Agent-Based Approach to Simulating Mobility as a Service

by

David D. Li

B.S. Computer Science and Engineering, Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author .....
Department of Electrical Engineering and Computer Science
January 20, 2022

Certified by .....
Moshe Ben-Akiva
Edmund K. Turner Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by .....
Ali Shamshiripour
Research Scientist
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Agent-Based Approach to Simulating Mobility as a Service

by

David D. Li

Submitted to the Department of Electrical Engineering and Computer Science
on January 20, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As a result of the changing transportation landscape, Mobility-as-a-Service (MaaS) was developed to be a streamlined operator of emerging on-demand transportation services and traditional modes of transport. However, much of MaaS's impact on end-user's activities and travel patterns remain unknown and require further investigation. Due to its complex nature, a tool is necessary to help us reliably quantify and evaluate the broader impacts of MaaS. To this end, we introduce MaaS into the activity-based, agent-based travel simulation platform: SimMobility. Prioritizing flexibility and compatibility with different cities, we provide a generic implementation on which users can define configurations according to desired scenarios.

Thesis Supervisor: Moshe Ben-Akiva
Title: Edmund K. Turner Professor of Civil and Environmental Engineering

Thesis Supervisor: Ali Shamshiripour
Title: Research Scientist

Acknowledgments

I would like to thank my advisor Ali Shamshiripour for being a great mentor. To Professor Moshe Ben-Akiva, I am deeply grateful for the opportunity to work in your lab. To Ford Motor Company, thank you for sponsoring the MaaS project.

Additionally, I would like to thank Lampros Yfantis, Emma DeSoto, Carlos Lima Azevedo, Ravi Seshadri for all the helpful discussions.

Finally, I would like to thank my family for their unconditional support.

Contents

1 Introduction **13**

1.1 Motivation 13

1.2 Background 16

2 Overview of SimMobility Infrastructure **19**

2.1 Plan Versus Action 20

2.2 Controllers 21

3 Enhancements **23**

3.1 MaaS Accounts 24

3.2 MaaS Controller 25

3.2.1 Communication Channels 25

3.2.2 Menu Construction 27

3.3 Modifications on Existing Controllers 30

3.4 Agent-Side Modifications 31

3.5 Attribute Estimation 32

4 Results **35**

5 Conclusion **39**

A Tables **41**

B Code **43**

List of Figures

2-1 SimMobility Framework 20

2-2 SimMobility Mid-Term Model 21

3-1 Integration of the MaaS controller in SimMobility 23

3-2 Supported modes in SimMobility and their MaaS variants 24

3-3 High-level example of menu usage 28

3-4 Sequence diagram of the MaaS protocol in SimMobility 30

4-1 Trip completion rate at different MaaS penetration levels 36

4-2 SimMobility runtime at different MaaS penetration levels 36

List of Tables

3.1 Example of existing message types 26
3.2 New message types 27

A.1 SimMobility maas_accounts table 41
A.2 SimMobility maas_plans table 41
A.3 Relevant columns of precomputed attributes tables (public transit and multimodal variants) 42
A.4 Relevant columns of precomputed attributes tables (cars) 42

Chapter 1

Introduction

Private mobility tool ownership has long been a cornerstone in transport systems. Mobility tools such as cars and motorcycles usually feature a high up-front cost and relatively low marginal cost to use. As a result, long-term ownership decisions can have a strong influence on short-term travel behaviors (e.g. car owners are less likely to take bus rides) [9]. In recent years, the growing prominence of smartphones and the Internet of things (IoT) in the digital age has fueled a shift away from private ownership and towards digitized, flexible transport models, which holds promise in reducing sunk cost biases and enhancing mobility.

1.1 Motivation

The rising popularity of the ride-hailing model and applications such as Uber and Lyft showcases the market demand and shift towards digitized traveler mobility solutions. These Mobility-on-Demand (MoD) systems have laid the foundations and began to address concerns with mode choice biases. Without long-term mobility tools being considered, mode decisions can be driven more by short-term costs. Despite these innovations, most mobility services remain largely disjointed - different services are accessible only through their own platforms.

Mobility-as-a-Service (MaaS) is a user-centric, digital and intelligent mobility service provision model with which passenger travel needs are met via a single, integrated

platform. Building upon the momentum and innovation created by existing mobility services, the concept offers multiple mobility services on one central smartphone application, often supported by a pay-per-use or subscription payment model [16][15]. The reform aims to bridge the gap between public and private operators by integrating the currently fragmented services required to conduct a trip. In doing so, proponents of MaaS argue successful integration of these components can reduce reliance on private vehicles, potentially driving down congestion and harmful external costs [12][5][2][8].

Building on MoD, MaaS works towards minimizing sunk cost and bias while offering more options and enabling integration of multiple modes into one trip. In particular, a MaaS provider must fulfill two requirements [9]:

1. integrate supported mobility services strategically and operationally.

2. support a user interface, with which users can make payments and interact with all services.

Due to its novelty, more research is required in order to determine the role of MaaS in future mobility systems. There have been numerous pilots, designs, and consumer surveys on MaaS in recent years. Another approach is through simulations, which has been popular in assessing the future of transport [17][14][3][13]. Based on the relatively small, but growing number of agent-based simulations looking at the impacts of MaaS, one can identify a key gap: lack of native support for both MaaS requirements on simulators. Consequently, existing simulation-based MaaS studies rely on specialized augmentations to simulators. A generalized MaaS simulator implementation with both mobility service integration and user interface may be beneficial. Such an approach would expand the set of input cities and enable simulations on different MaaS configurations (e.g. supported modes, payment models, etc.) without requiring changes to the underlying software.

Transport simulators have proven to be an efficient and sustainable approach to studying the impacts of MaaS on large city systems. SimMobility is an activity-based, agent-based travel simulation platform on which emerging modes can be evaluated. It

integrates various mobility-sensitive behavioral models with simulators to predict the impact of mobility demands on transportation networks, intelligent transportation services and vehicular emissions. The platform enables the simulation of the effects of a portfolio of technology, policy and investment options under alternative future scenarios. In particular, SimMobility encompasses the modeling of millions of agents, including pedestrians, drivers, trains, etc. [1].

