Flexible Software Support for Mobility Services

Ali Akyol[1], Jantje Halberstadt[1], Kimberly Hebig[2], Jan Jelschen[2], Andreas Winter[2], Alexander Sandau[3], Jorge Marx Gómez[3]

Abstract: Demographic change and growing urbanization are essential reasons for the increasing demand for mobility in rural areas in Germany. Municipalities in sparsely populated counties are confronted with the problem of providing a basic supply of public mobility services. Especially for rural areas, established public transport means have to be complemented by additional, new, and innovative mobility services.

NEMo (Sustainable satisfaction of mobility demands in rural regions) pursues the development of sustainable and innovative mobility services based on tailored business models for rural areas. Traveler information systems are enhanced to handle a range of active and passive mobility services in a very flexible way. Within this paper, three flexible cases are demonstrated. The SENSEI framework for software evolution provides improved software support by adding and modifying capabilities of the existing services to re-orchestrate and re-implement mobility services.

Keywords: mobility services, sustainability mobility, flexible software architecture, software integration, service oriented architecture.

1 Introduction

In Germany, more than 60% of the population lives in rural areas [Bu15]. For rural municipalities and counties, it is becoming increasingly more difficult to ensure a basic supply of public mobility services, such as buses and trains, without asking the question of necessary social participation, meaningful regional creation of value, and last but not least feasible environmental objectives. As rural areas often lack their own infrastructure, such as medical services or shopping malls, a greater dependency on the nearby cities arises. The important connection to these institutions cannot be achieved reliably by public transport [Zä00] [Mo06]. With the help of information and communications technology (ICT), the interdisciplinary and transdisciplinary research project NEMo offers a new opportunity for citizens in rural areas, to simplify their mobility [Je16]. One objective of the NEMo is to provide solutions for sustainable mobility and at the same time ensure an eco-friendly mobility system [CP15]. Nevertheless, it is very important to achieve maximum flexibility in the software system

[1] Leuphana University Lüneburg, Social Entrepreneurship, {ali.akyol, jantje.halberstadt}@leuphana.de
[2] Carl von Ossietzky University Oldenburg, Software Engineering, {hebig,jelschen,winter}@se.uni-oldenburg.de
[3] Carl von Ossietzky University Oldenburg, Very Large Business Applications, {alexander.sandau, jorge.marx.gomez}@uni-oldenburg.de

in order to implement new mobility services and increase the flexibility to integrate new business models, components and processes [AH12; Sa13]. The NEMo project focuses on an interdisciplinary and transdisciplinary approach with different departments, such as software engineering, business information systems, law, service management, social entrepreneurship, etc. [NE17]. Therefore, it is essential to create one taxonomy which integrates several heterogeneous business models and their software support in the NEMo information system.

The next chapter will explain the intention of the taxonomy, clarify the benefits, and show its importance. Afterwards, the third chapter will present the business models, which can be included into the mobility platform by using SENSEI which provides flexibility in orchestrating service [Je15a]. Its flexibility will be illustrated by three examples: adding capabilities, extending orchestrations, and changing component mappings. The conclusion will summarize the relationship between flexible software design and sustainability.

2 Taxonomy of mobility services

In the last chapter, the problem state was shown. With the taxonomy in this new chapter there are the means to build a foundation for communication in the NEMo project. The taxonomy outlines the layers of NEMo and highlights the associations between them. It constitutes the frame to describe any mobility service from the business process to the technical view. One can differentiate between four layers composed equally of composite patters: Mobility Services, Business Models & Processes, IT Services and IT Components.

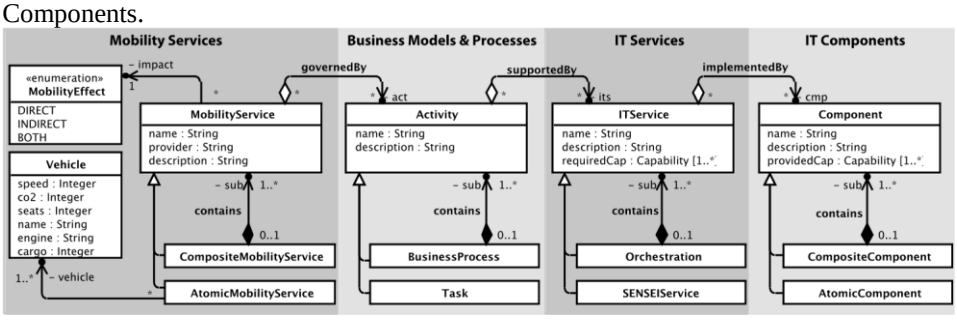

Figure 1: UML class diagram depicting the main concepts of the NEMo taxonomy.

