Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

myTaxiService

SOFTWARE ENGINEERING 2

Gatti Federico
Luca Fochetta

Requirements Analysis and Specification Document

Purpose

This document describes:

- goals
- components of the system
- functional and non-functional requirements
- domain properties and assumptions

It Also provides a description of the system using UML diagrams.

Goals

- Allow a taxi driver to inform the system about his availability
- Allow a taxi to change its status
- The system must handle in a fair way taxis queue
- The system must handle the distribution of taxis in the city

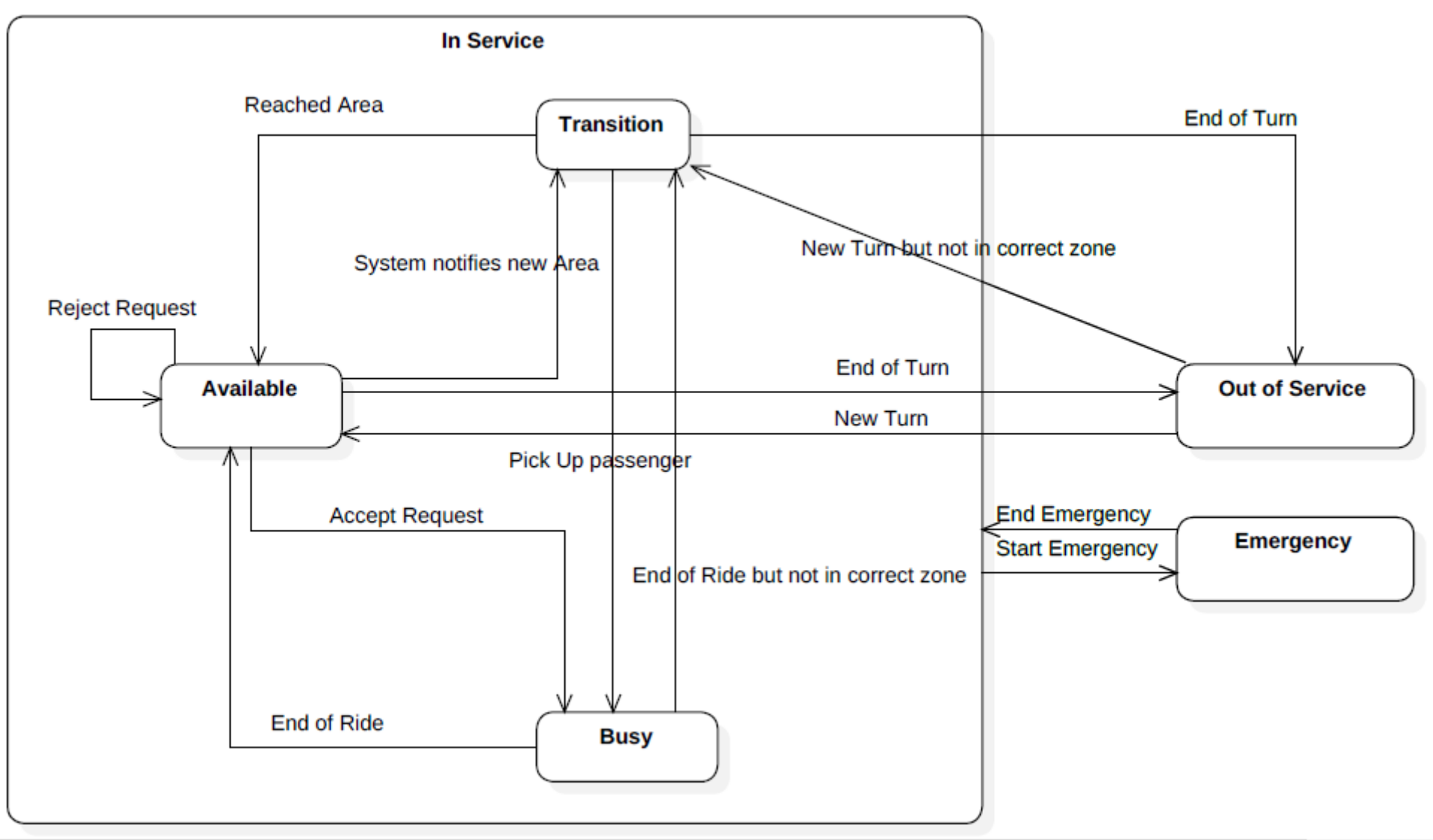
In our system the taxi driver can't select his area of competence autonomously but the system informs the taxi about it.

Taxi Status

The system must know the Taxi availability, so is necessary introduce a state for each taxi. The system provides 5 status for the taxi:

- Available: when the taxi doesn't carry passengers and it is situated in its competence area
- Busy: from request's acceptation by the taxi until the end of the ride
- Transition: when the taxi doesn't carry passengers and it isn't situated in its area of competence
- Out of Service: when the taxi is not in service
- Emergency: when a taxi occurs in an accident until the taxi solves it and decides to restart its turn

