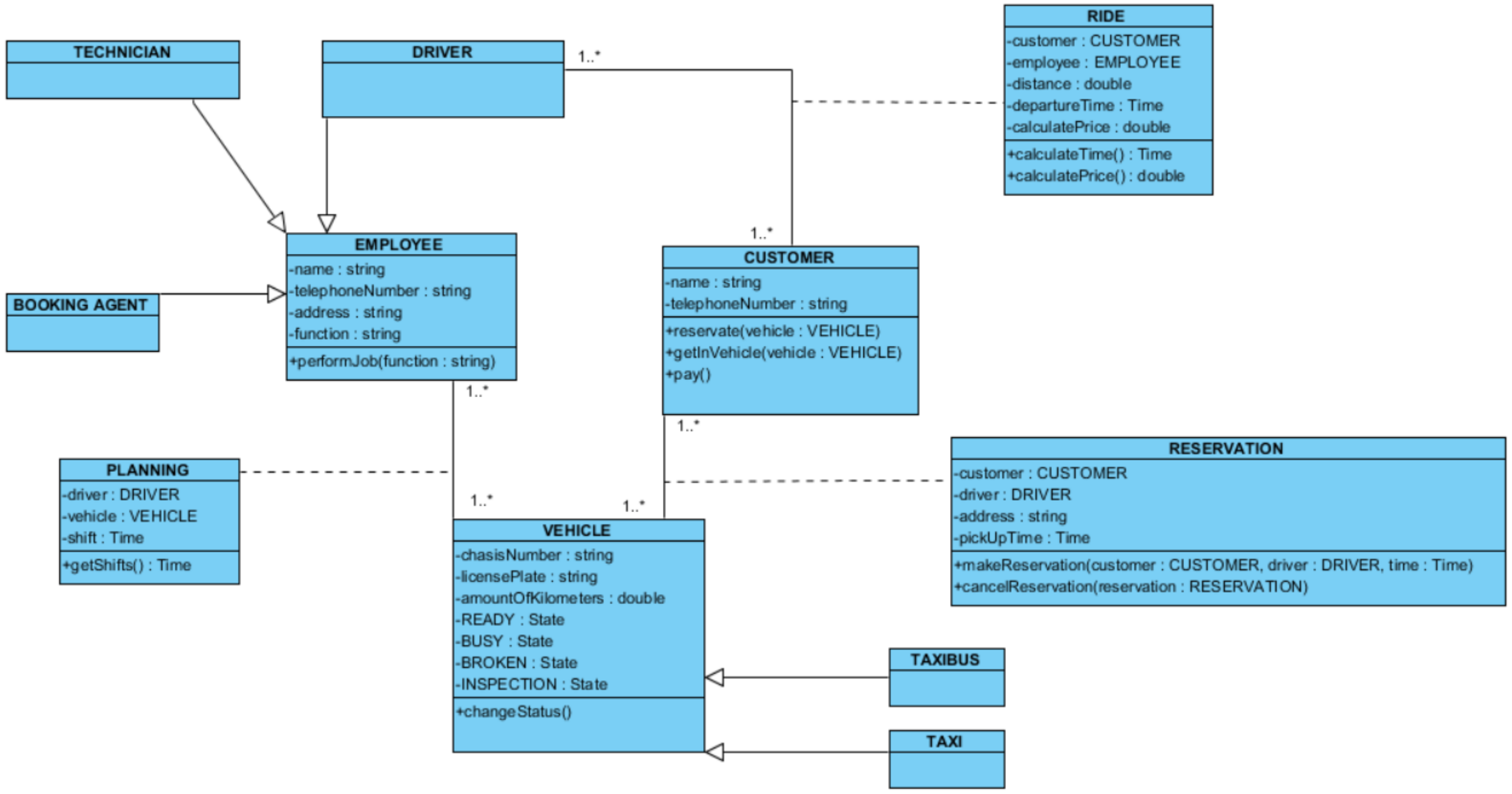
Activity diagram



State diagram



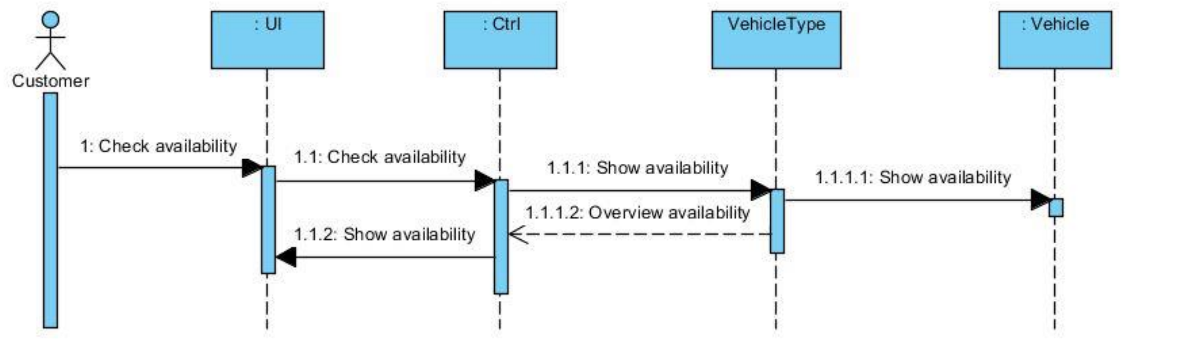
Updated class diagram



