Model-driven development of SOA-based Driver Assistance Systems

Marco Wagner
Heilbronn University
Heilbronn
Germany
Marco.Wagner@hs-heilbronn.de

Dieter Zöbel
University of Koblenz-Landau
Koblenz
Germany
Zoebel@uni-koblenz.de

Ansgar Meroth
Heilbronn University
Heilbronn
Germany
Ansgar.Meroth@hs-heilbronn.de

ABSTRACT
This paper describes an approach towards model-driven development of SOA-based Driver Assistance Systems. In the field of assistance systems for truck and trailer combinations Service-oriented Architecture (SOA) is a promising approach to handle the heterogeneity and the high degree of distribution of these systems. Through connecting or disconnecting trailers the system is very likely to change at runtime which sets up the demand of runtime adaption. This paper illustrates a process model to use SoaML for modeling the components and architectures of these systems. Based on these models, model-driven runtime adaption can be carried out.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**]: Computer-added software engineering – *model-driven development, process model, SOMA*. D.2.11 [**Software Architectures**]: Service-oriented architecture (SOA) – *runtime adaption.* J.7 [**Computers in other systems**]: Consumer Products – *Embedded Systems, Driver Assistance Systems.*

General Terms
Design, Standardization.

Keywords
Embedded Systems, Driver Assistance Systems, Runtime Adaption, Process Model, Service-oriented Architecture (SOA), SOMA.

1. INTRODUCTION
Modern Driver Assistance Systems (DAS) are supporting the driver in many situations. They, for example, assist the driver while changing lanes or influence the brakes of a car in order to keep it on the track. A new category of DAS are systems which support the driver backing up articulated vehicles. From an architectural point of view these systems are quite special. This is mainly because the components needed are distributed over at least two separate units. The connection of these units is not permanent and one pulling vehicle may be hooked-up to several different trailers over time. An example of these systems is the visual assistance as shown in Figure 1. The idea is to calculate the trajectories of the trailer and the overall vehicle and to overlay these on the picture of a rear view camera mounted on the trailer [16]. In order to do so, the steering angle and the angle between truck and trailer, the so-called bending angle, are determined to calculate these trajectories.

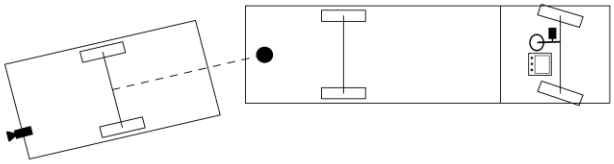

Figure 1.
Components of the steering assistance for one-axle trailers.

The previously mentioned system is just one option. Within the real time systems group of the University of Koblenz-Landau several other assistance approaches have been developed. Besides a visual human computer interface the modality used could as well be acoustical or tactile. An approach using modified semantics of the steering wheel has been also investigated [17]. The number of possible variations of the system is highly increased considering the different types of trailers that could be used. This multitude of variations combined with a high degree of distribution of the heterogeneous subsystems and the possibility that the system could change at runtime through disconnecting or connecting one or more trailers cannot be handled by state-of-the-art software architectures in the automotive domain. Therefore, we proposed a novel approach in [1], using service-orientation combined with software agents. In this approach, all functionality is encapsulated in fine-grained services. These services may be located on any device within the articulated vehicle or even on a nomadic device like a Smartphone. In order to set up the assistance one or more software agents discover the currently available services, determine the possible types of assistance and orchestrate the services. This re-configuration is done every time the configuration of the vehicle changes, for example, by connecting a trailer or in case an electronic control unit (ECU) or sensor system fails to work.

In this paper, we focus on identifying a modeling language as well as a process model to specify adaptive embedded systems. As a case study we are using the assistance approaches introduced earlier. With the chosen language and methodology, we aim on collecting and formalizing the assistance approaches being developed so far. We also want to identify the functionalities

needed in each type of assistance in order to find similarities. In the next step, these functionalities are converted into service specifications. Using these specifications architectures can be developed for each type of assistance. By merging the modeled architectures a library of SOA-based assistance systems is formed. This library along with the specifications of the services provides the basis for deploying model-driven runtime adaption.
The remainder of this paper is organized as follows: Section 2 introduces and categorizes several process models for developing Service-oriented Architectures. In Section 3, the customized process model we use is presented and differences to state-of-the-art approaches are pointed out. Section 4 concludes the paper and provides information on the future work within this project.

