Integration of Private and Carsharing Vehicles into Intermodal Travel Information Systems

Christian Samsel[1,3], Markus Christian Beutel[1,3],
David Thulke[1], Detlef Kuck[2] and Karl-Heinz Krempels[1,3]
[1]*Information Systems, RWTH Aachen University, Aachen, Germany*
[2]*Ford Research and Innovation Center, Aachen, Germany*
[3]*Fraunhofer FIT, St. Augustin, Germany*

Keywords: Carsharing, Intelligent Transportation Systems, Intermodal Travel Information, Web Information Systems.

Abstract: In the last years, intermodal mobility platforms offering combinations of various modal types, like trains, buses, carsharing and ride sharing, have emerged. These platforms often also offer a smartphone-based door-to-door navigation and a sophisticated travel assistance. Unfortunately, these smartphone-based services cannot be used by the travelers as soon as they are driving a car themselves, e.g., a carsharing vehicle or their private car, due to road safety regulations. The driver is essentially disconnected from the service. In addition, modern cars have a lot of configuration options a driver might want to set up. This discourages using shared vehicles in an intermodal itinerary. In this work we identify use cases of how an integration of carsharing vehicles into intermodal travel information systems can enhance travel experience, introduce a system architecture to allow the necessary information exchange and present a preliminary prototype to demonstrate its technical feasibility.

1 INTRODUCTION

In the last years, the demand for intermodal mobility has been increasing and especially in younger generations, car ownership becomes less popular (Klein and Smart, 2017). Reasons for this include high maintenance cost, the declining value of cars as a status symbol and environmental concerns (Kalmbach et al., 2011). In urban areas, other modes of transportation offer higher availability and flexibility by avoiding parking problems and congestions. In many situations, traditional public transport provides an adequate replacement. But areas with insufficient coverage of traditional public transport services or special demands (e.g., high reliability) are requiring additional mobility services. One service to complement traditional public transport is carsharing. Carsharing became more and more popular in recent years (Shaheen and Cohen, 2007).

To make these complementary services appealing, a seamless integration of various transportation systems is needed. One approach for such an integration are advanced Travel Information Systems (TIS) offering intermodal itineraries (Beutel et al., 2016a). Intermodal itineraries consist of a mix of different modes of transportation, e.g., using both trains and a carsharing vehicle in the same journey. This requires complex planning, which is enabled by integrating heterogeneous information from multiple mobility service providers. This information can include for example timetable information, vehicle availability or delays. Besides traditional public transport providers like bus or railway companies, integrated mobility services can include, among others, car, bike and ridesharing providers, parking services, taxi services or innovative services like Uber[1]. In addition to the offering of an intermodal itinerary, some of these systems also provide integrated booking and billing services for all legs of the itinerary. In this way, the user does not have to book them individually. Examples for these systems are Qixxit[2] by Deutsche Bahn (German railways), Moovel[3] by Daimler AG or "Mobility Broker" as suggested in (Beutel et al., 2014).

The rising number of features in modern cars allows an increasing amount of possibilities for customization and personalization. Besides standard fea-

[1]https://www.uber.com
[2]https://www.qixxit.de/en
[3]https://www.moovel.com/de/en

Samsel, C., Beutel, M., Thulke, D., Kuck, D. and Krempels, K-H.
Integration of Private and Carsharing Vehicles into Intermodal Travel Information Systems.
DOI: 10.5220/0006347803120319
In *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2017)*, pages 312-319
ISBN: 978-989-758-242-4

tures like seat and mirror positions, air conditioning and radio, cars can integrate the drivers' smartphones, offer connected infotainment systems or even driving assistants. Furthermore, many parts of current vehicles become electronically controllable and interconnected. In this way, the infotainment system is enabled to control settings like the seat or mirror position.

Motivation

Currently, TIS's are not integrated into cars booked via carsharing providers. When users arrive at their booked carsharing vehicles, these vehicles do not know anything about the users nor about their route. Unlike in transport modalities like trains, it is difficult for users to interact with their smartphone or to use it for navigation, as it is a security hazard and often forbidden by law. Necessary equipment like hands free systems are often not available in carsharing vehicles. Thus, the travel assistance provided by the TIS is disrupted and the user has to manually enter the destination address into the car navigation systems, which is a major inconvenience.