This distinction is needed because the mobility services are part of the mobility while the business models and processes take place in information management. IT-services provide abstract descriptions of the software which are implemented by IT-components.

Mobility Services
A mobility service aims at transporting people (or things) by using vehicles economically and ecologically [AH06]. In other words, having environmentally

conscious behaviour while earning money. A mobility service is also a combination of atomic mobility services. This is needed because of intermodal route planning. Whenever a person wants to get to one point and move on to another, a combination of mobility services is used. Any mobility service can be used to directly transport people, indirectly transport things, or to both transport people and things simultaneously. This is called the "impact of the mobility service". Moreover, a mobility service has a provider who offers it and a description where the business model is shown. The business model is how the mobility service strategically acts to earn money [Ba96].

Business Models & Processes
No mobility service can act without people using it. The latter describes human actions. It defines the tasks they must take before, during, and after using the mobility service. Activities that are made up of more than one task are called business processes [Ro05]. Every activity has a description and a name showing what people are doing in the business process and the tasks themselves. The activities can be supported by IT-services. In chapter 3 the user oriented layers, mobility services, and activities will be characterized in detail.

IT-Services
An IT-service is a description of how a software component should work and how a software component should work and instead of focusing on activities related to human behavior, IT-services focus on appropriate functionalities. [Je15b]. An IT-service is not the implementation as one part of the software but its description. The description is sufficiently complex and granular to be understood from the technical and from a more activity based view. The atomic ones are small parts of the software which must be orchestrated to IT-services. The required capabilities are needed because of the SENSEI-approach, as it can generate components with same capabilities.

Components
A component is the technical implementation of the IT-Service. Many components are based on atomic services that are combined by components leading to composed components. This is necessary because many functions are provided by combinations of smaller components [Je15b]. It is also needed to have components that implement SENSEI-services because of the recoverability a component has. Moreover, even if the SENSEI-services are different the components could be the same. In the fourth chapter, these two technical layers will be characterized in detail.

Summarized, the taxonomy shows four layers building the frame to describe a mobility service from the business model, the user interactions, the ICT support, and the software components. This taxonomy is needed for the following chapters in order to properly describe mobility services.

3 Business Models of Mobility Services

In NEMo, business models with direct and indirect impact on mobility services (Fig. 1) are developed and generate the additional value to simplify the mobility and to improve sustainability. The following table shows mobility services separated by the two different impacts each can have. Mobility services 1-3 have a direct effect while mobility services 4-6 have an indirect effect on the mobility of citizens in rural areas. The mobility services with a direct effect address the mobility of the citizens and try to improve the ecological balance. The others are avoiding mobility by providing surrogate services. All of the shown mobility services are used in the software built into the NEMo project as support for the user, a citizen in a rural area. Business model 1 is the mobility service of carpooling realizing an intelligent coordination platform.

No	MS with direct impact	No	MS with indirect impact
1	Carpooling	4	Supermarket delivery service
2	Collection service for groups	5	Babysitting vs. Shopping
3	Supermarket collection service	6	Social cook

Table 1: Mobility services sorted in types

A user having a car can integrate the routes into the software he usually takes (e.g. to work). He must state information such as start location, start time, and destination. A user without a car has the opportunity to search for routes, he wants to take. The NEMo platform will identify possible carpools and initiate the communication between driver and passenger. A user, searching for available routes will get the results of his search request shown on a dashboard. The request can be accepted or denied by the car owner.