Due to the complex nature of MaaS systems, explicit modeling and simulation of relevant components is necessary to ensure consistent evaluations. One proposal details a roadmap aimed at replicating the intricacies of MaaS systems and its components. The roadmap outlines five major components: MaaS market model, demand modeling framework, multi-modal network model, MaaS integration controller, and mobility service controllers [4]. Most relevant is the paper's description of a MaaS integration controller, which fits well within the SimMobility framework and is a major component of this project. The MaaS controller is described as a model designed explicitly for the purpose of interacting with the demand (users) and supply (mobility service controllers).

Currently, SimMobility does not feature the ability to simulate the effects of MaaS services. For this project, we extend existing mobility implementations and integrate MaaS functionalities within SimMobility's demand and supply framework. Prioritizing fidelity, the project aims to faithfully recreate MaaS design characteristics such as integration with mobility services and traveler-facing interfaces within the SimMobility simulation environment. To support future studies on MaaS or including MaaS, all added modules support a degree of user configurability. Successfully implementing generalized MaaS simulation capabilities in SimMobility will allow us to more closely evaluate and quantify the broader impacts and implications of MaaS in cities around the globe.

1.2 Background

Fueled by a generational shift in consumer attitude towards ownership and developments in technology, the past decade saw the boom of the “as-a-service" model [12]. Notably, rapid success and innovation in mobility services such as bike-sharing and MoD illustrate a trend away from private mobility tool ownership and towards digitization of personal transportation solutions [7]. These developments are significant as private cars have long dominated transportation in industrialized countries. However, these “as-a-service” schemes bear its own set of shortcomings.

As noted by [6] and [18], the new mobility options show promise in extending existing public transport networks. In areas without robust mobility options, the on-demand format can serve as connections to fixed transit routes. However, mobility providers still largely operate independently, making mode combinations difficult [9]. In addition, such a scheme has not been tested on a large scale. Nonetheless, this idea of “Uberization” of public transport presents an opportunity for a more inclusive transport model [10]. Indeed, these developments have led to the popularization of a more unified system aimed at serving individualized travel needs.

The proposed MaaS system hopes to solve fragmentation between the providers in both the public and private sector. Moreover, it is possible that such an approach would create more attractive travel alternatives to private vehicles. Early, small-scale studies yielded results supporting this expectation. For example, a 2016 field trial conducted in Gothenburg, Sweden allowed individuals to gain access to cars, car-sharing, and public transport through monthly credits [11]. The results show that these MaaS users tend to decrease private car usage in favor of using public transit. Another study looks at the possibility of using MaaS to promote shared modes of transportation. The survey finds that although shared modes are not particularly desired, a large portion of the respondents are willing to subscribe to MaaS plans with shared plans. Furthermore, a significant portion (60%) of potential subscribers expressed willingness to try new transportation modes included in the MaaS plan [12]. This finding points towards the potential of using MaaS to draw more travelers

to shared modes.

Aside from the integration of participating mobility services, the MaaS scheme also requires a consumer-facing interface, on which travelers can make payments, access trip information, and book rides:

1. For payment methods, MaaS schemes offer one or both of pay-per-use or pre-paid monthly subscriptions. Functionally, the pay-per-use model is identical to booking rides with providers individually; each leg is priced separately by the provider. However, this payment model is able to leverage the benefits of an integrated platform. MaaS applications can act as search engines, aggregating information from different providers and constructing optimal trips for users. With the subscription model, subscribers pay monthly for a bundle of services that best meets the traveler's needs. In turn, the MaaS operator purchases mobility services in bulk, thereby providing guarantees to the user. The structure of these subscription plans can be decided by the MaaS operator. A generic bundle example of a MaaS subscription plan would include a list of modes, each associated with an allowance and discount. One advantage of bundling is that an individual only needs to pay one price regardless if the trip includes multiple modes.

2. A simple representation of trip mode options is the idea of a menu. To better represent the complex dynamics between demand and supply entities with regards to MaaS, a menu is needed to fully capture their interactions. Namely, the menu serves as a bridge between the supplier (MaaS and mobility service providers) and travelers, with which users can make decisions based on the attributes of offerings. This is exemplified by the menu option and traveler mode choice system of the Tripod project, which heavily influences this project [19]. Tripod is a smartphone system that influences travelers' trip decisions based on real-time information and incentives. This is accomplished by having users access Tripod's personalized menu and make decisions based on the information shown. In addition to requesting mode options and booking rides, the interface

must also feature the ability to make payments.

3. Proper integration with mobility service providers imply both business and technical requirements are fulfilled (e.g. ride request APIs, mobile ticketing for buses, etc.). From the user's perspective, the process for booking rides with a MaaS provider is the same as booking with any individual provider. In this instance, the platform will act as both a planner and intermediary, satisfying requests and communicating with relevant providers.

As with all the recent emerging mobility services, much is still unknown about MaaS. There is limited knowledge regarding how these new services will interact with existing modes as well as with each other. Compared to established modes, these tech-driven modern mobility solutions are still in their infancy. Representative field trials can feasibly only be conducted on small scales, and the variety of concepts make predicting impact on large transportation systems strenuous. Additionally, knowledge gained from one study environment does not always transfer to another scheme or city [9]. It is likely that MaaS's impact on different communities will vary, making the results of studies conducted in specific cities difficult to generalize. These issues make simulations a favorable approach for insight discovery for emerging modes.

Chapter 2

Overview of SimMobility Infrastructure

SimMobility takes an agent-based approach to traffic simulation. In many cases, simulations can include millions of agents such as pedestrians and drivers. The simulator features three interconnected levels of temporal granularity: Short-Term (ST), Mid-Term (MT), and Long-Term (LT). The ST simulator considers events such as lane-changing and braking, which happen on the order of tenth of a second. The MT simulator focuses on agents' behaviors on a mesoscopic scale, including daily activities and travel patterns. The LT simulator captures land use and economic activities such as property developments and job relocations.