2. PROCESS MODELS FOR SERVICE-ORIENTED ARCHITECTURES

With the paradigm of Service-orientation getting more and more popular, the number of process models to develop such systems went up, too. In 2009 Thomas, Leyking and Scheid identified 21 different approaches in [2]. Most of the currently available models are built to suit for some special purpose, require a particular tool chain or concentrate on one field of application only. However, none of them suits to the domain of automotive SOA solutions. Instead of developing yet another model, we decided to find a process model that can be customized to this special scenario. In order to do so, criteria have been developed and the available approaches have been evaluated based on these. The following criteria have been defined:

1. Completeness of the specification phase
2. Independence of a specific field of application
3. Variability in the scenario of development
4. Tool support
5. Acceptance of the modeling language

Our first criterion is that the modeling approach has to allow a complete system specification which includes the specification of the services as well as the service architectures. This also implies that a detailed technical point of view should be assured rather than focusing on the business domain which is very common using SOA. Finally, concrete methods or techniques on how to carry out the steps within the process model should be proposed.
The second criterion is that the field of application should not be restricted. Specialized models, used for Web Services for example, are not very promising since their focus is too narrow. Converting these to suit embedded automotive systems would change too many of their essential ideas if possible at all.
Another criterion is that the starting position at the very beginning of the process should be variable. This is important because the process model should allow new developments as well as migrating existing systems into SOA.
The fourth criterion is that tool support should be given. Using a tool that for example allows modeling the system graphically reduces development time. In addition, implemented validation functionalities decrease the probability of semantic errors.
Finally, the last criterion is that the modeling language deployed is widely-used and hereby accepted. This demand is set up to ensure the readability of the models in the scientific community.
Using these criteria, eleven process models are analyzed. The first one is a model proposed by Stein and Ivanov in [3]. The model is based on ten phases starting with a business process model ending with the deployment of the developed system. It focuses on business processes and the modeling languages suggested belong to the domain of Web Services. A similar model, the Enterprise SOA Roadmap method is presented in [4]. This model also emphasizes on business modeling since only one of the five steps to be executed is technical. Both of the process models violate criteria two that they shouldn't restrict the area of application.
Other approaches lack concrete modeling techniques. Pingel [5] for example, introduces a technology independent five phase model extending well-known approaches. Another approach in this category is a proposal of Mathas [6] which extends the software lifecycle model by adding some SOA-specific tasks and roles while staying coarse-grained. The Service-oriented Modeling Framework developed by Bell is quite generic, too [7]. The idea of the author is to design a concrete process model for every case of application derived from his abstract methodology. Bell also proposes a special design notation which violates the criterion of using a widely-used modeling language. All these models are rather to be seen as suggestions on how a process model may be set up than being a concrete model itself.
Unlike the previously named ones the models "Service-oriented design and development (SOAD)" [8] by Papazoglou and van den Heuvel and "Creating Service-oriented Architectures (CSOA)" [9] developed by Barry are technical in nature. Both of them are phase-oriented and contain practical techniques to be performed in those phases. Through basing on modeling languages like the business-oriented "Business Process Modeling Language (BPML)" or the "Business Process Execution Language for Web Services (WS-BPEL)" they cannot be used for other fields of application without major changes. This fact violates criterion two.
Another approach is presented by Nadhan in [10]. The author describes a seven step procedure to migrate an existing solution into a SOA-based system focusing on technical issues. Targeting only on the migration scenario this model cannot be used for new developments. In doing so criterion 3 is violated.
Some highly interesting approaches are using the Service-oriented modeling language (SoaML), a notation created to model and design SOA-based systems. This is a promising approach because the language itself satisfies the criteria set up in being not restricted to one field of application and being widely used since it is a profile of the popular Unified Modeling Language (UML). One of these process models is presented in [11]. The authors describe the development of a Service-based monitoring system by identifying and specifying the needed services. Although this is very promising, it does not allow specifying the architecture of the overall system which violates the criterion of enabling the user to carry out a complete system specification. Another methodology using SoaML introduced in [12] closely follows the processes defined in the Model-driven architecture (MDA) approach published by the Object Management Group. Tool support is granted by the modeling tool "Modelio". This process model defines several specification steps within the computational independent model and the platform independent model of MDA. The approach is very close to "Service-oriented Modeling and Architecture (SOMA)" presented in [13]. This phase-oriented lifecycle model is based on the "Rational Software Architect" and is also free of any restrictions with respect of the area of application. Both of the lastly named methodologies are fitting the criteria set up earlier in this paper. The reasons why SOMA is favored is being more focused on technical issues and offering a more straightforward workflow.

