Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures

Georg Macher
Institute for Technical Informatics
Graz University of Technology
georg.macher@tugraz.at

Eric Armengaud
AVL List GmbH
eric.armengaud@avl.com

Christian Kreiner
Institute for Technical Informatics
Graz University of Technology
christian.kreiner@tugraz.at

Abstract: Automotive embedded systems have become very complex, are strongly integrated, and the safety-criticality of these systems pose new challenges. Distributed system development, short time-to-market intervals, and automotive safety standards (such as ISO 26262) require efficient and consistent product development along the entire development lifecycle. The de-facto industry standard AUTOSAR aims to standardize an open automotive software architecture and framework to facilitate the exchange of information across company boundaries for the software development process. However, providing consistency of the safety concept during the entire product life cycle is a tedious task. The aim of this paper is to enhance a model-driven system and safety-engineering framework with AUTOSAR aligned software-architecture design. This approach is part of a tool-chain solution enabling the seamless description of safety-critical systems, from requirements at the system level down to software component implementation. To that aim a tool bridge is proposed in order to seamlessly transfer artifacts from system development level to software development level based on AUTOSAR exchange format files.

1 Introduction

Embedded electronic control systems are strong innovation drivers for the automotive domain. The introduction of these systems enables the deployment of more advanced control strategies, such as better driveability and driver assistance systems. With the replacement of well-established mechanical systems by electronic systems, beginning with throttle-by-wire in early 1990s, electronic control systems became the main driver of innovation in the automotive domain. At the same time, the higher degree of integration and the safety-criticality of the control application poses new challenges. The independence of different applications (with different criticality levels) running on the same platform must be made evident. In parallel, new computing architectures with services integrated in hardware enable the development of new software architectures and safety concepts.

Safety standards such as ISO 26262 [ISO11] for electrical and electronical systems for road vehicles have been established to provide guidance during the development of safety-critical systems. They provide a well-defined safety lifecycle based on hazard identification and mitigation, and they define a long list of work-products to be generated.

One important challenge in this context is to provide evidence of consistency during product development among the different work-products.

The contribution of this paper is to bridge the existing gap between model-driven system development tools and software engineering tools. More specifically, the approach relies on the automated generation of software architecture description files based on the existing high level control system description (e.g., based on SysML format). This description includes specific system level information such as ASIL or trace to related (safety) requirement. The approach relies on the AUTOSAR [AUT09] aligned integration of software architectures into the system development tools and the standardized information exchange format of AUTOSAR. The goal is to support a consistent and traceable refinement from the early concept phase to single safety-critical software component implementation.

The document is organized as follows: Section 2 presents an introduction to AUTOSAR. Then, model-based development and integrated tool chains are presented in Section 3. In Section 4 a description of the proposed approach for the generation of software architecture description file according to AUTOSAR standard is provided. An application of the approach is presented in Section 5. Finally, this work is concluded in Section 6 with an overview of the presented approach.

2 AUTOSAR Overview

Several approaches deal with model-based system development and AUTOSAR software development. Different tool vendors provide AUTOSAR tool chains, which support different development stages of AUTOSAR aligned development.

Nevertheless, in terms of safety-critical development, artifact traceability, and support of ISO 26262 safety features there is still room for improvement and ongoing development, e.g. [Eis11, SS12]. Several configurations of AUTOSAR basic software modules and ECU description files still need to be done manually with several non-interacting tools.

The deployment of the AUTOSAR paradigm in projects is usually resource and cost intensive due to (a) the paradigm change to component-based design and related new activities such as interface description in XML format, (b) the complexity of the standard and of the tools with the large number of components and configuration parameters, and (c) the high costs for configuration tools and basic SW layers proposed by the suppliers. In order to support the AUTOSAR deployment, three Implementation Conformance Classes (ICC) have been proposed. These Implementation Conformance Classes have been introduced to smooth the changing process from conventional software development to AUTOSAR software development and are fully in line with the AUTOSAR standard [AUT09].