Furthermore, every time a traveler uses the service, he or she has to manually customize most of the car settings to his or her personal preferences. If a car is unknown to them, they have the additional hassle of finding out how to apply these settings to the car. Otherwise they have no access to useful features. According to (Berylls Strategy Advisors, 2015), the second most common reason (28%) for people not to use carsharing is that they do not want to drive an unfamiliar vehicle. If carsharing vehicles could be already configured for these users, this may mitigate the problem. Additionally, misconfigured vehicles can become a security risk. For example, it is crucial that the mirrors of the car are correctly adjusted, to remove blind spots.

2 RELATED WORK

This section gives an overview over related work, including actual systems and technologies which are currently in use, as well as related academic research.

As mentioned in the introduction, Qixxit is an example for a TIS by Deutsche Bahn. Beside train services, it includes local public transport services, multiple carsharing providers, rental cars, ride sharing, taxis, bike sharing and even airlines. Qixxit offers its services via a webportal and a mobile application.

INRIX Intermodal Navigation[4] is a in-car service to integrate local public transport connections into journey planning. It is able to suggest users to switch to alternative modes of transportation if these are faster than the expected vehicle travel time. If users choose to use a different mode of transportation, the vehicle navigates them for example to the next train station. In contrast to the system proposed in this work the system is bound to the navigation system in the vehicle and is not accessible for example via the user's smartphone.

The problem of driver preferences in shared vehicles already has been recognized by others. Among other car manufactures, Ford has been approaching the problem by offering the so called "MyKey". Drivers can store several settings of the car, like the seat position, on their keys. If the car is opened using the key, the settings which are stored on it are initiated. This solution works for a limited number of drivers (e.g., for families) but requires that every driver has a personal key and does not allow settings to be shared between multiple cars.

In (Beutel et al., 2014), the authors introduce an architecture for a TIS called Mobility Broker. It is designed to allow heterogeneous mobility providers to collaborate on a joint platform. Besides intermodal routing, it allows querying pricing information and manages the booking and payment of intermodal routes for users. Additionally, the advantages for participating stakeholders are discussed. In (Beutel et al., 2016a) the architecture is refined by discussing the information flow between the TIS and the different transportation providers in more detail.

A protocol used to communicate with sharing providers, including car and bike sharing, is the Interface for X-Sharing Information (IXSI) protocol, proposed in (Kluth et al., 2015). It is designed to create a unified interface between sharing providers and TIS's. It defines different services to enable functionality like availability queries, exchange of pricing information and booking of vehicles.

A solution more adequate to carsharing regarding the problem of driver preferences in shared vehicles is proposed in (Kümmerling et al., 2013). The idea is to create a centralized platform where every driver can store his or her own *Mobility Profile* containing his preferred vehicle settings. The user applies the profile to the car by connecting his or her phone over a Near Field Communication (NFC) interface. The authors applied the concept to a prototype of a car (consisting of a middle console and an electrically adjustable seat) and developed a mobile app. An advantage compared to other solutions is that the profile can be used for an arbitrary amount of vehicles. It additionally suggests four categories of data which may

[4]http://inrix.com

be stored in such a *Mobility Profile*. These consist of information about the driver, car dependent (e.g., seat position) and independent (e.g., radio station) settings and logging data (e.g., consumed fuel or driven kilometers). The disadvantage of their solution is that the driver still has to manually apply the settings to the car with his or her phone.

3 USE CASES

To guide the design, we start with possible use cases. Each use case discusses a real world example comprising problems mentioned in Section 1. Moreover, these are used to provide an evaluation for the implementation.

Scenario 1: Synchronized Preferences. Every friday, Bob uses a carsharing vehicle to drive from Aachen to Düsseldorf (and back). Previously, he adjusted the seat positions and searched for his favorite radio station every time. It annoys him that he has to adjust the vehicle every week. With the integration of the carsharing vehicle and the TIS, the TIS can now automatically apply the preferences to the vehicle before Bob arrives at the car. Thus, he can immediately start driving without the need to apply any configurations, giving a similiar experience as a private car.