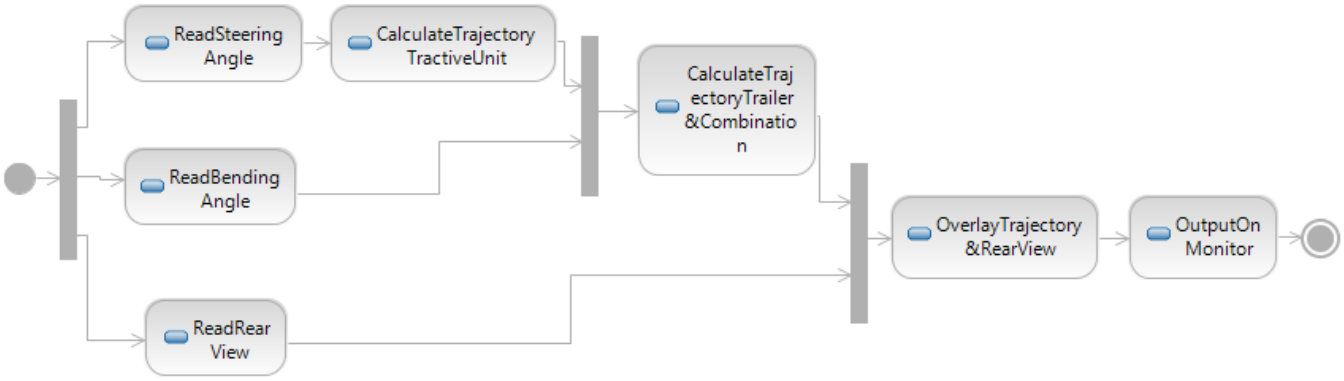

Figure 3. Activity model of the visual steering assistance.

In the next section SOMA is presented in detail and the changes to suit it to embedded automotive systems are explained.

3. CUSTOMIZING SOMA FOR AUTOMOTIVE EMBEDDED SYSTEMS

3.1 Introduction to SOMA

The SOMA methodology has been published by Arsanjani in 2004 [14]. The idea of this approach is to set up a phase-oriented process model that guides through the whole development process. The different phases of the model can be seen in Figure 2. Within the first step named "Business modeling and transformation", the requirements, namely the business processes are modeled and optimized to get a semi-formal description of the workflow. This is normally done using the Business Process Model and Notation (BPMN). Simultaneously, the concomitant project management processes are defined and the computation platform to be used is selected. In SOMA this step is called "Solution management".

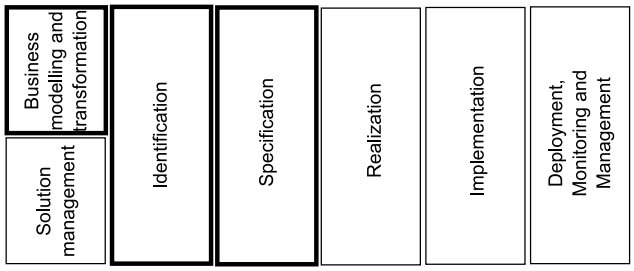

Figure 2. Overview of the SOMA phases [13].

In order to achieve an architecture model, first the service candidates are identified based on the components and flows of the business model. Therefore, SOMA recommends a number of identification techniques that might be used. Next, within the specification phase, the candidates are transformed into services. This is done by modeling the Service Interfaces, Service Contracts and the Participants which realize the functionality of the services. Also, the Service Architecture of the overall system is defined. In the next phase, the so called "Realization", the focus swaps from functional to non-functional requirements. This includes for example the development of the abstraction layers or the communication model. The following step "Implementation" is used to generate or write code that realizes the functionalities and in addition, the code is being tested to fulfill its requirements. In the last phase of SOMA the developed system is put into operation. The functionalities are monitored at runtime and the infrastructure and the network are managed to ensure stability and performance.