The business model „collection service for groups„ (business model 2) will improve the mobility for a group of people through the NEMo platform. The objective of this service is to provide a sustainable mobility service in rural areas. The service is designed as a pick-up and delivery service. This service is flexible to use and eco-friendly when citizens create groups to use mobility services.

The next business model is the „supermarket collection„ service (business model 3). This mobility service will collect a group of users for a shopping trip to a nearby supermarket. After their purchases, they will get home via the collecting service. The supermarket earns a profit by offering this mobility service because of rising the number of customers. Additionally, the citizens of the rural areas have a better and more comfortable opportunity to get to a supermarket.

The „supermarket delivery service„ (business model 4) is also a kind of delivery service but more eco-friendly. This business model is a mobility service with an indirect impact to the environment. The difference from a classic delivery service is, that the delivery starts, when several households order their goods. The supermarket will deliver twice a week and flexibly, if more households take an order.

The business model „babysitting vs. shopping„ (business model 5) is also a mobility service with indirect impact. This mobility service creates an additional value for users. People, who are not able to drive to the shopping mall themselves, can register on the NEMo platform for babysitting. Parents with children can search on the platform for a babysitter. They have to state the times a babysitter is needed. In return for the work of the babysitter, the parents will do the shopping for them.

An extended version of the previous mobility service is the business model of „social cooking„ (business model 6) which also has an indirect impact. The scenario is similar to the babysitting model. The main difference is the work of the person having no opportunity to get to the supermarket has to do. They will prepare a dinner and calculate the costs for the visitors signing in. In return the dinner, the visitor will do the shopping for them. To implement the mobility services mentioned before as IT components, it is necessary to describe the IT services. The IT services show how an IT component should work functionally. In the next chapter one IT service is described in detail: findRoute. The IT service was chosen because it takes part in each mobility service. The SENSEI approach is used to build up a catalog of IT services generating a higher level of flexibility. These flexibilities can be mapped to the mobility services (sec. 4).

For example, the first flexibility scenario for findRoute is to add capabilities, add a vehicle, and choose the shortest route with the vehicle (sec. 4.1). The second flexibility scenario shows it is possible to support intermodal route finding (sec. 4.2). Different mobility services, such as carpooling and supermarket collection service can be orchestrated, whereby the user gets the maximum amount of flexibility to plan a route. The third software flexibility support of SENSEI enables components to be modified, updated or replaced (sec 4.3). Therefore, it is possible to add a new route finding, modify the current route finding, or replace the current route finding component with a new one.

4 SENSEI

The last chapter gave an overview about mobility services and their business models. This chapter will show the way IT services are building the foundation to implement IT components in order to provide software for citizens in rural areas. Moreover, the SENSEI approach will be introduced.

SENSEI (Software Evolution Services Integration) is a service-oriented approach and framework towards building and integrating highly flexible applications from reusable components [Je15b]. On the one hand, it prescribes a strict separation of conceptual, technology-independent services, and the implementation by components on the other hand. Much more than generally practiced in service-oriented architectures, let alone the technical standards commonly associated with them for their realization (e.g. SOAP-based web services, or RESTful services). The service and component layers are bridged

using model-driven techniques, to automatically derive processes and integration logic from high-level models – either by means of code generation, or by a model interpreter at runtime.

Using SENSEI, basic units of functionality, their inputs, outputs, and capabilities are defined, described, and collated in a **service catalog**. Subsequently, desired application behavior is modeled as **orchestrations** of services: SENSEI defines a process-oriented, graphical language to specify control and data flow between instances of services selected from the catalog as needed.

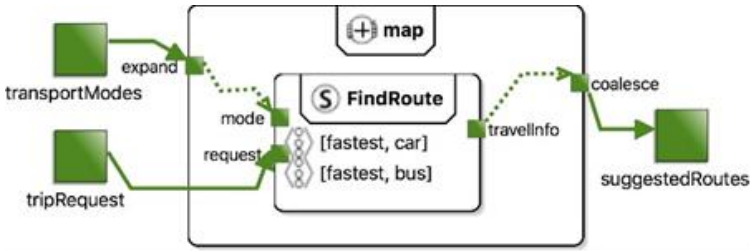

Figure 2: Simplistic SENSEI service orchestration for basic route planning.

