Customer-Centred Intermodal Combination of Mobility Services with Conversational Interfaces

Daniel Braun[1], Adrian Hernandez-Mendez[1], Anne Faber[1], Manfred Langen[2], Florian Matthes[1]

[1] Technical University of Munich, Department of Informatics, Munich, Germany
{daniel.braun,adrian.hernandez,anne.faber,matthes}@tum.de

[2] Siemens AG, Corporate Technology, Munich, Germany
manfred.langen@siemens.com

Abstract. The way we consume mobility is drastically changing. Mobility is no longer only provided by traditional public transport and goods we own, like cars and bicycles, but also by service providers which offer mobility as a service, like car sharing companies. This new mobility ecosystem enables more flexibility, but also introduces additional complexity, especially for intermodal travel, i.e. reaching one's destination by using multiple means of transportation. In this paper, we present an approach to combine mobility services in a customer-centred way, which enables intermodal journey planning through an intuitive natural language interface. In contrast to other approaches, our architecture does not rely on the cooperation of mobility service providers, which has proven to be one of the main obstacles towards an integrated solution of multiple mobility services in the past.

Keywords: mobility, conversational interfaces, modelling

1 Introduction

With the growth of the sharing economy, mobility is transforming from something provided by owned goods, like cars and bicycles, and traditional public transport, to a service that can be consumed on demand. This development leads to a new flexibility which encourages so called intermodal mobility, i.e. the combination of multiple means of transportation during one journey. Renting a bike to get to the train station, taking a train to another city and renting a car at the destination is just one example, how a combination of multiple means of transportation can look like. Depending on the preferences of the user, different combinations could provide different benefits, like being cheaper or faster.

However, finding the most suitable combination of means of transport can be a very time consuming and tedious task. Currently, customers have to check multiple proprietary apps or websites and calculate transfer times and connections manually. In this paper, we present an approach to combine mobility services in a customer-centred, easy to use fashion in order to facilitate intermodal mobility.

Multikonferenz Wirtschaftsinformatik 2018,
March 06-09, 2018, Lüneburg, Germany

This approach benefits from another major business trend called API economy. A growing number of mobility companies, including traditional players from the public transport industry, as well as new players, which are part of the sharing economy, offer APIs to retrieve the information which is necessary to plan a journey. However, since there is no common standard for representing this data, these APIs have different formats and are not compatible with each other.

By introducing an additional abstraction layer, we want to create a system which is able to combine different existing mobility service APIs and is easily adaptable to new service providers. In order to make this combined information accessible and easily consumable for users, we developed a conversational interface which enables user to communicate with our system in the most natural way, using natural language. Moreover, we developed a platform which is able to store, process and share contextual information, like user profiles, which can be used to provide information tailored to the needs of the user.

2 Methodology

This work follows a design science research approach [7,8]. This approach was particularly suitable for our research, since one of the main parts of our work is a research prototype. The artefacts we have designed are the aforementioned prototype, a conversational interface for intermodal mobility, a meta-model for mobility services, and a reference architecture, which is a more generalized abstraction of the architecture of our prototype. According to Hevner's three-cycle view of design science [7,9], our research contributes in the following ways to each of the cycles:

- **Relevance Cycle**: The changing mobility ecosystem and the changed demands of customers regarding mobility (i.e. the increasing desire for flexibility and the decreasing desire to own mobility providing goods such as cars) establish a need for the intermodal connection of mobility services. Existing solution approaches, as we show in Section 3, have not been adopted in practice. Therefore, there is still a need for an applicable solution to this problem. Insights gathered from the prototypical implementation are used to assess the utility and viability of our approach.
- **Design Cycle**: The design artefact is the prototypical implementation, based on the architecture described in Section 4. In its current stage, the system can already connect to different mobility services, but we plan to add support for more services in the future, to test the scalability of the routing algorithm with regard to the number of mobility services. The designed artefact is evaluated as part of the relevance cycle by conducting a prototypical field test and including expert feedback which might stimulate further developments.
- **Rigor Cycle**: We have studied and evaluated the existing literature and artefacts in the domain of conversational interfaces, and service-oriented architectures. We draw on the current state of the art in system design approaches for conversational interfaces. In addition to the prototype, as design artefact, we contribute to the

knowledge space by designing a reference architecture and describing benefits and limitations of our approach.

