Towards Transactional Self-Adaptation for AUTOSAR on the Example of a Collision Detection System

Christian Berger and Matthias Tichy
Department of Computer Science and Engineering
Chalmers | University of Gothenburg, Sweden
[christian.berger|tichy]@chalmers.se

Abstract: Within the last decade important automotive OEMs have created and released the system architecture standard AUTOSAR, and tools to support the development process are widely available. However, the resulting system architecture, which is logically modeled in the Virtual Functional Bus (VFB) and realized by generating a Runtime Environment (RTE), which corresponds to the concrete network of ECUs, is static in terms of runtime adaptability. As long as vehicle functions dominate which are executed exclusively on separated ECUs, there is only a limited demand for runtime adaptation. However, as soon as several vehicle functions are grouped to run on one ECU, which is enabled by AUTOSAR, or one vehicle function is composed of several independent software components, runtime adaptation is getting increasingly interesting. Potential use cases are for example energy-level based function adaptation or vehicle-to-X communication related function adaptation. In this paper, a concept of self-adaptation for AUTOSAR is outlined on the example of a collision detection and warning system, for which the timing correctness during the self-adaptation process is verified with timed automata and the model checking tool UPPAAL.

1 Introduction and Motivation

Modern vehicle functions are developed more and more according to AUTOSAR [FMB+09] to benefit from the standardization effort from the last decade. The reason is that all involved partners may profit at different stages during the development: AUTOSAR enables an OEM to define and simulate a logical communication architecture between different software-intense vehicle functions without considering the final ECU network at the early stages; next, a vehicle software function supplier can develop more independently its software components (SwC) according to consistent and standardized application programming interfaces (API); and finally, an ECU supplier can design and optimize its ECU operating system and the hardware abstraction layer which is realized in the basic software (BSW) as described in the AUTOSAR standard. Thus, on the one hand, all participants can develop with a greater flexibility and independence, and on the other hand, all resulting parts can be integrated more easily because their interaction can be simulated at first to get more information about runtime performance for example.

However, AUTOSAR's concept, the static architecture at design time and its manifestation at runtime through RTE generation, prevents the realization of vehicle functions which are

meant to be structurally adaptive to parameters which are available only at runtime. For example, self-adaptation [C+09] by structural reconfiguration [OMT98] at runtime would enable to start components only when necessary. A potential use case for example is to modify the complexity of the running vehicle functions to react on the available remaining vehicle's energy. Another example would be the modification of vehicle functions by data, which is available from vehicle-to-X (V2X) communication channels: E.g., an intersection assistant could base in its standard variant on the vehicle's surroundings' sensors only but if the vehicle is approaching an intersection that is equipped with a V2X system, the vehicle could enhance its internal intersection's representation by using data from its environment to get a more reliable world model.

Within AUTOSAR, the logical system architecture is modeled within the VFB. Thus, it is static in terms of modifications during the runtime of the involved components. At first, this property leads to a greater confidence that the resulting architecture complies with the confirmed requirements specification and the predefined task scheduling. But this property may prevent adaptations in the aforementioned examples when parameters have to be considered that are available only at runtime. In the following, selected reasons are outlined, which make runtime adaptation difficult for vehicle functions that are implemented with AUTOSAR.

RTE Generation. AUTOSAR follows a component-based software architecture and a layered architectural style to enable a high decoupling between the different components of the automotive software. This approach yields the advantages that components can be easily deployed to different ECUs or can be easily connected by third-parties to build new systems for example. The AUTOSAR standard enables this by combining the component-based approach with a code generation step where all decisions about, e.g., component deployment and component connections, are hard-wired into the generated RTE code. This enables to reap the benefits from a component-based software development approach and still results in a system which is optimized with respect to performance, memory footprint, etc. Unfortunately, the resulting component system is static, i.e., it cannot be structurally adapted as is typically done in self-adaptive systems.

Basic Software. The BSW contains application independent software which may be reused in different applications. Examples are the operating system, the microcontroller abstraction layer, and the watchdog driver. Especially, the watchdog places heavy constraints on implementing self-adaptation. The watchdog is awaiting periodic keep alive signals from the software components. Depending on the configuration, when one or more signals are missing, the watchdog will perform a reset or a functional degradation. In the case of a reconfiguration that takes several periods the watchdog might misinterpret the omission of keep alive signals as a failure. Consequently, the watchdog either has to know about the impeding reconfiguration and best and worst case execution times, or keep alive signals have to be sent even during the reconfiguration.