As the preferences are transfered to the TIS, which offers multiple carsharing services, his favorite radio station is even preset when he uses a different model and carsharing operator at the weekend.

Scenario 2: Preconfiguring the Navigation System. Alice plans a trip from her home in Cologne to a lake in the Eifel. The TIS recommends her to take the train to Aachen and to rent a carsharing vehicle to get to the lake from there. Without an integration of the TIS with the infotainment system of her carsharing vehicle, she had to manually enter the address of the lake into the navigation system of the vehicle. Because of the unknown interface, this was time consuming and frustrating for her. With the integration of the TIS and carsharing system respectively vehicle, the TIS can now preconfigure the navigation system of the vehicle. At the time Alice arrives at the vehicle, the address of the lake in the Eifel is set as destination and she can start driving immediately.

Scenario 3: Changing the destination while driving (using car sharing). Bob has to travel from Aachen to a meeting in Cologne. He booked a car sharing vehicle using his preferred TIS. An unexpected road accident on the highway causes a delay of over an hour. Due to this, he would be late to his meeting. Fortunately, his TIS is now integrated with the carsharing vehicle. Using the Infotainment System, he is notified about the possibility to stop the car and change to a train in Düren. Bob can then accept this itinerary change and thus still arrive at his meeting in time. The train booking is done automatically and the required ticket is transferred to his smartphone.

Current Coverage of Use Cases

The key-based preference systems are not applicable to carsharing but only to situations where a small number of persons share a vehicle. The concept of Mobility Profiles in (Kümmerling et al., 2013) requires the interaction of the user to manually apply his preferences to the vehicle. As a result, the process is not as seamless as described in Scenario 1.

The integration of intermodal travel information of INRIX in vehicles enables the navigation system to suggest the driver to change to other modes of transportation. In Scenario 3, it could suggest Bob to switch to a train. But there is no full integration of a TIS, a booking of the service is not possible. The component in the vehicle is not connected to other mobility applications and thereby is not able to support any of the other use cases.

4 APPROACH

As concluded in the last section, none of the solutions discussed so far supports all use cases presented and integrate carsharing vehicles into TIS. The approach presented in this section is similar to (Kümmerling et al., 2013) but does not require the user to manually apply his settings.

4.1 Architecture

For *Scenario 1*, the TIS has to be able to synchronize the user's preferences with the car. Before the start of the booking, it has to send the vehicle all applicable preferences of the user it knows. The carsharing vehicle has to apply these preferences before the user arrives at the car. During a trip it has to propagate all changes in the preferences back to the TIS. The TIS then stores these settings to reapply them at the next booking. For *Scenario 3*, the TIS has to be able to track the position of the carsharing vehicle. On a regular basis, it should check whether the user is still taking the optimal route with regard to the current traffic

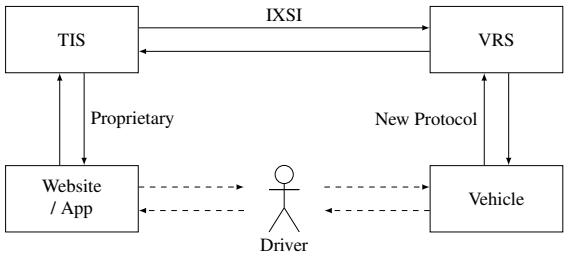

Figure 1: Sketch of the proposed architecture. Arrows indicate information flow, dashed arrows indicate user interaction.

and the current location of the vehicle. If the optimal route changes it has to propagate this change back to the carsharing vehicle which notifies the user. Thus, the part of the architecture deployed to a carsharing vehicle has to be able to identify its location and to expose it to the TIS. Additionally, it has to be possible to change the current destination of the navigation system, preferably with the option to notify the user about this change and to ask for approval.