3 Related Work

There have been multiple proposals, how mobility services could be connected to enable intermodal mobility. Beutel et al. [3] for example suggested a mobility broker architecture which introduces an independent mobility broker operator, which receives data from mobility service providers, combines them and offers a unified API to providers of webportals and mobile apps.

In contrast to our approach, Beutel et al. assume that mobility service providers will cooperate with the mobility broker and adapt der APIs according to the needs of the mobility broker. However, given the development we saw in the past, we do not think this is a realistic scenario in the near future. With a growing number of competitors, mobility service providers tend to be more closed up, rather than open and a cooperation of a significant number of service providers to use one unique interface seems at the moment unlikely.

The platform architecture proposed by Masuch et al. [10] makes the same assumption, that mobility service providers will actively adapt their services to fit the needs of a platform provider. However, in addition to the architecture of Beutel et al., the architecture from Masuch et al. contains a central storage for user information, a component which is also included in our architecture.

It is also worth mentioning that both, Beutel et al. and Masuch et al., only describe a high-level architecture, while we have also implemented a prototype based on the architecture we developed, in order to gather insights about the suitability of the approach.

From a commercial perspective, the most prominent example for a platform enabling intermodal journey planning is Google Maps. With its market power, Google was able to establish a format called General Transit Feed Specification (GTFS) [2], in which public transport companies have to provide their information in order to be included in Google Maps. However, GTFS only support street- and rail-based public transport. Hence, Google Maps is not able to include car and bike sharing in their journey planning or even a combination of trains and flights.

On a more general level, the TUM-LLCM project [5] researches open mobility platforms and their ecosystem, with a focus on the topics governance, architecture and core services, use cases, and geospatial-temporal analytics.

While there is also a huge amount of literature about chatbots and conversational interfaces, from past and present, there is only a few publications when it comes to conversational interfaces within the domain of transport and mobility. An early example of such a system is TRIPS (The Rochester Interactive Planning System) from Allen et al. [1]. TRIPS helps emergency services to plan the transport of patients to a hospital. The conversational interface takes the site of the accident and the hospital to which the patient should be transported as input and suggests which mode of transportation (e.g. helicopter or ambulance) is appropriate. Compared to the NLP

technologies available nowadays, Allen et al. were very limited and had to rely on a purely rule-based system.

4 Architecture

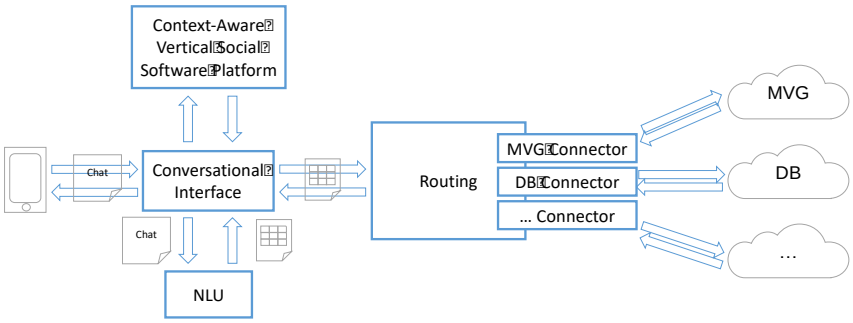

Figure 1. Architecture

The architecture we propose is shown in Figure 1. It contains five main elements. The Conversational Interface, which is described in more detail in Section 5, handles all interaction with the user. Incoming requests (i.e. chat messages) from the users are processed here and outgoing information is transformed from data to text within this element. For the processing of incoming messages, an external NLU component is used, which is connected to the conversational interface through a REST API. This component is also described in more detail in Section 5.