The first class, ICC1, introduces (a) software component development according to the AUTOSAR standard, a feature supported by almost all common software development tools, and (b) the AUTOSAR runtime environment (RTE). This feature already supports two of the main benefits of AUTOSAR aligned development, which are hardware independent software development, and allocation of dedicated processing cores at a very late development state. Basic software drivers and operating systems features remain unchanged and can be reused as they stand. Solely an interface wrapper for RTE standard interface needs to be adopted. This approach is frequently used when getting started with AUTOSAR, when porting applications to an AUTOSAR environment, or when the project

budget does not support full AUTOSAR tooling. Additionally, this approach is also perfectly suitable for introducing new technologies (e.g. new multi-core technologies) or if special hardware features (e.g. specific safety and security hardware modules) are not yet available in standard AUTOSAR implementation.

On the other hand, AUTOSAR Implementation Conformance Classes 3 (ICC3) implies full AUTOSAR conformity, implementation of the AUTOSAR interfaces, and standards for all structures of the software architecture. Furthermore ICC3 requires a higher level of granularity of the software architecture. This approach requires complete AUTOSAR tooling and vendor support. Nevertheless, ICC3 gives maximum flexibility and highest accuracy of the interface specifications.

ICC2 is an intermediate step between ICC1 and ICC3. Vertical subdivisions are already introduced and monolithic blocks of the basic software are already separated. The clustering on ICC2 level is not restricted by AUTOSAR and could lead to different optimizations. This ensures optimization strategies in terms of reduced execution time and memory footprint. ICC2 clustering can be used anytime when ICC3 structuring leads to non-negligible overhead and flexibility of ICC3 approach is not required.

The main benefit of the ICC1 approach clearly relies on the time-saving in terms of no additional familiarization with usually very complex and time-consuming AUTOSAR tools. Nevertheless, this approach does not take advantage of the AUTOSAR benefits of standardized component interfaces for exchange of components, supporting tools for RTE configuration, and communication interfaces. But in terms of safety-critical software development special attention has to be paid to the manual interface description (such as Excel files), manually coded interfaces, and manual configurations (see the parts with 'manual rework' marking in Figure 1). Manual document changes are difficult to trace and therefore require additional alertness to be paid in terms of safety-critical system development.

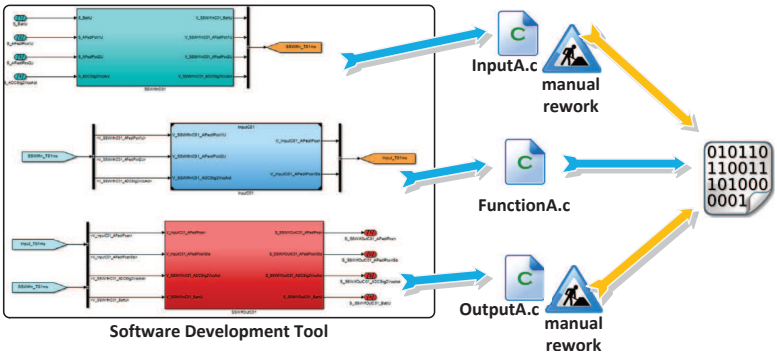

Figure 1: ICC1 AUTOSAR Approach Methodology with Required Manual Intervention

ICC1 mainly focuses on SW engineering and more specifically on the integration of ASW into a given SW architecture. However, the aspects related to control systems engineering (including HW/SW co-design) are not integrated and aspects such as HW/SW interface definition must be performed manually. The proposed approach in this work enhances this aspect and provides a framework for the visualization of interface wrapper configuration

and automated generation of the interfacing C files. Furthermore, the approach enhances the standard AUTOSAR software component description with links to involved development artifacts of prior development processes (e.g. traces to requirements, ASIL level, additional constraints).