The carsharing operator uses a Vehicle Rental System (VRS), which is connected to the vehicles to manage bookings and to track the state of the car (location, mileage, etc.). This is done via an onboard unit, which is installed in the car. One of these solution is "CloudBoxx"[5] by Invers GmbH. Among others, it offers the management of vehicle access and connectivity between the vehicles and the servers of the carsharing provider. Given these capabilities, it is possible to deploy an additional software to such an onboard unit to manage vehicle preferences. Once deployed in a vehicle, such a software component is difficult to access and thus should be designed to require low maintenance. To minimize the possible need for updates, it should contain a minimal amount of logic. Our approach uses this existing carsharing infrastructure and does not require additional hard- or software. This also has some further advantages. Instead of giving one TIS full access to all carsharing vehicles, the access rights can be managed by the carsharing provider. Also the scalability is better, as a TIS does not have to communicate with potentially thousands of vehicles, and the vehicles do not have to communicate with multiple TIS's.

By including the driver, the architecture depicted in Figure 1 emerges. The driver communicates with the TIS over the website of the TIS or its mobile application. Additionally he or she indirectly communicates with the module installed in his or her carsharing vehicle by modifying preferences or by responding to destination changes.

[5]http://cloudboxx.invers.com/index.en.html

4.2 Data Models and Protocols

After introducing particular components of the architecture, the next step is to discuss data models and protocols used by these components for communication. The communication between TIS and website / mobile application is not covered here as it is usually proprietary.

The update of the status of the current trip is modeled by a *TripProgress*. A *TripProgress* consists of a coordinate which represents the current location, the progress on the route as percentage or the estimated arrival time, depending on the preferences of the carsharing provider and the user. Futhermore, it contains a timestamp to match it to a specific point in time.

The modeling of Preferences is not as straightforward. The value of a *Preference* can have an arbitrary type. They can range from a simple decimal value to configure the volume of the speakers to a set of angles to configure the seat position. To keep the handling of these various types of values simple, the idea is to store them as one decimal or one string value. For most preferences either one or both of these properties are sufficient (e.g., the volume or temperature). All other settings can be encoded in these properties (the set of angles for example could be encoded as a string value). Different preferences are distinguished by their type property. A value of a type is a fixed string identifying a preference across different vehicles, carsharing providers and TIS's. In order for this to be possible, types have to be standardized, an appropriate standardization could be based on the Automotive Ontology (Feld and Müller, 2011). Finally, each preference has a boolean property to encode whether its value depends on certain vehicle components or whether it is universal. The vehicle components are determined by definition of its type and context of the booking. In case a preference is universal, its value has to be encoded as specified by its type.

TIS to VRS Communication

As mentioned in Section 2, one protocol designed for the communication between a TIS and a car sharing provider is IXSI. It defines different loosely coupled services. These services include static data, availability of vehicles, booking and pricing information. The smart car extension (Beutel et al., 2016b) adds two additional services to IXSI to allow the remote configuration of preferences (referred to as service 10 in the standard specification) and to set and track the current trip (service 11). Each service offers the possibility to set preferences or the destination via a request and the possibility to subscribe to changes of preferences

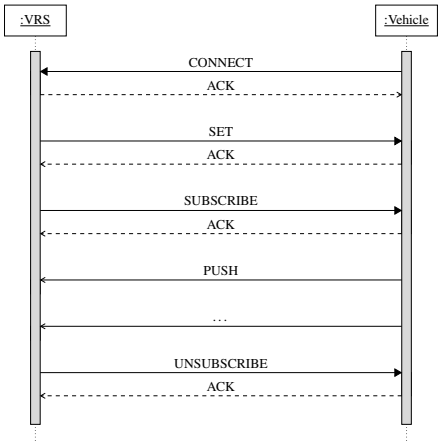

Figure 2: Interaction protocol between a VRS and a vehicle.