Once the incoming messages are processed and transformed from text to structured data, this data is used to generate search queries. Before these search queries are sent to the routing component, they are enhanced by information stored in the Context-Aware Vertical Social Software Platform, which is based on the architecture described by Hernandez-Mendez et al. [6]. This platform can for example be used to replace phrases like “home” or “work” in queries like “How can I get home from the main station?” or to incorporate user preferences (e.g. prefer trains over planes because of aviatophobia).

The enhanced query is subsequently sent to the routing component. In the simplest case, the routing component, described in Section 7, just chooses the appropriate mobility service. If the user is for example looking for a bus in Berlin, the routing component will connect to the BVG (Berliner Verkehrsbetriebe) API, if the user is looking for a bus in Munich, it will connect to the MVG (Münchner Verkehrsgesellschaft) API. In more complex cases, the routing algorithm will choose multiple mobility service providers to allow intermodal traveling.

In order to be able to query APIs from different mobility service providers and connect their results, the routing module needs the fifth element of our architecture, the different mobility service connectors. These connectors can be easily plugged into the routing component and are based on the meta-model for mobility services we describe in Section 6.

5 Conversational Interface

Natural language is one of the most intuitive ways for humans to interact with their environment. The recent hype regarding chatbots and conversational interfaces in both, industry and science, was fuelled not only by advances in machine learning and natural language processing, but also by the emergence of centralized messenger platforms. Messenger apps like Telegram or Facebook Messenger start to become the one-stop shop for users to interact with a range of different services, making individual apps for specific services obsolete.

Therefore, we decided to make our system available through a conversational interface which allows users to interact with the system from their favourite messenger. The user could for example ask “How can I get from Platz der Republik in Berlin to Marienplatz in Munich?” or set his preferences by writing “I live in Hamburg”. The design of our conversational interface follows the architecture described by Braun et al. [4], shown in Figure 2. We will give a more detailed view on our implementation of the architecture in this section.

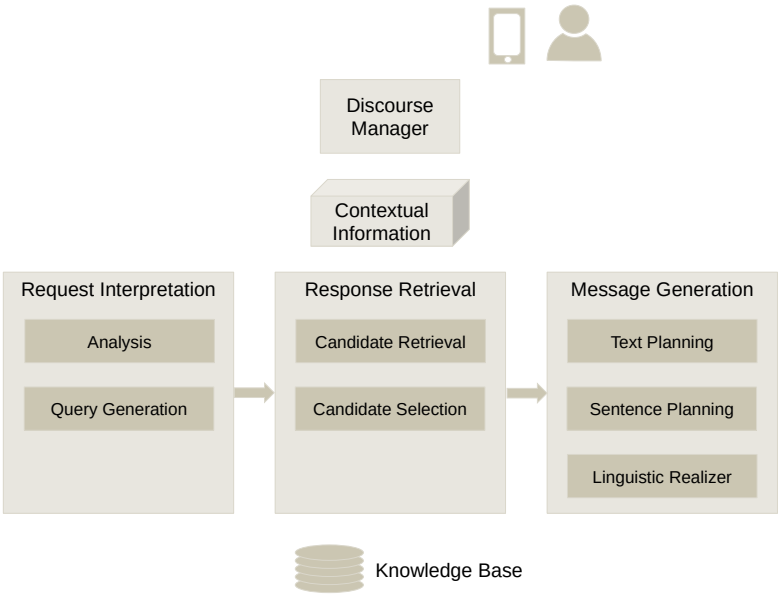

Figure 2. General Architecture for Chatbots by Braun et al. [4]

5.1 Discourse Manager

The Discourse Manager keeps track of the current state of the conversation, which includes the history, possible next steps, and missing information. In order to do routing, for example, we need a start and an end point for the trip. If a user just enters a destination ("I want to travel to Hamburg"), the conversational interface should detect the missing information and react accordingly, e.g. by sending a message like "From where do you want to travel?". If the user now responses for example "From Cologne", the conversational interface has to know that this is the start point for a trip to the destination Hamburg. This is what the Discourse Manager does. In the prototype we developed, we use Rasa Core[1], an open source discourse manager which is currently in closed beta.