Verification and Validation. Verification and validation is of utmost importance in cars as in all safety-critical systems; especially during the development and testing for safety-critical vehicle functions in the context of ISO-26262 [SHGB11]. Reconfiguration adds another degree of freedom to the resulting system as the system may behave differently with respect to the different configurations. Additionally, the reconfiguration itself may

contain failures. Consequently, the standard verification and validation activities have to account for the different configurations and the reconfiguration itself.

In our contribution, we focus only on the last two aspects whereas we will address the first aspect in future works. Thus, we present reconfiguration principles for one single single-threaded SwC with respect to supervision of the BSW's watchdog. Furthermore, we address transactional behavior in terms of a failsafe rollback when the reconfiguration unexpectedly fails. We model our reconfiguration behavior using timed automata [BY03] and formally verify it by using UPPAAL [BLL+95]. As the running example, we utilize a collision detection and warning system (CDW) inspired by [BCG+09] which is depicted in Fig. 1. The system consists of three data processing stages S_1, S_2, and S_3 which are composed into two adaptable systems $S_A = S_1 \bigoplus S_2$ and $S_B = S_1 \bigoplus S_2 \bigoplus S_3$. S_A system continuously localizes the own vehicle within a given annotated map while detecting unclassified obstacles within the current driving trajectory by using a LIDAR scanner. The annotated map provides information about the current vehicle's surroundings like rural or highway environments or positions of crosswalks.

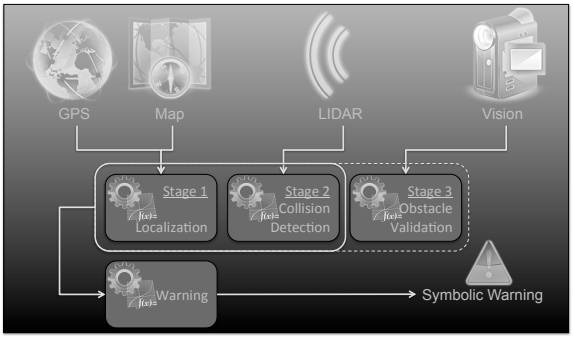

Figure 1: Context-adaptable collision detection and warning system.[1]

The system S_B relies additionally on a vision system to validate the data from the LIDAR system. Therefore, the detected obstacles are used by the vision system to narrow the region of interest for the computationally intense image processing algorithms. However, this classification depends on information from the annotated map to use the vision system only while not driving on highways. Finally, a symbolic warning is used to alert the driver. Let's assume that according to the VFB both vehicle functions S_A and S_B are deployed to one single ECU which should execute only one system configuration at the same time due to energy-efficiency reasons for example. Therefore, S_A must be transformed into S_B depending on the information from the GPS.

The remaining paper is structured as follows: First, we discuss approaches which are related to our concept. Next, we present our approach for a hierarchical, transactional self-adaptation in the context of AUTOSAR. We propose a possible architecture and discuss the necessary behavior during the reconfiguration stage. Afterwards, we apply that concept

[1] Thanks to KDE-Look for the contribution of symbols.

to our running example and show in a sequence chart the information and control flow during a successful and an unexpectedly failing reconfiguration, respectively. Finally, we prove the timing correctness of our concept and ruled out that there were not deadlocks.

2 Related Work

As mentioned in the introduction, a once deployed VFB is not reconfigurable anymore. Thus, reconfiguration is actually not supported by AUTOSAR and connections between SwCs which are generated within an RTE cannot be changed at runtime. Becker et al. propose to model as many different models as necessary to cover all required reconfigurations and subsequently merge them into a single deployable configuration [BGN+09]. However, this proposal creates many very similar models which differ only slightly.

Haber et al. present an approach [HRR+11] which addresses this problem by focusing on the similarity of the required reconfigurations and modeling only their differences to manage them more easily. Nevertheless, all possible reconfigurations are loaded into memory and, thus, are available during runtime which causes an overhead in the required resources like memory and energy. Instead of having all reconfigurations statically available at any time, it is more efficient to have only those components loaded into memory which are required at runtime and modify the runtime model when necessary.

First steps towards a self-adaptation during runtime in the context of AUTOSAR are described by [THP+07]. The authors propose a so-called organic middleware which is similar to AUTOSAR and realizes self-configuration and self-healing features. In contrast to our approach, Trumler et al. focus on application independent self-adaptive behavior. However, we focus on reconfiguration of the application, e.g., in response to changed contexts. Similarly, Zeller et al. also address the application independent self-adaptation [ZPW+11]. They focus on the formal specification of constraints which the adapted system has to satisfy with respect to the correct deployment of software components to ECUs.