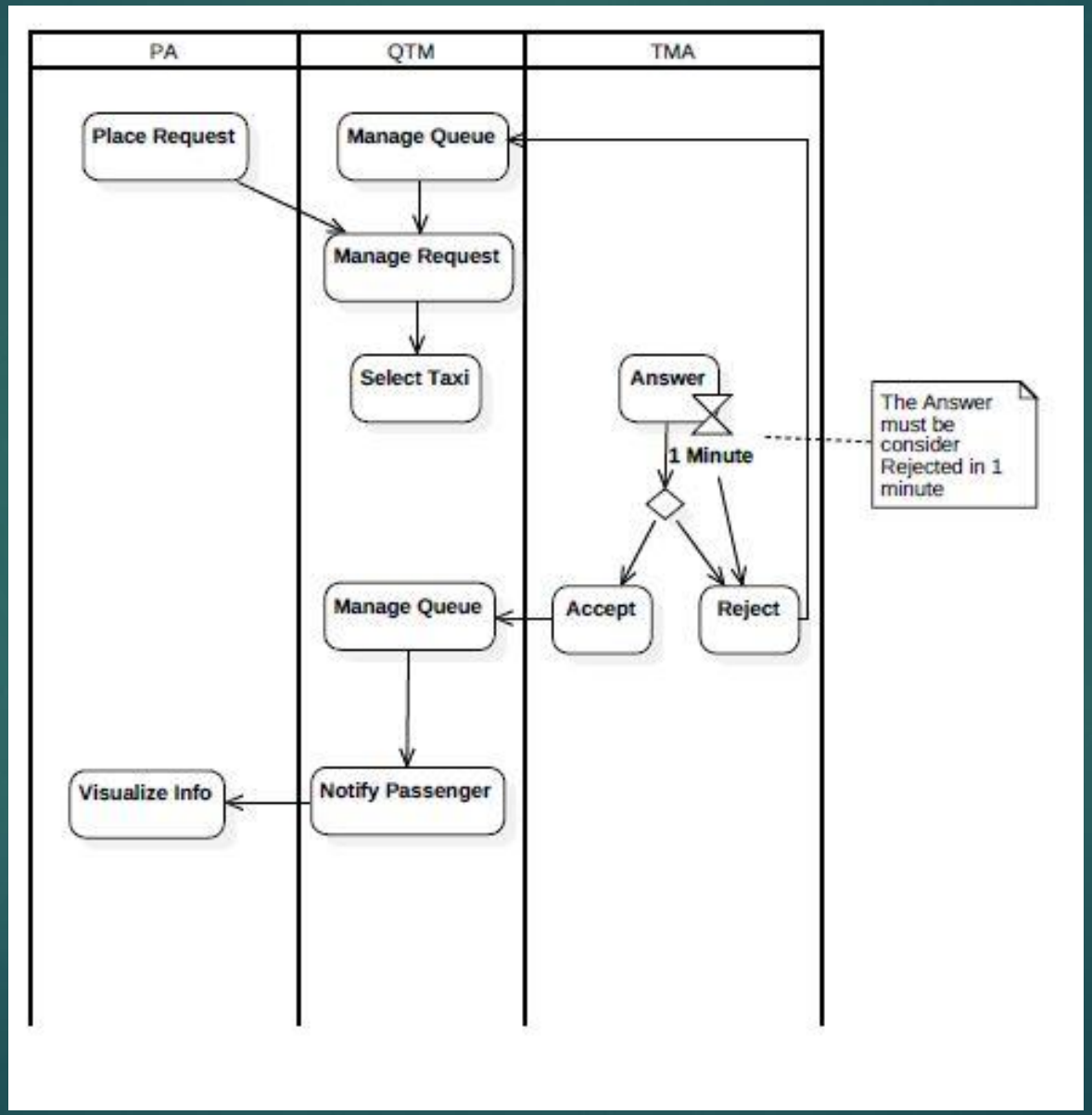
State diagram

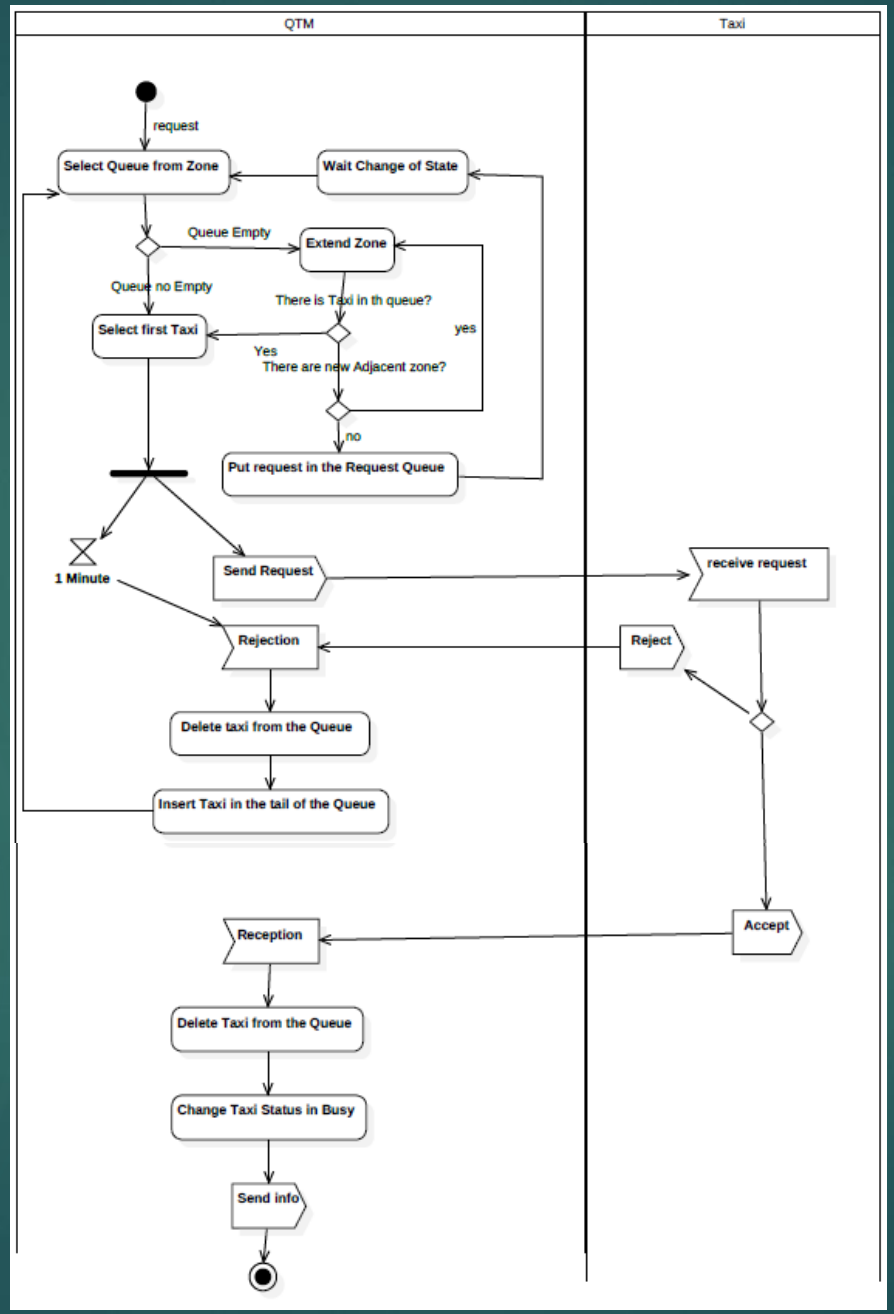


Queue Manager

- The request is forwarded to the first taxi in the queue associated to the zone from which the request comes
- If there aren't available taxi in the first queue the request is forwarded to the surrounding zones
- If the taxi doesn’t answer in 1 minute the request must be considered Rejected
- If a taxi accepts a request it will be deleted from the queue and its status changes in to busy
- If a taxi rejects a request it will be deleted from the top of the queue and it will be added in the tail

Activity Diagram: request




QTM
Taxi
request
Select Queue from Zone
Wait Change of State
Queue Empty
Extend Zone
Queue no Empty
There is Taxi in th queue?
Select first Taxi
yes
Yes
There are new Adjacent zone?
no
Put request in the Request Queue
1 Minute
Send Request
receive request
Rejection
Reject
Delete taxi from the Queue
Insert Taxi in the tail of the Queue
Reception
Accept
Delete Taxi from the Queue
Change Taxi Status in Busy
Send info

Taxi Distribution Manager

The system provides an algorithm to maximize the Taxi distribution in the city.