and the trip progress. Users' preferences are modeled by a set with elements of the type *BookingSetting*. Each *BookingSetting* contains a *BookingSettingsClassType* defining the type of the preference, a decimal, a string value, the id of the current booking and a boolean value indicating whether it depends on the current booking target. This boolean value can be used by the carsharing provider to indicate a preference depending on a specific vehicle component. The trip progress is modeled by the type *BookingProgressType*. Its properties are the time stamp of the point of time the progress was determined, the ETA, the current position and progress. Because of the privacy considerations discussed in the design of the carsharing provider, the latter two properties are optional. After the booking of the vehicle, the TIS sends a *SetNavigationDestinationRequest* to the VRS containing the coordinates of the destination of the current trip. If the destination is successfully applied, the VRS sends the associated response. The next step of the TIS is to send a *BookingProgressSubscriptionRequest*. This enables the subscription of trip updates. Subsequently, the VRS sends messages containing the current booking process. If the TIS intends to change the destination, it simply sends a *SetNavigationDestinationRequest* again. The VRS continues to send updates of the booking process until the vehicle arrives at the destination or the TIS terminates the subscription.

VRS to Vehicle Communication

For the communication between the VRS and the vehicle, an information exchange protocol has been developed. Its structure is inspired by the smart car extension of IXSI discussed in the last section. Similar to IXSI, it allows to set preferences and the destination in a request / response scheme and allows

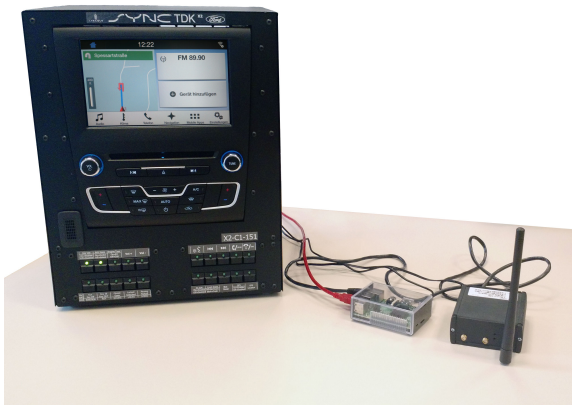

Figure 3: Implementation setup with the Ford TDK, a Raspberry Pi (a mini computer) and a cellular network router.

asynchronous subscriptions to listen for trip updates and for changes in the preferences. Because of the requirement of asynchronous subscriptions and bidirectional communication, the protocol is designed to work on top of WebSockets. The data model of the protocol outlined in this section is similar to the one discussed at the beginning of this section and the one used in IXSI. Each message consists of an ID, a type and a body. The ID of a message is unique in regard to the current session. A type can be one of the following: *CONNECT*, *SET*, *SUBSCRIBE*, *UNSUBSCRIBE*, *PUSH*, *ACK* or *NACK*. The body of a message depends on its command. A possible protocol interaction is illustrated in Figure 2.

5 IMPLEMENTATION

The goal of the implementation is to create a prototype of the proposed architecture to prove its feasibility. To limit the scope of the prototype, the goal is to only implement a client on the vehicle side and a component for a VRS.

5.1 Vehicle Module

Instead of an actual car, a Technical Development Kit (TDK), the Ford SYNC Gen3 TDK X2, is used for the implementation. It contains the current generation of infotainment systems which are deployed in vehicles built by Ford and Lincoln. The included infotainment system consists of a large touchscreen, built-in speakers, a CD player and buttons to control the radio, the heating and the air conditioning. As in a normal vehicle, the communication between the components works over a Controller Area Network bus (CAN bus). Beside the messages necessary for the functionality of the infotainment system, the TDK

contains a module which simulates CAN traffic for further modules of the car which are not included in the TDK. These include, for example, the state of the locking system or the ignition of the vehicle. A set of buttons at the bottom of the TDK allows to send messages over the simulated CAN bus.

Figure 3 shows a picture of the implementation setup with the TDK, a Raspberry Pi and a cellular network modem. The Raspberry Pi represents the car-sharing onboard unit.

