Using Model-based Development for ISO26262 aligned HSI Definition

Georg Macher, Harald Sporer, Eric Armengaud, Eugen Brenner, Christian Kreiner

Using Model-based Development for ISO26262 aligned HSI Definition

Georg Macher*‖,Harald Sporer*, Eric Armengaud‖, Eugen Brenner* and Christian Kreiner*

*Institute for Technical Informatics, Graz University of Technology, AUSTRIA
Email: {georg.macher, sporer, brenner, christian.kreiner}@tugraz.at

‖AVL List GmbH, Graz, AUSTRIA
Email: {georg.macher, eric.armengaud}@avl.com

***Abstract*—The ISO 26262 safety standard for road vehicles, among other engineer standards, was established to provide guidance for the development of safety-critical systems. Providing evidence of consistent, correct, and complete system specification covering different work-products is crucial in this context. One of these required work-products is the hardware-software interface (HSI) definition. The HSI definition is especially important since it defines the interfaces between different engineering domains (such as system development, HW development, and SW development). Model-based development (MBD) is the most promising approach to support consistent description of the system under development in a structured way. This paper thus presents an ISO 26262 aligned hardware-software interface definition tool approach which bidirectionally combines the versatility and intuitiveness of spreadsheet tools (such as Excel) and the properties of MDB tools (e.g. different views, levels of abstraction, central source of information, and information reuse). The approach presented is capable of defining an ISO 26262 compliant HSI and enables automatic derivation of basic software configurations according to these interface definitions.**

***Index Terms*—model-based development, ISO 26262, hardware-software interface.**

I. Introduction

Electronic control units (ECU) have steadily increased in value on the vehicle market over the years. The range of premium cars software implementations were close to 1 gigabyte code and were spread over more than 90 electronic control units (ECU) in 2009. By 2018 the cost of vehicle electronics is forecast to have risen to 30% of the overall vehicle costs [5]. At the same time the higher degree of integration and the safety-criticality of the control application is raising new challenges. Evidence of correctness of the different applications and safety concepts must be guaranteed and the increasing demands for safety and security require additional development efforts. Safety standards such as ISO 26262 [6] for electrical and electronic systems for road vehicles have been established to provide guidance during the development of safety-critical systems. The standards provide a well-defined safety lifecycle based on hazard identification and mitigation, and define a long list of work-products to be generated. One important work-product among the many that are defined here is the hardware-software interface (HSI) definition. The HSI specifies the hardware and software interactions in consistency with the technical safety concept, which includes hardware components that are controlled by software and support the software execution. ISO 26262 states the importance and essentiality of the HSI specification by highlighting its definition during the system design phase and its further refinement during hardware and software development phase [6]. The HSI document is the last development artifact of the system development and the starting point for parallel development of hardware and software. The HSI definition thus requires mutual domain knowledge of hardware and software and is usually the result of a collective workshop of hardware, software, and system experts. The HSI is the linkage between different levels of development and is used to align topics relevant to both hardware and software development. Insufficient definition of the HSI can cause several additional iteration cycles and communication issues between development teams. Model-based development (MBD) supports the description of the system under development in a more structured way, which enables different perspectives for different stakeholders, different levels of abstraction and a central source of information, moreover it also appears to be the best approach for handling these HSI definition issues. Nevertheless, seamless model-based solutions have not been achieved so far, mainly due to inadequate tool-chain support (e.g. redundancy, inconsistency and lack of automation), which hampers the MBD approach in tapping its full potential. This paper thus presents a tool approach for ISO 26262 aligned hardware-software interface definition. The approach presented combines the versatility and intuitiveness of spreadsheet tools (such as Excel) and the properties of MDB tools (e.g. different views, levels of abstraction, central source of information, and information reuse) bidirectionally.

The document is organized as follows: The next section describes related works and the state of the art of hardware/software interface definition. The III section provides a description of the proposed enhancement for HSI definition. Section IV evaluates the presented approach with an automotive use case. Finally, the last section concludes this work with an overview of what has been achieved.

II. State-of-the-Art HSI Definition

Although the topic of defining hardware/software interface definitions is of great importance for the automotive domain, only few recent publications exist. King et. al [7] postulate the problematic of defining HW/SW interfaces in early development steps for of System on Chip (SoC)development. First, a detailed interface is difficult to specify without detailed knowledge of software and hardware. Second, these detailed interfaces prevent a later migration of interface functionality and addition of features.

In the automotive domain hardware and software development cycle times differ significantly in length and software development is typically separated into several abstraction layers (such as application software, microcontroller abstraction layers, basic functionality drivers). This approach conceals hardware specific details and enables the establishment of focused software development teams (e.g., basic software developer, application software engineers, software integrators), but on the other hand it sometimes obfuscates the importance HSI development.