The logic handled by the Discourse Manager can best be visualised by a tree structure. Figure 3 shows a simplified version of the subtree which is used if the user asks for a departure time. The only mandatory information that is needed to search for departure times is the place (e.g. "When can I leave from Sendlinger Tor?"). Therefore, the system has to ask for the place, if it is not given as part of the first message. However, the user might also decide to ask for something different when asked for the place. In this case, the current state (or memory) of the conversation tree is erased and the system jumps back to the top of the root node of the conversation tree. Other information, like a vehicle number ("When does ICE 123 depart from Berlin Hauptbahnhof?"), are optional.

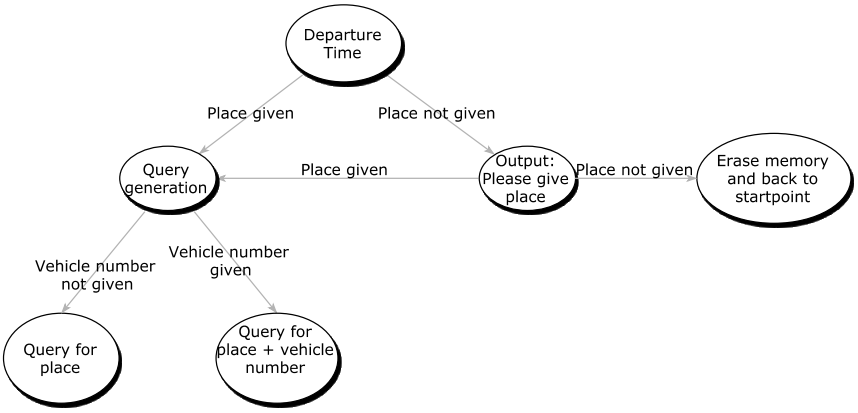

Figure 3. Simplified version of the representation of the subtree for the intent "Departure Time" in the Discourse Manager

5.2 Request Interpretation

In order to analyse incoming requests, we use so called Natural Language Understanding (NLU). Braun et al. [4] evaluated four different NLU services:

[1] https://rasa.ai/products/rasa-core/

Microsoft LUIS, IBM Watson Conversation, API.ai, and Rasa NLU. In their evaluation, Microsoft LUIS performed best, followed by Rasa NLU. Since Rasa NLU was the only tested NLU service which is open source and can be hosted on premise, we decided to use it for our prototype.

We trained the NLU service with two main intents: “Find Connection”, for finding a connection between two places and “Departure Time” for getting the departure time of public transport at a certain station. Moreover, we trained five entities: “Location Start”, “Location Destination”, “Mode of Transportation”, which allows the user to search for a certain mode of transportation, like a train, “Line”, which allows the user to search for a certain line, like U6 or ICE 1234, and “Criterion” which can be used to specify preferences for the routing algorithm, like “fastest”, “cheapest” and so on.

Additionally, we trained some auxiliary intents, like “Set Preferences” or “Greeting” and “Farewell”, which are used to make the interaction with the conversational interface more natural. However, it is important to underline that we do not try to mimic human behaviour and our prototype is built to be a conversational interface, not a chatbot.

For training, we used a corpus with 206 utterances, collected from a mobility chatbot for Telegram and manually annotated by the authors. Overall, the corpus consists of 706 labels, 206 intents and 500 entities.

