A Comprehensive Framework for Modelling Taxi Driver Behaviour in SimMobility:

Technical Specifications

Author: **Bat-hen, Arun, Ravi, Simon**

Kakali, HP, Jabir,

Status: Draft

Version: 0.5

Date: 31th Mar 2017

TABLE OF CONTENTS

ABSTRACT

A majority of past research efforts have been devoted to modelling and optimizing the taxi fleet operations. Much less attention has been made to the decentralized nature of taxi systems [Martinez et al., 2015]. In the context of multi-agent simulation technology, although some facets of taxi operations —such as street pick-ups, taxi queueing, and taxi dispatch—have been modelled, there is no platform that captures the dynamics between decentralized driver's decision-making and the centralized dispatching decision making. Such decentralized perspective is critical in modeling taxi fleets because taxi drivers can only be incentivized or coordinated but not centrally controlled [Cheng and Nguyen, 2011].

This paper aims to fill this gap by introducing a comprehensive framework that models various facets of taxi driver's behavior along with a taxi dispatching system within SimMobility Mid-Term simulator. SimMobility Mid-Term simulator is an agent based, fully econometric, and activity-based demand model integrated with a dynamic traffic assignment model [Lu et al., 2015]. It is capable of simulating daily travel at both the household and individual levels. The traffic dynamics are simulated using a mesoscopic simulator. SimMobility Mid-Term simulator is part of a much larger simulation platform that also contains long term and short term models [Adnan et al., 2016].
Simulating taxis is extremely challenging because of complex interactions between independent taxi drivers, the central controller, and travelers decision processes. To facilitate the study of such a complex and partially decentralized system, we propose an event-based modelling framework. In this framework, the taxi-driver, the controller, and the traveler are represented as separate decision agents. The taxi driver can exist in six states: booked, occupied, queuing, cruising, out-of-service and direct-to-destination. The events that can trigger change in state include pick up, drop off, cruising for too long, queueing for too long, joining a queue, controller request reception, and controller information reception. The specific behaviors of the taxi drivers are modelled within a discrete-choice framework. Specifically, the following models are used: (i) *Taxi driver Decision Model*, a binary choice model, that predicts the probability of a taxi driver to cruise or not to cruise; (ii)*Taxi Stand Choice Model* which estimates the probability of a taxi driver to choose a specific taxi stand among taxi stands stations scattered around the network; (iii) *Destination Choice Model*, which estimates the probability of a taxi driver to choose a specific operation destination among all possible destinations; (iv) *Route Choice Model*, which estimates the probability of a taxi driver to choose a specific route while she is cruising and her destination is set in advance;(v) *Link Based Mode, which e*stimates the probability of taxi driver to choose her next link while she is cruising; (vi) *Compliance Model*, which estimates the probability of a taxi driver to change her/his initial choices according to the information sent by the controller regarding a specific destination where the demand for taxis there is high and her/his presence is requested. (vii) *Acceptance Model,* which estimates the probability of taxi driver to accept or reject a pickup request made by the controller. Pick-up and drop-off events are handled without a model, when a taxi reaches the pick-up or drop-off point, taxi state will be changed accordingly.

On the other hand, the controller will process traveler’s service requests and assign the travelers to taxis after considering taxi driver’s choices and service costs. The costs include the overall time/distance required to dispatch, pick-up, and drop-off to client’s destination, the occupancy of the vehicle, and the demand generation potential of nearby hotspots. The controller captures the taxi service status, updated taxi locations, and movement through real-time communication with supply simulator over the simulation time, to react to the incoming ride and re-directing request accordingly. In this paper we also present the design features of the proposed controller that will allow us to extend the simulation of the operations of multiple on-demand services—Uber, Lyft—in conjunction.

We integrate this taxi framework to the calibrated SimMobility model of Singapore [Lu et al, 2015; Adnan et al., 2016]. In the estimation of SimMobility models, data from multiple sources are used. The variables related to network performance and level of service comes from network skims and GPS data. The GPS-enabled travel time data along with network skims, after application of data fusion techniques, allows generation of more realistic travel time data for the whole day. Land use data, which forms the basis of attraction variables for destination choice models is zonal based. The population was synthesized from household interview travel survey data (HITS 2012). The taxi demand was estimated by the Mid-Term activity-based pre-day model, which generates the day activity schedules for each passenger. The traveler can independently make decision according to controller’s suggestions. The suggested model can reflect strategic changes made by the driver such as the decision to stop cruising and move to a known location, the decision to stop queuing and start cruising, the decision to accept booking or not, the decision to move towards a high demand spot according to controller information, etc. Real-world operational data sets for Singapore are used, which allows us to construct a highly realistic simulation environment. For the estimation of each of the proposed taxi decision dimensions, a huge GPS dataset collected from a major taxi fleet operator in 2010 was used, containing position and service status data. This proposed framework will allow researchers and policy makers to study and evaluate potential mechanisms, policies, and new services for improving taxi services.

The current work has three major contributions. (1) Development of a comprehensive event-based framework that addresses complex behaviors and interactions of taxi drivers, fleet controllers, and travelers. (2) Incorporation of the proposed framework within a highly realistic agent-based simulation platform, SimMobility. (3) Evaluation of the suggested framework against real-world data. (4) Evaluation of taxi driver’s strategies, policies, and the dispatch mechanisms of the complete taxi eco-system.

1. Introduction

2. Literature Review

3. Taxi Driver Behaviour Framework

3.1 Initialization

Assumptions for modeling taxi driver agents in pre-day:

- For a given taxi driver agent (identified by LT), start time of work is pre-specified. Shift duration is pre-defined using taxi GPS data (see Figure A2: Shift duration distribution among taxi drivers.

- For the population of taxi drivers, the distribution of work start times across time of day will be determined/defined based on the Taxi GPS data.
- Given the start time and end time of the work shift, a day activity pattern will be generated using the pre-day model for the rest of the day (modifications to exclude work tours/intermediate stops)

Assumptions for modeling taxi driver agents in within-day:

- The creation of taxis will be made gradually, in 30 min interval, to reach the capacity of 85% of the fleet (the maximum capacity of the taxis on the road) according to the pattern identified by taxi GPS data (see Figure A1: the distribution of taxi fleet by time of day).
- The taxies "death" will be made according to the taxi drivers shift duration, which will balance the number of taxies on the road.

3.2 System Architecture

The taxi roaming model is designed to simulate taxi driver's behaviors within the MT and ST simulators of SimMobility. The simulation is focused in reproducing taxi driver's behaviors in a typical working day in a city. This framework is an event driven, and the taxi-driver, the controller, and the traveler are represented as separate decision agents. The taxi driver can exist in six states: booked, occupied, queuing, cruising, free-to-additional passenger, and direct-to-destination. The events that can trigger change in state include pick up, drop off, cruising for too long, queueing for too long, joining a queue, controller request reception, and controller information reception. The specific behaviors of the taxi drivers are modelled within a discrete-choice framework. Specifically, the following models are used: (i) *Taxi driver Decision Modl*, a binary choice model, that predicts the probability of a taxi driver to cruise or not to cruise; (ii)*Taxi Stand Choice Model* which estimates the probability of a taxi driver to choose a specific taxi stand among the taxi stands scattered around the network; (iii) *Destination Choice Model*, which estimates the probability of a taxi driver to choose a specific operation destination among all possible destinations; (iv) *Route Choice Model*, which estimates the probability of a taxi driver to choose a specific route while she is cruising, and her destination is set in advance; (v) *Link Based Mode, which e*stimates the probability of taxi driver to choose her next link while she is cruising; (vi) *Compliance Model*, which estimates the probability of a taxi driver to change her/his initial choices according to the information sent by the controller regarding a specific destination where the demand for taxis there is high and her/his presence is requested. (vii) *Acceptance Model,* which estimates the probability of taxi driver to accept or reject a pickup request made by the controller. Pick-up and drop-off events are handled without a model, when a taxi reaches the pick-up or drop-off point, taxi state will be changed accordingly.