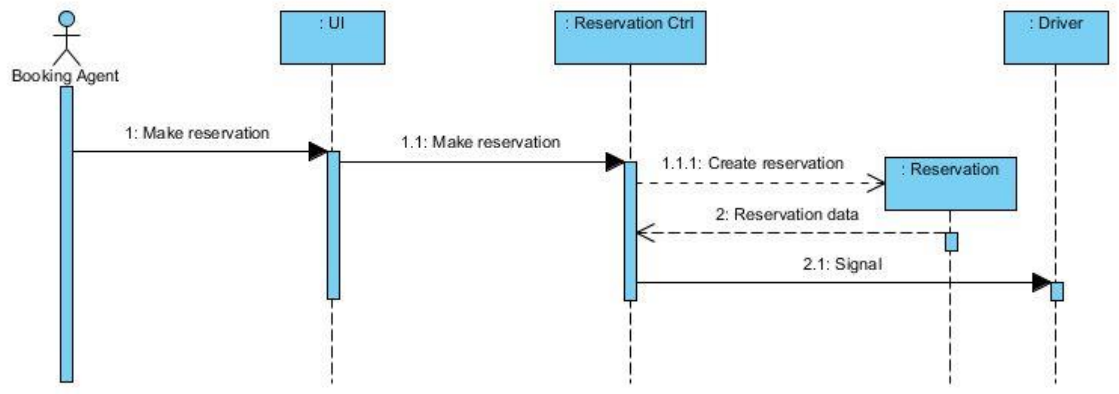
Application layer

Sequence diagrams

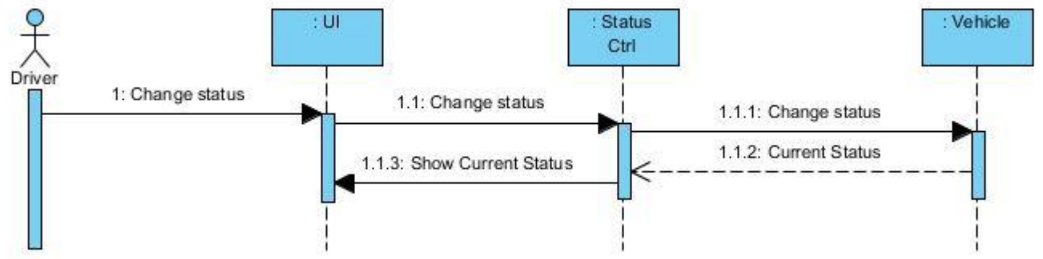
Check availability:



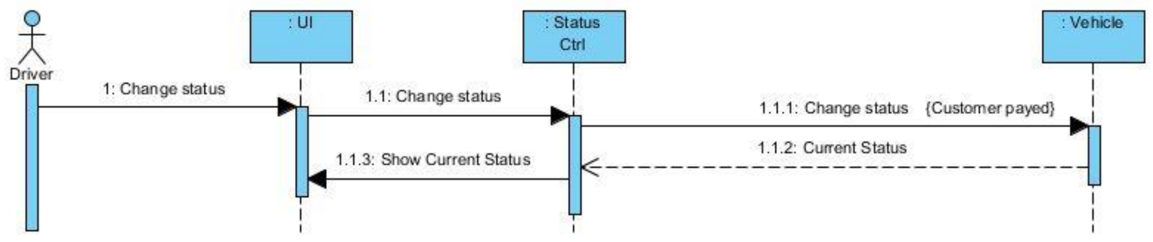
Dispatch:



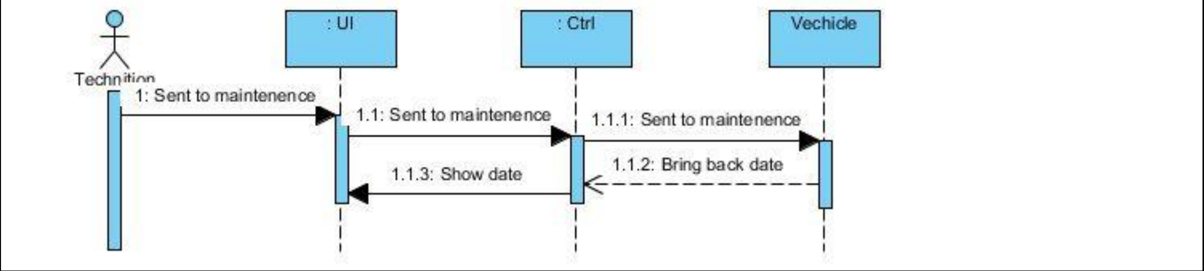
Pick up customer:



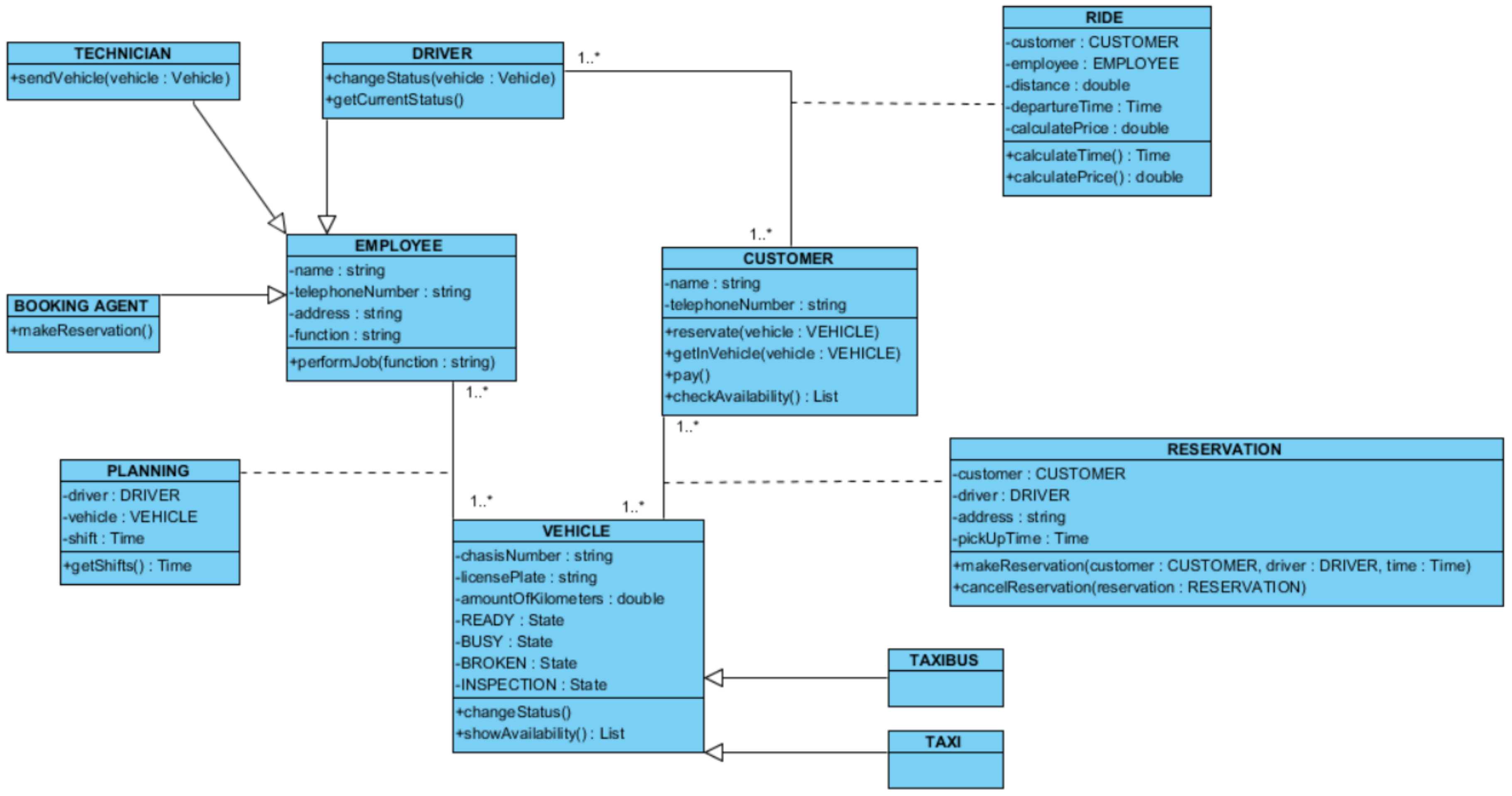
Arrives at destination:



Inspects:

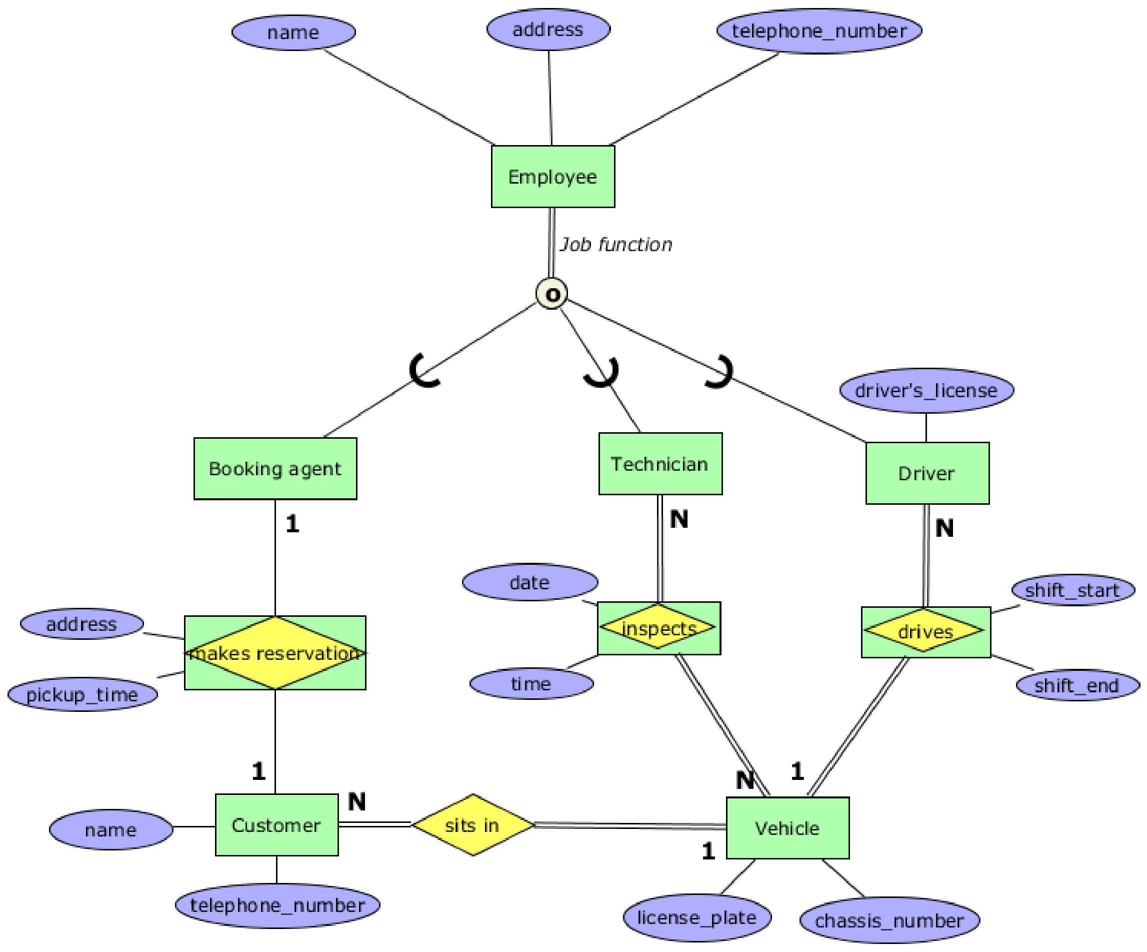

Technition
: UI
: Ctrl
Vechicle
1: Sent to maintenence
1.1: Sent to maintenence
1.1.1: Sent to maintenence
1.1.2: Bring back date
1.1.3: Show date

