Software Components for Reliable Automotive Systems

H. Heinecke
BMW Car IT GmbH
Munich, Germany

W. Damm, B. Josko, A. Metzner
OFFIS
Oldenburg, Germany

H. Kopetz
Technical University of Vienna
Wien, Austria

A. Sangiovanni-Vincentelli
Univ. of California Berkeley
Berkeley, CA, USA

M. Di Natale
Scuola Superiore Anna
Pisa, Italy

Abstract

System-level integration requires an overall understanding of the interplay of the sub-systems to enable component-based development with portability, reconfigurability and extensibility, together with guaranteed reliability and performance levels. Integration by simple interfaces and plug-and-play of sub-systems, which is the main objective of AUTOSAR, requires solving essential technical problems. We discuss to what degree the existing AUTOSAR standard can support the development of safety- and time-critical software and what is required to move toward the desirable goal of timing isolation when integrating multiple applications into the same execution platform.

1. Introduction

Today's automotive electronics systems are often developed by car makers (or OEMs) by assembling *components* that completely or in part have been designed and developed by suppliers. The value chain is traditionally targeted at simple, black-box integrated subsystems, where the requirements capture and the specifications issued to the OEMs consist of the message interface with their periods and general performance requirements, often without a detailed definition of the timing and synchronization properties and requirements of the communication protocols.

When considering the increasing complexity of automotive architectures, the increased distribution of active-safety and future safety-critical functions, including by-wire systems, and the interdependency of these functions, the burden on the integrators is rapidly becoming unbearable. The OEMs are generally struggling to understand and control the emerging behavior of the complex distributed functions, resulting from the integration of subsystems.

The source of these problems is clearly the increased complexity but also the difficulty of the OEMs in managing the integration and maintenance process with subsystems that come from different suppliers who use different design methods, different software architecture, different hardware platforms, different (and sometimes proprietary) Real-Time Operating Systems and middleware layers. Furthermore, there is limited understanding of how to control the para-functional behavior of interacting modules, including the timing and reliability properties emerging from the composition.

Therefore, there is a need for standards in the software and hardware domains that may allow plug-and-play of subsystems. The ability to integrate subsystems will then become a commodity item, available to all OEMs. The competitive advantage of an OEM will increasingly reside on novel and compelling functionalities. The possibility of defining components (subsystems) at higher levels of abstraction and with well defined interfaces allows separation of concerns and improves modularity and reusability. Furthermore, the availability of verification tools gives the possibility of a design-time verification of the system properties. Possibly, components could be defined in such a way that their fundamental properties are preserved after composition, or the properties of the aggregate can easily be derived from abstract properties of its components. In this way, systems could be built in such a way that they are correct-by-construction.

The essential technical problem to solve for this vision is the establishment of standards for interoperability among IPs, both software and hardware, and tools. AUTOSAR [5], a world-wide development partnership including almost all players in the automotive domain electronics supply chain, has been created with the purpose of developing an open industry standard for automotive software architectures.

To achieve the technical goals of modularity, scalability, transferability and re-usability of functions, AUTOSAR provides a common software infrastructure based on stan-

978-3-9810801-3-1/DATE08 © 2008 EDAA

dardized interfaces for the different layers.

The AUTOSAR project has been focused on the concepts of location independence, standardization of interfaces and portability of code. While these goals are undoubtedly of extreme importance, their achievement will not necessarily be a sufficient condition for improving the quality of software systems. As for most other embedded system, car electronics are characterized by functional as well as non functional properties, assumptions and constraints. In complex systems, component-based design may provide encapsulation and separation of concerns, and therefore improve reuse, upon condition that information hiding is implemented so that the component model allows the following properties ([13])

- **Composability** ability of guaranteeing that a component property is preserved across integration
- **Compositionality** ability of deducing global properties (of the composed object) from the properties of its components

Clearly, there are technical and business challenges to overcome. In particular, from the technical point of view, while sharing algorithms and functional designs seems feasible at this time, the sharing of *safety-critical* and *hard real-time* software requires substantial improvements in design methods and technology. Several issues need to be solved for function partitioning and subsystem integration, in the presence of real-time and reliability requirements, including [7]:

- **Time predictability.** This issue is related to the capability of predicting the system-level timing behavior (latencies and jitter), resulting from the synchronization between tasks and messages, but also from the interplay that different tasks can have at the RTOS level and the synchronization and queuing policies of the middleware. The timing of end-to-end computations depends, in general, on the deployment of the tasks and messages on the target architecture and on the resource management policies.
- **Dependability.** The deployment of the functions onto the system ECUs and the communication and synchronization policies must be selected to meet dependability targets. A system-level methodology should integrate support for design patterns suited to the development of highly-reliable components with fault containment both at the functional level and at the timing level. Time triggered architectures can provide timing isolation, but require careful planning and tool support to optimize resource availability against future changes.
- **Composability and Extensibility vs. Efficiency.** The timing of software tasks depends on the presence or absence of other tasks (a similar reasoning applies to messages). A scheduling policy that could prevent timing variability in presence of dynamical changing task characteristics can be conceived (for example, timing isolation or resource reservation policies) but it will carry overhead, albeit potentially not prohibitive. The previous situation is the standard trade-off between efficiency and reliability but it has more important business implications than usual. In fact, if software from different sources has to be integrated on a common hardware platform, in absence of composition rules and formal verification of the properties of composed systems, who will be responsible for the correct functioning of the final product?

In the future scenario, in which application tasks from multiple Tier-1 suppliers are integrated into the same ECU, leveraging the standardization of interfaces allowed by AUTOSAR, protecting the tasks of each IP from the functional and timing errors of other IPs is of fundamental importance. Timing isolation is therefore required to provide for additional separation of concerns and protection.

The following sections will provide additional details on the major challenges that the development of component-based methodologies and standards must face today. Section 2 provides a reasoned overview of the fundamental concepts and the main challenges in the development of the AUTOSAR standard. Section 3 highlights the main principles of a component-based methodology that not only allows the static checking of interfaces, but also verification of behavioral properties and timing constraints. The approach is an extension of the AUTOSAR component model and includes a contract-based approach to interface specification that allows the use of timing analysis tools for the verification of end-to-end latencies. Finally, Section 4 defines the needs of a component-based methodology for the realization of integrated architectures. Focus is on the concepts of encapsulation, fault isolation and error containment, both at the functional and temporal level.

2. AUTOSAR: facts and fancies on the impact in system design

AUTOSAR is a global development partnership which achieved a significant drive for the industry-wide handling of functional complexity [5]. The core objectives of the partnership are concentrating as well on a domain independent standardized software architecture as on a standardized methodology in the software system design enabling a solid development basis with large design bandwidth.

Since the foundation of the partnership in 2003 the maturity of the resulting specifications has been significantly increased from release to release. The recent release 3.0 contains more than 130 specifications which are now utilized for industrial exploitation in the automotive industry.

There is a clear commitment in the automotive industry to migrate now from individual proprietary solutions to the standardized AUTOSAR Architecture and methodology. Such a move and joint undertake is unique in the automotive industry because it is triggered by some key targets. The most prominent ones are: Hardware abstraction, domain wide standard interfaces, reusability of software implementations, transferability of functions and facilitated multi user development and exploitation as well as a support in system design by defining and utilizing the relevant functional and system data for the configuration process. Today, the AUTOSAR set of specifications deliver for the first time a platform for a OEM-wide realization.

In this standardization activities a consolidation of existing automotive design solutions were integrated together with several innovative concepts, see Figure 1 [1]. The new concepts include:

Extended configuration concept The support of multiple configuration variants enables pre-compile, link-time and post-build parameters configuration classes or a combination thereof.

Error handling A consistent and non ambiguous error handling supports effective communication to application layer functionality and can also be used as a means for mode management and diagnostic purposes. Use cases include broken sensors, communication errors and memory failures.

Runtime Environment (RTE) From an abstract point of view (i.e. at system design time) the RTE is the run-time implementation of the Virtual Functional Bus (VFB) on a specific ECU. The notion of the VFB translates to all communication mechanisms that are relevant to the application layer software. Conclusively, the RTE abstracts the application layer from any implementation details of the basic software and hardware aspects. With this in mind, the RTE is a middleware layer technology that enables transferability of application layer software components across the network.

Functional interfaces In order to facilitate integration of application software components in a system, clear semantics of the interface are being published in function catalogues. Descriptions of the interface that conform to the standardized AUTOSAR format are used as an input to the development process without the need to disclose any internal design, which is often sensitive to competition.

Methodology The AUTOSAR meta model precisely defines the concepts used to describe a self-contained system with AUTOSAR. A direct derivation of the meta model are the exchange formats (based on 'templates'), which are thus inherently consistent. Now, the AUTOSAR methodology covers all major steps of system development, starting with the input descriptions for

- Software components
- ECU resources
- System constraints

All subsequent development steps up to the generation of executable code are supported by the AUTOSAR methodology by defining exchange formats and work methods for these steps.