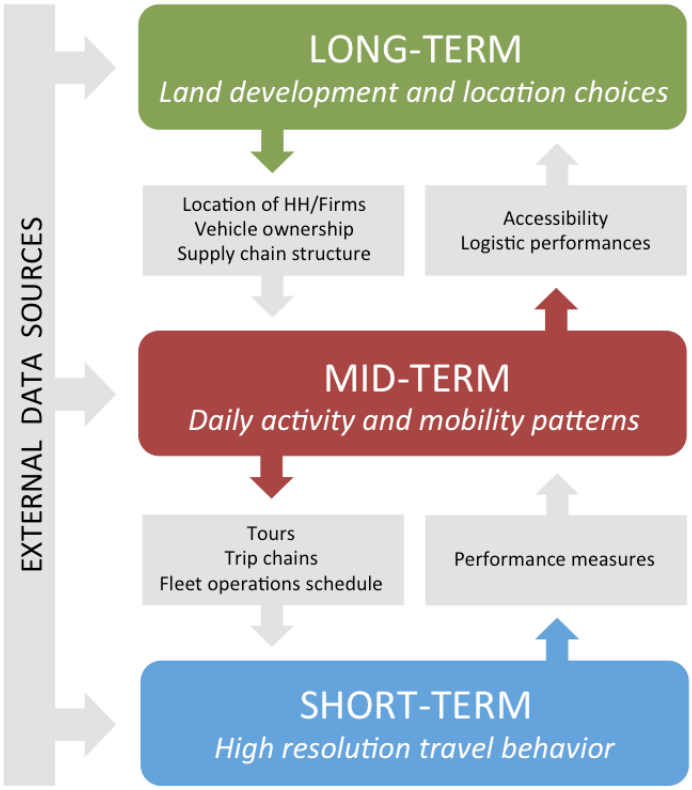

Figure 2-1: SimMobility Framework

2.1 Plan Versus Action

MaaS will largely impact agents' daily travel behavior. As such, all designs and implementations for this project occur under the scope of the MT simulator. Taking a closer look at the MT simulator framework, demand is modeled by two component groups: pre-day and within-day. At the high-level, agents' daily plans are generated using pre-day models, while the actual execution of these plans occurs during within-day simulations.

SimMobility takes an activity-based approach; pre-day models decide agents' initial daily plan. This plan - also known as a Daily Activity Schedule (DAS) - includes information such as the agent's activity sequence, preferred modes, and departure times. With the DAS determined, individuals utilize within-day models to further decide routes for trips and execution plans. The within-day is where demand meets supply. One of the focal points of this project is to alter these within-day mecha-

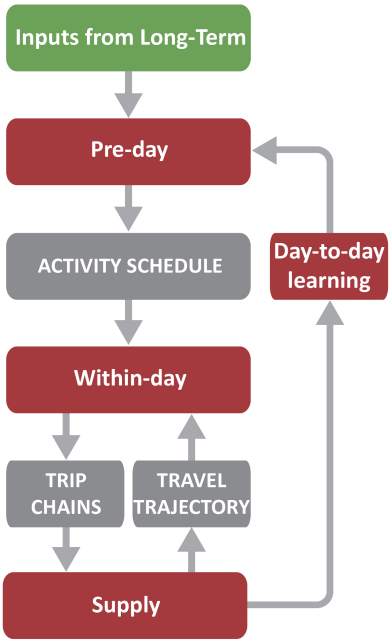

Figure 2-2: SimMobility Mid-Term Model

nisms to accommodate the MaaS infrastructure and newly added models. On the supply side, we introduce the MaaS operator (implemented as a new controller in SimMobility) and update associated existing components.

2.2 Controllers

Controllers are one of the central components driving SimMobility's supply side abstraction. The MT simulator framework allows for the plug and play of different controllers, each in charge of some mode(s). Public transport bus and train controllers are similar in that both handle vehicle movements on the network. Buses are dispatched by the bus controller at scheduled times while trains will move along predefined tracks and stop at station platforms. Another set of controllers - generalized as mobility service controllers - represent mobility services providers such as taxi services and Uber.

In SimMobility, this group of mobility service controllers are broken down further into type OnHailTaxi (on-hail controller) or type OnCall (on-call controller) respectively. On-hail controllers are simple in that their only functionality is to authorize

taxi drivers to operate on the streets. Additional considerations such as who to pick up and where to cruise are decided by the drivers independently. On-call controllers have an additional layer of complexity; they have to fulfill requests from passengers and coordinate with drivers regarding where to pick up, drop off, etc. In SimMobility, on-call controllers are the only controllers that require this ride booking mechanism (to mimic ride-sharing applications). All other modes can be carried out without notifying the provider (e.g. riders can board trains independently, passengers can hail cabs without prior communication, etc.). SimMobility users can define a set of settings for controllers in the simulation configuration XML file.

Controllers run in parallel and are configurable by the user. In theory, Individual agents have access to any of the enabled controllers. In practice, however, agent communication with controllers is largely limited to trip requests, which are only sent to the controller associated with the predetermined mode specified for the trip. This behavior will change with the integration of MaaS, as MaaS users will query information from all associated controllers through the MaaS controller.

Once a trip request has been received and successfully processed by a on-call controller, the actual trip will be assigned to and carried out by a driver. This backend coordination is done only between the on-call provider and individual driver(s). For MaaS users, trip requests are sent to the MaaS controller as opposed to the on-call controller. In this instance, the MaaS controller simply acts as an intermediary between the agent and the relevant on-call controller. As MaaS providers and on-call providers are separate entities, internal functions of these controllers are kept private and not exposed unless otherwise specified. Note that the integration of MaaS does not degrade the utility of existing controllers. Furthermore, for MaaS users, the MaaS controller preserves the functionalities expected of all service providers.

Chapter 3

Enhancements

SimMobility must be able to mimic MaaS's real-life functionalities and properly simulate the application's interactions with users. This involves creating a MaaS controller and integrating it with all the existing mobility service controllers. The proposed architecture is as follows:

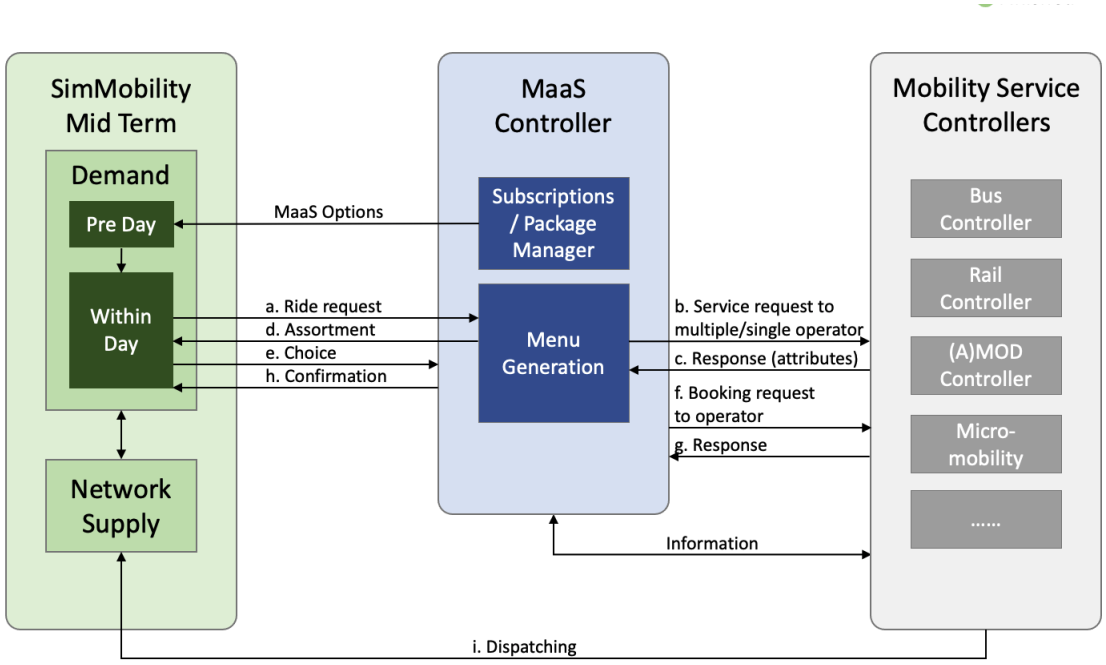

Figure 3-1: Integration of the MaaS controller in SimMobility

The resulting supported modes are shown in Figure 3-2. Functionally, modes

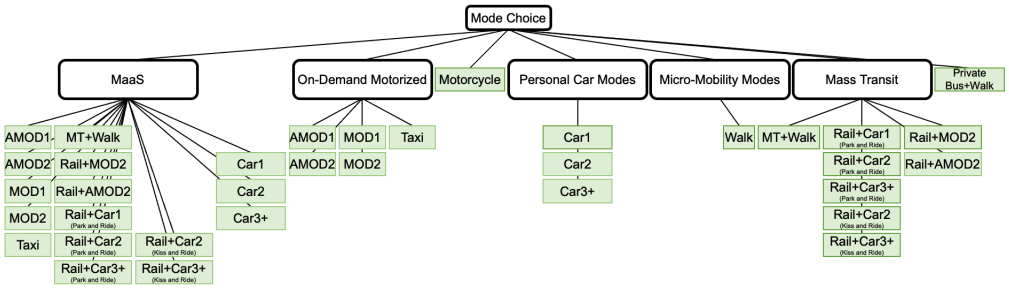

Figure 3-2: Supported modes in SimMobility and their MaaS variants

accessed through the MaaS operator are tagged with a "MaaS" prefix to differentiate between MaaS and non-MaaS variants of the same mode.

3.1 MaaS Accounts

Current MaaS proposals are structured around collections of discounts and allowances bundled into MaaS subscription plans. Individuals can then subscribe to any one specific plan. Discounts and allowances are allocated on a per-mode basis, where MaaS plans may include different modes, each may be at a different discounted price (discount) and number of uses (allowance). An agent's MaaS account describes the agent's current allowance with regards to available MaaS usage and associated MaaS plan. In our implementation, SimMobility users can define these MaaS plans. For example, it is possible for a MaaS plan to have the same discount rate across all modes and infinite allowances. Conversely, it is also possible to have a different MaaS plan in the same simulation with well-defined discount rates and allowances per mode.

In SimMobility, we model whether an individual subscribes to a MaaS plan in the pre-day simulation, and act accordingly in the within-day. The MaaS controller must handle each of these cases; if a user is subscribed to a MaaS plan, the MaaS controller will first check the individual's remaining allowance for each mode. If the user has enough, the proper discount will be applied to the cost shown in the menu. Otherwise, no discounts will be applied. Non-subscribers are still able to ask the MaaS controller for mode alternatives. However, no discounts would be given. This

case is implicitly accounted for since those who are not subscribed to a MaaS plan will have an allowance of zero for all modes.

In our implementation of the MaaS account, we utilize several SQL tables to ensure the MaaS controller's ability to properly calculate cost per mode. Namely, the two tables we use are maas_accounts and maas_plans, described in detail in Tables A.1, A.2. Agent accounts are loaded using an on-demand policy (i.e. an agent account is queried from the database only if that agent has an upcoming trip in the simulation). Since SimMobility is agent-based and many scenarios can include millions of agents, this paradigm is critical to ensure scalability.

3.2 MaaS Controller

Since MaaS is fundamentally an aggregator of different mobility offerings, its underlying structure differs from other service providers in SimMobility. It is an interface that sits between the user and all mobility services controllers. Its functionality relies on its connection with individual users and service providers. Thus, our main task is to integrate the MaaS operator into the existing supply infrastructure and build communication channels around the MaaS controller and relevant components (as shown in Figure 3-1).

3.2.1 Communication Channels

In SimMobility, entities such as agents and controllers communicate via messages sent over a message bus. Since SimMobility is a multi-threaded program, a message bus is necessary to transfer messages across contexts. These messages are structures that can hold a variety of information, ranging from driver sign-ons to completed trip statistics. As a generalized message can serve a host of uses, each message's purpose is differentiated by a message type tag. Some examples of message types are listed in Table 3.1.

Since coordination with drivers is done privately between the driver(s) and on-call controller, MaaS is only concerned with passenger-facing message types such as

Message Type	Description	Message Endpoints
MSG_DRIVER_SUBSCRIBE	Adds the driver to the service provider's list of available drivers.	Driver → Controller
MSG_DRIVER_UNSUBSCRIBE	Removes the driver to the service provider's list of available drivers.	Driver → Controller
MSG_TRIP_REQUEST	Passenger requests a origin/destination ride with an on-call controller.	Passenger → Controller
MSG_SCHEDULE_PROPOSITION	On-call controller sends a schedule proposition to a driver, specifying pickups and dropoffs.	Controller → Driver

Table 3.1: Example of existing message types

MSG_TRIP_REQUEST. As described in Section 2.2, the MaaS controller will not compromise the functionality of existing controllers. Additionally, as the intermediary abstraction between individual controllers, the MaaS controller will be able to handle all relevant requests resolvable by the controllers it represents. That is, any passenger request satisfiable by a mobility controller is also satisfiable by the MaaS controller. Furthermore, recall that trip requesting is only necessary for on-call controllers. Thus, the MaaS controller must be able to handle all expected MaaS functionalities along with all passenger-facing functionalities of on-call controllers.