Weiss et al. address in [WZEK09] self-organization for infotainment systems in the car. They note similar restrictions in AUTOSAR and current automotive software systems as we did in the introduction. However, their proposed system targets the infotainment domain. Thus in contrast to our approach, they do not have to meet hard real-time requirements nor are concerned with safety.

3 Hierarchical, Transactional Self-Adaptation

We outline in this section our approach for the hierarchical and transactional self-adaptation for AUTOSAR systems. We first describe the architectural structure. Then, we present how the reconfiguration is executed.

3.1 Architectural Design

Several different architectural patterns for self-adaptive systems have been proposed in the past, e.g., MAPE-K [KC03] for business information systems, the Operator-Controller-Module (OCM) for mechatronic systems [HOG04], or the Three-Layer-Architecture [KM07] as an application of the three layer architecture from robotics to self-adaptive software systems. All these architectural approaches have in common that they follow the separation of concerns design principle by separating the application behavior from the adaptation behavior. Additionally, they all draw from the field of control engineering by incorporating a feedback control loop into their adaptation behavior, i.e., they measure the state of the application behavior, plan appropriate adaptation actions, and finally execute these adaptation actions on the application behavior.

Automotive software systems have specific non-functional requirements, e.g., hard real time and safety. The OCM has been particularly designed for these systems. It decouples the hard real time application behavior from the soft real time adaptation behavior as well as the safety-critical behavior from the non safety-critical behavior.

We propose to employ a similar kind of architecture for the structural reconfiguration of AUTOSAR systems based on the architecture outlined in [HPB12] for mechatronic systems. Particularly, we propose to separate the reconfiguration behavior from the application behavior and to employ a hierarchical approach for the reconfiguration because the application is typically hierarchically structured. A non-hierarchical approach to reconfiguration would break the capsulation of components [HPB12, THHO08].

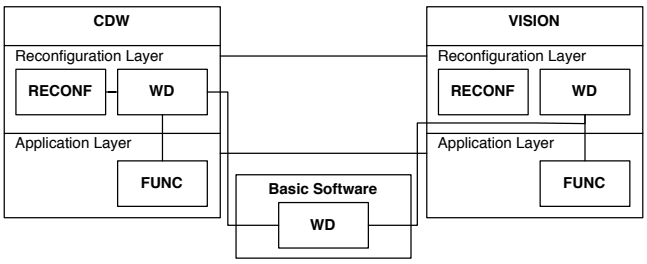

Figure 2: Architectural design for our running example.

Fig. 2 shows an overview of our proposed architecture with respect to our running example. Each component (CDW, VISION) contains two layers – one for the application itself and one for the reconfiguration behavior. Communication with respect to reconfiguration is restricted to the reconfiguration layers. Each component contains a reconfiguration component which is concerned with the reconfiguration of each local component. We restrict our example in Fig. 2 to a single hierarchy level but it can be naturally extended to multiple levels.

As mentioned in the introduction, we particularly consider the BSW watchdog as a very important part to achieve a reliable system. Thus, every component contains an internal watchdog proxy (WD) in addition to the reconfiguration component. This internal proxy

sends keep alive signals to the BSW watchdog even during the reconfiguration. It knows about the reconfiguration, the affected components as well as the worst case execution time for the reconfiguration. Therefore, it has all required knowledge to, on the one hand, act as a proxy for sending keep alive signals and, on the other hand, detect when a reconfiguration failed and stop sending keep alive signals.

3.2 Behavioral Design

Architectural reconfiguration has been introduced in [OMT98]. Component addition, component removal, component replacement, and reconnection of existing components have been presented as basic actions during the architectural configurations. These basic actions are rather low-level. Instead, it is beneficial to group several of these basic actions into higher-level actions. In the past, we presented approaches to either use a state-based reconfiguration [BGT05] or a rule-based reconfiguration using Component Story Diagrams [THHO08]. We abstract in the following from the reconfiguration approach but focus on the process of reconfiguration.

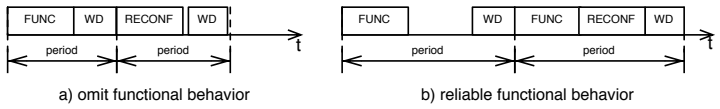

Figure 3: Alternatives for scheduling reconfigurations.