In the next subsection, our approach for a Model-driven development of SOA-based Driver Assistance Systems will be described in detail. Focusing only on the functional issues of the system, the business process based SOMA approach is customized towards a methodology suitable of handling embedded automotive applications. Therefore, the phases "Business modeling and transformation", "Identification" and "Specification" are refined.

3.2 Customized phases of SOMA

In its first phase, Service-Oriented Modeling and Architecture conducts the development of a business process model. This step aims at identifying the tasks and parties within the workflow. Therefore, SOMA recommends the usage of BPMN as a graphical representation to specify business processes. This makes sense for developing SOA solutions in a business context. BPMN is, however, not created to describe technical systems like DAS. To solve this issue, we propose to use an UML 2 Activity Diagram. Similar to BPMN models, Activity Diagrams describe workflows consisting of a number of activities. These activities are important here because they accumulate the functionalities of the system. As they are not restricted to the business domain, Activity Diagrams allow modeling embedded systems without violating the semantics of its components. Figure 3 shows the Activity Diagram of the visual steering assistance system introduced in Section 1. All actions that have to be done to carry out the assistance are modeled as activities. The control flow describes how they cooperate to represent the DAS.

The Activity Diagram itself may be modeled using any kind of description of the system. This includes a specification as well as a systems requirement model or a description in natural language. Within a migration scenario, code may be analyzed to create the diagram.

Having finished the modeling of the workflow using an Activity Diagram, the next step is to identify the service candidates. In SoaML they are called Capabilities [15]. These Capabilities represent entities that offer some distinct functionality and therefore are predestined to become services. Following the recommendations of SOMA, one of the most straightforward ways to identify these service candidates is to analyze the BPMN model created in the first step. This is done by extracting the lanes of the

model which represent some participating party and transfer them into Capabilities. The tasks executed by these parties are modeled as the operations those Capabilities provide. Eventually, this leads to a coarse-grained model with a relatively low number of services.

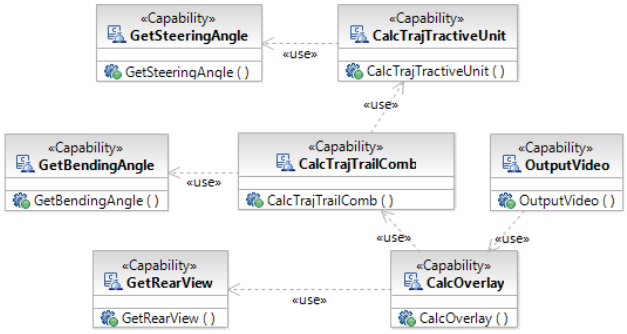

Figure 4.
The Service Candidates derived from the Activity Diagram.

In order to design a highly flexible SOA-based Driver Assistance System, we are confident that the services identified with the SOMA method are too coarse-grained. Within this context, the granularity should be enlarged to a certain extend. Considering this demand and the fact that the starting point of this step has changed from a BPMN model to an Activity Diagram, the identification phase has to be changed. In our modified phase, we transfer each activity of the Activity Diagram shown in Figure 3 into a service candidate. The result of this transformation in the case of our example system can be seen in Figure 4. For example, the Capability "GetSteeringAngle", which reads out the current steering angle, may be used in other assistance scenarios as well. A more coarse-grained modeling might prohibit such re-use.
The next phase of the SOMA methodology is the specification. Since this phase is very extensive, it is split up into four sub-phases; specification of the Service Interfaces, Service Contracts, Participants and the Service Architectures.
The specification of the Service Interfaces in SOMA is done by deriving them from the Capabilities. In order to do so, each Capability is represented by a single Service Interface. Furthermore SOMA recommends to specify a number of sub-interfaces of the UML type "Interface" and to assign the operations of the capability to one of these sub-interfaces. Beyond that, no rules or guidelines are given on how many sub-interfaces should be created or how the operations should be distributed onto these.

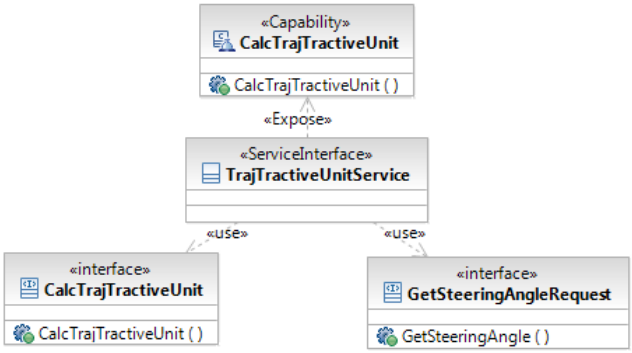

Figure 5. The Service Interface of a service to calculate the trajectory of the drawing vehicle.