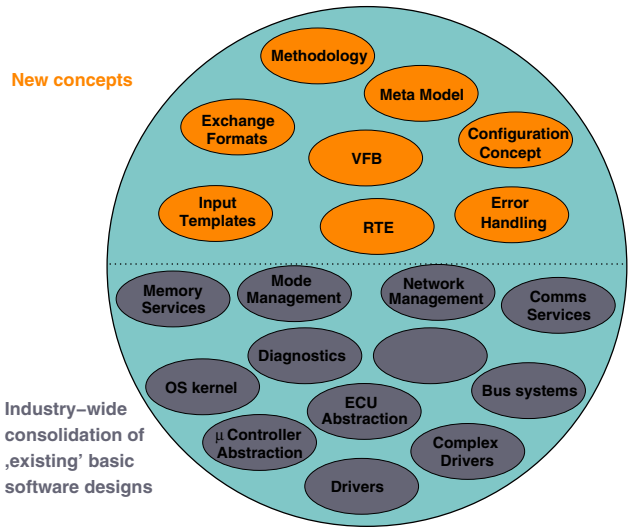

Figure 1. AUTOSAR concepts.

Whilst the AUTOSAR methodology is not a complete process description itself, it can be applied as a foundation for multipartner development processes. In analogy to the benefits of technical interface compatibility, the AUTOSAR methodology has the potential to enable synergies and improved effectiveness when applied properly in a collaborative business context.

It is important to point out here some major limitations:

1. The enabling role of AUTOSAR with respect to reusability of software modules is frequently misunderstood. The AUTOSAR methodology and infrastructure is not intended to specify, design or generate the reusability of Software components; it is organized to handle in a standardized design process finalized software components (SWC) which are designed to be

reused at clear boundary conditions. This means the fraction of the functional network represented in such a SWC must be integrable in the different networks of SWC. This holds true as well for the logical as well as for the dynamic signal networking. This limitation is an intrinsic one.

2. In order to allow for an improved system design methodology where the mapping of SWC (and therefore the mapping of fractional function networks) to computer nodes is supported, the handling of timing and scheduling requirements is mandatory besides the already described items in the AUTOSAR templates. Consequently the extension of the AUTOSAR metamodel and the templates is a must for the implementation of system generators enabling the possibility for prior to implementation system configuration checks [2].

3. Limited support of Release 3.0 in the infotainment and telematics domain so far.

Being aware of these limitations and pushing forward the timing handling in AUTOSAR will certainly push AUTOSAR in a leading software platform in the automotive industry.

3. Building blocks of a design methodology for distributed real-time automotive applications

The Autosar Design Methodology offers key benefits, but also poses major challenges:

- Separating application specification and development from implementation on specific hardware platforms is a key enabler for enhanced cross platform re-use of applications, allowing cost-tuned hardware implementations for the specific set of functions and features integrated in a particular vehicle development.

- This same separation, however, inherently limits the capabilities for system-level analysis, which strives for an early assessment of key functional and extra-functional requirements in order to avoid deep design iterations and assess risks and realizability constraints for new functions.

A design methodology for distributed real-time automotive applications should reconcile the advantage of early system-level analysis with the overall Autosar objective of decoupling function design from its implementation. As key enabler towards this objective, we propose to conservatively extend the Autosar component model towards what we call rich component interface specification . Here, "richness" refers to three dimensions:

- the capability to express the multitude of non-functional constraints, which are central to embedded systems development (including resource constraints, dependability, end-to-end latencies, costs, weight, volume)

- sufficient expressiveness of interface specification language, enabling interface compatibility analysis beyond pure static checking, and cross-viewpoint system analysis

- Contract-based interface specifications, allowing in particular the use of so-called vertical assumptions for capturing resource requirements at system-level.

Jointly, these allow system-level analysis to be performed up to a degree of confidence characterized by the collection of vertical assumptions of system-level design units, such as those decorating so-called "runnables" and virtual communication channels. In particular, assumptions can be annotated with confidence levels, reflecting design experience on the ability to meet e.g. expected resource constraints. Such vertical assumptions can also be used to guide the search for cost-efficient hardware structures supporting the joint resource constraints, as well as exploring allocation decisions with respect to their impact on extra-functional requirements.

When committing to a given system configuration mapping runnables on ECUs and logical communications to inter- or, respectively, intra ECU communication, we can propagate bottom-up extra-functional characteristics of ECUs and bus-systems to assess compliance to the vertical assumptions of system-level design units.

The Speeds project [4] has developed a layered metamodel of heterogeneous rich components and standardized approaches for the integration of commercial industry standard modeling tools to assemble system-level design models with rich interface specifications by combining models expressed in any authoring tool compliant to the integration standard, including Matlab-Simulink/Stateflow [8], Rhapsody [10], and Scade [11]. It is currently integrating a range of analysis methods supporting interface compliance testing and dominance analysis between contracts expressed in extended automata model, subsuming timed automata.