3 Model-based Software Development and Integrated Toolchains

Model-based Systems and Software development as well as tool integration are engineering domains and research topics aimed at moving the development steps closer together and thus improving the consistency of the system over the expertise and domain boundaries. Broy et al. [BFH+08] mention concepts and theories for model-based development of embedded software systems. The authors also claim model-based development the best approach to manage the large amount of information and complexity of modern embedded systems with safety constraints. The paper illustrates why seamless solutions have not been achieved so far, they mention commonly used solutions, and arising problems by using an inadequate tool-chain (e.g. redundancy, inconsistency and lack of automation).

Chen et. al. [CJL+08] presents an approach that bridges the gap between model-based systems engineering and the safety process of automotive embedded systems. The systematic approach uses the EAST-ADL2 architecture description language to develop safety cases in close relation to the system model and analysis of malfunctions. Although the work provides a system model for keeping various engineering information across multiple levels of abstraction and concerns consistent, their approach ends just before the definition of software architecture design. More recently the MAENAD Project [1] is focusing on design methodologies for electric vehicles based on EAST-ADL2 language.

The work of Holtmann et al. [HMM11] highlights process and tooling gaps between different modeling aspects of a model-based development process. Often, different specialized models for specific aspects are used at different development stages with varying abstraction levels. Traceability between these different models is commonly established via manual linking. The authors claim that there is a lack of automation for those linking tasks and missing guidance which model should to be used at which specific development stage. The proposed tool-chain mentions two important gaps: First, missing links between system level tools and software development tools. Second, several very specific and non-interacting tools which require manual synchronization, are therefore often inconsistent, rely on redundant information, and due to a lack of automation require redundant manual work.

This issue is also addressed by Giese et al. [GHN10]. System design models have to be correctly transferred to the software engineering model, and later changes must be kept consistent. The authors propose a model synchronization approach consisting of tool adapters between SysML models and software engineering models in AUTOSAR representation. One drawback of this approach stems from the bidirectional model transformation, each transformation step implies potential sources for ambiguous mapping and model mismatching.

An important topic to deal with is the gap between system architecture and software architecture - especially while considering component-based approaches such as UML and

[1] http://maenad.eu/

SysML for system architecture description and AUTOSAR for SW architecture description.

Pagel et al. [PB06] mention the benefit of generating XML schema files directly from a platform-independent model (PIM) for data exchange via different tools. Performing extra transformation steps would only add potential sources for error and ambiguous mappings could result in unwanted side-effects. We also spotted a potential drawback for the previously mentioned approach of Giese et. al. [GHN10].

Boldt [Bol09] proposed the use of a tailored Unified Modeling Language (UML) or System Modeling Language (SysML) profile as the most powerful and extensible way to integrate an AUTOSAR method in company process flows. The author highlights the option of UML to link requirements to ECU, SWC or runnables.

An automotive tool-chain for AUTOSAR is also presented by Voget [Vog14]. The work focuses on ARTOP, a common platform for innovations which provides common base functionality for development of AUTOSAR compliant tools. Unfortunately the Eclipse based ARTOP platform serving only as a common base for AUTOSAR tool development, is not a tool solution, and also requires time-consuming initial training to even get started to develop a desired tool.

Among these, we evaluated several commercial available tools. The following paragraph provides a brief overview of tools supporting the AUTOSAR approach, this list is not intended to be exhaustive. The tools Matlab/Simulink and Embedded Coder are widely used for automotive software development. It is possible to import and export AUTOSAR software component descriptions and generate AUTOSAR software code in an integrated environment with both tools. However, this tool focuses solely on software development. Dassault System AUTOSAR Builder is a software platform for system and ECU (Electronic Control Unit) design, based on the ARTOP tool environment. The tool suite imports model-based design (MBD) descriptions and generates AUTOSAR compliant C code. The embedded software platform Arctic Core includes a real-time operating system, communication services, memory services, and drivers for different microcontroller devices. Continental's CESSAR-CT Tool-Box integrates the AUTOSAR methodology in an already existing process landscapes. The modular tool includes several editors, an AUTOSAR conformance validation, and code generation framework. Elektobit offers a complete product line, called EB tresos. One part of this product line, EB tresos AutoCore, is an AUTOSAR compliant basic software, supporting AUTOSAR compliant OS for single and multicore ECUs. ETAS AUTOSAR Solutions are designed to cooperate with ETAS's development tools, such as ASCET, INCA, and others. Vector's AUTOSAR tool chain consists of PREEvision, DaVinci Developer, and DaVinci Configurator Pro. These tools provide tool support for model-based development, AUTOSAR basic software, and RTE configuration.