Seen above, messages cover a range of endpoints: passenger-to-controller, controller-to-driver, driver-to-controller. Most notably, there are no controller-to-passenger and controller-to-controller communication channels in SimMobility. In order to fully represent the MaaS protocol and simulate agent decisions, we must establish a bi-directional communication channel amongst passengers and service providers. To accomplish this, we create channels and endpoints such that:

1. Passengers can receive messages from controllers (more details in Section 3.4).

2. The MaaS controller can send and receive messages from both passengers and other controllers.

3. Mobility service controllers can send and receive messages from the MaaS controller (more details in Section 3.3). Recall that although mobility service controllers do not directly send messages to passengers, transmitting information through the MaaS controller is sufficient.

We introduce a set of new message types, shown in Table 3.2, to facilitate these requirements.

Message Type	Description	Message Endpoints
MSG_TRIP_QUERY	Used when users request mode alternatives and information for a origin/destination pair.	Passenger → Controller Controller → Controller
MSG_MENU	Identifies messages containing menu information.	Controller → Controller Controller → Passenger
MSG_MENU_RESPONSE	Sent by the user to the MaaS controller when a mode decision has been made.	Passenger → Controller
MSG_TRIP_COMPLETE	The MaaS controller is notified once a user completes a trip.	Passenger → Controller
MSG_DRIVER_ASSIGNED	On-call controller notifies the MaaS controller a driver has been properly assigned to a requested trip.	Controller → Controller

Table 3.2: New message types

3.2.2 Menu Construction

Using the new message types, we are able to simulate the data (attribute) collection process of a functional MaaS service application. The MaaS trip request and booking protocol starts with the user initiating a menu request. This process reflects the act

of using a mobile application to check how to get from one location to another. A menu is an aggregation of possible transportations choices and relevant information associated with each choice. Attributes of different modes are queried from their respective mobility controllers. Once all information is gathered, they are compiled into an optimized menu of service alternatives that is then sent back to the user. The user then makes a decision based on the information presented in the menu and notifies the MaaS controller. The purpose of the menu is to help users understand the real-time mobility attributes and allow them to make grounded travel decisions. Design of the menu is heavily inspired by the Tripod project [19]. In our implementation in SimMobility, a menu is a collection of menu items, which consists of the estimated waiting time, travel time, cost, and associated path.

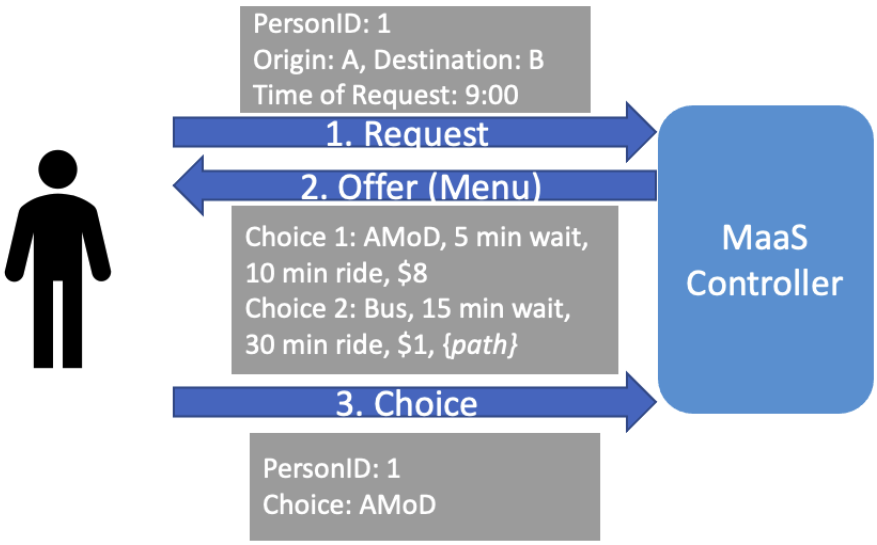

Figure 3-3: High-level example of menu usage

Once the MaaS controller receives a menu request via a MSG_TRIP_QUERY request with relevant information such as the requesting user's unique ID, origin, and destination, it will begin computing the menu. The menu generation process includes:

1. The MaaS controller sends out MSG_TRIP_QUERY requests to relevant, enabled service controllers.

2. Build up a personalized menu for the agent as the MSG_MENU messages come back.

3. Update menu item costs based on the requesting agent's MaaS plan and allowance.

4. Apply the user-defined filter function once all relevant service controllers have responded.

5. Sending a MSG_MENU message to the requesting user with a finalized menu.

At the end of the menu information aggregation process, we allow an additional layer of modularity via a user-defined menu filtering function. Without this filtering capability, it is likely that the menu will contain many different routes and options for each mode alternative, which will scale poorly as the network grows. For example, public transit options often have many different routes to get from point A to point B, some of them are strictly less optimal compared to others of the same mode. This is similar to how popular navigation applications typically only show the quickest or shortest distance options by default. To reflect this, SimMobility users can define a filter in the Lua format, a light-weight and beginner-friendly programming language. This filter serves to reduce the clutter in an unfiltered menu. It is possible for the user to define the filter as seen fit for any particular simulation setup. In our case, we filter each mode for the cheapest option by default, resulting in a final menu with one option per mode. Listing B.1 shows an example of a Lua function filtering for only the cheapest option per mode.

Once the agent makes a decision from the choices presented in the menu, a MSG_MENU_RESPONSE message will be sent to the MaaS controller. For transportation modes such as bus or rail, this notification is simply for bookkeeping purposes. For other modes such as SMS and other on-demand variations, the MaaS controller will continue by forwarding the trip request to the appropriate on-call controller. Once an agent completes a MaaS-affiliated trip, the agent will notify the MaaS controller with a MSG_TRIP_COMPLETE message. Again, this is for bookkeeping