3.3 Agents in Taxi Driver Behaviour Framework

3.3.1 Taxi agent

The taxi-drivers are modelled as agents with their own interactions with the travelers and the controller. In this section we present the variables associated with the taxi-driver. We assume that taxi drivers work in 8 hours shifts and each taxi is associated with only one controller. The behavior of the taxi-driver is dependent not only on her characteristics but also the associated controller. For example, a specific controller might not allow for driver's preferences while assigning travelers, thus eliminating the taxi-drivers freedom to choose client.

As taxi-driver and controller interactions are appropriately modelled as events, we first present the state-vector which defines the state of the taxi-driver.

State vector of the taxi-driver

A taxi's state is determined by its state vector. The state vector is composed of the following variables: booked, occupied, queueing, cruising, direct-to-destination, waiting and break. All of these are Boolean variables and are described in detail below:

Booked (B)
This variable describes if a taxi is on its way to pick up a costumer. The taxi is booked after it interacts with the controller.

Occupied (O)
This variable describes the situation when the client is in the taxi, and the taxi is heading towards the client's destination.

Free-to-additional passenger (F)
This latent variable characterize Occupied (O) state, it allow booking or pick up with the following three conditions: (1)current distance to $Client_2$ should be less than 5km, (2) current distance to $Client_2$ should take less than 10 min, (2) distance gap between two destinations should be less than 5km. If satisfy, the taxi with O state can be O|(F==1), and the taxi is able to consider i) booking request from controller or ii) client pick up from the street.

Queueing (Q)
This variable describes the situation when the taxi is in a queue at a taxi stand to pick up a client. We allow the taxis to stop and queue at any stopping bay; we treat different bays as a taxi stand. We will not allow the taxis to stop and queue on the lane[1].

Cruising (C)
This variable describes the situation when a taxi is searching the streets to find the next client. More generally it represents that the vehicle is ready for a passenger.

Direct-to-destination (D)
The variable describes a state where the taxi driver choses to or is direct to a specific destination. She will not make intermediate stops or pickups until she reaches the destination.

Waiting (W)
The variable describes a state where the taxi driver is waiting on the network for either request or direction from the controller.

Break (K)
The variable describes a state where the taxi driver is on a break; the driver temporarily will not be available to accept any kind of requests.

The above seven variables together —which define the state vector— describe different taxi states. The following table describes various possible traffic states and their representation in the state vector.

State	**Description**
Booked (B)	On the way to pick up a booked client
Occupied (O)	Serving a client
Queuing (Q)	Queuing at a taxi stand

[1] This part will be fully implemented when SimMobility parking model will be implemented, after mapping all parking lots.

Cruising (C)	Cruising in search of a client; represents the vehicle is ready to pick-up
Direct-to-destination (D)	Going to a specific destination (node, zone, stand etc.) for potential clients. No intermediate stop is allowed
Free-to-additional passenger (F)	Internal state of taxi, allowing to take additional passenger.
Waiting (W)	Waiting on the network (for either request/direction from controller)
Break (K)	On a break, temporarily will not be available to accept requests

Table 1: The different states of the taxi as explained by the state vector

3.3.2 Traveller agent

The traveler will meet the taxi in home location or in the walking network. When the traveler book a taxi, she will meet the taxi driver in the traveler's residential location. Else, they will meet on the street either by hailing a taxi or by walking to the most desired taxi stand nearby. This demand data are collected by the system to provide information to the taxi driver about the historical space and time distribution of the clients at each node. The information is then used by taxi drivers to choose the most adequate taxi stands to stop at different hours of the day and choose the most attractive routes for finding clients in the street.

The traveler searching for available taxi, is added to a first in, first out array of potential clients for a taxi at that link and waits for the taxi acceptance. At each node of the road network, the traveller can be picked up after visualizing him at the end link of the node.

Traveler agent in shared mode

Our framework allow to have a shared ride option (to be implemented in a later stage). From the demand side predicted by pre-day, we will add two more modes to the existing mode choice model, so that the new modes which are relevant to a taxi choice will include "Taxi as a driver", "Taxi as a passenger", and "Taxi in a shared mode". While booking a taxi in shared mode the controller will assign two travelers to the same taxi if a similar path was identify. The taxi will move to the pickup point of the first traveler and pick her up. Then, the rout choice model will be activated and the taxi will move according to travelers 1 destination, drop her off and continue to travelers 2 drop off point.

3.3.3 Controller agent

The controller agent is responsible for managing the operation of the taxi fleet, taxi request and taxi dispatch. Please refer to document “Smart Mobility Controller in SimMobility”.

3.4 Events influencing the taxi-driver agent

3.4.1 List of events

The designed model is an event driven and there are seven major event types:

Pick-up event:
This event occurs when taxi-driver pick-up a traveler. The pick-up can happen on-road or at any stopping bay; we treat different bays as a taxi stand a taxi-stand.

Drop-off event:
This event occurs when the taxi reaches the destination of the traveler. After this event, the taxi-driver reevaluates his strategy.

Join queue event:
This event occurs when the taxi is reaching the taxi-stand. The taxi can join the queue at the stand only if the number of taxies already queueing at the stand is no more than 5.

Cruising too long event:
This event is triggered when the taxi-driver has been cruising unsuccessfully for too long. If the taxi-driver has not been cruising unsuccessfully for too long, the taxi-driver reevaluates his strategy. if the taxi-driver has been cruising unsuccessfully for too long, the taxi-driver will choose “do not cruise”.

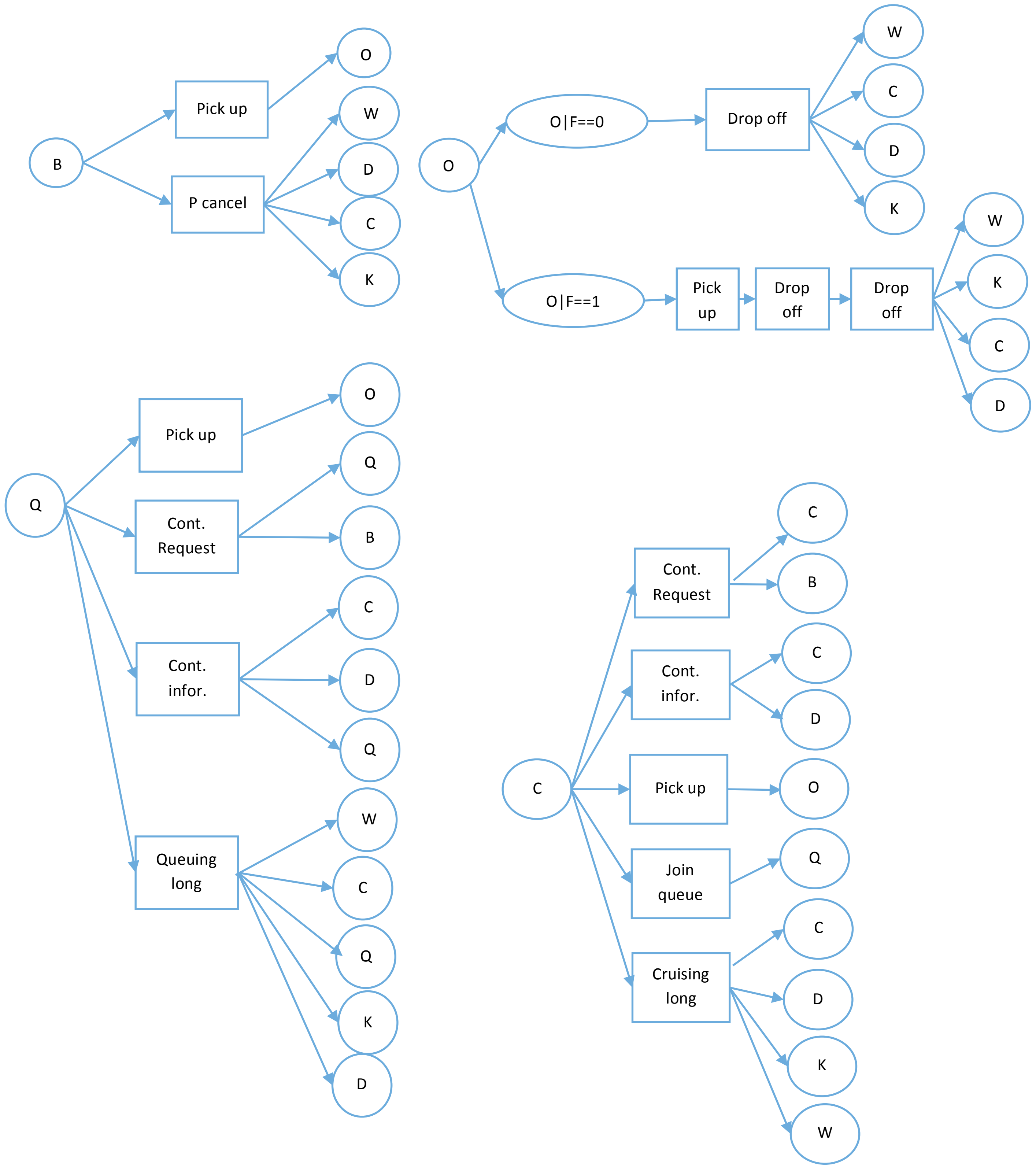
Queuing too long event:
This event is triggered when the taxi-driver has been queuing unsuccessfully for too long at a taxi-stand. After this event, the taxi-driver reevaluates his strategy.