Figure 2 shows a very simple orchestration, using only a small IT service, *FindRoute*, to implement basic route planning functionality. The service catalog defines *FindRoute* to require two inputs, the desired *mode of transportation*, and the *trip request*, which encodes starting point, destination, time of departure or arrival, and potentially further travelling constraints. The service's output contains the *traveling information* (e.g. a list of driving directions). The orchestration nests the service instance within a map control flow construct. This is basically a (potentially concurrent) loop that executes its body once for every element in the input stream (expand), and collects the individual results (*coalesce*). As a result, this orchestration calculates a route for each of the specified transportation modes, and returns them as collection of *suggested routes*.

SENSEI services can further have *capabilities* to model specific features which implementations may or may not support. For *FindRoute*, there are *two capability classes* defined in its catalog entry, which represent variation points for implementing components: an *optimization goal*, with individual capabilities to find (e.g. the *fastest*, *shortest*, or *cheapest* route), and a *transport mode*, whose capabilities representing support for finding (e.g. *car*, *bus*, or *walking* routes). In orchestrations, designers can declare their required capabilities for each instantiated service: in Figure 2, the *FindRoute* function is required to support finding the fastest routes for private cars and public busses.

The actual implementation of the functionality, defined by the IT services, is implemented by components, listed in SENSEI's **component registry**. Besides each IT component implementing specific IT services, registry entries also declare *provided capabilities* to further specify the extent of the provided functionality. SENSEI's

capability model is leveraged by its tooling to automatically match orchestrated IT services and their required capabilities to appropriate IT components that provide them.

Through the clear separation of specification and actual implementation, the service layer is kept clean of technical details and complexities, making it easier to develop and evolve. The separation also enforces a strict encapsulation of individual IT services and IT components, respectively, ensuring that no accidental dependencies "creep in", an otherwise common effect of long-term software evolution. Integration logic is either generated, or provided generically by an interpreter, reducing manual effort and further improved flexibility. Three basic flexibility mechanics can be identified in SENSEI and for each of these mechanics, an example scenario is given in the following section to provide details of their respective workings.

1. Adding or modifying required capabilities.
2. Extending or modifying existing orchestrations, or re-orchestrating existing services.
3. Mapping orchestrated IT services to different IT components, or partitioning functionality in IT components in different ways.

4.1 Adding Capabilities

The NEMo project aims at supporting mobility in rural areas in a sustainable, environmental friendly manner. Mobility services like the <carpooling> could therefore be improved if route planning would also provide *shortest* routes, instead of *fastest* routes, assuming the former are generally more ecological than the latter due to lower carbon emissions.

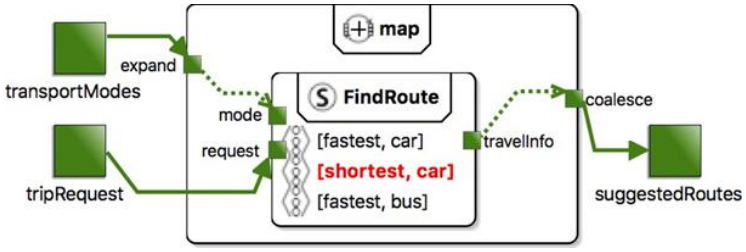

Figure 3: Basic route planning with an additional required capability.

Figure 3 shows an orchestration that has been extended to support this use case. The only difference from Figure 2 is the additional required capabilities for private cars, *shortest* routes will also be provided. In the optimal case, this minimal change is all that is required to add this functionality. SENSEI will automatically integrate additional IT components as needed. In simple cases, as with this example, it is not unlikely that the originally used implementation might also already support this additional functionality. However, SENSEI does not depend on a single component to provide all the required capabilities, but is able to map a small service instance to combined components. The right IT component will be chosen at runtime, based on the declared or provided capabilities and the actual input data.