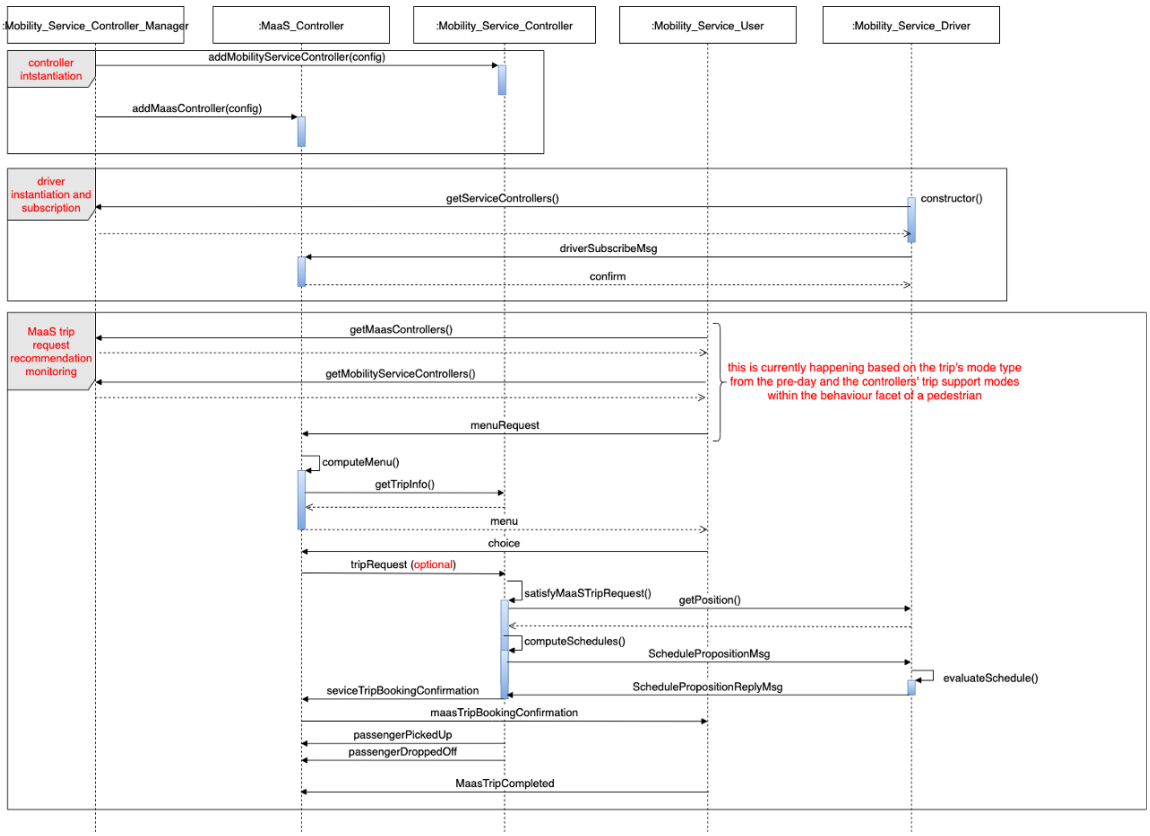

Figure 3-4: Sequence diagram of the MaaS protocol in SimMobility

purposes such as updating the passenger's MaaS account. The full sequence of events is as shown in Figure 3-4 above.

3.3 Modifications on Existing Controllers

To finalize the aforementioned bi-directional communication channels, we augment relevant controllers to be able to send messages to other controllers. Many controllers have existing message handlers capable of receiving messages. Thus, we build upon these features, and incorporate the ability to handle messages from both users and other controllers. The goal is to empower relevant controllers with the ability to handle MSG_TRIP_QUERY messages and properly reply with MSG_MENU messages.

Once the service controllers are capable of sending messages to the MaaS controller and users, we continued by modifying each controller such that trip attributes such as travel time and cost can be broadcasted. Doing so enables the MaaS controller

to query information about a trip from individual mobility service controllers. Ultimately, this querying functionality will allow the MaaS controller to systematically generate menus at the users' request.

3.4 Agent-Side Modifications

Unlike the previous implementation of trip generation in SimMobility (described in Section 2.1), MaaS trips do not have a determined plan regarding mode and route until time of execution. Instead of determining the entire trip at load time, MaaS trips initially only include the action of querying the MaaS controller. Further subtrips are generated only when a mode choice is made. This is done so that decisions regarding MaaS trips are made with the most up-to-date information.

With the addition of the MaaS communication framework, passengers can receive messages from other entities. Most importantly for this project, passengers receive menus from the MaaS controller via messaging. From the passenger's perspective, the sequence for using MaaS is as follows:

1. Send a menu request (implemented as a MSG_TRIP_QUERY message) to the MaaS controller and wait for a response.

2. Once a MSG_MENU is received, the passenger evaluates each menu item using a SimMobility user-defined Lua model to make a mode choice.

3. Build the rest of the current trip (routes, subtrips, etc.) with the decided mode.

4. Send a MSG_MENU_RESPONSE message to the MaaS controller informing the mode decision.

5. Send a MSG_TRIP_COMPLETE message to the MaaS controller once the trip is complete.

Similar to the filtering functionality described in Section 3.2.2, passenger decision on the choices presented can be modeled by a user-defined Lua script. Refer to Listing B.2 for an example.

3.5 Attribute Estimation

For performance, origin/destination (O/D) pair attributes are precomputed and stored in DB tables according to the mode. Public transport and their multimodal variants (e.g. Rail and MoD, Rail and Car, etc.) have individual tables. These mode-specific tables are structured so that each row contains information about a particular route. Since public transit lines are known and there can be more than one viable route, each O/D pair query can match multiple entries in the table, and therefore produce multiple routes with different attributes. As a result, each generated public transport menu item presented to the traveler is associated with a particular route.

For other modes like taxi, MoD, AMoD, and car, attribute DB tables are divided based on time of day instead of mode. The current implementation specifically precomputes three tables, each representing one of: morning rush, afternoon rush, and off-peak conditions and estimations. Since paths are decided by the driver, entries in these tables are not linked to any specific route. The associated controller of these modes will estimate attributes based on a combination of information retrieved from the table(s) and configuration set by the SimMobility user. Refer to Tables A.3 and A.4 for more detail. Attributes in the menu are generated as follows:

1. Taxis: Waiting time is 0 by default unless otherwise configured by the user. Depending on the time of day, query tables for travel time t from origin zone to destination zone. Estimating cost requires querying for distance d and computing the following function:

$$cost = B + fare_d * d + fare_t * t$$

   where base fare B, distance fare $fare_d$, and time fare $fare_t$ are defined by the SimMobility user.

2. Cars and shared cars: Procedure for estimating waiting time and travel time is the same as the procedure for taxis. O/D specific distance d, electronic road

pricing $cost_{erp}$, and parking fee P is queried to compute cost as follows:

$$cost = (cost_o * d + cost_{erp})/n + P$$

where operational cost $cost_o$ is defined by the user. n represents the number of individuals in the car.