Controller request event:
This event describes the messages the (controller) dispatcher sends to the taxi driver regarding the pick-up request. This event can occur at any time.

Controller information event:
This event describes the messages the (controller) dispatcher sends to the taxi-driver regarding the current high-demand areas or to redirect her to some other locations.

3.4.2 Influence of events on taxi-driver state

The changes in the state of the taxi (see section 3.3.1.1) in response to the events are given in the following figure.


B
Pick up
P cancel
O
W
D
C
K
O
O|F==0
Drop off
W
C
D
K
O|F==1
Pick up
Drop off
Drop off
W
K
C
D
Q
Pick up
Cont. Request
Cont. infor.
Queuing long
O
Q
B
C
D
Q
W
C
Q
K
D
C
Cont. Request
Cont. infor.
Pick up
Join queue
Cruising long
C
B
C
D
O
Q
C
D
K
W

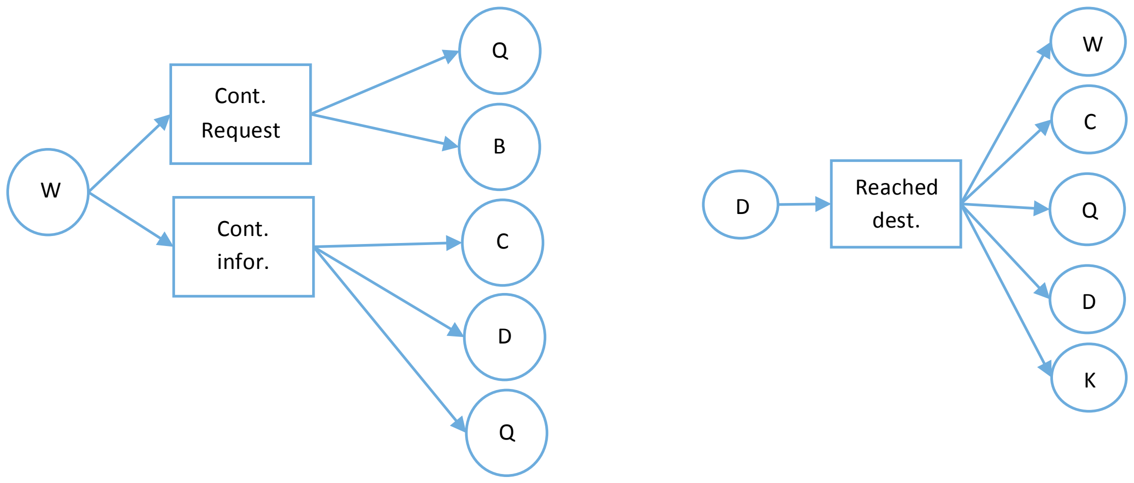


Figure 1: The changes in taxi state in response to events (Note - B: Booked; O: Occupied; Q: Queuing; C: Cruising; D: Direct-to-destination; F: Free-to-additional passenger; K: Break; W: waiting)

For each initial state the set of possible events that can affect the state and the corresponding final states are given:

Case(i): initial state is booked (B). The taxi's state changes to occupied (O) when it picks up the traveller. The traveller agent can cancel the booking, in that case taxi's state changes to direct-to-destination (D), cruising (C), waiting (W) or break (K).

Case (ii): initial state is occupied (O). If the taxi is occupied, then the only event that can change its state is the drop-off when the traveller's destination is reached. After the drop-off, the taxi-driver can change to any of the following states: cruising, queuing or direct-to-destination. The event where there already exists a passenger at the drop-off point is modelled as follows. First, the taxi-driver changes state to cruising. Then, she becomes aware of the passenger and then decides whether or not to pick-up. While sustaining O state, if there is another request (either from controller or street), taxi can make decision based on F variable already defined in previous section. Once the driver is allowed and decides to take additional passenger, the taxi driver returns to C, W, K or D state after two consecutive drop off events.

Case (iii): initial state is direct-to-destination (D). In this state the taxi-driver choses to or is directed to go to a particular destination. By the nature of this variable, the vehicle can only change state after it reaches the destination , the taxi driver can decide to change his destination only when he receive messages or requests from the controller. At destination, the vehicle can change state to queuing (Q), cruising (C), direct-to-destination (D), waiting (W) or break (K).

Case (iv): initial state is cruising (C): In this state the driver is actively looking to pick-up the passengers. She can respond to controller request and controller information. Alos to on-street pick-ups. When she is close to a taxi-stand, the join queue event might be triggered making her join the queue. If she is cruising for too long, she might revaluate her strategy.

Case (v): initial state is queueing (Q): As the driver is at a taxi-stand, she might pick up a passenger or revaluate her strategy if she is waiting for too long, than taxis state can be changed to cruising, direct-to-destination, waiting or break and she can also choose to queue. Taxi driver can also respond to controller request and controller information, and switch to any of the states: cruising, booking, direct-to-destination, queueing, waiting or break.

Case (vi): initial state is waiting (W): As the driver is waiting at a parking area, she might get a controller request or controller information; then she can switch to any of the states: cruising, booking, direct-to-destination, queueing, waiting or break.

3.5 Taxi Driver Agent Behaviour framework

Figure 2 describes the taxi driver agent behavior framework. The blue boxes describes the taxi driver behaviour, and the yellow boxes describes the relationship between the controller and the driver. The squares represent a model and the others represent a choice. We start by the taxi driver initial decision who may choose to cruise or not to cruise, meaning to drive directly to a taxi stand.

- If the taxi driver chooses to drive to a specific taxi stand then the destination choice model and the route choice model are been activated. After reaching to a taxi stand the taxi driver will join a queue; if she finds a passenger there, she will pick him up, than the route choice will be activated, and the taxi will choose the shortest path to traveler's destination. If the taxi driver did not find a costumer while queuing she may reevaluate her plans and the simulation will go to the starting point.

- If the taxi driver chooses to cruise, she will search the streets to find a client. She can do so by moving towards a predetermined node or stand, than the route choice model and the destination choice model will be activated. The taxi driver can also cruise link by link. The taxi will choose a path or the next link according to demand (see section 3.6 which describes the attractiveness variable). If a client was found, the taxi driver will pick him up; the route choice model will be activated, and the taxi will move towards client destination. If the taxi driver is cruising for too long, she will change her strategy to “do not cruise”, else, the simulation will start from the starting point.

The controller can constantly send messages to the taxi driver, the driver can choose to accept or reject these messages. The controller can send a booking request or information request:

- If the controller is sending a booking request, and the taxi driver accepts the request, he will moves towards the customer's location using the route choice model.

- If the controller is sending information request regarding a specific station or node where the demand for taxis there is high, then the compliance model will be activated and the driver may choose to move that particular station or node using the rout choice model, otherwise the simulation will go to the starting point. “Centralized Taxi Operation in SimMobility document will describe in detail the way in which the controller is modeled.

