# Component Based Communication Middleware for AUTOSAR

Dietmar Schreiner[1],
Karl M. Göschka[2] (Faculty Mentor), and Jens Knoop[1] (Faculty Mentor)
[1]Institute of Computer Languages, Compilers and Languages Group
[2]Institute of Information Systems, Distributed Systems Group
Vienna University of Technology
Vienna, Austria
Email: {schreiner,knoop}@complang.tuwien.ac.at
Email: {k.goeschka}@infosys.tuwien.ac.at

**Abstract** — *Due to market demands and technological progress automotive electronic systems have become highly complex, distributed, and heterogeneous systems. In consequence, costs and time-to-market tend to out-grow spendable budgets, whereas quality seriously suffers. To overcome this situation, automotive manufacturers agreed upon a new standard for their electronic systems—AUTOSAR. In AUTOSAR applications are built in conformance to the component paradigm by assembling prefabricated application components that utilize standardized component middleware. Hence, AUTOSAR compliant systems provide increased reusability, maintainability and thus reduced time-to-market and over-all costs. However, the middleware itself is still a coarse grained layered software architecture that is hard to adapt and to optimize w.r.t. application specific needs. Within this paper we provide an overview on our work on how to apply the component paradigm to the component middleware itself, and how to capitalize on this re-designed middleware architecture by automatically synthesizing custom-tailored, light-weight middleware that still complies to the AUTOSAR standard.*

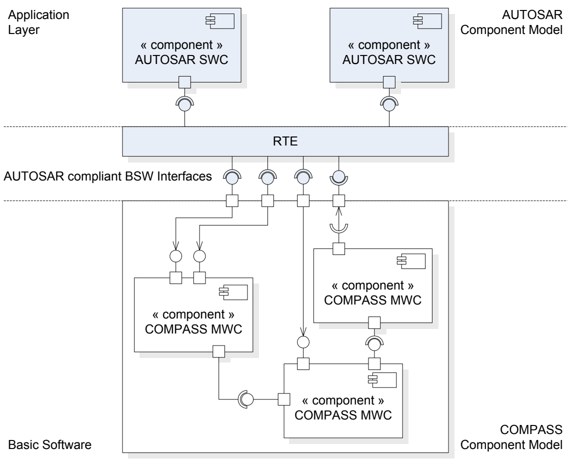
Application Layer
« component » AUTOSAR SWC
« component » AUTOSAR SWC
AUTOSAR Component Model
RTE
AUTOSAR compliant BSW Interfaces
« component » COMPASS MWC
« component » COMPASS MWC
« component » COMPASS MWC
Basic Software
COMPASS Component Model

Figure 1: Component based AUTOSAR middleware

## I. Motivation

Driven by steadily increasing requirements of innovative applications, automotive electronic systems have reached a level of complexity that requires a technological breakthrough in order to manage them cost efficiently and at high quality. In contrast to other domains, like e.g. avionics, automotive electronic systems are produced in comparatively large quantities (69,1 million vehicles in 2006). Therefore, the price per unit has to be as low as possible, forcing manufacturers to assemble economical, and thus resource constrained electronic building blocks. In consequence, automotive software has to cope with harsh restrictions, especially with limitations of available memory and processing power.

The Automotive Open System Architecture (*AUTOSAR*) [1], an upcoming industry standard within the automotive domain, reflects these facts by constituting Component Based Software Engineering (*CBSE*) [2, 3, 4] as development paradigm for automotive applications. CBSE is well accepted within the embedded systems community, as it provides a clear separation of concerns, and hence facilitates extensive software reuse. In *AUTOSAR*, application concerns are covered by software components, while infrastructural ones are handled within layered component middleware. This design leads to an increase in application quality, reusability, and maintainability, and consequently to a reduction of costs and time-to-market. However, the *AUTOSAR* component middleware is specified as layered software that is only customizable on a coarse-grained level.

In consequence, conventional *AUTOSAR* middleware tends to be heavy-weight, which is problematic within resource constrained automotive embedded systems. Therefore, our work contributes by applying the component paradigm to *AUTOSAR* beyond the application layer—to the component middleware. *AUTOSAR*'s conventional, layered middleware architecture is replaced by a component based design that completely resembles the standard's functionality, but is more flexible in terms of application specific customization.

To capitalize on the benefits of a component based architecture, we designed a model driven development process, that incorporates generation of component based communication middleware. As a result, custom-tailored, resource saving middleware can automatically be synthesized from application models and prefabricated, reusable middleware components.