3. MoD, MoD Pool, AMoD, AMoD Pool: According to mode and time of day, query the appropriate precomputed table for waiting time, travel time, and distance. To estimate cost, apply the following function:

$$cost = max(fare_{min}, B + fee_{service} + fare_d * d + fare_t * t)$$

where minimum fare $fare_{min}$, base fare B, service fee $fee_{service}$, distance fare $fare_d$, and time fare $fare_t$ are all defined by the user. Note that variables B, $fare_d$, $fare_t$ are different from those used to calculate cost for taxis.

4. Bus and rail: Waiting time, travel time, cost, and route are all determined by the result of querying the appropriate attribute table.

5. Multimodal trips: Given the requested O/D pair, query the combination-specific table for valid routes. Break the resulting route down into individual legs such that each leg uses only one mode. Estimate attributes according to the mode and O/D of each leg. For example, consider a multimodal route requiring an AMoD ride to a rail station before taking a train to the final destination. Firstly, separate the route into an AMoD leg and rail leg. Then, estimate attributes according to the procedure for each mode and intermediate O/D pairs (initial origin to rail station and rail station to final destination). Final attributes are calculated by taking the summation of individual attributes from all legs.

These methods for estimation are implemented with modularity in mind, meaning any of the procedures above can be easily swapped out by future users.

Chapter 4

Results

This section describes tests and results to ensure our implementation is correct, robust, and scalable. To this end, we run simulations using artificial test demands at different levels of MaaS penetration (i.e. 0%, 10%, 20%, etc. of all trips are MaaS based). Demand at each level features all supported modes, fed into SimMobility as 463000 trips and 224911 persons. As the MaaS penetration level increases, trips in this demand are randomly set to their MaaS mode variants. For the base case of 0% MaaS penetration, the MaaS controller is disabled to provide context on how SimMobility performs in the absence of MaaS entirely. Using an exemplary virtual city as input, all tests are conducted on a machine with a Intel Xeon CPU E5-2695 v4 @ 2.10GHz processor and 256 GB of RAM. In these test runs, all travelers planning to use a MaaS mode are subscribers of a generic bundle providing 50% discount to all supported modes.

For correctness, we examine whether MaaS trips are completed at the end of the simulation, where a trip is completed when a traveler has reached the requested destination. This indication is necessary and sufficient as a completed MaaS trip implies that the MaaS protocol is functional (i.e. the traveler is able to request information on trips, make decisions, book rides, etc.). Recall that MaaS trips are not fully constructed until a final mode choice has been made. The completion of such trips would also indicate that the real-time trip construction is correct. For our tests, we expect the trip completion rate to stay the same or increase as we increase the level

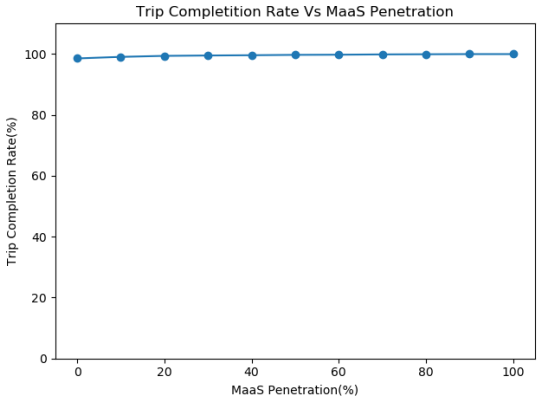

Figure 4-1: Trip completion rate at different MaaS penetration levels

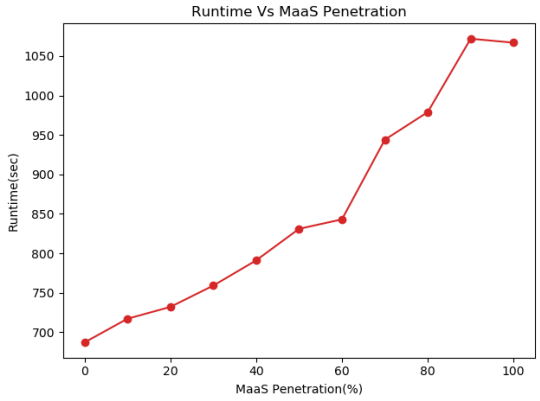

Figure 4-2: SimMobility runtime at different MaaS penetration levels

of MaaS penetrations. Confirmation of this trend shows that trips updated to their MaaS variants can still execute properly. Additionally, completion rate is likely to go up due to the MaaS controller only presenting viable options. As shown in Figure 4-1, our simulation runs confirms these expectations. Trip completion rate trends upwards as MaaS penetration increases. Specifically, running SimMobility at the base case of 0% MaaS penetration yielded a completion rate of 98.57%. All simulations at higher MaaS penetration levels resulted in even higher completion rates, ultimately reaching 99.99% completion rate at 100% MaaS penetration.

For scalability, we run our tests at each MaaS penetration level five times and take the average run time. We expect the performance of SimMobility to decrease given the additional components necessary for the MaaS protocol. As shown in Figure 4-2, we see a relatively steady increase in runtime as the mode share of MaaS

modes increase in the input demand. In the extreme case of 100% MaaS penetration, SimMobility experiences a 55.31% slowdown compared to the base case runtime. Given the complexity of MaaS compared to other protocols, this level of performance degradation is expected and acceptable.

Chapter 5

Conclusion

We implemented MaaS in SimMobility by linking all the components together and fully integrated the MaaS controller into the existing supply infrastructure. This will allow the MaaS controller to be fully capable of receiving user requests, generating menus, and executing user mode choice decisions. More importantly, researchers will be able to leverage SimMobility and prototypical city simulation models developed by the ITS Lab. A key problem studying potential impacts is the ability to transfer findings between cities studied, as each city's urban structure, network and transit capacity, and population socio-demographics are different. The prototype cities approach allows findings to be generalized to all cities within a typology category, and therefore scaled. These models represent the heterogeneity in population, land use, passenger mobility patterns, and supply services across cities. Thus, we hope that the combination of a MaaS-enabled SimMobility and the prototypical city simulation paradigm will bring upon new insights on cities around the world.