Nevertheless, these tools mainly target the AUTOSAR ICC3 approach and usually focus on the configuration and analysis of the AUTOSAR stack and SW architecture. The proposed contribution in this paper is (a) the formalization of control system description (including BSW and HW) especially interesting for ICC1 and (b) the import / export to AUTOSAR XML files for the traceability of architecture artifacts such as ASIL or timing constraints from system description down to SW implementation, which are interesting for all ICCs.

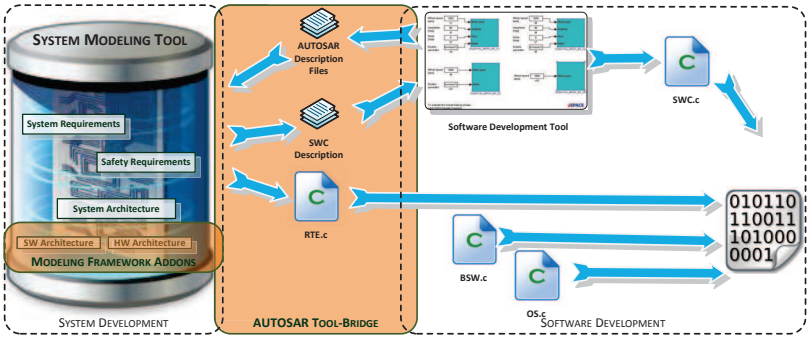

Figure 2: Portrayal of the Bridging Approach Transferring System Development Artifacts to SW Development Phase

4 AUTOSAR Tool-Bridge for Model-based Software Development Approach

The basis of our approach stems from the CESAR Project [RW12] and was further improved by a feasibility study [Mad12] to support continuous safety related system development according to ISO 26262 at concept phase and system development level. For a more detailed overview of the tool-chain as a whole see [MAK14] and [Mad12].

The contribution proposed in this work is an extension of this framework towards software development in the context of AUTOSAR. More specifically, our contribution consists of the following parts:

- *AUTOSAR UML modeling framework*: Enhancement of an UML profile for the definition of AUTOSAR specific artifacts, more precisely, for the definition of the components interfaces (based on the virtual function bus). This is required for consistent SW system description, see Figure 2 – modeling framework addon.
- *BSW and HW module modeling framework*: Enhancement of an UML profile to describe BSW components and HW components. This is required to ensure consistency of the specification and implementation for the entire control system, see Figure 2 – modeling framework addon.
- *SW architecture exporter*: Exporter to generate the resulting SW architecture as AUTOSAR XML files for import in third party tools for further detailed development, see Figure 2 – AUTOSAR tool bridge.
- *AUTOSAR file importer*: Importer to integrate refined SW architecture as AUTOSAR XML file (e.g., as a result of round-trip engineering), see Figure 2 – AUTOSAR tool bridge.

This proposed approach closes the gap, also mentioned by Giese et al. [GHN10], Holtmann et al. [HMM11], and Sandmann and Seibt [SS12], between system-level development at abstract UML-like representations and software-level development modeling tools (e.g.

Matlab Simulink/Targetlink). This bridging supports consistency of information transfer between system engineering tools and software engineering tools. Further, the approach minimizes redundant manual information exchange between these tools and contributes to simplify seamless safety argumentation according to ISO 26262 for the developed system. Benefits of this development approach are highly noticeable in terms of re-engineering cycles, tool changes, and reworking of development artifacts with alternating dependencies, as also mentioned by Broy et al. [BFH+08]. Closing this gap creates a seamless tool-chain from initial design, to software architectures in a model-based development environment and final decisions in code implementation in compliance with ISO 26262. Our approach supports the ICC1 AUTOSAR approach and relies on the AUTOSAR specification [AUT09](currently implementing AUTOSAR $R3.2$ due to compatibility with Matlab 2011 based SW development tools) for architectural approach, definition of application software interfaces, and exchange formats.