Figure 2 illustrates the taxi driver agent behavior framework.

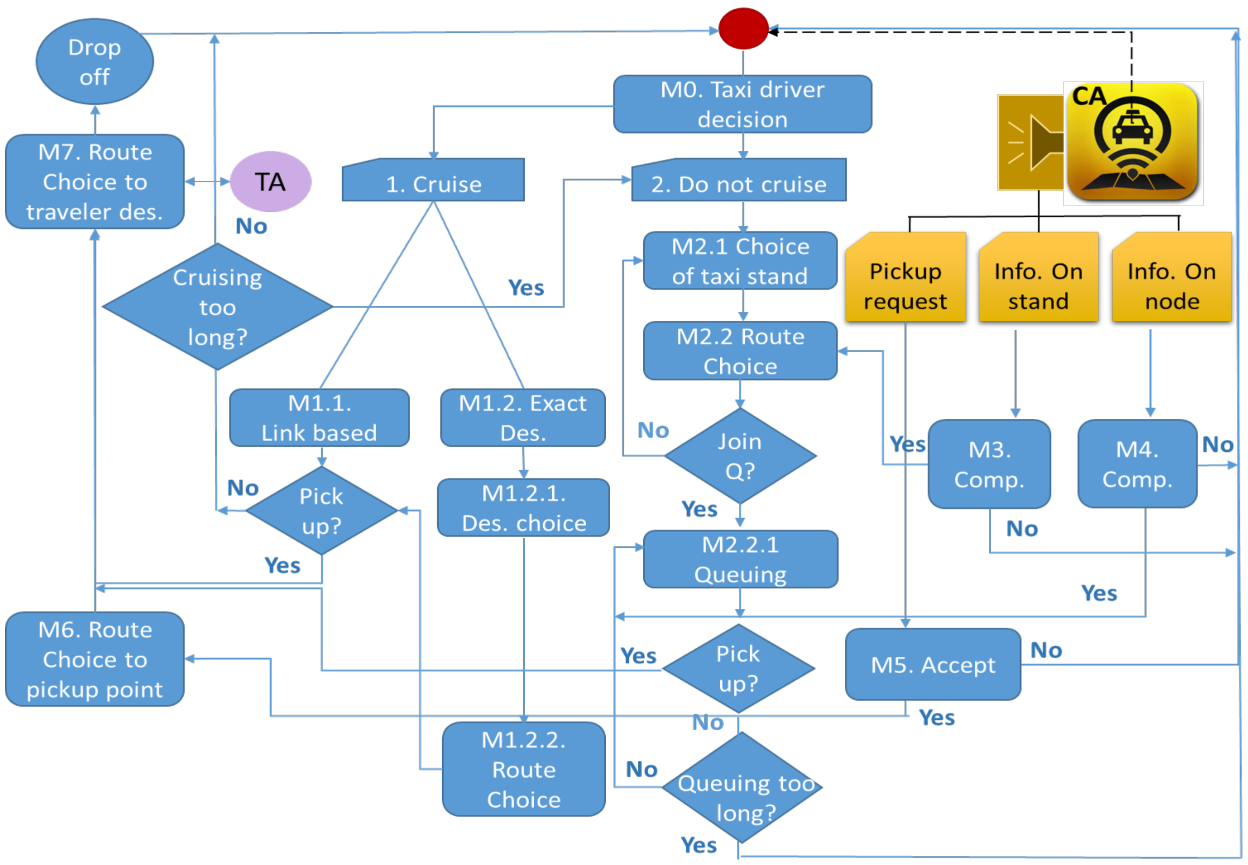


Figure 2: Taxi driver agent behavior framework

3.6 Models

In this section, the taxi driver behavior framework models and variables are described in detail.

M0. Taxi driver decision of cruising or not in a specific zone
This model is a binary choice model that will predict the probability of taxi driver to cruise or not to cruise.

Identified variables:

1. Characteristics of the taxi driver
2. Overall attractiveness (total demand divided by travel time) of the streets using historical data[2]
3. Overall attractiveness (total demand divided by travel time) of the taxi stands using historical data[1]
4. CBD (yes/no)
5. Expected travel cost to the destination
6. Congestion levels in the zone/ expected travel time to the destination
7. Remaining time in shift

M2.1 Taxi Stand Choice Model
This model will estimate the probability of taxi driver to choose a specific taxi stand (at the node level) among X stations scattered around the network.

Identified variables:

1. Characteristics of the taxi driver
2. Attractiveness (demand divided by travel time) of the taxi stand using historical data[1]
3. Expected waiting time at the stand
4. CBD (yes/no)
5. Journey time to the potential stations
6. Expected travel cost to the destination

Integrated Model Destination and Route Choice Model

[2] The attractiveness aj of street node/taxi stand j is the total number of trips dj departing from the street node or taxi stand point j in relation to expected travel time Tj between the street node or taxi stand points to the taxi driver location: $a_j = \frac{d_j}{T_j}$
$p_j = \frac{a_j}{\sum_j a_j}$ is the probability that street node/taxi stand j will be chosen.

This model is a simple nested model, at the upper level of this model, the individual chooses her destination among all possible nodes, and at the lower level she chooses a specific route to reach this destination.

M1.2.1. Destination Choice Model

This model will estimate the probability of taxi driver to choose a specific destination among X possible destinations.

<u>Identified variables:</u>

1. Land use variables (if any)
2. CBD (yes/no)
3. LS from the route choice model

M1.2.2. Route Choice Model

This model will estimate the probability of taxi driver to choose a specific route while she is cruising and her destination is set in advance.

<u>Identified variables:</u>

1. Characteristics of the taxi driver
2. Attractiveness (total demand divided by travel time) of the route using historical data[1]
3. Journey time to the potential destination
4. Expected travel cost to the destination

M1.1. Link Based Model

This model will estimate the probability of taxi driver to choose her next link while she is cruising.

<u>Identified variables:</u>

1. Characteristics of the taxi driver
2. Attractiveness (demand divided by travel time) of the next link using historical data[1]
3. Appearance of taxi stand at the next link

M3/M4. Compliance model

This model will estimate the probability of taxi driver to accept or reject the information coming from the controller.

<u>Identified variables:</u>

1. Characteristics of the taxi driver
2. Total attractiveness (demand divided by travel time) of the taxi stand or node using historical data[1]
3. Total attractiveness (demand divided by travel time) of the taxi stand or node using real time data[1]

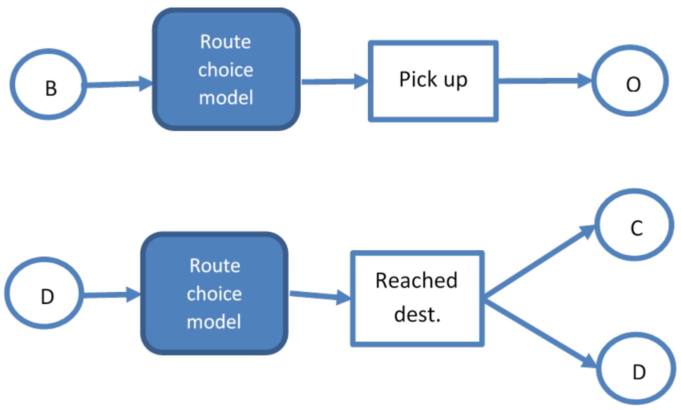
4. CBD (yes/no)
5. Journey time to the potential destination

M5. Acceptance model
This model will estimate the probability of taxi driver to accept or reject a pickup request which comes from the controller.

Identified variables:
1. Characteristics of the taxi driver
2. Expected revenue from the trip[3]
3. CBD (yes/no)
4. Journey time to destination
5. Time to end of shift
6. Willingness/availability to take an additional passenger[4]

3.7 The links between the taxi-driver state, events and models

The relationship between the taxi-driver state and events is given in section 3.3.1, in the following figures we link the events and states to the taxi driver behavior models introduced in this section.