## II. Component Based Middleware

As adumbrated in Fig. 1, we aim at the construction of component based *AUTOSAR* middleware. This so called *Basic Software* is assembled from standardized and hence replaceable middleware components, but still has to provide the same interfaces as its conventional counterpart, to allow seamless integration into existing environments.

To build component based middleware we developed a new component model (*COMPASS CM*) for the middleware, which can not resort to middleware itself. Our component model specifies a component definition, and a composition- and interaction-standard. Composition is based on procedural- and on data-interfaces (function calls and shared memory); interaction may adhere to the client-server or the sender-receiver paradigm. The type-system of *COMPASS CM* is kept close to that of *AUTOSAR* to ensure full compatibility [5].

## III. Component Identification

After specifying a sound component model for component based middleware, we identified standardized middleware building-blocks as much as pared-down versions for each of them. Therefore we investigated a full-fledged industry implementation of a *Basic Software* stack. First we did a manual decomposition based on expert knowledge. In a second step we developed a context-insensitive static analysis that, guided by few manual source-code annotations, automatically identified a set of valid decompositions containing the manually created one [6, 7].

## IV. Automated Middleware Synthesis

One of the advantages of component architectures is the ability to easily replace single building-blocks by isomorphic ones exposing the same interfaces. Heavyweight components may be replaced by pared-down ones, if the application does not require the components' full functionality. Based on this fact, we developed a model driven process that automatically synthesizes *AUTOSAR* middleware from prefabricated middleware building blocks and application models [8, 9]. The core of the defined model driven process is the so called *Connector Transformation*. It transforms platform independent models (PIMs), describing the application's component architecture, into platform specific models (PSMs), containing application components and component equivalent middleware structures. The PSMs moreover render the system's physical structure, so one PSM is created for each system node.

## V. Results

To prove the proposed methodology, and to assess the reduction in size of the synthesized communication middleware, we implemented a simple automotive application. The read-only-memory (ROM) footprint of the application's custom-tailored, synthesized communication middleware was compared to that of the conventional industry implementation to substantiate the claimed improvement. Results showed a reduction of the middleware's ROM footprint by nearly 30% for the implemented application, while a worst-case scenario, where the full *AUTOSAR*'s functionality was required within the middleware, led to an identical footprint as that of the conventional one. In addition the middleware's execution time was reduced by 10% due to omitting forwarding calls between prescribed layers.

## Acknowledgments

This work has been partially funded by the FIT-IT [embedded systems initiative of the Austrian Federal Ministry of Transport, Innovation, and Technology] and managed by Eutema and the Austrian Research Agency FFG within project COMPASS under contract 809444, and by the 7th EU R&D Framework Programme within project ALL-TIMES under contract No 215068.

## References

[1] AUTOSAR. *Automotive Open System Architecture*. `http://www.autosar.org/`.

[2] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, January 1998.

[3] Ivica Crnkovic and Magnus Larsson, editors. *Building Reliable Component-Based Software Systems*. Artech House, 2002.

[4] Rob C. van Ommering, Frank van der Linden, Jeff Kramer, and Jeff Magee. The koala component model for consumer electronics software. *IEEE Computer*, 33(3):78–85, 2000.

[5] Dietmar Schreiner and Karl M. Göschka. A component model for the *AUTOSAR* Virtual Function Bus. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference*, volume 2, pages 635–641. IEEE, 2007.

[6] Dietmar Schreiner and Karl M. Göschka. Building component based software connectors for communication middleware in distributed embedded systems. In *Proceedings of the 2007 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications (MESA07)*, ASME/IEEE, 2007. CD-ROM.

[7] Dietmar Schreiner, Markus Schordan, Gergö Barany, and Karl M. Göschka. Source code based component recognition in software stacks for embedded systems. In *Proceedings of the 2008 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA08)*, IEEE. IEEE, 2008. to appear.

[8] Dietmar Schreiner and Karl M. Göschka. Synthesizing communication middleware from explicit connectors in component based distributed architectures. In *Proceedings of the 6th International Symposium on Software Composition (SC 2007)*, LNCS. Springer, 2007. to appear.

[9] Dietmar Schreiner and Karl M. Göschka. Explicit connectors in component based software engineering for distributed embedded systems. In *SOFSEM 2007: Theory and Practice of Computer Science, Proceedings*, volume 4362 of *LNCS*, pages 923–934. LNCS, Springer, Jan 2007.