4.1 AUTOSAR UML Modeling Framework

The first contribution is the development of a specific UML modeling framework enabling software architecture design in AUTOSAR like representation within the system development tool (Enterprise Architect). This EA profile takes advantage of the AUTOSAR virtual function bus (VFB) abstraction layer and enables an explicit definition of AUTOSAR components, component interfaces, and connections between interfaces. This provides the possibility to define software architecture and ensures proper definition of the communication between the architecture artifacts, including interface specifications (e.g. upper limits, initial values, formulas). In addition to standard VFB AUTOSAR artifacts our profile supports graphical representation of ASIL, assignment to dedicated signals and modules, and supports specification of runnables with respect to timing constraints (such as WCET), ASIL, priority, and required stack sizes. This meta information enables mapping of tasks to a specific core and establishment of a valid scheduling in a later development phase.

The proposed approach thus supports the ISO 26262 requirements of traceability along the development process, even for ICC1 AUTOSAR development. Hence, the AUTOSAR-aligned representation can be linked to system development artifacts and traces to requirements can be easily established. These explicit links can be further used for constraints checking, traceability of development decisions (e.g. for safety case generation), and reuse. Figure 3 shows an example of a safety-relevant software module (AUTOSAR Composition) and its ASIL decomposition in two components with lower ASIL levels, represented in Enterprise Architect. This integrated definition of system artifacts and software module in one tool furthermore supports the work of safety engineers by adding values and visual labels for safety-relevant software modules.

4.2 BSW and HW Module Modeling Framework

Special basic software (BSW) and hardware module representations are assigned to establish links to the underlying basic software and hardware layers. The AUTOSAR architectural approach ensures hardware-independent development of application software modules until a very late development phase and therefore enables application software developers and basic software developers to work in parallel. Nevertheless, safety-critical sys-

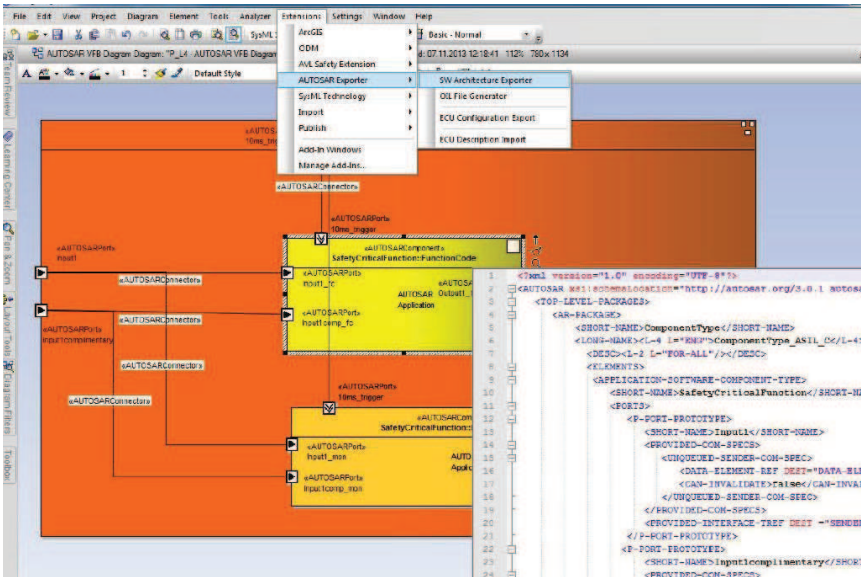

Figure 3: Screenshot of the SW Architecture Representation within the System Development Tool and Extension of Bridging Approach

tem development concerns also hardware development and support for hardware-software co-design is absolutely essential. The hardware profile allows a graphical representation of hardware resources (such as ADC, CAN), calculation engines (core), and connected peripherals which interact with the safety-critical software. This additional information enables the mapping of tasks to a specific core and establishment of a valid scheduling in a later development phase. Furthermore, the profile enables an intuitive graphical way of establishing software and hardware dependencies and a hardware-software interface (HSI), as required by ISO 26262. In combination with the ICC1 AUTOSAR development approach this profile enables the possibility of a traceable automatic RTE configuration generation instead of the typical manual software component interface definition.