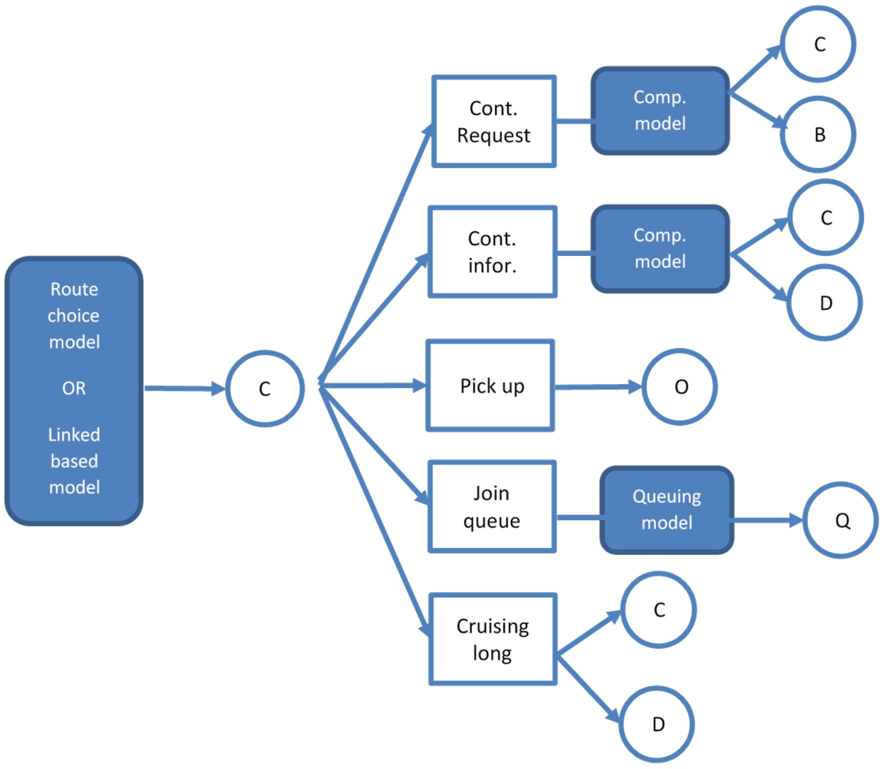
[3] Distance based revenue.

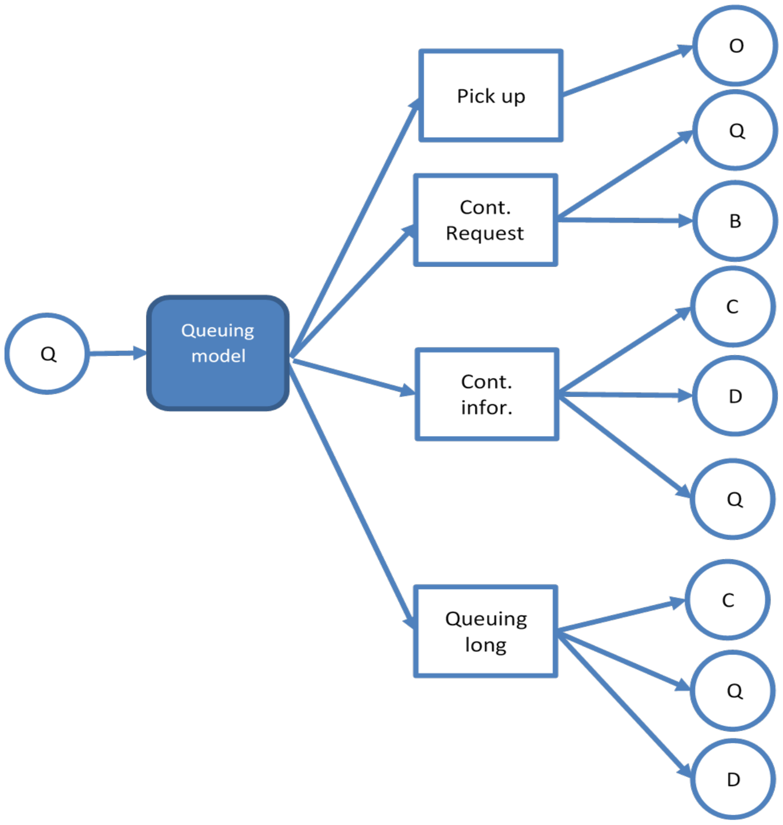
[4] Binary variable that allow booking with following three conditions:

$$\text{Distance to Client}_2\ (S) \leq 5\text{km}$$

$$\text{Travel} - \text{time to Client2} = \frac{\text{Distance to Client}_2}{\sum_{s=1}^{S} \bar{v}_s / S} \leq 10min$$

$$\text{Distance gap} = \sqrt{((X_{Des_{Client1}} - X_{Client2})^2 + (Y_{Des_{Client1}} - Y_{Des_{Client2}})^2)} \leq 5km$$


Route choice model
OR
Linked based model
C
Cont. Request
Comp. model
C
B
Cont. infor.
Comp. model
C
D
Pick up
O
Join queue
Queuing model
Q
Cruising long
C
D


Q
Queuing model
Pick up
O
Cont. Request
Q
B
Cont. infor.
C
D
Q
Queuing long
C
Q
D

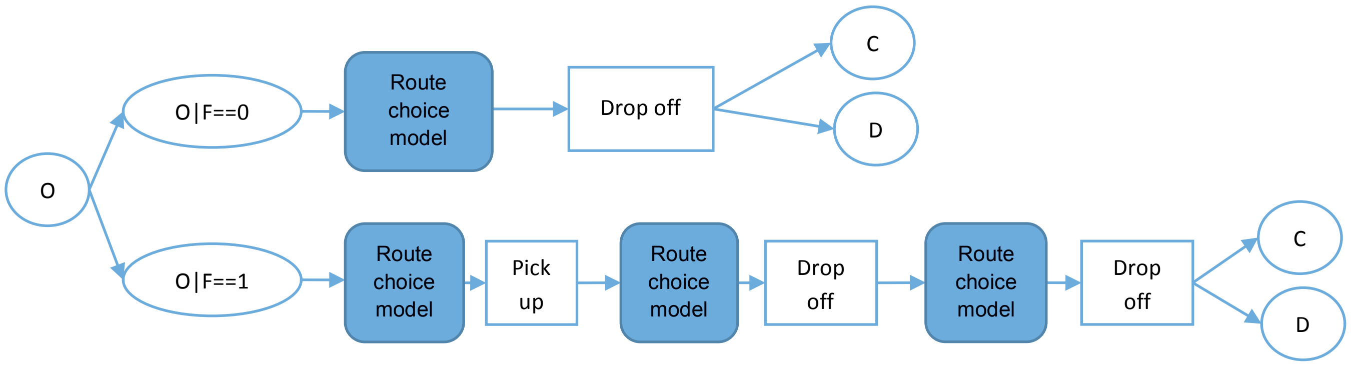


Figure 3: The links between the taxi-driver state, events and models

3.8 Taxi parking

Within this framework, taxis will able to stop at any stop bay, which will be considered as a taxi stand; this movement will be similar to bus stop simulation in SimMobility. Taxis do not park on the street when they pick up client. As mentioned, the traveller agent will be teleported to the cab without stopping. Taxis do not park at the end of the shift, they pass from one driver to anther; some of them (randomly picked) are “killed” to balance the number of active taxies on the road (see Appendix A: the distribution of taxi fleet by time of day).

4. IMPLEMENTATION

4.1 Introduction

SimMobility is a agent-based simulation application, in order to implement taxi roaming, some kinds of agents should be created to perform different behaviors, those agents will collaborate together to realize the simulation of taxi roaming and at the same time some statistics can be collected for the analysis about simulation result of taxi roaming .