After an incoming message is processed, the NLU services returns a JSON object containing the extracted information, which could look like this:


```
{
    "text": "Can you find a connection from Berlin to Munich?",
    "intent": "FindConnection",
    "entities": [
      {
        "entity": "Location Start",
        "start": 6,
        "stop": 6,
        "text": "Berlin"
      },
      {
        "entity": "Location Destination",
        "start": 8,
        "stop": 8,
        "text": "Munich"
      }
    ]
}
```


Afterwards, this structured information and our mobility service model, which is described in Section 6, is used to create a query for the routing algorithm, which is described in Section 7.

5.3 Message Generation

Our prototype uses a template-based approach for the generation of response messages, instead of a full NLG pipeline, as suggested by Braun et al. [4]. The data which is used as input for the message generation is returned by the routing algorithm. In order to offer some text variation, multiple templates have been created for each possible response.

An example responses for the intent "Find Connection" is: *"First, take the bus N40 from Haderner Stern to Karlsplatz (Stachus) at 13:08. You will arrive at 13:28. Then, walk from Karlsplatz (Stachus). You will arrive at 13:38. Your journey will take 30 minutes."* An Example response for the intent "Departure Time" is: *"Bus 197 departs at 00:36 from Quiddestraße to Innenring Neuperlach."*

Whenever a user connects to the conversational interface for the first time, a welcome message is send including example questions which can be asked, in order to help the user to get an idea how he can interact with the conversational interface.

6 Meta-Model for Mobility Services

In order to build a system which does not rely on the assumption that mobility service providers will cooperate and provide a certain, unified, API format, we did need to find an abstraction which allows us to subsume the majority of the existing mobility services and their APIs. This abstraction needs to provide sufficient information for intermodal routing but at the same time, it can only rely on information that is provided by almost all mobility service APIs.

In order to find such an abstraction, we analysed multiple APIs, public and private, from different mobility service providers, including Deutsche Bahn, Lufthansa, Drive Now, and MVG. Based on the inspection of these APIs, we developed the three-layered model for mobility services shown in Figure 4.

The first layer of the meta-model, called *MobilityService*, contains all information that is mandatory for every mobility service, this includes the name of the service, the means of transportation offered by the service (e.g. subways, buses, cars, etc.), and its level of operation, which can be either regional, national, or international. In Section 7, we will explain why this information is necessary. We can also check for every mobility service, whether a specific place is reachable with the service or not, and we can ask for a connection from a start point to an end point, which is used by our routing mechanism, described in Section 7.

In the second layer, we differentiate between public transport services and individual transport services like car and bike sharing. For public transport services, there is a function which enables the system to find stations next to a given place and to get a list of departures at a certain station. For the individual transport services, we can find the vehicle closest to a given place. On the third layer, we have concrete models, derived from the above-mentioned layer, for a specific service. Since we do not rely on the cooperation of mobility service providers, an individual model has to be created for every mobility service and has to be maintained in case the API changes. However,

once this model exists, the mobility service can be immediately plugged into the system and will be included in the routing process without any additional changes.