The AUTOSAR architectural approach [1] explicitly forces an approach of this kind to support hardware independent development of application software modules until a very late development phase. The AUTOSAR specifications intention is to support exchange and reuse of software, by defining software architectures, interfaces, and exchange formats and enable parallel working of application software developers, basic software developers, and hardware developers.

A comprehensive overview of hardware/software co-design methods and tools can be found in [9]. Teich's work exploits the synergies of hardware and software with focus on cost and performance constraints and highlights major research directions and achievements.

Contract-based design paradigms are an emerging domain-independent paradigm for interface definition. The contracts specify the input and output behavior of a component and provide a guaranteed behavior [3]. Such an approach can be used for software component safety contracts [8] as well as contract-based embedded system development [4]. These contract-based approaches foster model-based development and traceability of development decisions. Nevertheless, this approach is not simple and easy enough to be used of HSI definition workshops. Because of its simplicity and easy to use nature many hardware/software interface definitions are still done within spreadsheet tools or in textual form within a requirement management tool. Although Chen et. al [2] claim that social and text-based communication does not scale for the handling of future embedded automotive systems and their advanced interface definition constraints.

III. The HSI Definition Approach

The HSI definition tool presented in this work allows the definition of ISO 26262 aligned hardware-software interfaces in a practicable and intuitive way in a spreadsheet tool (such as Excel). The tool presented enables the transformation of these HSI definitions in a reusable and version-able model

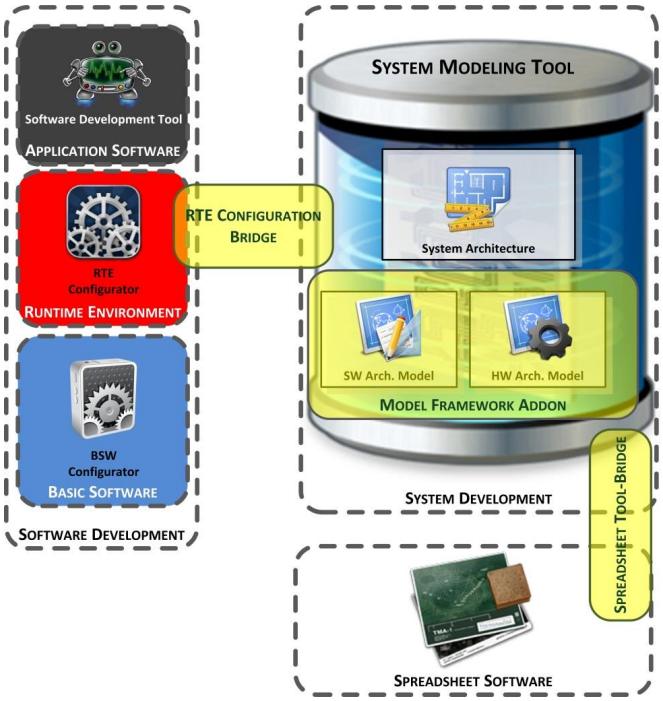

Fig. 1. Conceptual Overview of the HSI Definition Approach

representation in a MDB tool(such as Enterprise Architect). The spreadsheet tool and the MBD tool can be bidirectionally aligned via program-specific APIs, which supports the tool-independence of the approach presented.

An overview of the main parts of the contribution is shown in Figure 1, the following sections describe each part in more detail.

A. Application SW Modeling Framework

The first part of the approach is a specific UML modeling framework developed to enable an AUTOSAR-like representation of software architecture designs within the system development tool (Enterprise Architect). The UML 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. The AUTOSAR-aligned representation can be linked to system development artifacts and requirement representations, which eases the traceability between these different types of development artifacts. These explicit links can further be used for automated constraints checking and ease the confirmability of development decisions (e.g. for safety case generation). This provides the possibility to define software architecture and ensures a proper definition of the communication between the architecture artifacts, including interface specifications (e.g. upper limits, initial values, formulas).