Updated class diagram

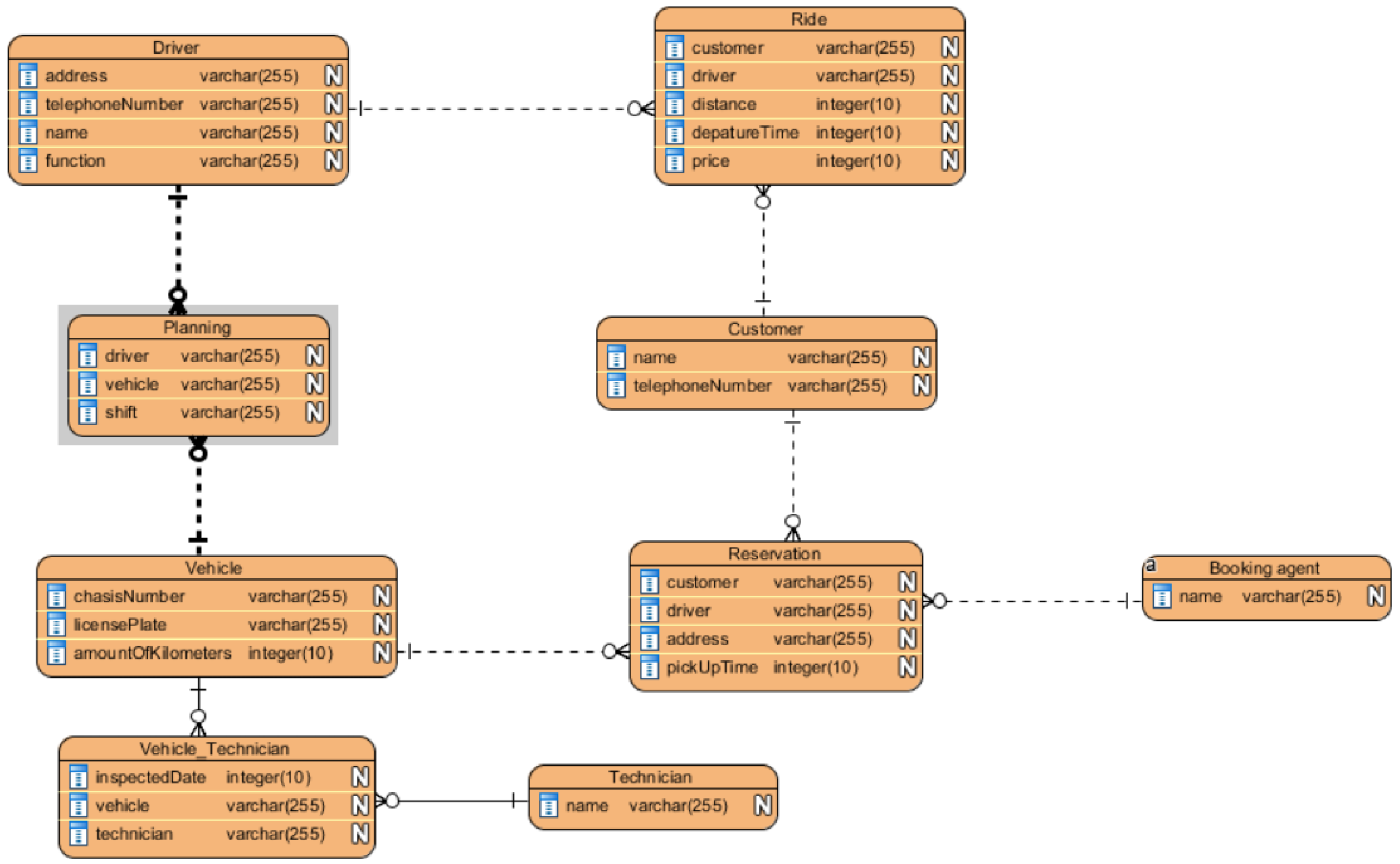


Technical layer

Conceptual model



ERD



Driver
address varchar(255) N
telephoneNumber varchar(255) N
name varchar(255) N
function varchar(255) N
Ride
customer varchar(255) N
driver varchar(255) N
distance integer(10) N
depatureTime integer(10) N
price integer(10) N
Planning
driver varchar(255) N
vehicle varchar(255) N
shift varchar(255) N
Customer
name varchar(255) N
telephoneNumber varchar(255) N
Vehicle
chasisNumber varchar(255) N
licensePlate varchar(255) N
amountOfKilometers integer(10) N
Reservation
customer varchar(255) N
driver varchar(255) N
address varchar(255) N
pickUpTime integer(10) N
a
Booking agent
name varchar(255) N
Vehicle_Technician
inspectedDate integer(10) N
vehicle varchar(255) N
technician varchar(255) N
Technician
name varchar(255) N

Hypothetical concepts

- State pattern :

  This pattern can be used for implementing the states of the taxis. When the driver changes his - and the status of the taxi - the system automatically executes steps to avoid that some methods of the application can be executed while in a certain state.

- Observer-Subject pattern:

  When the driver changes the status, the display of the booking agent must be updated automatically. Herefor we will use the observer pattern. The display of the booking agent will be the observer, while the taxi will be the observable.

- Strategy pattern

  This can be used to allow for different ride / reservation types.

- Factory pattern

  We can use the factory pattern to make the creation of new rides and reservation easier. Therefore the code will be less complex to maintain, even if another programmer has to do changes.

- MVC pattern

  The model-view-controller pattern allows the UI and the domain model to interact with each other without them knowing. This is also easier to maintain the code.

- Facade pattern

  With the facade pattern we will make a top-class for the UI and the model. The UI class knows every UI aspect, the same goes for the model class. Like the previous two patterns, this will contribute to a better maintainable program.