Appendix A

Tables

column	data type	explanation
account_id	integer	Identify unique account
individual_id	string	Identify individual associated with account
subscribed_maas_controller_id	integer	Identify MaaS controller recognizing account
active_plan_id	integer	Identify MaaS plan associated with account
active_plan_expiration_due	integer	Define end of subscription
active_plan_balance	string	Define remaining balance for each bundled mode

Table A.1: SimMobility maas_accounts table

column	data type	explanation
plan_id	integer	Identify unique plan
maas_controller_id	string	Identify MaaS controller recognizing plan
plan_cost	integer	Define cost of MaaS plan
plan_duration	integer	Define duration of MaaS plan
product_providers	integer	Identify controllers associated with bundled modes
product_types	string	Define bundled modes
product_allowance_type	string	Define unit of allowance (e.g. # of trips)
product_allowance	string	Define allowance included in plan
product_discount_type	string	Define unit of discount (e.g. percentage discounts)
product_discount	string	Define value of discounts

Table A.2: SimMobility maas_plans table

column	data type	explanation
pathset_origin_node	integer	Identify starting node
pathset_dest_node	string	Identify desired destination node
path	string	Define path associated with O/D pair
path_travel_time_secs	double	Define travel time
total_distance_kms	double	Define distance
total_cost	double	Define cost
total_waiting_time	double	Define estimated waiting time

Table A.3: Relevant columns of precomputed attributes tables (public transit and multimodal variants)

column	data type	explanation
origin_zone	integer	Identify starting zone
destination_zone	string	Identify desired destination zone
distance	string	Define estimated distance between the two zones
car_cost_erp	double	Define ERP
car_ivt	double	Define travel time

Table A.4: Relevant columns of precomputed attributes tables (cars)

Appendix B

Code

```
function filter_menu(menu, N_choice)
    local cheapest_option_indices = {}
    local cheapest_option_values = {}
    for i = 1,N_choice do
        local mode = menu:mode(i)
  end
        local cost = menu:cost(i)
        if fasted_option_indices[mode] == nil then
            cheapest_option_indices[mode] = i
            cheapest_option_values[mode] = cost
        else
            if cost < cheapest_option_values[mode] then
                cheapest_option_indices[mode] = i
                cheapest_option_values[mode] = cost
            end
        end
  end
    local result = {}
    local count = 0
    for _, index in pairs(cheapest_option_indices) do
        result[count + 1] = index
        count = count + 1
        result[0] = count
    end
```

```
    return result
end
```

Listing B.1: Example Lua menu filtering function

```
function choose_maas_wdme(pparams, mparams, N_choice)
  computeUtilities(params, dbparams)
  computeAvailabilities(params, dbparams, numMode)
  local probability = calculate_probability("mnl", choice, utility,
   availability, scale)
  local final_choice = make_final_choice(probability)
  return final_choice
end
```

Listing B.2: Example Lua menu choice

Bibliography

[1] "Smart-Fm/Simmobility-Prod." GitHub, github.com/smart-fm/simmobility-prod.

[2] Lasse Nykänen Aki Aapaoja, Jenni Eckhardt. Business models for maas. *Conference: 1st international conference on Mobility as a Service*, November 2017.

[3] Markus Lienkamp Benedikt Jäger, Fares Maximilian Mrad Agua. Agent-based simulation of a shared, autonomous and electric on-demand mobility solution. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, October 2017.

[4] Lampros Yfantis Jakub Muscat Moshe Ben-Akiva Carlos Lima Azevedo, Maria Kamargianni. Incorporating the mobility as a service concept into transport modelling and simulation frameworks. *Transportation Research Board Annual Meeting*, January 2019.

[5] Daniel J. Graham Daniel Hörcher. Maas economics: Should we fight car ownership with subscriptions to alternative modes? *Economics of Transportation*, June 2020.

[6] Narelle Haworth Elliot Fishman, Simon Washington. Bike share's impact on car use: Evidence from the united states, great britain, and australia. *Transportation Research Part D*, August 2014.

[7] Elliot Fishman. Bikeshare: A review of recent literature. *Transport Reviews*, April 2015.

[8] I.C. MariAnne Karlsson Göran Smith, Jana Sochor. Mobility as a service: Development scenarios and implications for public transport. *Research in Transportation Economics*, April 2018.

[9] Francesco Ciari Kay W. Axhausen Henrik Becker, Milos Balac. Assessing the welfare impacts of shared mobility and mobility as a service (maas). *Research in Transportation Economics*, September 2019.

[10] David A. Hensher. Future bus transport contracts under a mobility as a service(maas) regime in the digital age: Are they likely to change? *Transportation Research Part A*, February 2017.

[11] Helena Strömberg Jana Sochor, I. C. MariAnne Karlsson. Trying out mobility as a service: Experiences from a field trial and implications for understanding demand. *Transportation Research Record*, January 2016.

[12] Maria Kamargianni Melinda Matyas. The potential of mobility as a service bundles as a mobility management tool. *Transportation*, page 18, August 2018.

[13] Nicolai Mallig Martin Kagerbauer Peter Vortisch Michael Heilig, Tim Hilgert. Potentials of autonomous vehicles in a changing private transportation system – a case study in the stuttgart region. *Transportation Research Procedia*, August 2017.

[14] Krishna M. Gurumurthy Joschka Bischoff Michele D. Simoni, Kara M. Kockelman. Congestion pricing in a world of self-driving vehicles: An analysis of different strategies in alternative future scenarios. *Transportation Research Part C: Emerging Technologies*, January 2019.

[15] Corinne Mulley. Mobility as a services (maas) – does it have critical mass? *Transportation Review*, March 2017.

[16] Anna-Maria Feneri Shima Ebrahimigharehbaghi María J. Alonso González Jishnu Narayan Peraphan Jittrapirom, Valeria Caiati. Mobility as a service: A critical review of definitions, assessments of schemes, and key challenges. *Smart Cities – Infrastructure and Information*, June 2017.

[17] Felix Becker Emilio Frazzoli Kay W. Axhausen Sebastian Hörl, Claudio Ruch. Fleet operational policies for automated mobility: A simulation assessment for zurich. *Transportation Research Part C Emerging Technologies*, May 2019.

[18] Joseph Y. J. Chow Shadi Djavadian. An agent-based day-to-day adjustment process for modeling 'mobility as a service' with a two-sided flexible transport market. *Transportation Research Part B Methodological*, October 2017.

[19] Carlos Lima Azevedo Arun Prakash Akkinepally Bilge Atasoy Kyungsoo Jeong Ravi Seshadri Moshe Ben-Akiva Yifei Xie, Mazen Danaf. Behavioral modeling of on-demand mobility services: General framework and application to sustainable travel incentives. *Transportation*, December 2019.