4.2 Taxi Agent Class Diagrams

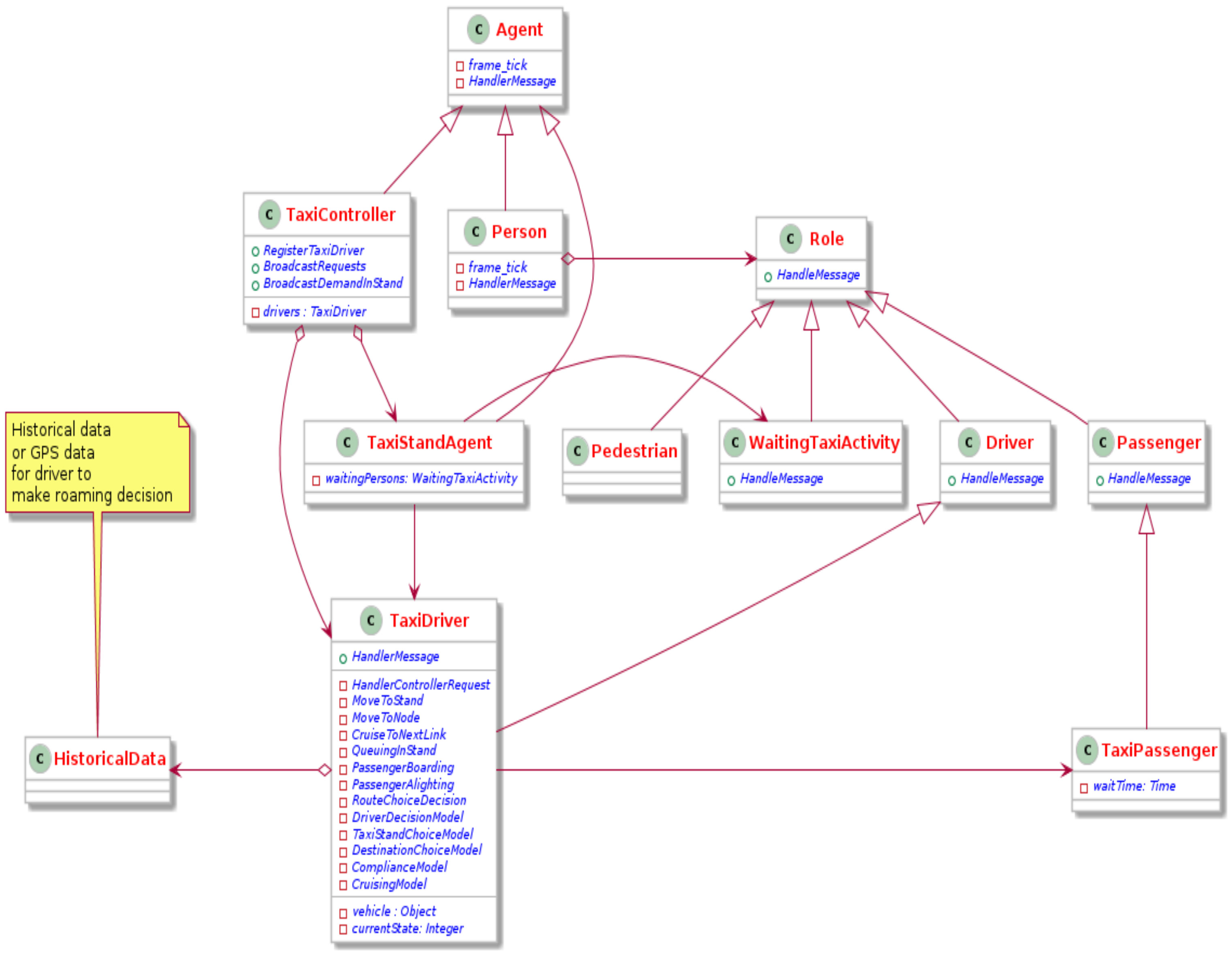
Some new classes are created to represent different objects: TaxiController, TaxiDriver, TaxiPassenger, DemandCollector. TaxiController include all the references to the active taxi-drivers and can send request to each individual taxi-driver. TaxiDriver will perform taxi roaming simulation on the road network; TaxiPassneger will wait on the road or taxi-stand for a taxi and record waiting-time; DemandCollector will collect space and time distribution of passengers.

- The class “TaxiDriver”, this class represent taxi driver, which is inherited from software agent and will update himself at every frame tick. In this class, many functions are designed to perform different tasks:
- The function 'HandleControllerRequest' will process the request message broadcasted from TaxiController.
- The function 'MoveToNextStand' will perform the movement to the selected destination taxi-stand.
- The function 'MoveToNextNode' will perform the movement to the selected destination node.
- The function 'Cruise' will perform the cruising movement from one link to next link until the behaviour is changed.
- The function 'QueuingInStand' will perform the queuing model when this driver arrive at a given taxi-stand.
- The function 'PassengerBoarding' will move a waiting passenger inside the taxi and switch its state to the occupied state.
- The function 'PassengerAlighting' will release a passenger to the destination and swith its state to the idle state.
- The function 'RouteChoiceDecision' will decide a concrete route when a behaviour is made or a passenger is picked up.
- The function 'DriverDecisionModel' will estimate next behavior between cruising and Direct-to-destination.
- The function 'TaxiStandChoiceModel' will choose one particular taxi-stand from many stands.
- The function 'DestinationChoiceModel' will choose one particular node as the destination from many nodes.
- The function 'ComplianceModel' will make a decision whether to accept the order when some requested messages are received from taxi controller.

- The function 'CrusingModel' will estimate cruising behavior and decide next link.

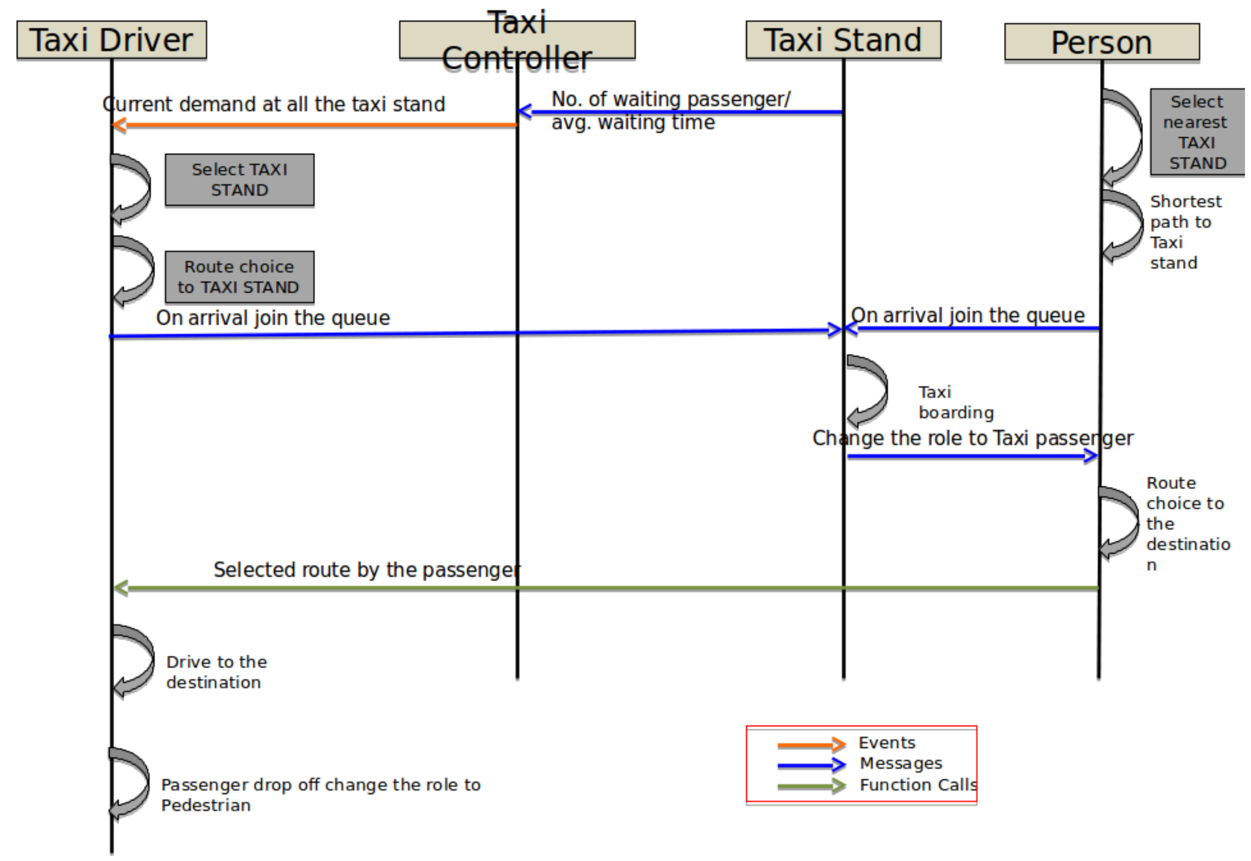
- The class “TaxiController”, this class represent a control component, which will keep all the references of active taxi-drivers in order to communicate with them.
- The function 'RegisterTaxiDriver' will register a reference inside the taxi-controller for future communication.
- The function 'BroadcastRequests' will send a request message to some drivers
- The function 'BroadcastDemandInStand' will send demand information at taxi-stand to some drivers for compliance decision
- The class “TaxiStandAgent” this class represent a taxi-stand, which collect waiting person for taxi-drivers and record waiting-times.
- The class “WaitingTaxiActivity” this class represent the waiting person for an taxi and record waiting-time at the same time.
- The class “TaxiPassenger” this class represent the taxi-passenger in an taxi and record travel-time at the same time.