The number of taxis in a zone must be $\frac{Nm_z}{\sum_{z=1}^{Z} m_z}$

Where:

- N identifies the number of available taxi in the city
- z identifies a single zone
- Z identifies the set of the zones
- m_z identifies the number of taxis request in the zone z

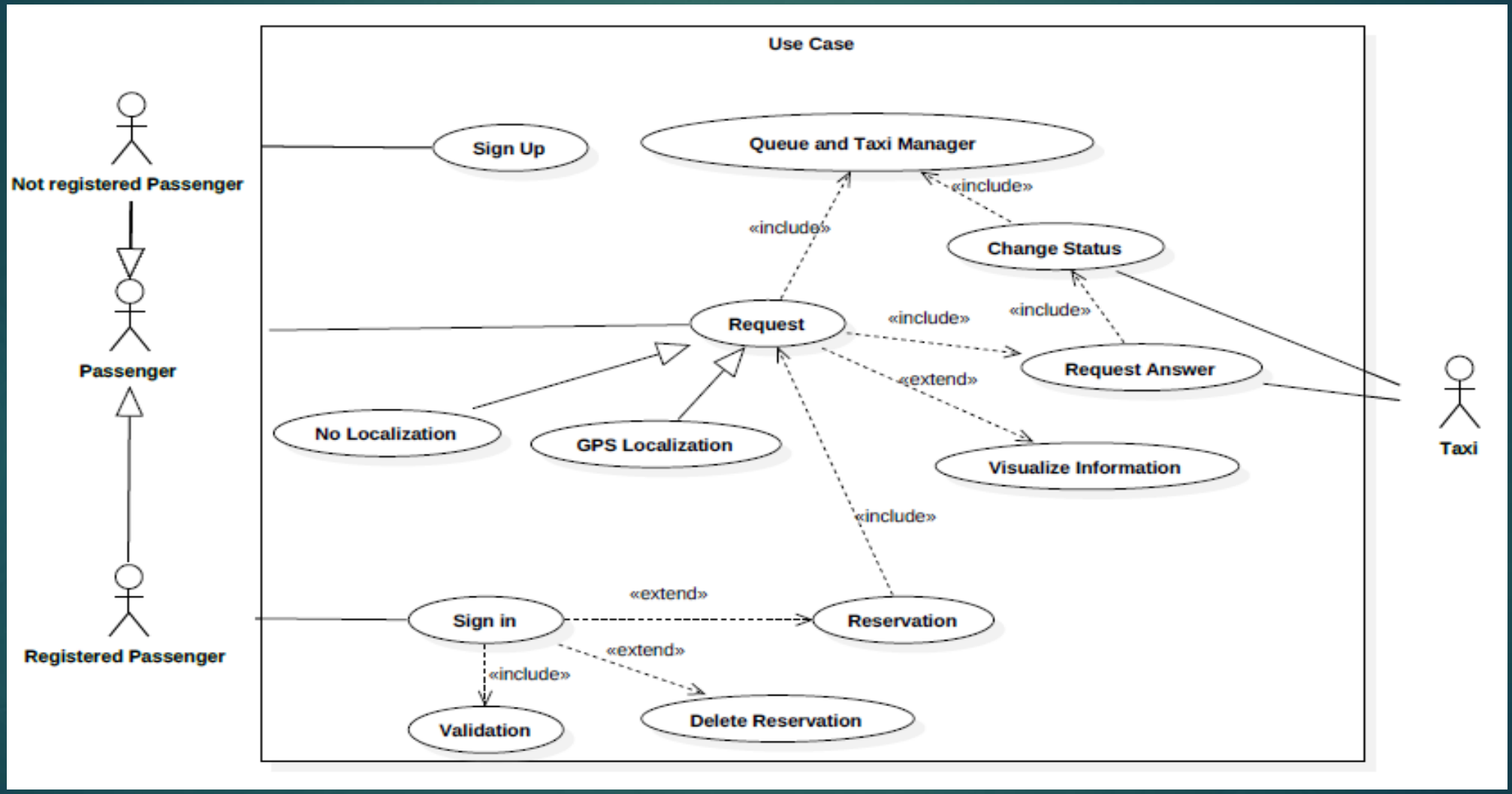
With a tolerance of 25%

- The algorithm considers Taxi in Transition in the correct zone for computes the distribution, but the taxi isn't inserted in the queue until it reaches the competence zone