In the same manner, support for additional transportation modes can be added. What kind of features can be added or modified depends on how IT services are modeled, i.e. what capability classes they define. Of course, this flexibility mechanism assumes that the actual functionality is already available, implemented in registered components, so that SENSEI will be able to integrate them into the modified software solution, fully automatic. Even if IT components are missing and have to be implemented, another benefit of using SENSEI is that the standardization provided through the service catalog facilitates reuse: Once a service is implemented in an IT component, it becomes available for future usage in different contexts. In the long run, the set of components will grow, reducing the need for manual implementation and the effort involved. This makes this scenario more viable over time.

4.2 Extending Orchestrations

An important consideration to take regarding NEMo is the combination of mobility services to provide comprehensive, demand-based mobility options. The basic route planning example is insufficient for this: instead, the ability to provide *inter-modal routing* is needed, meaning the combination of multiple modes of transportation within a single route [Je16].

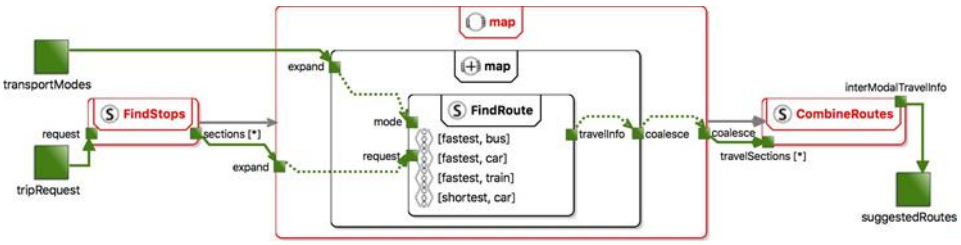

Figure 4: Inter-modal route planning orchestration.

A straight-forward way to evolve the SENSEI-integrated software system that provides the route planning functionality is to extend the existing orchestration with additional IT service instances. Figure 4 shows an orchestration for *inter-model route planning*. At its core, it still contains basic route planning, but has two additional IT service instances, and uses another *map* control flow construct.

The first step is now performed by an instance of the *FindStops* service, which takes the trip request and tries to partition it by determining reasonable places to switch transport mode (e.g. bus stops and train stations). This results in a set of sub-requests, with the discovered stops as new points of origin or destinations. The outer map construct iterates these, so that the *FindRoute* instance will now be invoked once for each pairing of sub-request and transportation mode, yielding multiple, partial routes. The last step is to stitch these partial routes together again, so that they satisfy the original travel request,

which is done by *CombineRoutes*.

The high abstraction level of orchestrations simplifies the task of defining and evolving the desired processes. Orchestration designers do not have to address different interface or binding technologies, disparate data formats, and technical incompatibilities between IT components of different vendors or providers. SENSEI shifts the burden of providing interface adapter and data transformation logic from software integrators to component developers, and imposes a structure that promotes its reusability, as opposed to fusing it to individual components. The fact that integration logic is either auto-generated and thus "discardable", or provided at runtime by an interpreter, prevents SENSEI-based software systems from becoming entangled in hard-wired dependencies.

4.3 Changing Component Mappings

To realize the functionality modeled in orchestrations, SENSEI maps the instantiated IT services to the IT components providing them. There is an n-to-m relationship between IT services and IT components: as stated earlier, a small service can be implemented by combined components employing different capabilities, and the opposite is also possible, with an atomic component providing multiple, different services.

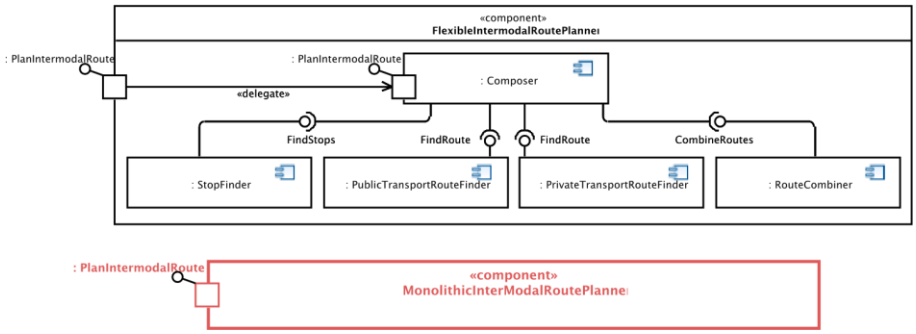

Figure 5: Composition of combined components, and a monolithic component, both implementing the PlanIntermodalRoute service