Obtaining a common structure is essential to be able to use the model for runtime adaption. Therefore, we have to extend SOMA in this phase, too. This is done by defining two extra rules. At first, any functionality that is provided by the service is mapped into its own sub-interface. These sub-interfaces are so called provided interfaces. The second rule is to create a sub-interface for any functionality that is needed by the service in order to fulfill its tasks. The interfaces modeling the need for a particular service are called requested interfaces. The result of these guidelines can be seen in Figure 5. As an example, the interface of a service is shown that offers to calculate the trajectory of the pulling vehicle. This provided service can be seen on the left encapsulated into its own sub-interface. To be able to calculate the trajectory, it needs the current value of the steering angle. This necessity is expressed by the sub-interface on the right.
In a second specification step, Service Contracts are defined. For interacting with a Service Interface the consumer needs to know how to access it. Therefore one or more Service Contracts are defined. Service Contracts formalize the exchange of information between the provider and the consumer of a service [15]. SOMA develops contracts by specifying two attributes: the roles and the protocol of such a service call. The roles can either be "Service Interface", "Interface" or "Class" types according to the SoaML specification. Describing the protocol, any adequate diagram defined in UML may be used such as interaction or state diagram.

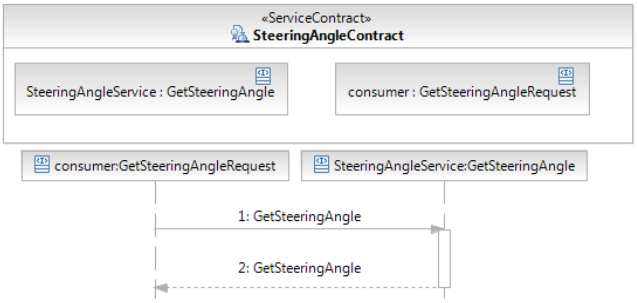

Figure 6. The Service Contract of a Service to determine the steering angle of the drawing vehicle.

We decided to add several constraints to the generic SOMA approach, in order to use the contracts for runtime adaption. One of these constraints is the obligatory use of a Sequence Diagram to model the protocol. This regulation helps to keep a common structure while the Sequence Diagram is able to model further attributes such as time limits. The second constraint is that the messages exchanged are in Remote Procedure Call (RPC) style. Compared to the document style exchange used for Web Services, this method keeps the amount of data transmitted low which is crucial for embedded computing. Figure 6 presents such a contract created by the changed SOMA methodology using the example of a service developed to determine the steering angle of the vehicle. In this contract, two roles are defined: a provider called "SteeringAngleService" and a consumer. The RPC style protocol is defined in a dedicated Sequence Diagram pictured at the bottom of the figure.
Step three within the specification phase of SOMA is the introduction of Participants. In the systems domain a Participant might be a system, application or component that offers or consumes a service [15]. SOMA uses this type as some kind of particular unit. Therefore, a Participant is created and assigned with one or more ports where each port represents a Service Interface. The idea is to map the functionalities encapsulated

within the services to hardware units to use these units for the Service Architecture specified in the last step of the specification phase. The Service Architectures being developed by this approach are rather System Architectures. This is because they do not only illustrate the relations between the software components but also between the hardware components hosting the software.

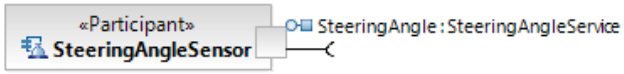

Figure 7. The Participant realizing the steering angle Service.

Since one of the goals of our approach is to allow the services to be distributed on any ECU within the vehicle combination, we do not want to map them onto hardware entities at this point. Therefore, SOMA has to be modified at this step, too. In doing so we are using the broadly framed specification of a Participant in SoaML. Since a Participant is defined to be a unit that provides or consumes services, it is also possible to specify it to be an instantiated process. This process may run on any hardware system of the vehicle. This definition avoids mapping the services to a particular hardware device without violating the specification of SoaML. For example, the Participant realizing the steering angle service is shown in Figure 7.