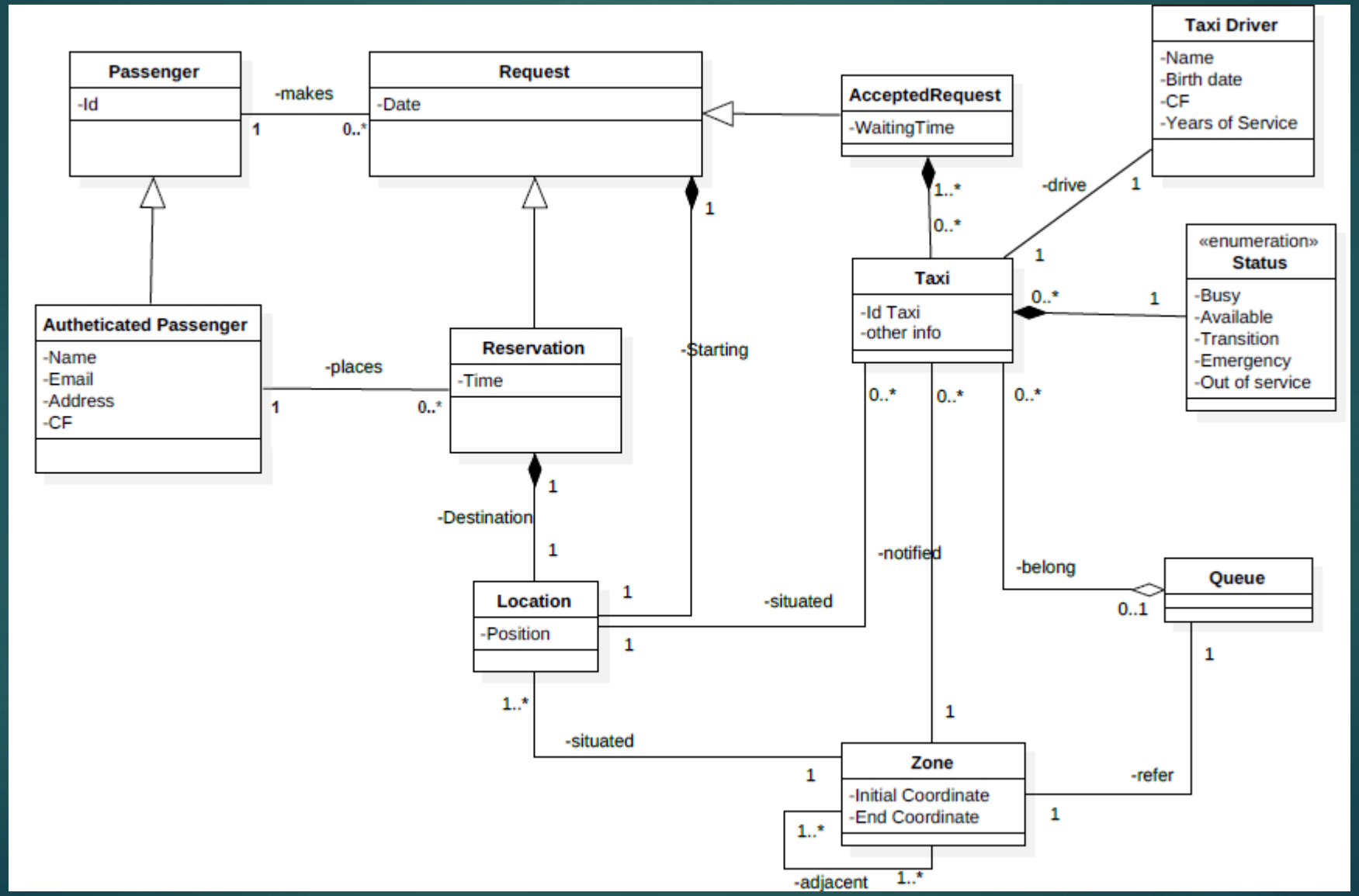
Some Functional Requirements

- The system shall handle the taxi status
- The system shall handle the taxi queue
- A taxi can be available in only one zone at a time
- The system shall consider Rejected a request if the Taxi doesn't answer in 1 minute
- A Taxi can't reject for 2 times the same request
- The system shall notify taxi its area of competence by TMA to obtain the best distribution in the city

Use Case


Use Case
Not registered Passenger
Passenger
Registered Passenger
Sign Up
Queue and Taxi Manager
«include»
«include»
Change Status
Request
«include»
«include»
Request Answer
«extend»
No Localization
GPS Localization
Visualize Information
«include»
Taxi
«extend»
Sign in
Reservation
«extend»
«include»
Validation
Delete Reservation

Class Diagram



Domain properties and Assumptions

- A Taxi driver respects the system decisions
- Every request is correctly forwarded
- There is a single queue for each zone
- The system doesn't forward the request from out of town
- The system is notified by GPS when the taxi reaches its competence's zone
- A Taxi Driver communicates the status Available to the system when a passenger leaves the taxi
- A Taxi Driver communicates the status Busy to the system if a passenger takes a taxi without using the PA
- A Taxi drivers has always access to Internet

Design Document

SOFTWARE ENGINEERING 2

Purpose

The purpose of the Design Document is to give a detailed description, analysis and specication of the software and hardware structure for the myTaxiService system.

The main objective of the Design Document is to achieve good understanding among analysts, developers, testers and customers.

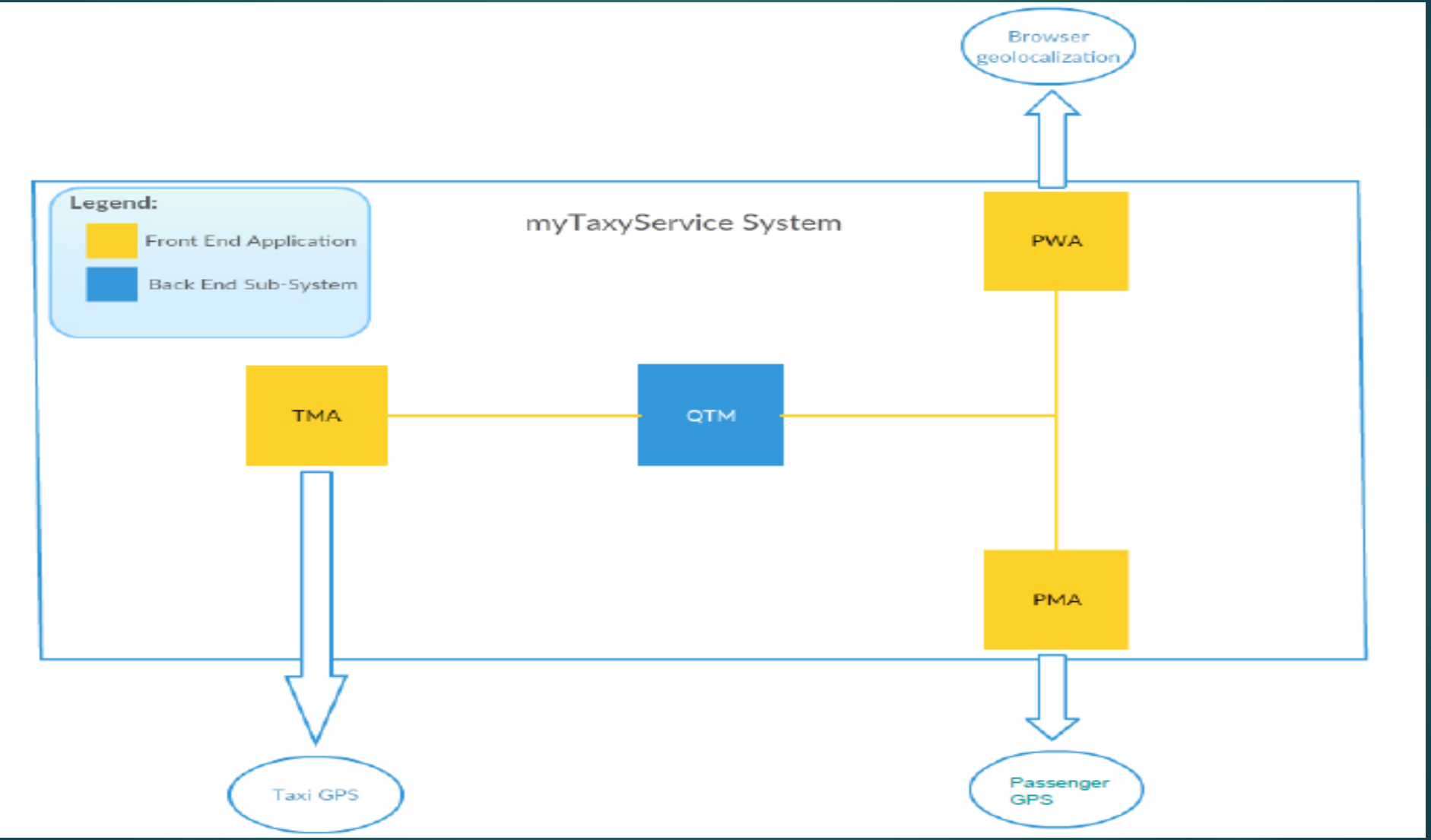
It is also aimed to be a solid base for project planning, software evaluation and possible future maintenance activities.