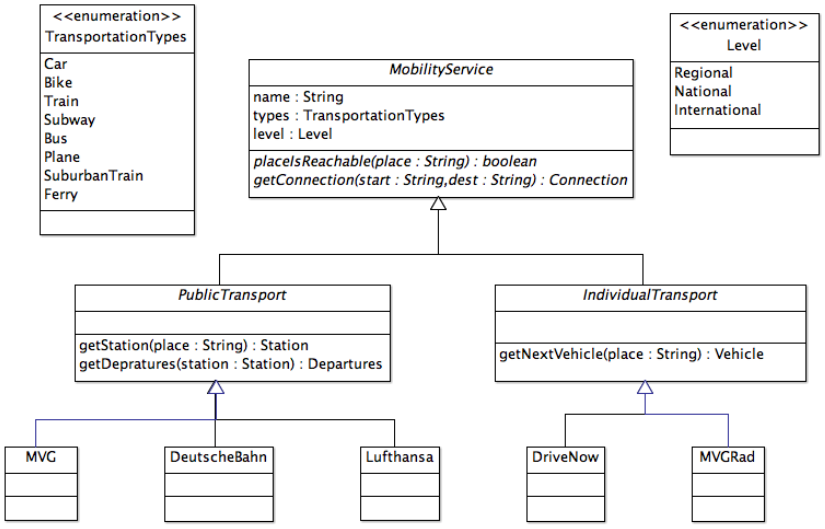

Figure 4. Mobility Service Model

7 Routing

Compared to other approaches, which we introduced in Section 3, our system does not rely on the cooperation of mobility service providers. One drawback of this approach is the fact that we have only very restricted access to information. This specifically limits us with regard to the routing, which is necessary to connect different modes of transportation and enable intermodal mobility.

Since most mobility services offer their own routing algorithm through their API, they do not offer all the information which would be necessary to do the routing manually. Therefore, we have to rely on these routing algorithms and connect them, rather than creating our own routing algorithm based on geospatial data. While this might result in some routes not being found, it enables us to integrate as much services as possible, without relying on the cooperation of their operators.

The main challenge for our routing process is therefore the shift of modes of transportation. If a user, for example, wants to travel from Platz der Republik in Berlin to Marienplatz in Munich, she first has to walk from Platz der Republik to the subway Station “Bundestag” and take the U55 from there to Hauptbahnhof, then take e.g. ICE 1513 from Berlin to Munich and in Munich S1 from Hauptbahnhof to Marienplatz. In this section, we describe, how our routing process solves this problem.

First, we calculate the distance between the start location and the end location. We do this by using the Google Maps API to geocode the extracted addresses (e.g. street names) to coordinates of latitude and longitude. Afterwards, we can use the Haversine

formula, to calculate the distance between the two locations. If the distance is less than 50 km, the trip is classified as "regional". Based on this classification, the system decides which mobility service providers should be checked. For a travel within Berlin, for example, we do not want to check flight or long distance train connections. Therefore, we only check mobility service providers which are also labelled as regional and operate in this region. In Berlin, this could be for example BVG or Car2Go.

A trip from Berlin to Munich, on the other hand, will be labelled as national. Therefore, the system would for example check Deutsche Bahn first. The Deutsche Bahn API will return a route from Berlin Hauptbahnhof to Munich Hauptbahnhof. Afterwards, the system will recursively try to find a route from Platz der Repbulik to Berlin Hauptbahnhof and from Munich Hauptbahnhof to Marienplatz, until a direct connection is found.

One additional challenge of this approach is the fact that most mobility service APIs "greedily" try to deliver an answer for every input. The Münchner Verkehrsgesellschaft (MVG), for example, obviously does not offer any connections to Berlin, however, if the API gets Berlin and Marienplatz as input, it will return a route from "Berlinerstraße" (Berlin street) in Munich to Marienplatz. Therefore, we cannot only rely on the results from the APIs, to check if a place is really reachable with a given service.

We tested different approaches how to deal with this issue, e.g. by calculation how different the returned result (e.g. "Berlinerstraße") is from the actual input (e.g. "Berlin") with linguistic metrics like the the Levenshtein distance. However, we achieved the best results with the same method we also use for the routing itself: we geocode the input and the returned result and calculate the spatial distance between both points with the Haversine formula. If it is more than one kilometre, we assume that the returned place is not identical with the input.

8 Advantages and Limitations

As we have stressed before, one of the key points of our approach is the fact that we do not rely on the cooperation of mobility service providers. While we think, this is a major advantage, it is also the cause for most of the limitations of our approach.