Fig. 3 shows two single-threaded alternatives to schedule the reconfiguration behavior (denoted as RECONF) in addition to functional behavior (FUNC) and the watchdog proxy (WD). On the left hand side, a variant is shown which omits the functional behavior in case of a reconfiguration. This enables to use the complete time reserved for functional behavior for the reconfiguration which results in faster reconfigurations but also loss of functional behavior in that period. On the right hand side, the reconfiguration is interlaced with the regular functionality consuming several periods. It is even possible that a reconfiguration spans more than one slice. For both, best and worst case execution times of all components and behaviors have to be known to validate the scheduling.

4 Realizing Self-Adaptation for an AUTOSAR-based Vehicle Function

In the previous section, theoretical principles for the runtime reconfiguration are outlined. These are now applied to the introductory example of the CDW system. For example due to energy-efficiency reasons, the vision system, necessary data structures, and computationally intense algorithms should be created or activated only when necessary.

4.1 Schematic Sequence Chart

In Fig. 4, the regular information and control flows between the BSW watchdog (BSW_WD), the CDW system which itself consists of three modules (internal watchdog (WD), actual functionality (FUNC), and reconfiguration component (RECONF)), the localization component (LOC), and the vision are depicted; for the sake of clarity the scheduler within the RTE is omitted. Furthermore, the reconfiguration is issued by FUNC for the sake of simplicity– another possibility would be to use a dedicated component within CDW for this purpose. The diagram is divided into different stages. First, the data is processed with LIDAR only by carrying out `p_lidar`: After completing that computation cycle, FUNC sends a *ping* to the CDW's internal WD; afterwards, WD itself sends a *ping* to BSW_WD which validates that CDW is alive and its time slice consumption Δt is within the specification; otherwise, CDW would be deactivated.

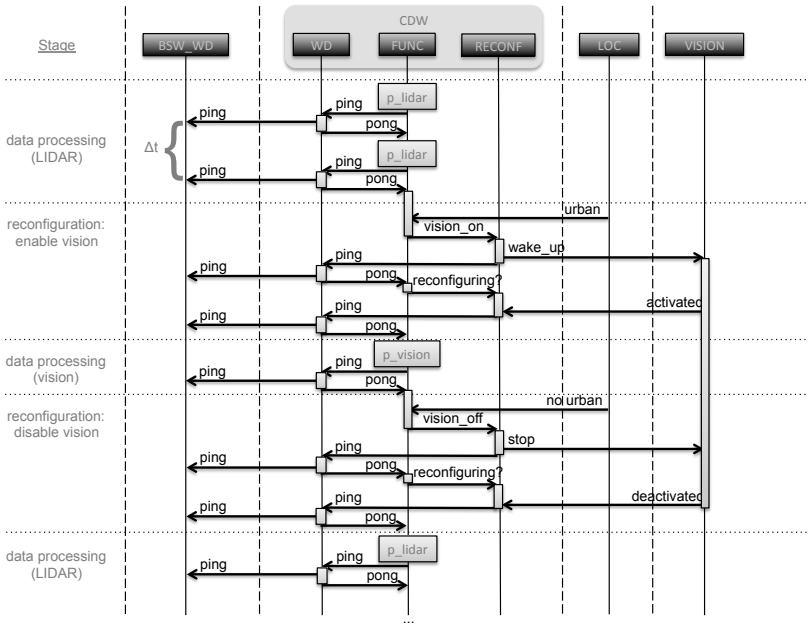

Figure 4: Schematic sequence chart which shows the regular LIDAR-based processing, the successful reconfiguration for using the vision system, and the reconfiguration to disable the vision system again.

The second stage shows exemplarily the reconfiguration after entering an urban environment. Due to CDW's single-threaded architecture with a predefined scheduling according to Fig. 3a), FUNC requests its reconfiguration from RECONF which itself awakes VISION. Let's assume the reconfiguration consumes more than one computation cycle, RECONF's internal state remains *reconfiguring*; however, to fulfill the overall timing specification, RECONF signals its liveliness to BSW_WD using WD. In the following computation cycle, FUNC delegates the control flow back to RECONF due to the single-threaded architecture, and RECONF completes the reconfiguration to provide `p_vision` object structures to

FUNC for computation in the next turn; technically, this could be done by setting a binary flag in FUNC. The deactivation of the vision is carried out in an analogous manner.