Architectural Design
High level Architecture

myTaxiService system is composed by:

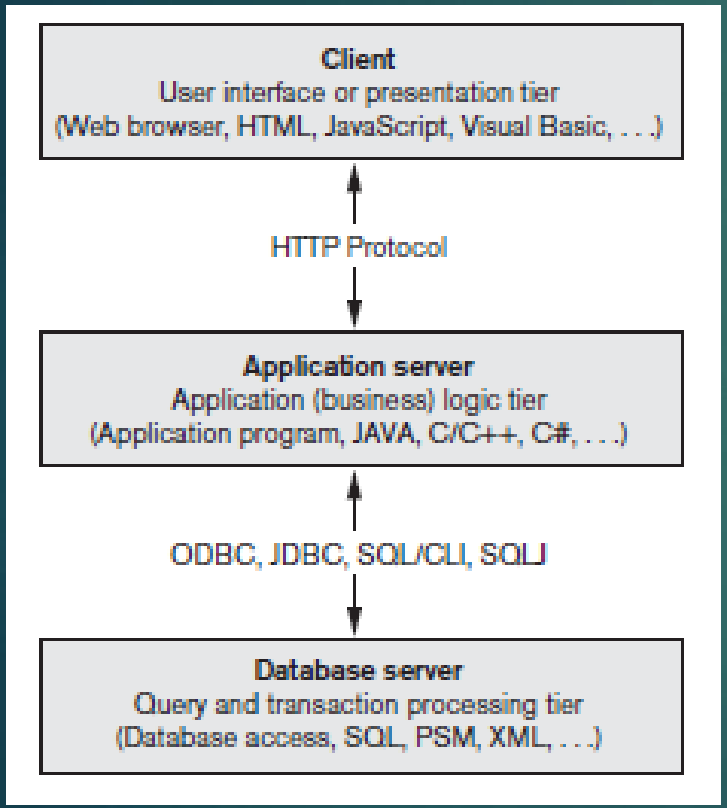
- 2 front-end applications used by the passengers called PMA and PWA
- 1 front-end application used by the Taxi Drivers called TMA
- 1 back-end subsystem called QTM

High level components



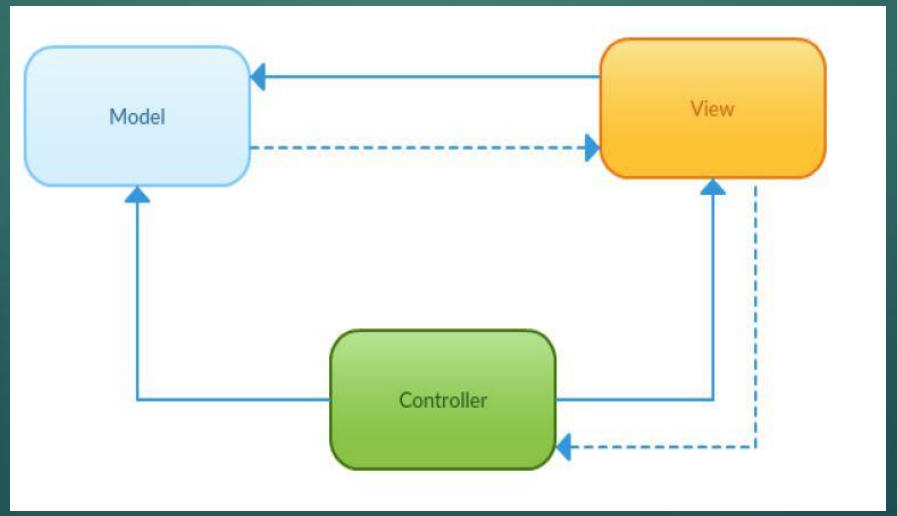
Architectural styles and pattern

Three-tier architecture

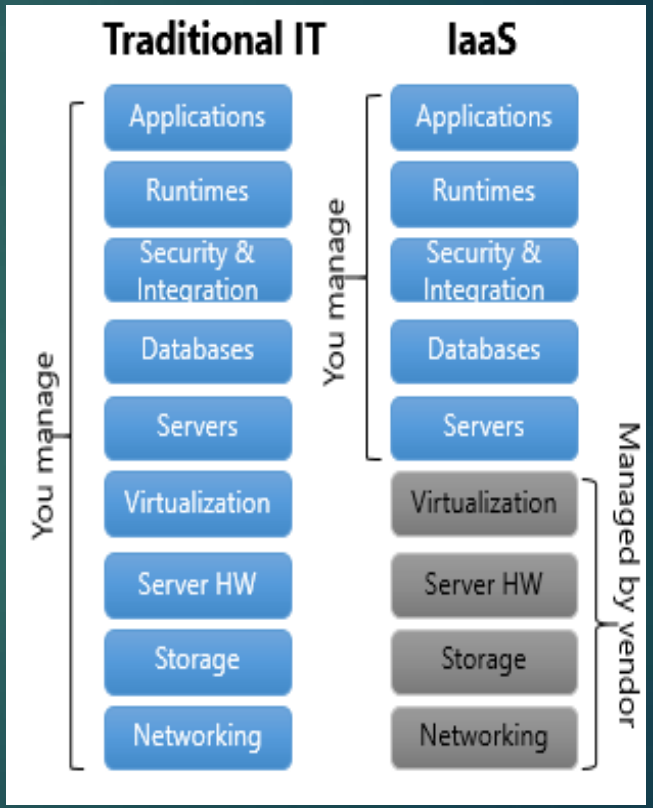


In order to improve performance, scalability, security and maintanability we have choice a Client-Server Architecture with this feature