A key component of the proposed methodology is the availability of real-time analysis methods, which allow to assess realizability of end-to-end latencies at system level, exploring the design space of possible system configurations meeting vertical resource assumptions, and assessing compliance to such vertical assumptions based on distributed real-time schedulability analysis for FlexRay- and CAN bus-based target architectures [6][9].

4. From a Federated to an Integrated Automotive Architecture

Substantial economic and dependability benefits could be realized, if the different existing distributed application subsystems (DAS) within a car (e.g., the power-train control subsystem, the braking and chassis control subsystem, the comfort electronic subsystem, the multi-media system etc.,) were integrated into a unified automotive architecture, with a consequent reduction in the number of Electronic Control Units, physical wires and physical contact points. In such an architecture, DASes of different criticality that are developed by different organizations and may use different platform services would have to reside in a single ECU and would have to share a single physical communication channel.

In our opinion the key obstacles that hinder the move to such an integrated architecture are the limited encapsulation, the weak fault-isolation and the poor error containment capabilities of the prevailing communication protocols and of existing ECU hardware/software architectures that partition the services of a single CPU to a set of nearly independent DASes.

Recent technology advances in the area of communication protocols and SoC (System-on-a-Chip) architectures support the shift from a federated to an integrated automotive architecture. In the area of communication, the recent move of the industry to time-triggered protocols, such as FlexRay [6], TTP [12] or Time-triggered Ethernet, supports the partitioning of a single physical communication channel into nearly independent sub-channels that are free of logical or temporal interference and thus provide the encapsulation and error-containment services that are required in an integrated distributed architecture.

In the area of processor architectures, the advent of Multiprocessor MPSoCs that link a number of independent IP Cores on a single chip by a proper Network on Chips (NoC) provides an execution environment, where each component of a DAS can be hosted on its own IP-Core, containing a processor and local memory, such that fault-isolation, and error containment, both in the logical and temporal domain, are achieved by design. Since the IP-Cores within and among DASes communicate solely by the exchange of messages, either via the NoC or the off-chip time-triggered network, the NoC design must support the four composability requirements:

1. precise interface specification: the interfaces between the IP-Core and the NoC must be precisely specified in the temporal and logical domain,
2. stability of priori services: the integration of an IP-core into the SoC must not invalidate the established correctness of the prior services of the IP-Core,
3. non-interfering interactions: There may be no temporal interference among the messages exchanged by the NoC, and
4. error containment: An IP Core that becomes faulty (either because a hardware or software fault) may not interfere with the interactions of the other (non-faulty) IP Cores.

On top of these core services of such a platform SoC architecture, higher-level application specific services can be implemented in middleware such that the APIs (application program interface) that are visible to the application software conform with the requirements of existing legacy applications (e.g., a CAN overlay network) and support the seamless integration of this existing legacy software into the new integrated architecture.

5. Conclusions

The automotive electronic industry is undergoing a deep change in its supply chain, which requires support for the modeling, analysis, development and integration of components. Issues on component design, composability requirements, safety and timing protection and a quick look at the opportunities and the limitations of the existing AUTOSAR standard are discussed.

References

[1] H. Heinecke et al, Current results and preparations for exploitation, Euroforum May 2006 Stutttgart; see www.autosar.org

[2] M.Rudorfer et al. Being on time using AUTOSAR methodology, Elektronik automotive S2(2006)

[3] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, E. Böde, Boosting Re-use of Embedded Automotive Applications Through Rich Components, Elsevier's Electronic Notes in Theoretical Computer Science, Elsevier Science B.V., 2005

[4] Speeds: Speculative and Explorative Design in Systems Engineering, IST project 033471, www.speeds.eu.com

[5] AUTOSAR Consortium web page, www.autosar.org.

[6] Flexray, Protocol Specification V2.1 Rev. A, available at http://www.flexray.com, 2006.

[7] A. Sangiovanni Vincentelli, M. Di Natale, Embedded System Design for Automotive Applications IEEE Computer, Vol 40 (10), Oct. 2007, pp 42-51

[8] Mathworks The Mathworks Simulink and StateFlow User's Manuals, web page: http://www.mathworks.com.

[9] R. Bosch GmBh, CAN Specification, Version 2.0, Stuttgart, 1991.

[10] Telelogic Rhapsody product web page, http://modeling.telelogic.com/modeling/products/rhapsody

[11] Esterel Scade product web page, http://www.esterel-technologies.com/products/scade-suite/

[12] Hermann Kopetz, Günter Grünsteidl, TTP-A Protocol for Fault-Tolerant Real-Time Systems, Computer, v.27 n.1, p.14-23, January 1994

[13] G. Gossler and J. Sifakis, Composition for Component-Based Modeling, Science of Computer Programming, March, 2005, vol. 55, pp. 161–183