Figure 2 shows the EA model profile for the modeling of the software module (AUTOSAR composition) and its signal interface definitions. As can be seen in the figure, software modules contain a declaration of their related ASIL and a

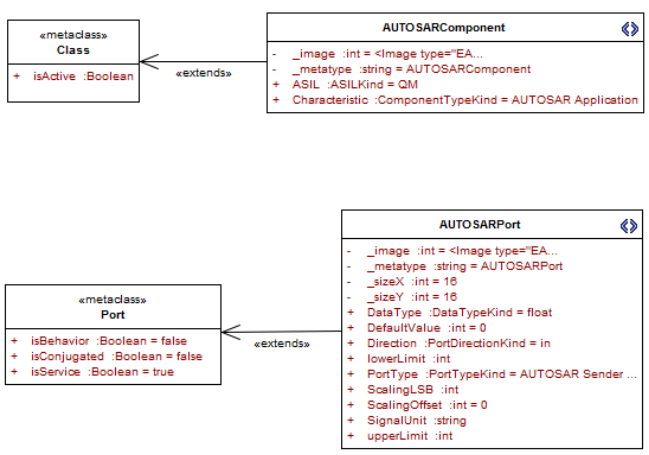

Fig. 2. EA Model for Modeling of Software Architectures

characteristic declaration. This characteristic determines the module either as a hierarchical composition or an atomic software module, which comprises additional benefits for SW allocation and analysis. The ASIL assignment supports the work of safety engineers by adding values and visual labels for safety-relevant software modules. The port configuration artifact (lower part of figure 2) implies the definition of the interface specification and enables the automatic generation of interfaces to BSW modules. The model representation of SW architectures and module interfaces in the MBD tool enables constraint checking features and supports traces to HSI definition and requirements.

B. Basic SW and HW Module Modeling Framework

To also model basic software (BSW) and hardware module representations additional model artifacts are defined. These representations are assigned to establish links from ASW modules to the underlying basic software and hardware layers. Thereby the hardware profile enables an intuitive graphical way of establishing hardware-software interfaces (HSI) and linking of SW signals of BSW modules to HW port pins via dedicated mappings. Figure 3 depicts the HW and BSW artifacts required to complete HSI model representation. The BSW module representation is shown in the upper part of the picture, this enables the modeling of interfaces between ASW and BSW layer (runtime environment). The second artifact is a representation of HW pins with a listing of the possible pin configurations. HSI configurations of the HW pin can be modeled using the third artifact.

C. HSI Definition Exporter and Importer

The HSI exporter MBD-tool extension establishes a links to the spreadsheet tool via API calls and enables the export of modeled HW/SW interfaces to spreadsheet documents (*.csv* or *.xls* files). The MBD-tool extension is developed in form of a dll class library and via API links, which provides

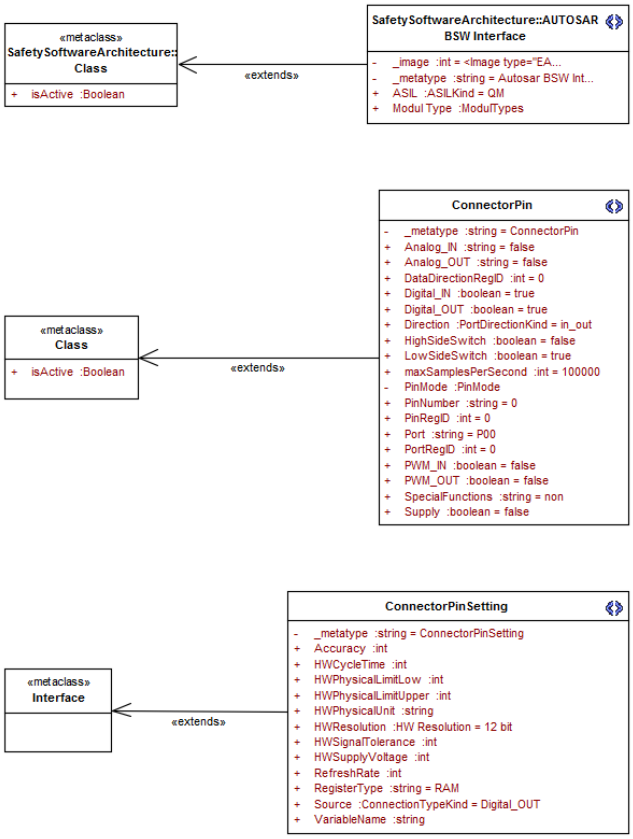

Fig. 3. EA Model for Modeling of Information of BSW and HW Representation

means for reuse by multiple programs and ensures MDB-tool independence of the exporter respectively of the specific spreadsheet tool.

The HSI spreadsheet importer is the HSI exporter's counterpart. Also implemented as a dll class library using the spreadsheet tools API it enables the import of information and selective update of HW/SW interface model artifacts, which enables round-trip engineering between spreadsheet and MDB tool HSI representations. Common spreadsheet file extensions (such as *.csv* or *.xls*) are importable, which correspond to the generic structure of the spreadsheet template.