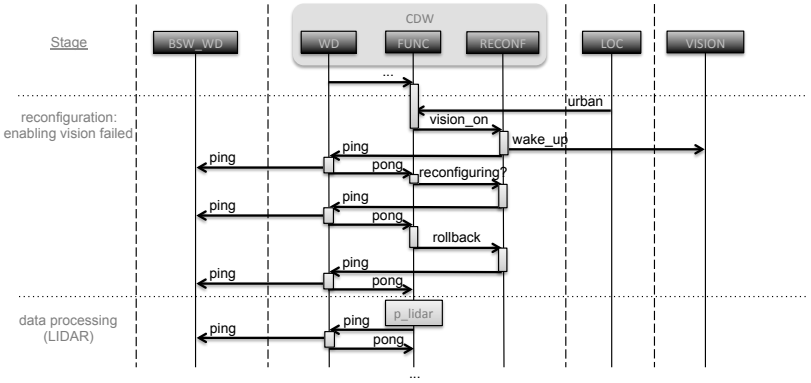

Figure 5: Schematic sequence chart which shows the failing attempt to activate the vision system.

However, to react appropriately on a malfunctioning VISION, and thus to realize a transactional behavior for CDW, Fig. 5 shows the corresponding information and control flows. In contrast to the previous sequence, RECONF does not receive the proper operating state from VISION which itself signals to FUNC. Therefore, FUNC requests a *rollback* from RECONF in the next cycle to fallback to `p_lidar`.

4.2 Verification of Model Properties using UPPAAL

It is important to verify that the timing constraints are fulfilled at anytime to prevent deactivation by BSW_WD. Therefore, we have modeled CDW, a rudimentary vision component, and the BSW watchdog with timed automata and used the UPPAAL tool to formally verify its timing correctness and to rule out deadlocks. For further inspection, we provided the UPPAAL model at `http://goo.gl/iPe62`.

Component	**BCET [ticks]**	**WCET [ticks]**
LIDAR data processing	210	215
vision data processing	230	250
enabling vision	425	440
disabling vision	410	420
rollback	430	445
processing watchdog signals	6	10
permitted scheduling from BSW_WD	200	260

Table 1: Estimation of timings for the CDW's components.

According to Sec. 3.2, we had to specify best case and worst case execution times (BCET, WCET). For our verification we assumed the values as outlined in Tab. 1; additionally,

the last row describes the periodic processing time slice within which the algorithms of the CDW must be successfully computed; otherwise the independently running watchdog BSW_WD deactivates the component due to an unexpected behavior. It can be easily seen that the reconfiguration processes need more than one time slice and thus, they must be divided into two slices as outlined in Sec. 3.2. Based on this timing specification we could successfully verify that the modeled CWD does not cause a deadlock and thus fulfills the defined timing. Furthermore, we showed that the system performed a rollback after detecting a malfunctioning vision to transition to a safe operating state again.

5 Conclusion and Future Work

In this paper, we have outlined concepts to realize self-adaptation within the widely used AUTOSAR system architecture. Therefore, we have discussed principle limitations within the standard and possibilities for circumvention. On the example of a collision detection system, we have shown the general information and control flow with respect to a predefined scheduling; finally, we have modeled the system alongside with a watchdog and vision component by using UPPAAL to verify its timing correctness during the self-adaptation stage and for a failed reconfiguration attempt. As outlined in Sec. 3.2, there are different scheduling policies possible; for our example we decided to use the single-threaded model with a predefined scheduling for timing correctness verification. Future work should both analyze the timing correctness in a multi-threaded environment and interlace the self-adaptation process with the regular execution of the SwC's functionality for an optimal usage of the available time slice. Furthermore, the results should be validated by implementing a proof-of-concept in a demonstration vehicle. Next, a more generic self-adaptation approach could be derived in an abstract manner at the design stage while its realization could be achieved during the generation of the corresponding RTE. Thus, even a distributed and complex self-adaptation in which several ECUs are involved could be realized in a safe way.

References

[BCG+09] Alberto Broggi, Pietro Cerri, Stefano Ghidoni, Paolo Grisleri, and Ho Gi Jung. A New Approach to Urban Pedestrian Detection for Automatic Braking. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):594–605, December 2009.

[BGN+09] Basil Becker, Holger Giese, Stefan Neumann, Martin Schenck, and Arian Treffer. Model-Based Extension of AUTOSAR for Architectural Online Reconfiguration. In Sudipto Ghosh, editor, *MoDELS Workshops*, volume 6002 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2009.