The Class Diagram of Taxi Driver In SimMobility

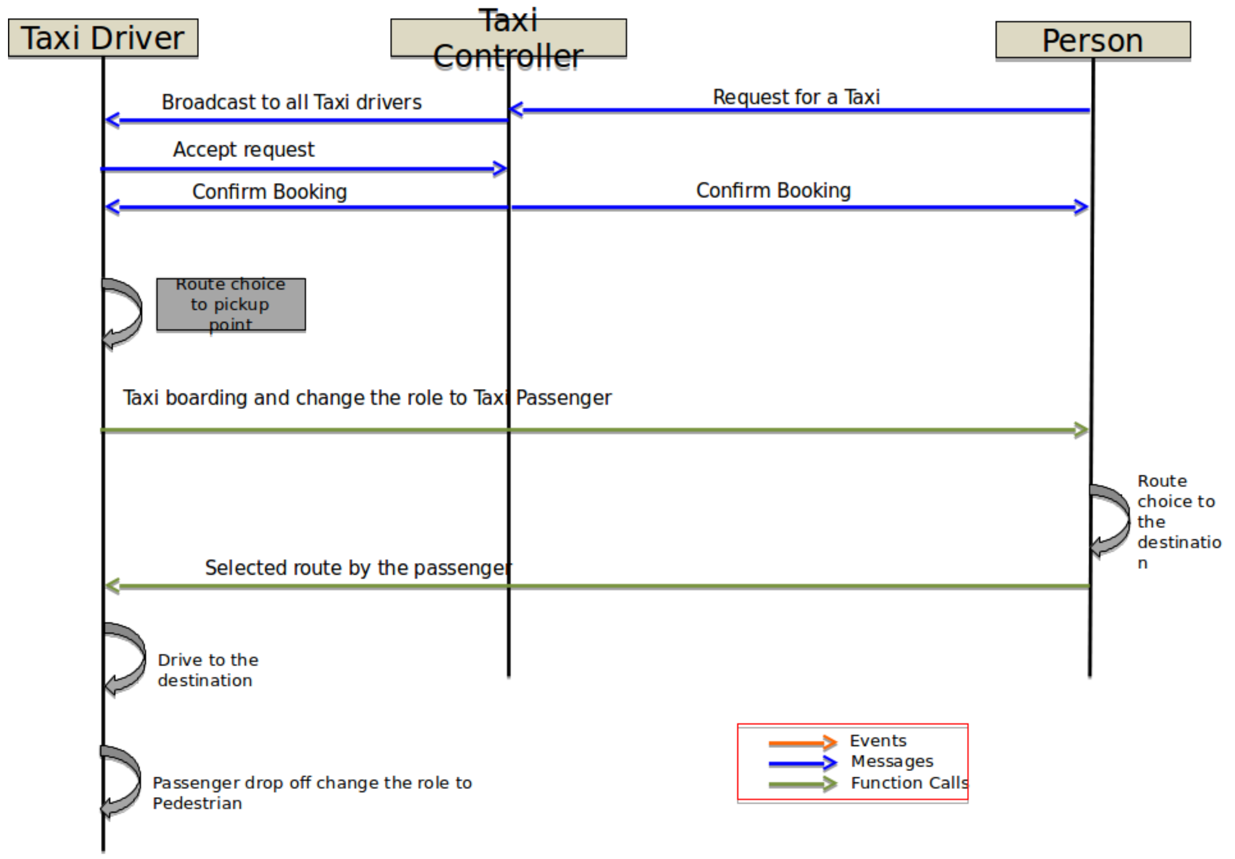


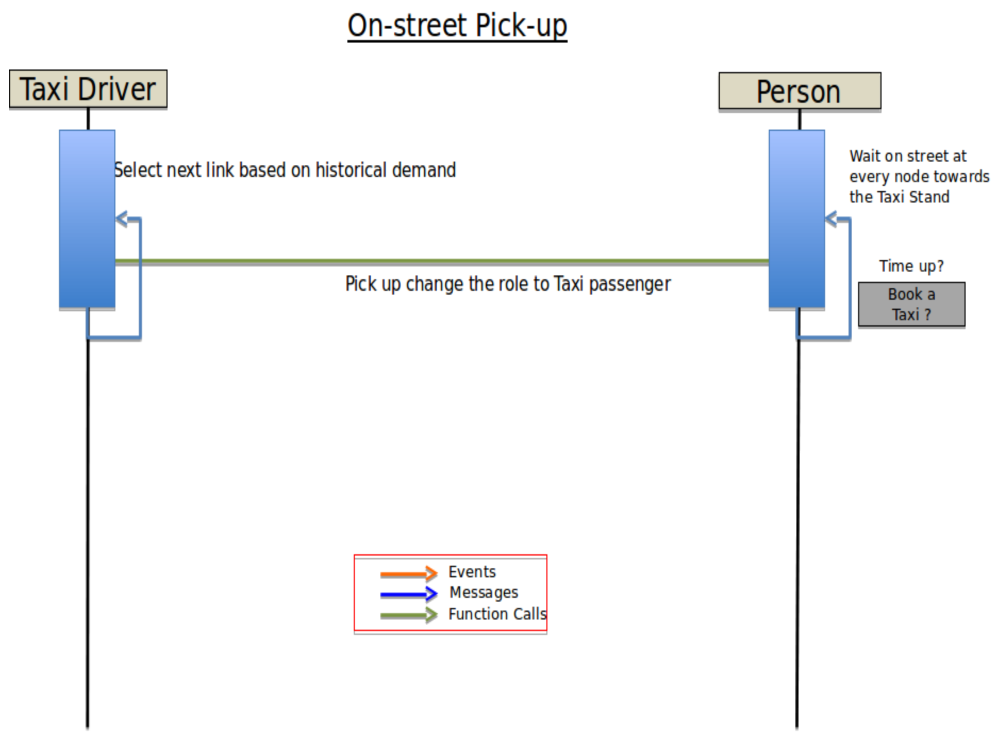
4.3 Sequence diagram between agents related to Taxi-roaming

Sequence Diagram at Taxi Stand


Taxi Driver
Taxi Controller
Taxi Stand
Person
Current demand at all the taxi stand
No. of waiting passenger/ avg. waiting time
Select nearest TAXI STAND
Select TAXI STAND
Shortest path to Taxi stand
Route choice to TAXI STAND
On arrival join the queue
On arrival join the queue
Taxi boarding
Change the role to Taxi passenger
Route choice to the destinatio n
Selected route by the passenger
Drive to the destination
Passenger drop off change the role to Pedestrian
Events
Messages
Function Calls

Book a Taxi


Taxi Driver
Taxi Controller
Person
Request for a Taxi
Broadcast to all Taxi drivers
Accept request
Confirm Booking
Confirm Booking
Route choice to pickup point
Taxi boarding and change the role to Taxi Passenger
Route choice to the destinatio n
Selected route by the passenger
Drive to the destination
Passenger drop off change the role to Pedestrian
Events
Messages
Function Calls


On-street Pick-up
Taxi Driver
Person
Select next link based on historical demand
Wait on street at every node towards the Taxi Stand
Pick up change the role to Taxi passenger
Time up?
Book a Taxi ?
Events
Messages
Function Calls

4.4 Output

APPENDIX A: THE DISTRIBUTION OF TAXI FLEET BY TIME OF DAY AND SHIFT DURATION

Taxi fleet distribution according to LTA records is as follows:

6am – 7am and 11pm – 12mn: 60% of the fleet

7am – 11am and 5pm – 11pm: 85% of the fleet

The graph below shows the distribution of taxi fleet by time of day, on a typical day, according to taxi GPS data from 2013. The blue bares represent the distribution of the entire taxi fleet by time of day. The red bares represent the distribution of the taxis that are occupied only.