In this regard, SENSEI does not make any assumptions. In other words, invoking IT services for IT components always assumes that they have no knowledge of the overall process they are contributing to, and that they cannot, and will not, communicate directly with other IT components. All information goes through a central hub, the *composer*, which can be seen in the top half of Figure 5. This IT component diagram shows an IT component composition that realizes intermodal route planning, as specified by the orchestration in Figure 4. The *FindStops* service is mapped to the component *StopFinder*, and CombineRoutes is mapped to *RouteCombiner*. *FindRoute* is associated to two components: *PublicTransportRouteFinder* provides the capability for bus and train routes, while *PrivateTransportRouteFinder* provides routes for driving a private car and walking.

The composer realizes the control and data flow specified by the orchestration, invoking the right IT components in the right order, and passing along data. The composer is either fully auto-generated, or is embodied by an interpreter. The whole composition of composer and service-providing IT components provides the *PlanIntermodalRoute* service, which is also induced by the corresponding orchestration. This IT service can be instantiated in other orchestrations, forming the hierarchy of IT services and orchestrations seen in the IT *Services* layer of Figure 1.

This architecture of completely isolating individual components is essential to delivering SENSEI's stated goal of sustainable, high flexibility and reusability. Allowing direct communication between components and interdependencies would quickly undermine this objective. However, these benefits are traded in for a potential curtailment of other attributes, e.g. run-time performance. While a smart composer, and the overall SENSEI infrastructure, could certainly try to put optimizations in place (e.g. to reduce the data traffic through caching, in general, there will be an overhead). It is also a matter of finding the right level of granularity for IT services: more finely-grained IT services may be more reusable, but more coarsely-grained ones may allow for more internal cohesion, and optimizations that would otherwise incur a steep performance penalty because of the higher communication overhead.

While orchestrations are also IT service instances (*PlanIntermodalRoute*), they can also be mapped to an atomic monolithic IT component, as suggested in the bottom half of Figure 5. Such an IT component could, for example, realize a more complex route-finding algorithm that requires its constituents to be more closely attuned to each other and share large amounts of data. Comparable to the individual service-providing IT components above, such an atomic IT component is viewed by SENSEI as a black box, only having to adhere to the interface defined by the IT service, while the exact manner of implementation is left unconstrained. Specifically, a monolithic component does not have to follow the process prescribed by an orchestration. With this mechanism, SENSEI allows trade performance and scalability against flexibility and reusability. The latter only has to be sacrificed for select parts of an overall software system, for which performance is critical. Also, such monolithic implementations fit into SENSEI seamlessly, being treated like any other component, just providing a more coarse-grained service.

5 Evaluation

As previously introduced, the mobility landscapes change rapidly with ongoing market dynamics. As a result, new mobility providers with correlating mobility services arising have to be integrated into the corresponding traveler information system. The proposed integration of the SENSEI framework enhances the existing traveler information system of the NEMo project in terms of flexibility and leading to a more sustainable software

design. To show the benefits of this integration three different scenarios have been chosen to demonstrate the software support by adding/modifying capabilities of services, re-orchestrating, and re-implementing mobility services. To achieve the integration of the SENSEI framework, the existing IT services have to be cut up into small parts of functionalities. This enables the existing and the integration of new functionalities to be reused and support upcoming business models. In terms of the project, the existing public transport planning service will be divided into the proposed manner according to Figure 3 and Figure 4, resulting in a system architecture as shown in Figure 5. For the first step, existing functionalities of the traveler information system migrate to new architecture. During the following steps, the approach is tested with new business models that are developed in the NEMo project. The system is iteratively tested with citizen participation in field trials in rural areas.

6 Conclusion

The challenge of the inter- and transdisciplinary research project NEMo was to improve the insufficient mobility offers in rural areas. To prevent misunderstanding, a taxonomy was created for the project. Furthermore, business models with direct and indirect impact were developed to simplify the sustainable mobility. For realization, an intelligent and flexible software is required to integrate new business models. With the application of SENSEI, the software system is sustainable and supports three flexibility scenarios.