The last step of the SOMA specification phase is to specify the architecture of the overall system. The Service Architecture illustrates the relationships between the participants involved using ports and contracts. This is done by assigning the ports of the participants to roles within the contracts. This time, we are able to adopt the procedure recommended by SOMA. The idea of our approach is to create a Service Architecture model for every type of DAS for articulated vehicles developed. An example can be seen in Figure 8.

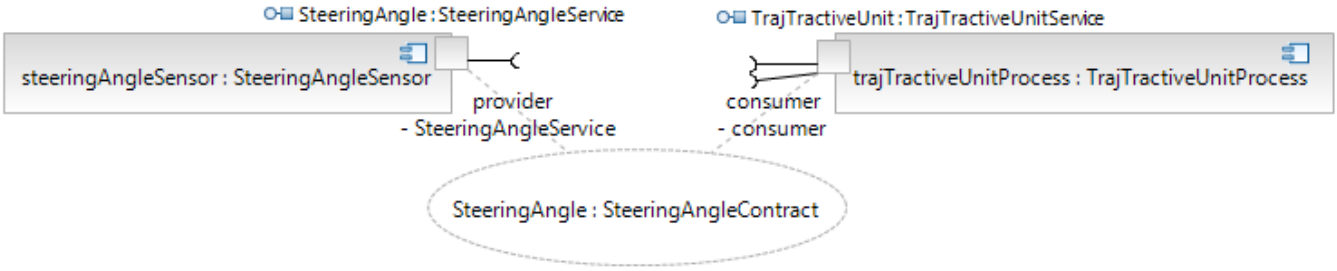

Figure 8. Part of the Service Architecture for the visual steering assistance.

In this figure only a small part of the architecture is shown in order to obtain lucidity. It shows how the Participant realizing the calculation of the trajectory of the drawing vehicle uses the contract of the steering angle service to obtain the data needed.

Finalizing the specification phase by modeling the Service Architectures, the phases of SOMA conducting the modeling of functional attributes is finished. Since the development of non-functional components is not in scope of this paper our modified SOMA process model is completed.

3.3 Model-driven adaption

As a result, this customized process model helps to build up two different databases that can be used for model-driven runtime adaption. First of all, a Service Inventory and hereby a catalog of functionalities is established. It contains a list of the services as well as a description of what they do and how they can be accessed. Second, a library of Service Architectures is created. This library forms a well-defined collection of assistance types. Using this data, the types of the services needed to represent a specific kind of assistance can be determined. These databases form the basis of two different adaption approaches.

The first idea is to pursue an architecture-driven approach. Using a software agent that overlooks the whole system, the currently available services are determined. This is followed by matching them to the catalog of Service Architectures. In doing this the types of assistance realizable can be detected. Additionally, the information about the relationships between the Participants realizing the service can be used to connect them and build up the assistance system.

The information modeled in the Service Interfaces could be used to execute adaption as well, using an interface-driven approach. Since every Service Interface contains not only the services provided but also the services consumed, it is able to explore whether the requested services are currently available within the system. Starting from a data sink, this could be used to determine possible assistance types as well. In the given example, the video out, which is able to offer visual assistance, would start looking for an overlay service which is modeled as its requested interface. The search for this service can be done by invoking the Service Discovery functionality. Having found such an overlay service in the current vehicle configuration, this service itself starts to search for its requested partners. If the chain can be finished and every service requested can be found, the kind of assistance is ready to be used. If some service needed is missing, the adaption mechanism stops.

4. CONCLUSIONS AND FUTURE WORK

By modifying SOMA, we have found an approach to model embedded automotive systems basing on Service-oriented Architecture. In order to evaluate the process model, several types of assistance systems have been specified. The systems successfully modeled so far, are the visual assistance for the one-axle and two-axle trailer as well as the acoustical and haptic assistance for the one-axle trailer. The assistance using modified semantics of the steering wheel has also been specified using this approach.

The process model presented fulfills the demands described in Section II. We are able to collect and formalize already existing assistance approaches as well as new developments. By finding service candidates, the functionalities needed within the different DAS are identified. Since theses functionalities are merged into a common database, similarities can be detected. Going through the

different steps of the specification phase these functionalities are converted into service specifications as well as architecture specifications. This data is collected within two databases and allows to be used for model-driven runtime adaption as described in Section III.

Having now established the process model and specified the functional attributes of the services and Service Architectures, we are now able to move on with the non-functional components.