It is an advantage, because in the past, many approaches failed, because mobility service providers could not agree on one data format or were not willing to cooperate with each other at all. Even Google, despite its market power, could only include a limited number of services. And all services included are almost exclusively from the "old" public transport industry. Car sharing services or similar offers are not included. One reason for this might be the fact that traditional public transport providers almost never directly compete with each other in one area, at least in Germany. In Munich, there is MVG, in Berlin, there is BVG, but no competing alternatives. On longer distances, there is a competition between trains and planes, however, Google Maps does not offer intermodal journey planning including flights.

The main disadvantage of our approach is the fact that connectors have to be maintained for all services. While the initial creation is relatively easy, partially thanks to the provided meta-model, every time a service provider changes its API, the

connector has to be adapted, which can lead to an interruption of service, until the connector is updated.

We hope that in the future, mobility service providers will be more willing to cooperate with each other and open standards will emerge for mobility service APIs. While this would make our routing module with the connectors obsolete, the rest of our architecture would still be valid. Moreover, we believe that our meta-model for mobility services and the insights we gained during its design could be a blueprint for the development of an open standard.

From a computational perspective, our approach has another advantage, because most of the routing is outsourced to the servers of mobility service providers. Our system only has to do the routing between different modes of transportation, which is usually a very short distance.

Another main advantage of our architecture is the usage of a conversational interface, which allows a very intuitive interaction with the system. This approach also eradicates the need of downloading and installing an additional app, which makes it more likely to be adopted by users. Additionally, it makes the system very easily incorporable with the popular becoming voice assistants like Apple's Siri or Amazon's Alexa.

9 Conclusion

In this paper, we have presented a customer-centred approach to the intermodal combination of mobility services, using conversational interfaces. Following a design science approach, we have analysed changes in the mobility ecosystem and existing approaches to the intermodal combination of mobility services. We designed a reference architecture, which, in contrast to the existing approaches, does not rely on the cooperation of mobility service providers. Based on this architecture, we have implemented a research prototype combining our findings with the latest research on conversational interfaces, natural language understanding, and context-aware social software. In the discussion of our results, we identified advantages and limitations of the approach.

In the future, we would like to expand our prototype to support additional mobility services. So far, our main focus was on the architecture and the meta-model for mobility services. In the future, an additional focus could be set on the interface, e.g. by comparing our conversational interface to a more classical, form-like interface.

References

1. Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A.: Toward conversational human-computer interaction. AI magazine, 22(4), 27 (2001)
2. Antrim, A., Barbeau, S.J.: The many uses of GTFS data–opening the door to transit and multimodal applications. Location-Aware Information Systems Laboratory at the University of South Florida (2013)

3. Beutel, M. C., Gökay, S., Kluth, W., Krempels, K. H., Samsel, C., & Terwelp, C.: Product oriented integration of heterogeneous mobility services. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, pp. 1529-1534. IEEE (2014)
4. Braun, D., Hernandez Mendez, A., Matthes, F., Langen, M.: Evaluating Natural Language Understanding Services for Conversational Question Answering Systems. Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany (2017)
5. Faber, A., Matthes, F., Michel, F.: Digital Mobility Platforms and Ecosystems. Technical Report, Technische Universität München (2016)
6. Hernandez-Mendez, A., Braun, D., Matthes, F., Langen, M.: Towards a Context-Aware Vertical Social Software Ecosystem. Proceedings of the 19 IEEE Conference on Business Informatics (CBI). Thessaloniki, Greece (2017)
7. Hevner, A.R.: A Three Cycle View of Design Science Research, Scand. J. Inf. Syst., vol. 19, no. 2, pp. 87–92 (2007)
8. Hevner, A.R., Chatterjee S.: Design Research in Information Systems: Theory and Practice, vol. 22. Boston, MA: Springer US (2010)
9. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q 28:75–105 (2004)
10. Masuch, N., Lützenberger, M., & Keiser, J: An open extensible platform for intermodal mobility assistance. Procedia Computer Science, 19, 396-403 (2013)