In the following, two methods are discussed which could enable the implementation of the proposed vehicle client. Smart Device Link (SDL) is a standard to connect a driver's smartphone with the infotainment system inside vehicles[6]. It was originally developed by Ford and integrated in their vehicles under the name AppLink[7]. SDL is designed to work with applications running on the driver's smartphone. Additionally, the connected device has to be linked with the infotainment system and the applications have to be manually started by the driver. Thus, SDL in its current form is not a suitable solution for the implementation of the vehicle module. In contrast to using SDL, accessing the vehicle over the CAN bus is a rather low level approach. Due to the standardization of OBD-II, a common interface to access the CAN bus exists in most vehicles. OBD-II is a standard to diagnose malfunctions in various vehicle components. Though the protocols used in OBD-II and the CAN bus are standardized, the implementation varies between different manufactures and between different vehicle generations. Unfortunately, none of the approaches presented above is an ideal solution to implement a vehicle client as designed in Section 4.1. To create a prototype for the vehicle client, the solution to access the vehicle via the CAN bus has been preferred to the approach of using SDL. To get access to the CAN bus of the TDK, the adapter OBDLink SX by ScanTool[8] has been used. It was possible to identify a few CAN bus messages to read and set the state of vehicle settings. The first preference is the volume of the infotainment system. The volume can be set by sending its value (between 0 and 30) to the CAN bus. Unfortunately, no way was found to directly query the value of the current volume. To be able to report the current volume to the VRS, the idea is to set the volume at the start of each session (either to the value provided by the TIS or a default one) and to track its changes. The second preference identified is the current frequency of the radio. Messages were found which are sent when the knob to change the frequency is rotated to the left or the right. Additionally, the current frequency is constantly broadcasted. This is sufficient to read and set the frequency.

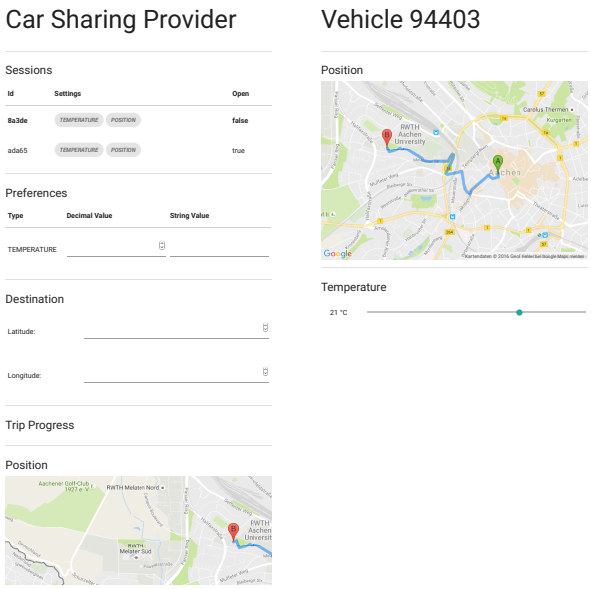

Figure 4: Web clients developed to test the implementation of the server.

The next step is to implement a prototype for the vehicle client. Python was chosen as programming language for the implementation. Python allows quick prototyping and libraries exist to communicate over the serial interface with the vehicle and over WebSocket with the VRS. The implementation is divided into two modules. The vehicle module implements the communication with the vehicle over the serial interface. It listens for changes in the preferences and exposes a method to apply a preference. The server module handles the communication with the VRS over the protocol defined in Section 4.2. The implementation is build on top of websocket-client[9], a WebSocket implementation for Python. The module manages subscriptions of the VRS and exposes *SET* messages to other components. Additionally, it offers a method to send changes in preferences via a *PUSH* message to the subscribed VRS. When starting the client, it configures the adapter as described above. Afterwards, it establishes a connection to the VRS server and sends a *CONNECT* message. Then it waits for changes in the preferences or for *SET* messages. As soon as one of them arrives, they are forwarded to the respective module.