4.3 SW Architecture Exporter

The third part of the approach is an exporter which is able to export the software architecture, component containers, and their interconnections designed in SysML to a software development tool (e.g. Matlab/Simulink) via AUTOSAR XML files (see Figure 3). Most of the state of the art software development tools are able to generate AUTOSAR-conform code and software component description files or support the import of AUTOSAR modules. This ensures flexibility of the approach in terms of the preferred software development tool in use (e.g. Matlab/Simulink or ETAS ASCET) and ensures tool-independence of the presented approach.

The exporter generates an AUTOSAR conform software component description files (currently AUTOSAR $R3.2$ to be compatible with Matlab 2011) enriched with system and

safety development artifact traces. Information that is not importable by default AUTOSAR import functions of third-party tools is transferred via description and long-name values of individual models and is therefore still available for the user of this particular tool. Using this exporter, the (safety) context can be efficiently exported and communicated to the software experts in their native development tools, thus improving the consistency of the product development.

4.4 Import of AUTOSAR Files

The fourth part of the approach is the import functionality add-on for the system development tool. This functionality, in combination with the export function, enables bidirectional update of software architecture representation in the system development tool and the software modules under development. The importer also re-imports additional information from the ARXML file stemming from software development level. On the one hand, this provides input for the previously mentioned timing estimations of task, and on the other hand it ensures consistency between system development artifacts and changes done in the software development tool. Finally, the import functionality enables reuse of available AUTOSAR software modules, guarantees consistency of information, and shares information more precisely and less ambiguously.

5 Application of the Proposed Approach

This section demonstrates the benefits of the introduced approach in terms of ISO 26262 aligned development. As a first step of ISO 26262 related safety-critical system development, the boundaries of the system and its interacting environment must be specified. For each system on each level of abstraction, system targets (requirements and use cases) and a system structure can be refined. The system design is completed by the definition of the hardware-software interface (HSI). This mapping provides a basis for concurrent development of software and hardware.

The definition of the software architecture is usually done by a software system architect within the software development tool (such as Matlab/Simulink). With our approach this work package is included in the system development tool in an AUTOSAR-aligned representation (already depicted in Figure 3). On one hand, this does not hamper the work of the software system architect and the AUTOSAR format based exporter ensures consistent transferring to the software development tool, in a way that the software unit design remains unaffected. On the other hand this change offers a significant benefit for development of safety-critical software in terms of traceability and replicability of design decisions.

The presented approach bridges the existing lack of tool support for transferring SW architectures between system design and software implementation tools. Due to the basis of information transfer via standardized AUTOSAR exchange formats this tool-independent approach can also be used to link additional tools to the development tool-chain. Furthermore, the approach guarantees traceable links of safety concept considerations throughout the entire development cycle, due to the single source of information principle (all relevant information and design decisions within the system development tool), as well as improved re-useability of software modules by adding information and safety constraints to

the AUTOSAR software component description file. Figure 4 depicts the add-on of traceable artifact links of the approach for a specific subsystem. Required ASIL and related requirements can be additionally stored within the ARXML file, as well as information about other dependencies (e.g. required HW resources or required core allocation diversity). The figure illustrates the transfer of artifacts between the separated development phases of SW development and system development, and indicates how traceability can be supported.