The first flexibility scenario demonstrated the ability to add capabilities in the mobility services. Despite the changes, the software system proposed the optimized route based on the customer's preferences. For example, the customer can set preferences like fastest or shortest trip, when the software calculates the route it will automatically include all aspects (vehicles such as bus, railway carpooling etc.) and return an optimized result. The SENSEI approach makes it possible, that components can be replaced and new components can be added without to change the whole software system (second flexibility). By using a software which is flexible, it is easier to change IT components, maintain the software, and update with new developed components or business models [Sa13]. For example, to clarify the advantage of the SENSEI framework, the public transport map for bus stops will be updated with new times and parameters which replace or change the maps. It is also possible to add new vehicles like e-cars using one's own map for e-car stations and a new system for calculating the best routes. The third flexibility scenario is the application of SENSEI support of the software in its flexibility and all included components. In summary, of the mentioned facts about the flexible software system, the life cycle of a software system will be extended when it is flexible and expandable [CP15].

Acknowledgement This work takes part in the project "NEMo - Sustainable satisfaction of mobility demands in rural regions". The project is funded by the Ministry of Science and Culture of Lower Saxony and the Volkswagen Foundation (VolkswagenStiftung)

through the “Niedersächsisches Vorab” grant program (grant number VWZN3122).”

References

[AH06] Ammoser, H.; Hoppe, M.: Glossar Verkehrswesen und Verkehrswissenschaften: Definitionen und Erläuterungen zu Begriffen des Transport- und Nachrichtenwesens, ger, Diskussionsbeiträge aus dem Institut für Wirtschaft und Verkehr 2006,2; 2/2006, 2006, url: http://hdl.handle.net/10419/22704.

[AH12] Ahrend, C.; Herget, M.: Umwelt-und familien-freundliche Mobilität im ländlichen Raum. Handbuch für nachhaltige Regionalentwicklung. Berlin/, 2012.

[Ba96] Baatz, E.: Baatz, E.B.: Will Your Business Model Float? In: WebMaster Magazine, Stand: 10.1996, 1996, url: http://www.cio.com/archive/webbusiness/ 100196_float.html.

[Bu15] Bundesamt, S.: Statistisches Bundesamt (ed.): Statistisches Jahrbuch Deutschland. Wiesbaden/, 2015.

[CP15] Calero, C.; Piattini, M.: Green in Software Engineering. Springer International Publishing, 2015.

[Je15a] Jelschen,J.:Service-OrientedToolchainsforSoftwareEvolution.9thInternational Symposium on the Maintenance and Evolution of Service-Oriented and CloudBased Environments (MESOCA)/, pp. 51–58, 2015.

[Je15b] Jelschen, J.: Service-oriented toolchains for software evolution. In: 2015 IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA). IEEE, pp. 51–58, Oct. 2015, isbn: 978-1-4673-7935-9,

[Je16] Jelschen, J.; Küpker, C.A.; Winter, A.; Sandau, A.; Wagner vom Berg, B.; Gómez, J.M.: Towards a Sustainable Software Architecture for the NEMo Mobility Platform. In: Proceedings of the 30th International Conference on Environmental Informatics Stability, Continuity, Innovation: Current trends and future perspectives based on 30 years of history (EnviroInfo 2016). Berlin, 2016.

[Mo06] Für Mobilitätsforschung, I.: Öffentlicher Personennahverkehr - Herausforderungen und Chancen. Springer, Berlin, 2006.

[NE17] NEMo: NEMo - Mobilität, Wissenschaftliche Leistung, Stand: 13.04.17, 2017, url: https://www.nemo-mobilitaet.de/blog/de/projektkonsortium/ wissenschaftliche-leitung.

[Ro05] Rosenkranz, F.: Geschäftsprozesse: Modell- und computergestützte Planung. Springer Berlin Heidelberg, 2005.

[Sa13] Sametinger, J.: Software Engineering with Reusable Components. Springer Berlin Heidelberg, 2013.

[Zä00] Zängler,T.:Mikroanalyse des Mobilitätsverhaltens in Alltag und Freizeit. Springer, Berlin, 2000.