[6]https://smartdevicelink.com
[7]https://developer.ford.com/pages/sdl
[8]https://www.scantool.net/obdlink-sx/
[9]https://github.com/liris/websocket-client

5.2 Server

To supplement the vehicle client, a server was created that mocks the behavior of a VRS in the proposed architecture. It was implemented in Java as a Spring Boot[10] application. The implementation is separated into three modules. The core module handles the management of vehicle sessions. If a new vehicle client connects to the server, a vehicle session is stored in the session store. A vehicle session is an abstraction of the connection between the server and the vehicle client which exposes the functionality of the underlying protocol. The core module offers interfaces to backend and frontend modules. Both types of modules are designed to be interchangeable and to allow multiple implementations to work in parallel. As a consequence it is possible to extend the server with compatibility for other protocols. The backend module handles the connection with the vehicle client. It implements the protocol used to communicate with the vehicle and provides it as an implementation of a vehicle session. When a vehicle establishes a connection and sends a *CONNECT* message, it adds the session to the session store of the core module.

To allow the testing of features not supported by the vehicle client described in the previous section, an additional web client, has been implemented which simulates a vehicle (Figure 4). It displays its current location on a map and provides simulated preferences as a slider. Besides testing additional features, this allows to test the system with multiple connected vehicles.

6 EVALUATION

The goal of this section is to evaluate how far the design and implementation support the use cases listed in Section 3 and how them compare to the systems presented in Section 2.

6.1 Methodology

The test setup for the integration test includes the TDK, a Raspberry Pi, a cellular modem and a test computer. The server component discussed at the end of the last chapter is deployed on the test computer. The vehicle client is deployed on the Raspberry Pi. The OBDLink SX adapter connects the TDK with the Raspberry Pi. The cellular modem is connected to the Raspberry Pi to allow network access under conditions similar to that in an actual vehicle. Over this connection, it can connect to the server on the test computer.

To initialize the preferences, a radio station and the volume is configured on the TDK. The vehicle client notices these changes and sends *PUSH* messages to the server. As a result, the table displaying the preferences gets updated. If one modifies the preferences in the web client of the carsharing provider, it sends a *SET* message to the vehicle client. This then appropriately applies the preferences to the TDK.

To test the system with multiple vehicles, the vehicle web client can be used to simulate additional vehicles. As soon as it opens in the browser, it connects to the server and is displayed as an additional session in the web client of the VRS. By clicking on this new session in the web client, it can be simulated that the user switches to another vehicle. The VRS releases the previous vehicle client of the TDK and terminates its subscriptions. As a second step, the session of the web vehicle client is acquired and subscriptions for preferences and the trip progress are created. Additionally, all currently known preferences (currently displayed in the preferences part of the screen) are sent in a *SET* message to the client. The client is then automatically configured with the known preferences.

The vehicle web client can then be additionally used to test the subscription of trip updates and the change of the destination. By entering a coordinate in the destination field of the VRS web client, a *SET* message with this destination is sent to the vehicle client. This moves the destination marker (red) in the map to the corresponding location. If the marker of the current location (green) is moved, the client sends a *PUSH* message with the appropriate trip progress to the server.

6.2 Results

The designed system supports all use cases presented earlier. It is able to set the destination of the navigation system before the user arrives at the vehicle and can change the destination during a trip, based on trip progress updates it receives from the vehicle. Therefore, it supports the Scenarios 2 and 3. Though the server implemented and the vehicle web client support the functionalities required, it was not possible to implement them in the vehicle client. We do not know whether it is possible to set the destination of the navigation system via messages over the CAN bus.

Additionally, the system is able to apply arbitrary preferences to a vehicle before the user arrives and is able to listen to their changes. In contrast to the Mobility Profiles proposed in (Kümmerling et al., 2013), the functionalities discussed above work without ad-

[10]https://projects.spring.io/spring-boot/

ditional interaction with the driver. Finally, the designed system works across the boundaries of different carsharing providers. The reason for this is that the core logic of the system is located at the TIS and thus preferences can be applied to vehicles of all carsharing providers supporting the system. Although, only a limited set of preferences has beeen implemented into the vehicle client, the supported preferences show that the general approach is working and therefore Scenario 1 is possible.