The next step is the implementation of a Quality of Service (QoS) parameter for each service. This parameter needs to reflect the performance of each service affected by influences from inside and outside the component. It should be easily computable and allow a comparison between services of the same functionality. The QoS parameter will also be taken into account when a service selection algorithm is set up. We also aim on using the parameterized model for online verification conducted using formal methods. This will be done by transferring the SoaML model into hybrid automata.

Another element of our future work is to define a communication model. State of the art in the modern automobile has the ECUs are connected using a mixture of automotive specific network systems. The communication system to be developed has to be highly flexible and independent from the kind of network used. At the same time the overhead produced should be minimal. Achieving this, the unique characteristics of automotive network systems will be taken into account.

The integration of the approach into AUTOSAR will also be discussed within the project.

The last step of the project will be to validate the architecture using a full scale prototype.

5. ACKNOWLEDGMENT

Marco Wagner has been supported by a grant of the "Thomas Gessmann-Stiftung", Essen, Germany.

6. REFERENCES

[1] Wagner, M., Zöbel, D. and Meroth, A. Towards an adaptive Software and System Architecture for Driver Assistance Systems. In *Proceedings of the 4th IEEE International Conference on Computer Science and Information Technology (ICCSIT 2011)* (Chengdu, China, June 10-12, 2011). Wiley-IEEE Press, Piscataway, NY, 2011, Vol. 4, 174-178.

[2] Thomas, O., Leyking, K. and Scheid, M. Serviceorientierte Vorgehensmodelle: Überblick, Klassifikation und Vergleich. *Informatik Spektrum*, *33* (4). 363-379.

[3] Stein, S., Ivanov, K. Vorgehensmodell zur Entwicklung von Geschäftsservicen. in Fähnrich, K.-P. and Thränert M. *Integration Engineering – Motivation, Begriffe, Methoden und Anwendungsfälle*, Leipziger Informatik-Verbund, Leipzig, 2007.

[4] Hack, S. and Lindemann, M. Enterprise SOA Roadmap. SAP Press, Bonn, 2007.

[5] Pingel, D. Der SOA-Entwicklungsprozess. in Starke, G. and Tilkov, S. *SOA-Expertenwissen*, DPunkt Verlag, Heidelberg, 2007, 187-200.

[6] Mathas, C. SOA intern. Hanser Verlag, Munich, 2007.

[7] Bell, M. Service-Oriented Modeling. John Wiley & Sons, Hoboken, NJ, 2008.

[8] Papazoglou, M. and Van Den Heuvel, W.J. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, *2* (4/2006), 412-442.

[9] Barry, D. Web services and service-oriented architecture. Morgan Kaufmann Publishers, San Francisco, 2003.

[10] Nadhan, E.G. Seven Steps to a Service-oriented Evolution. *Business Integration Journal*, *1* (2004), 41-44.

[11] Gebhart, M., Moßgraber, J., Usländer, T. and Abeck, S., SoaML-basierter Entwurf eines dienstorientierten Überwachungssystems. in *40. Jahrestagung der Gesellschaft für Informatik*, (Leipzig, 2010).

[12] Elvesæter, B., Carrez, C., Mohagheghi, P., Berre, A.-J., Johnsen, S. G. and Solberg, A. Model-Driven Service Engineering with SoaML. in Dustdar, S. and Li, F. *Service Engineering*, Springer-Verlag, Vienna, 2011, 25-54.

[13] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S. and Holley, K. SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, *47* (3), 377-396.

[14] Arsanjani, A., Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA, 2004. Retrieved August 12, 2011, from IBM developerWorks: http://www.ibm.com/developerworks/library/ws-soa-design1/.

[15] The Object Management Group (OMG) Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS). Specification Beta 2, OMG, Needham, MA, 2009.

[16] Berg, U. and Zöbel, D., Visual Steering Assistance for Backing-Up Vehicles with One-axle Trailer. in *Vision in Vehicles 11*, (Dublin, 2006), North-Holland Publishing.

[17] Berg, U. and Zöbel, D., Gestaltung der Mensch-Maschine-Interaktion von Lenkassistenzsystemen zur Unterstützung der Rückwärtsfahrt von Fahrzeugen mit Anhänger. in *Mechatronik 2007 Innovative Produktentwicklung*, (Wiesloch, 2007), VDI, 575-588.