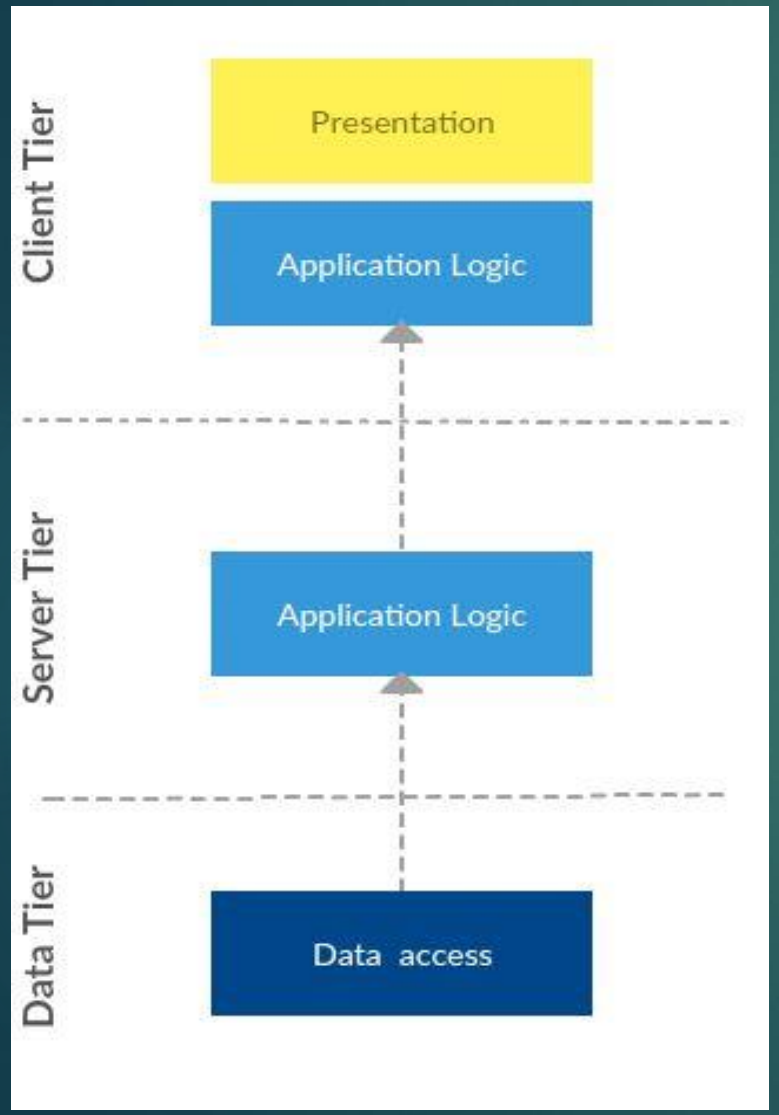
MVC



IaaS

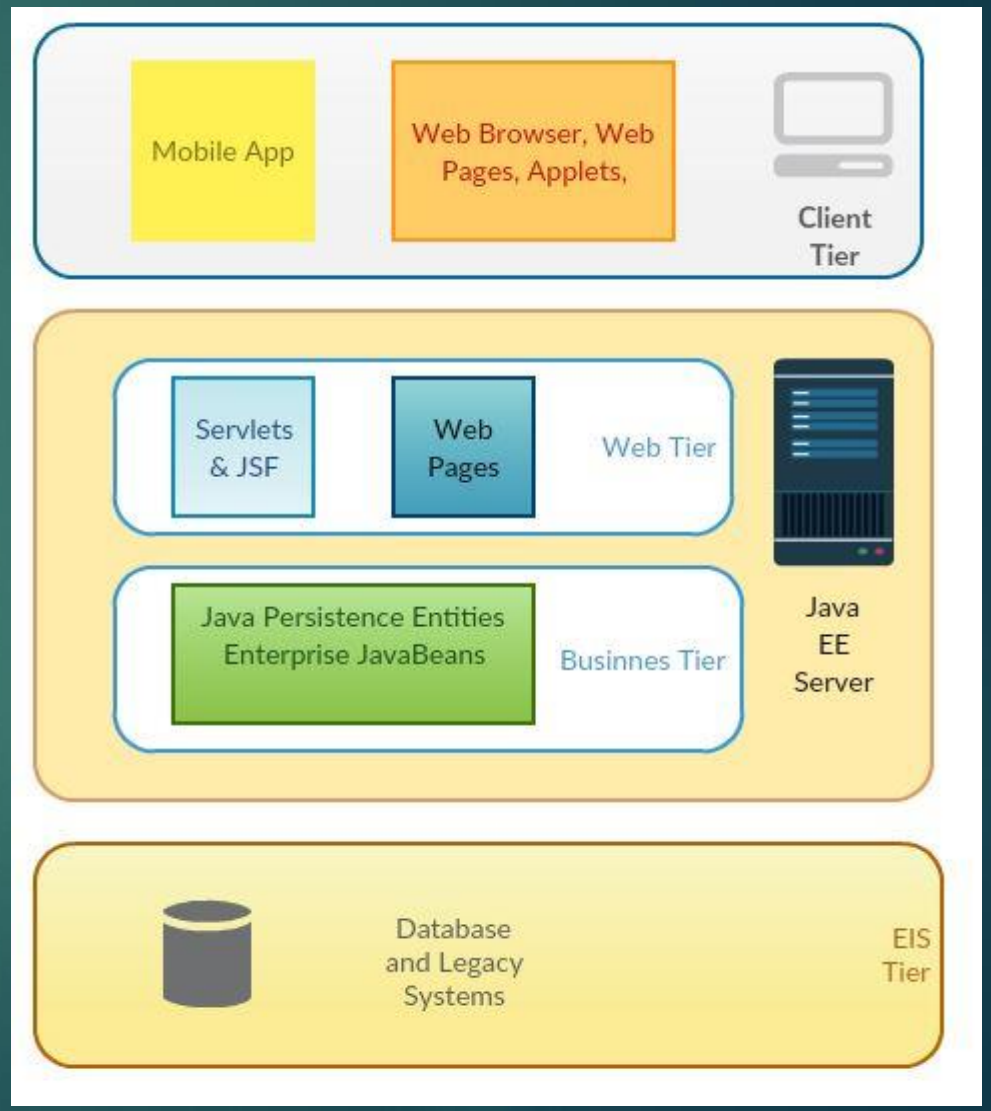


Architectural and Commercial Choice

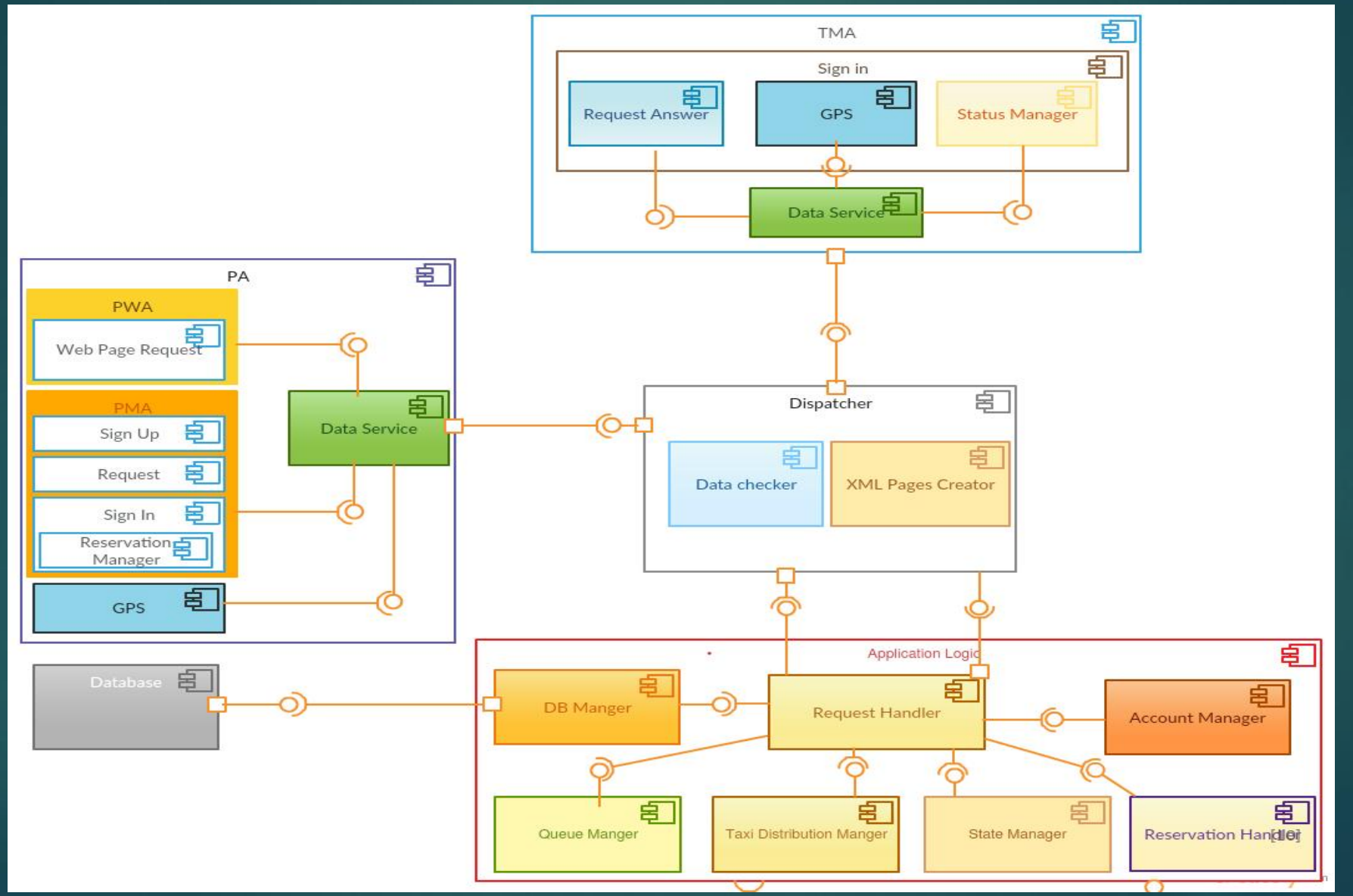