D. Spreadsheet Template

The spreadsheet template defines the structure of the data representation in a generic, project-specific and customizable way. This ensures the practicable and intuitive method of engineering HSI definitions with spreadsheet tools. Additionally, the machine- and human-readable notation of spreadsheets enables the transformation of this information to a reusable and version-able representation in the MDB tool. This approach thus unifies the project-dependent process of HSI definitions across the variety of different projects and contributing partners without requiring exactly the same development tools or processes to be in place which ensures a cost- and time-saving alternative to what are usually complex special-purpose tools.

TABLE I
ADDITIONAL FEATURES PROVIDED BY THE APPROACH AND NUMBER OF EMERGES FOR THE BMS USE-CASE

Provided features	**Number of emerges for BMS use-case**
extracted settings to Excel	437
consistency checks of ASW signal mappings	54
modeled ASW artifacts	10 modules + 86 signal ports
modeled BSW artifacts	7 modules + 38 signal ports
modeled HW/SW interface artifacts	19 HW pins + 19 pin configurations
generated LoC for ASW/BSW mapping	33

E. SW/SW Interface Generator

The SW/SW interface generator generates .c and .h files defining SW/SW interfaces between application software signals and basic software signals from the modeled HSI artifacts. This eliminates the need of manual SW/SW interface generation without adequate syntax and semantic support and ensures reproducibility and traceability of these configurations. This generator is another dll class library based MBD tool extension, which can also be implemented by other frameworks or reused by other tools.

IV. EVALUATION OF THE HSI DEFINITION APPROACH

For evaluation of the approach an automotive use-case of a central control unit (CCU) in a battery management system (BMS) prototype for (hybrid) electric vehicle has been chosen. Project-specific details have been abstracted for reasons of commercial sensitivity. The CCU SW architecture of the use-case consists of 10 SW modules on ASW layer and 7 SW modules on BSW layer. The ASW modules count 54 inputs and 32 outputs, which define 48 SW/SW interfaces on ASW layer and 19 interfaces to BSW and HW connections.

This adds up to more than 30 lines of code (LoC) for HW/SW interface definition, which can be generated automatically into interface.c and interface.h files with the approach presented. Consistency checks for the 32 output interfaces and 54 input interfaces on the ASW layer can ensure point-to-point consistency of these signal routings. This adds up to 774 definitions, for 9 definable features per signal, which are automatically checked for consistency with this approach.

The HW/SW interface mapping consists of 19 interfaces for this specific SW architecture. The mappings include 23 settings per pin, which can be automatically exported into a spreadsheet and kept consistent with the model-representation via the importer functionality. This ensures actuality of dependent development artifacts and simplifies tracing of development decisions.

Table I sums up the additional features provided by the presented approach for the BMS use-case.

V. CONCLUSION

This paper presented a tool approach for ISO 26262 aligned hardware/software interface definition. The approach combines the versatility and intuitiveness of spreadsheet tools (such as Excel) and the properties of MDB tools (e.g. different views, levels of abstraction, central source of information, and information reuse) and does so in a bidirectional manner. This, enables a practicable, tool-independent, and intuitive method of engineering HSI definitions with spreadsheet tools and transformation of the generated information into a reusable and version able model representation. The machine- and human-readable notation of spreadsheets ensures a cost- and time-saving alternative to what are usually complex special-purpose tools (such as AUTOSAR tools). Furthermore, the capability of the approach for defining an ISO 26262 compliant HSI and automatic derivation of basic software configurations according to these interface definitions has been evaluated with a brief BMS use-case.

ACKNOWLEDGMENT

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). We are grateful for the contribution of the SOQRATES Safety AK experts and the expertise gained in SafEUr professional trainings. 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

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

[2] 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.

[3] A. Cimatti and S. Tonetta. A Property-Based Proof System for Contract-Based Design. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 21–28, Sept 2012.

[4] W. Damm, H. Hungar, B. Josko, T. Peikenkamp, and I. Stierand. Using contract-based component specifications for virtual integration testing and architecture design. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011.

[5] Christof Ebert and Capers Jones. Embedded Software: Facts, Figures, and Future. *IEEE Computer Society*, 0018-9162/09:42–52, 2009.

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

[7] Myron King, Nirav Dave, and Arvind. Automatic Generation of Hardware/Software Interfaces. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 325–336, New York, NY, USA, 2012. ACM.

[8] A. Soderberg and R. Johansson. Safety contract based design of software components. In *Software Reliability Engineering Workshops (ISSREW), 2013 IEEE International Symposium on*, pages 365–370, Nov 2013.

[9] Juergen Teich. Hardware/Software Codesign: The Past, the Present, and Predicting the Future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1411–1430, May 2012.