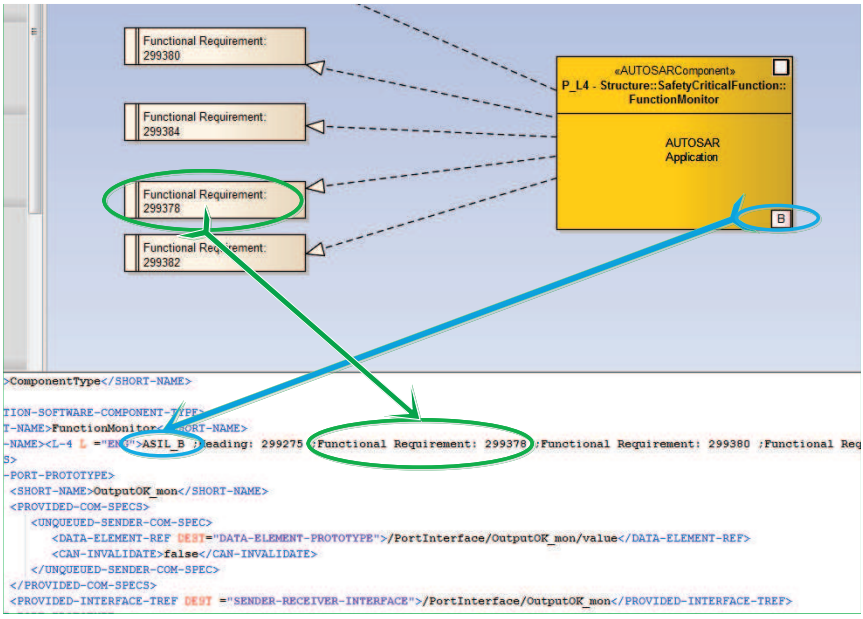

Figure 4: Traceability of Artifacts between System Engineering and SW Development

These AUTOSAR software module description files can be imported by software development tools (such as Matlab/Simulink), thus minimizing the effort of error-prone manual work without adequate tool support. The possibility of re-importation of information from changed AUTOSAR software component description files into the system model ensures round-trip engineering and consistency of the implementation and the system model. The previously mentioned definition of hardware-software interface enables the automatic generation of interface wrapper files (*.c* and *.h* files), thus also reducing the amount of manual changes of source code files without adequate tracing of changes.

To provide a comparison of the improvements of our approach we defined the three layer E-Gas monitoring functionality accordingly [ZS07] as use-case. This elementary use-case is well-known in the automotive domain, but is nevertheless representative in our opinion, due to the fact that several safety-critical systems base on this approach. Table 1 gives an overview of the improvements indicated compared to AUTOSAR ICC1 approach.

In terms of getting started with AUTOSAR aligned development our approach does not rely on full AUTOSAR tooling support, but rather features the smooth first step approach of the ICC1 AUTOSAR approach. In terms of safety-critical development the presented

approach supports round-trip engineering by tool-supported information transfer between separated tools and links to supporting safety-relevant information. Furthermore, the approach eliminates the need of manual interface source code rework and ensures reproducibility and traceability arguments for this task. These indicators infer that the presented approach surmounts the main drawbacks of the ICC1 AUTOSAR approach.

Evaluation Criteria	**ICC1 AUTOSAR Approach (reference)**	**AUTOSAR Tool-Bridge Approach**
11x Interface definitions (consisting of limits, data type, scaling, unit, default value)	usually done with SW engineering tool, no representation within system development tool	graphically with option list support
Interface consistency evaluation	needs to be done manually at review without tool support	automated consistency checks possible for point-to-point connections
SW Architecture definition (3 levels, 7 modules, 30 connections)	usually done with SW engineering tool, no representation within system development tool	graphically establish-able within system development tool, automatic export to ARXML
SW Architecture update	usually no representation within system development tool	import and export functionality within system development tool
ASIL assignment	not supported	for each module and each BSW mapping possible
RTE configuration	manually	mapping automatically generated