7 CONCLUSION AND OUTLOOK

In this work, a system was proposed to integrate carsharing vehicles into intermodal travel information systems. To guide the design process and to create a base for evaluation, use cases were defined. The system was designed to enable continuation of a user's navigation via the navigation system integrated in a carsharing vehicle and furthermore to configure carsharing vehicles according to the preferences of the user. The progress compared to similar approaches lies in the integration of the navigation system and the removal of necessity of any user interaction. To prove the feasibility of the proposed architecture, a prototype of a vehicle client and a prototype of a VRS were built. Comparing the features provided by the prototype with the listed use cases showed that the approach is a viable first step to integrate carsharing vehicles into intermodal travel information and by that improve the travel experience.

The next step is to complete the implementation of the proposed architecture. This includes an extension for an existing TIS to manage the user's preferences. Such a system should be able to reason about preferences to infer them for new vehicles. Additionally, a robust implementation of vehicle clients is necessary. The most promising way of realization is that manufacturers implement these functionalities in their vehicles natively. As mentioned in the introduction, the popularity of carsharing increases. If the system proposed in this work is able to accelerate this trend, which has to be evaluated, an automobile manufacturer could gain a competitive advantage in the market of carsharing vehicles. Last but not least, the system is to be assessed for user feedback in a realistic scenario with a real car.

ACKNOWLEDGMENTS

This work was conducted in cooperation with Ford Research and Innovation Center Aachen.

REFERENCES

Berylls Strategy Advisors (2015). Carsharing – der große Durchbruch steht noch bevor [german]. http://www.berylls.com/media/informationen/downloads/presse/150216_Berylls_Carsharing_-PM_Slides.pdf, visited 2017-08-19.

Beutel, M. C., Gökay, S., Kluth, W., Krempels, K.-H., Ohler, F., Samsel, C., Terwelp, C., and Wiederhold, M. (2016a). Information Integration for Advanced Travel Information Systems. *Journal of Traffic and Transportation Engineering*, 4(4):177–185.

Beutel, M. C., Gökay, S., Kluth, W., Krempels, K.-H., Samsel, C., and Terwelp, C. (2014). Product oriented integration of heterogeneous mobility services. In *Proceedings of the 17th International Conference on Intelligent Transportation Systems (ITSC 2014)*, pages 1529–1534, Qingdao, China. IEEE.

Beutel, M. C., Gökay, S., von Grumbkow, P., Hillbrecht, D., Krempels, K.-H., Samsel, C., Terwelp, C., Twele, H., and Wagner, H. (2016b). Mobility-Broker Interface with Smartcar Extension based on IXSI - Interface for X-Sharing Information Version 4. https://rwth-i5-idsg.github.io/downloads/ixsi/ixsi-docu-smartcar-english-version.pdf, visited 2017-08-20.

Feld, M. and Müller, C. (2011). The Automotive Ontology: Managing Knowledge Inside the Vehicle and Sharing it Between Cars. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2011)*, pages 79–86, Salzburg, Austria. ACM.

Kalmbach, R., Bernhart, W., Grosse Kleinmann, P., and Hoffmann, M. (2011). Automotive landscape 2025: Opportunities and challenges ahead. *Roland Berger Strategy Consultants*. http://www.rolandberger.com/media/pdf/Roland_Berger_Automotive_Landscape_2025_20110228.pdf, visited 2017-08-18.

Klein, N. J. and Smart, M. J. (2017). Millennials and car ownership: Less money, fewer cars. *Transport Policy*, 53:20–29.

Kluth, W., Beutel, M. C., Gökay, S., Krempels, K.-H., Samsel, C., and Terwelp, C. (2015). IXSI - Interface for X-Sharing Information. In *Proceedings of the 11th International Conference on Web Information Systems and Technologies (WEBIST 2015)*, Lisbon, Portugal.

Kümmerling, M., Heilmann, C., and Meixner, G. (2013). Towards Seamless Mobility: Individual Mobility Profiles to Ease the Use of Shared Vehicles. In *Proceedings of the 12th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, pages 450–454, Las Vegas, USA. IFAC HMS.

Shaheen, S. and Cohen, A. (2007). Growth in worldwide carsharing: An international comparison. *Transportation Research Record: Journal of the Transportation Research Board*, (1992):81–89.