[BGT05] Sven Burmester, Holger Giese, and Matthias Tichy. Model-Driven Development of Reconfigurable Mechatronic Systems with Mechatronic UML. In Uwe Aßmann, Arend Rensink, and Mehmet Aksit, editors, *Model Driven Architecture: Foundations and Applications*, volume 3599 of *Lecture Notes in Computer Science (LNCS)*, pages 47–61. Springer Verlag, August 2005.

[BLL+95] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL - a Tool Suite for Automatic Verification of Real-Time Systems. In *Proceedings of*

Workshop on Verification and Control of Hybrid Systems III, pages 232–243, New Brunswick, NJ, USA, October 1995.

[BY03] Johan Bengtsson and Wang Yi. Timed Automata: Semantics, Algorithms and Tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003.

[C+09] Betty H. C. Cheng et al. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2009.

[FMB+09] Simon Fürst, Jürgen Mössinger, Stefan Bunzel, Thomas Weber, Frank Kirschke-Biller, Peter Heitkämper, Gerulf Kinkelin, Kenji Nishikawa, and Klaus Lange. AUTOSAR - A Worldwide Standard is on the Road. In *Proc. of the 14th International VDI Congress Electronic Systems for Vehicles 2009, Baden-Baden*, 2009.

[HOG04] Thorsten Hestermeyer, Oliver Oberschelp, and Holger Giese. Structured Information Processing for Self-Optimizing Mechatronic Systems. In Helder Araújo, Alves Vieira, José Braz, Bruno Encarnação, and Marina Carvalho, editors, *ICINCO (3)*, pages 230–237. INSTICC Press, 2004.

[HPB12] Christian Heinzemann, Claudia Priesterjahn, and Steffen Becker. Towards Modeling Reconfiguration in Hierarchical Component Architectures. In *15th ACM SigSoft International Symposium on Component-Based Software Engineering (CBSE 2012)*, 2012.

[HRR+11] Arne Haber, Holger Rendel, Bernhard Rumpe, Ina Schaefer, and Frank van der Linden. Hierarchical Variability Modeling for Software Architectures. In *Proceedings of the Software Product Line Conference 2011*, pages 150–159, Munich, Germany, 2011. IEEE.

[KC03] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41 – 50, jan 2003.

[KM07] Jeff Kramer and Jeff Magee. Self-Managed Systems: an Architectural Challenge. In *FOSE '07: 2007 Future of Software Engineering*, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.

[OMT98] Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor. Architecture-Based Runtime Software Evolution. In *ICSE*, pages 177–186, 1998.

[SHGB11] Sebastian Siegl, Kai-Steffen Hielscher, Reinhard German, and Christian Berger. Formal Specification and Systematic Model-Driven Testing of Embedded Automotive Systems. In *Proceedings of the Conference on Design, Automation, and Test in Europe*, pages 1530–1591, Grenoble, France, 2011. European Design and Automation Association.

[THHO08] Matthias Tichy, Stefan Henkler, Jörg Holtmann, and Simon Oberthür. Component Story Diagrams: A Transformation Language for Component Structures in Mechatronic Systems. In *Postproc. of the 4th Workshop on Object-oriented Modeling of Embedded Real-Time Systems (OMER 4)*. HNI Verlagsschriftenreihe, 2008.

[THP+07] Wolfgang Trumler, Markus Helbig, Andreas Pietzowski, Benjamin Satzger, and Theo Ungerer. Self-configuration and Self-healing in AUTOSAR. In *Proceedings of the 14th Asia Pacific Automotive Engineering Conference*, Hollywood, CA, USA, August 2007.

[WZEK09] Gereon Weiss, Marc Zeller, Dirk Eilers, and Rudi Knorr. Towards Self-organization in Automotive Embedded Systems. In Juan Gonzalez Nieto, Wolfgang Reif, Guojun Wang, and Jadwiga Indulska, editors, *Autonomic and Trusted Computing*, volume 5586 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin / Heidelberg, 2009.

[ZPW+11] Marc Zeller, Christian Prehofer, Gereon Weiss, Dirk Eilers, and Rudi Knorr. Towards Self-Adaptation in Real-Time, Networked Systems: Efficient Solving of System Constraints for Automotive Embedded Systems. In *Proceedings of the Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 79–88. IEEE, October 2011.