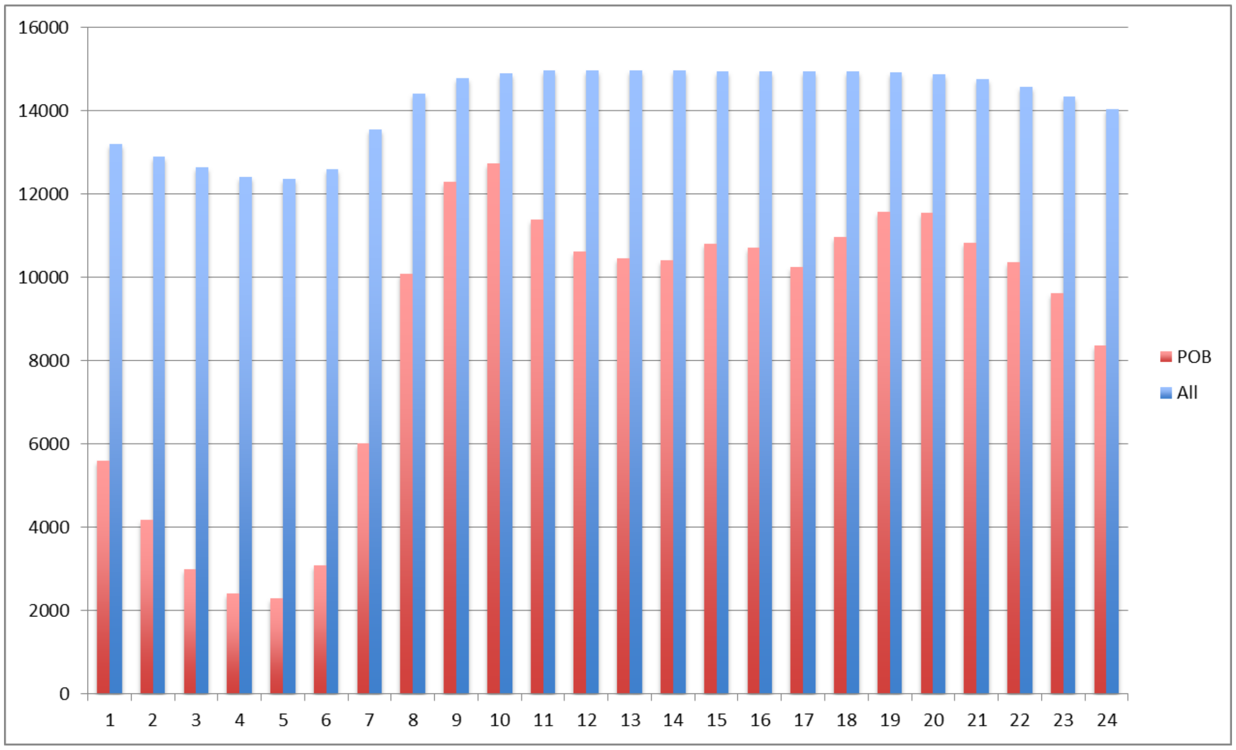


Figure A1: Taxi fleet dispersion by time of day

Taxi vehicle sharing among drivers are as follows:

- Taxi with 1 driver: 5866 (38%)
- Taxi with 2 drivers: 8718 (57%)
- Taxi with more than 2 drivers: 681 (5%)

The graph below shows the distribution of shift duration among taxi drivers, on a typical day, according to taxi GPS data from 2013.

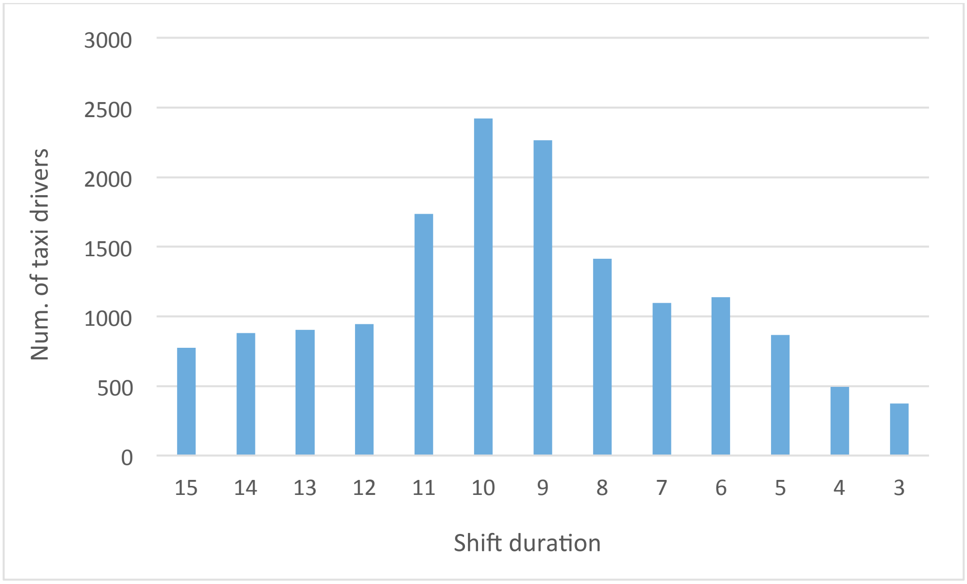


Figure A2: Shift duration distribution among taxi drivers

Taxi GPS data proccesing

Stage 1: cleaning

1. if BUSY then it should be the state before it became BUSY
2. if 'STC' then 'FREE'
3. if 'PICKUP' then 'ARRIVED'
4. if 'NOSHOW' then 'FREE'
5. if Break, off line, FORWARD or POWEROFF then Out of service

Stage 2: main script

State	Description	comments
Booked (B)	On call	
Occupied (O)	POB	
Queuing (Q)	If (Free and X,Y==taxi stand)	taxi stand is defined within a buffer of 5 meters
Cruising (C)	If (Free and X,Y<>taxi stand)	
Direct-to-destination (D)	If ((Free and X,Y<>taxi stand keep going until X,Y==taxi stand) and no POB before reaching taxi stand)	
Free-to-additional passenger (F)	In the future	
Out of service	Break, off line, FORWARD or POWEROFF	
Pick up	If (POB and before diff state, pick up)	
Drop off	If (Payment and after change in state, drop off)	
Rejection of request from controller	Busy	
End of shift	Off line and no driver ID	Filled the blanks v 1
Change of shift	Change in driver ID	Filled the blanks with -1

```
if driver id is blank and driver id before == driver id after then fill with driver id
else
if driver id is blank and driver id before <> driver id after then fill with -1
```

<u>Stage 3: improvements</u>

1. if FREE before payment change FREE to OCCUPIED
2. if driver ID is -1 the stat should be OUT OF SERVICE (we discussed already)
3. if there are CONSECUTIVE drop off, keep the latest and change the other drop off to OCCUPIED
4. if there are CONSECUTIVE pickup, keep the latest and change the other pick up to CRUISING
5. if there are CONSECUTIVE payment, keep the latest and change the other payment to OCCUPIED