We decide to use a fat client, in this way we can divide the computational weight over the client and the server.

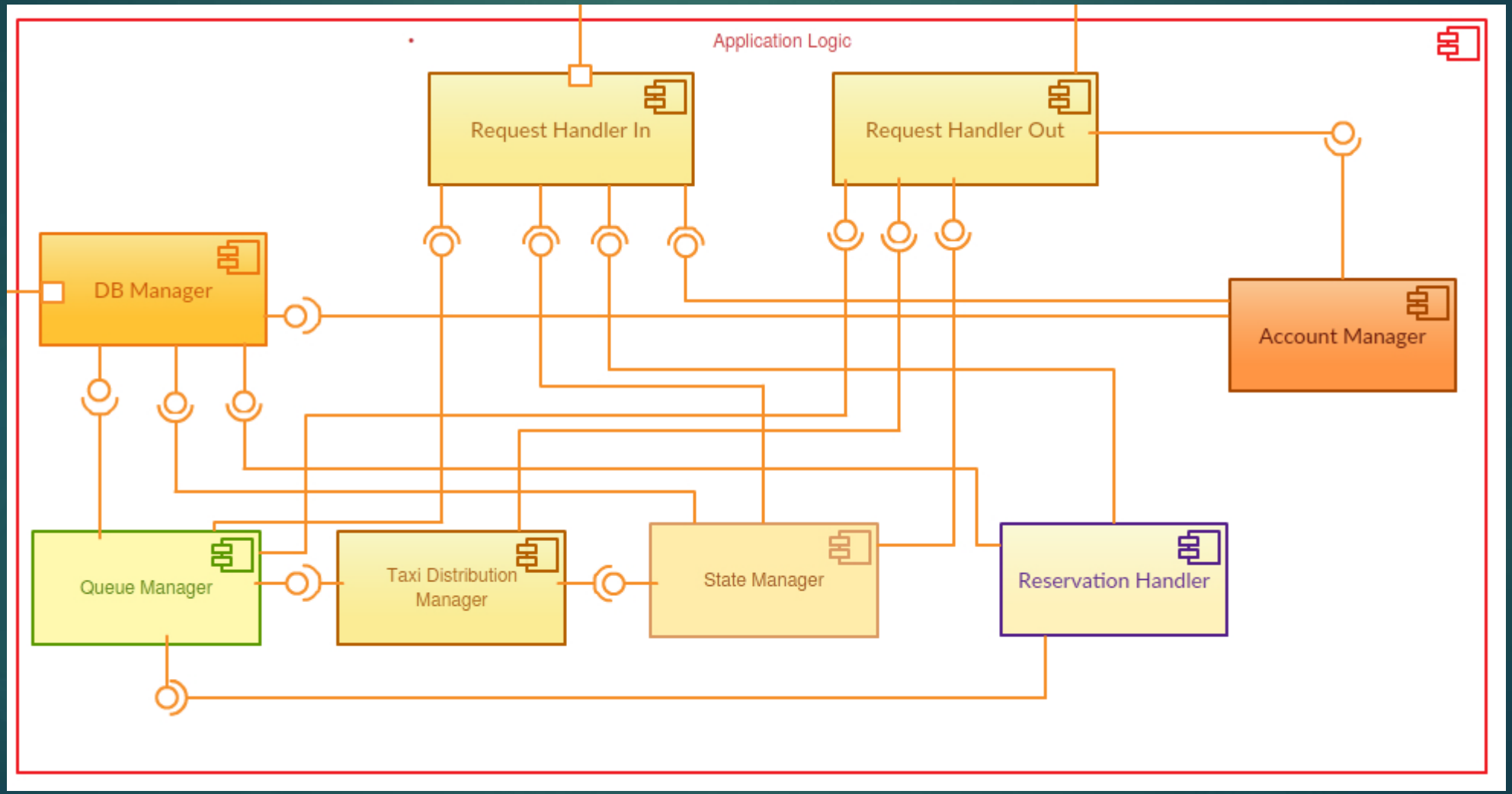
We can choose to use J2E for the real implematation of the system