Table 1: Approach Improvement Indicators

6 Conclusion

An important challenge for the development of safety-critical automotive systems is to ensure consistency of the safety relevant artifacts (e.g., safety goals, concepts, requirements and mechanisms) over the development cycle. This is especially challenging due to the large number of skills, tools, teams and institutions involved in the development. This work presents an approach to bridge tool gaps between an existing model-driven system and safety engineering framework and software engineering tools, based on domain standard AUTOSAR. The implemented tool extension transfers artifacts from system development tools to software development tools, thereby creating traceable links across tool boundaries, and relying on standardized AUTOSAR exchange files. The main benefits of this enhancement are: improved consistency and traceability from the initial design at the system level down to the single software components, as well as a reduction of cumbersome and error-prone manual rework along the system development path. Further improvements of the approach include the introduction of AUTOSAR without relying on full AUTOSAR tooling support, and progress in terms of reproducibility and traceability of safety-critical arguments for the software development.

Acknowledgments

The authors would like to acknowledge the financial support of the "COMET K2 - Competence Centers for Excellent Technologies Programme" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economy, Family and Youth (BMWFJ), the Austrian Research Promotion Agency (FFG), the Province of Styria, and the Styrian Business Promotion Agency (SFG).

Furthermore, we would like to express our thanks to our supporting project partners, AVL List GmbH, Virtual Vehicle Research Center, and Graz University of Technology.

References

[AUT09] AUTOSAR development cooperation. AUTOSAR AUTomotive Open System ARchitecture, 2009.

[BFH+08] Manfred Broy, Martin Feilkas, Markus Herrmannsdoerfer, Stefano Merenda, and Daniel Ratiu. Seamless Model-based Development: from Isolated Tool to Integrated Model Engineering Environments. *IEEE Magazin*, 2008.

[Bol09] Richard Boldt. Modeling AUTOSAR systems with a UML/SysML profile. Technical report, IBM Software Group, July 2009.

[CJL+08] DeJiu Chen, Rolf Johansson, Henrik Loenn, Yiannis Papadopoulos, Anders Sandberg, Fredrik Toerner, and Martin Toerngren. Modelling Support for Design of Safety-Critical Automotive Embedded Systems. In *SAFECOMP 2008*, pages 72 – 85, 2008.

[Eis11] Ulrich Eisemann. Modellbasierte Entwicklung in einer AUTOSAR-Werkzeugkette. www.elektroniknet.de/automotive/sonstige/artikel/74849/, Jannuary 2011.

[GHN10] Holger Giese, Stephan Hildebrandt, and Stefan Neumann. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *LNCS 5765*, pages pp. 555 –579, 2010.

[HMM11] Joerg Holtmann, Jan Meyer, and Matthias Meyer. A Seamless Model-Based Development Process for Automotive Systems, 2011.

[ISO11] ISO - International Organization for Standardization. ISO 26262 Road vehicles Functional Safety Part 1-10, 2011.

[Mad12] Roland Mader. *Computer-Aided Model-Based Safety Engineering of Automotive Systems*. PhD thesis, Graz University of Technology, 2012.

[MAK14] Georg Macher, Eric Armengaud, and Christian Kreiner. Bridging Automotive Systems, Safety and Software Engineering by a Seamless Tool Chain. In *7th European Congress Embedded Real Time Software and Systems Proceedings*, pages 256 –271, 2014.

[PB06] Mike Pagel and Mark Broerkens. Definition and Generation of Data Exchange Formats in AUTOSAR, process independent model. *lNCS 4066*, pages pp. 52–65, 2006.

[RW12] Ajitha Rajan and Thomas Wahl. *CESAR Project Book*. Springer Verlag, 2012.

[SS12] Guido Sandmann and Michael Seibt. AUTOSAR-Compliant Development Workflows: From Architecture to Implementation - Tool Interoperability for Round-Trip Engineering and Verification & Validation. *SAE World Congress & Exhibition 2012*, (SAE 2012-01-0962), 2012.

[Vog14] Stefan Voget. SAFE RTP: An open source reference tool platform for the safety modeling and analysis. In *Embedded Real Time Software and Systems Conference Proceedings*, 2014.

[ZS07] Thomas Zurawka and Joerg Schaeuffele. Method for checking the safety and reliability of a software-based electronic system, January 2007.