Component Diagram

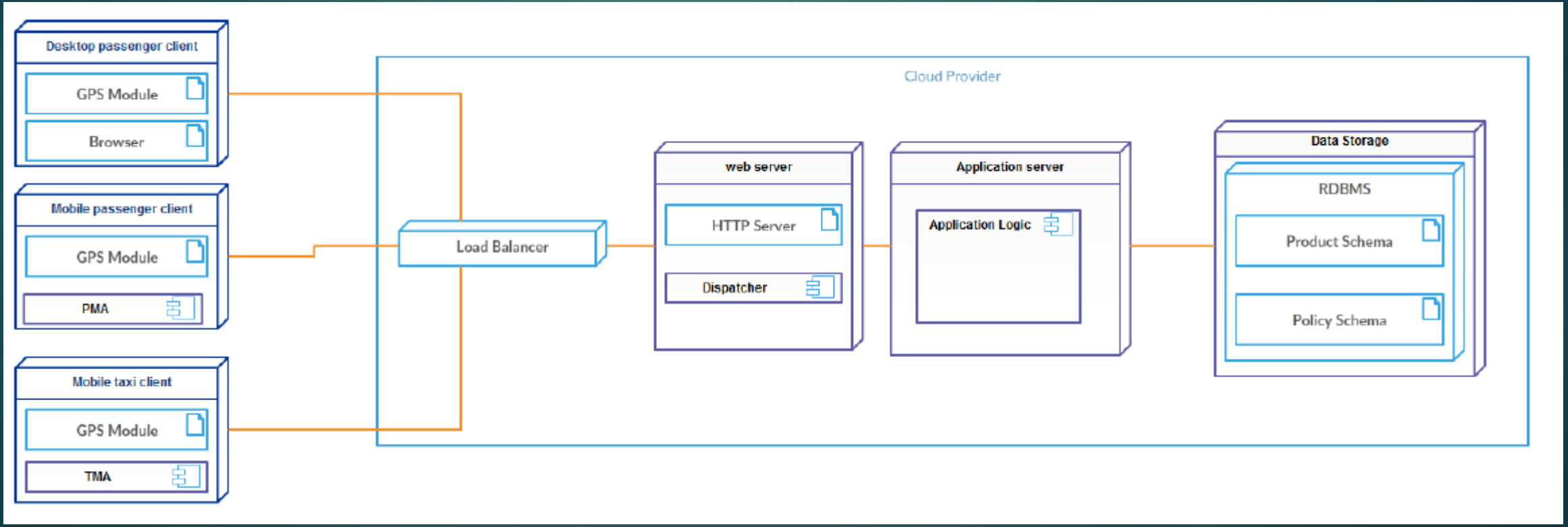


Application Logic Components



Deployment view

- Device nodes are physical computing resources with processing memory and services to execute software
- EEN is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements



Algorithm Design

Taxi Distribution Manager: has the task to maintain a optimal taxis' distribution in the city. The component must be able to take taxis from the zones that have a surplus of taxis and relocate them in the zones that have a deficiency.
We choose to use the minimum cost flow algorithm.
We know if a zone has a deficiency of taxis using some parameters:

1. m_z the number of request per minute in the zone z
2. r_z identies the number of taxis requested in the zone z to obtain a fair distribution
3. $T(r_z)$ is the value r_z considering the tolerance
4. N identifies the number of Available Taxis in the city

$$r_z = \frac{N m_z}{\sum_{z=1}^{Z} m_z}$$

If $a_z < T(r_z)$ we must perform this steps the minimum cost flow algorithm

Distribution Manager Steps:

1. Compute the value of the parameters and check if $a_z < T(r_z)$, if it's true continue, else stop
2. Construction the complete bipartite graph with the properties described in the DD
3. Find a feasible solution using the Maximum flow
4. Use the minimum cost flow to find a fair distribution of the taxis in the city
5. Notify the taxi of the new competence area and change their status in to Transition

Complexity: O(nmlog(n)log(nC))

Algorithm 1 Minimum cost flow

```
1: procedure NegativeCycleElimination(G, x)
2:     Gr ← ResidualGraph(x)
3:     while exist negative cycle C in Gr do
4:         UpdateFlow(x, C)
5:         updateResidualGraph(x)
6:     end while
7:     return
8: end procedure
```

Algorithm 2 Maximum Flow

```
1: procedure AugmentingPaths(G, x)
2:     x ← 0
3:     while P[t]!=0 do
4:         G_r ← Residual_graph(x)
5:         Search(G_r, s)
6:         if P[t]!=0 then
7:             AugmentFlow(P_st, x)
8:         end if
9:     end while
10: end procedure
```

Request and Reservation Handler

The component handles all the request and reservation forwarded to the system

Algorithm 6 Answer Time Out

```
1: procedure TimeOut(t, r)
2:     startTimeOut()
3:     if t no Answer before 1 minute then
4:         t.reject(r)
5:     else
6:         sendInfo(r,t)
7:     end if
8: end procedure
```

Algorithm 5 Count the number of Answer

```
1: procedure countNumberOfAnswer(t, r)
2:     if t has already answer to request r then
3:         t.accept(r)
4:         sendInfo(r,t)
5:     else
6:         sendRequest(r,t)
7:     end if
8: end procedure
```

State Manager

State manager's aim is handle the taxis status according to the State Diagram explained in the RASD

Algorithm 8 Change Status in Busy

```
1: procedure changeStatusInBusy(t, N)
2:     if t.status == Available || t.status == Transition then
3:         t.setStatus(Busy)
4:         deleteTaxiFromAQueue(t,t.getZone())
5:         updateTaxiAvailableNumber(t.oldStatus(),t.status())
6:     else
7:         textbfReturnError
8:     end if
9: end procedure
```

Algorithm 9 Change Status in Emergency

```
1: procedure changeStatusInEmergency(t)
2:     if t.getConfirmedRequest() is not empty then
3:         deleteRequest(r)
4:         sendRequest(r,r.getZone())
5:     end if
6:     deleteTaxiFromAQueue(t,t.getZone())
7:     t.setStatus(Emergency)
8:     updateTaxiAvailableNumber(t.oldStatus(),t.status())
9: end procedure
```