A Seamless Model-Transformation between System and Software Development Tools

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

A Seamless Model-Transformation between System and Software Development Tools

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 development of dependable embedded automotive systems faces many challenges arising from increasing complexity, coexistence of critical and non-critical applications, and the emergence of new architectural paradigms on the one hand, to short time-to-market intervals on the other hand. This situation requires tools to improve efficiency and consistence of development models along the entire development lifecycle. The existing solutions to date are still all too frequently insufficient when transforming system models with higher levels of abstraction to more concrete engineering models (such as software engineering models). Future automotive systems require appropriate structuring and abstraction in terms of modularization, separation of concerns, and supporting interactions between system, and component development.**

However, refinement of system designs into hardware and software implementations is still a tedious task. The aim of this work is to enhance an automotive model-driven system engineering framework with software-architecture design capabilities and a model-transformation framework to enable a seamless description of safety-critical systems, from requirements at the system level down to software component implementation in a bidirectional manner.

***Keywords*—Automotive, model-based development, reuse, traceability, model-based software engineering, ISO 26262.**

I. Introduction

Embedded systems are already integrated in our everyday lives and play a central role in all domains including automotive, aerospace, healthcare, manufacturing industry, the energy sector, or consumer electronics. In 2010, the embedded systems market accounted for almost 852 billion dollars worldwide, and is expected to reach 1.5 trillion by 2015 (assuming an annual growth rate of 12%) [18]. Current premium cars implement more than 90 electronic control units (ECU) per car with close to 1 Gigabyte software code [6], these are responsible for 25% of vehicle costs and bring an added value of between 40% and 75% [23].

The trend of replacing traditional mechanical systems with modern embedded systems enables the deployment of more advanced control strategies providing additional benefits for the customer and for the environment, but at the same time, the higher degree of integration and criticality of the control application is posing new challenges. These factors are resulting in multiple cross-domain collaborations and interactions in the face of the challenge of mastering the increased complexity involved and also to ensure consistency of the development along the entire product life cycle.

Model-based development supports the description of the system under development in a more structured manner, in the context of handling upcoming issues with modern real-time systems and also in relation to ISO 26262. Model-based development approaches enable different views for different stakeholders, different levels of abstraction and central storage of information. This improves the consistency, correctness, and completeness of the system specification. Nevertheless, such seamless integrations of model-based development still tend to be the exception rather than the rule and often fall short of target due to the lack of integration of conceptual and tooling levels [4]. Consequently, this work focuses on improving the continuity of information interchange from system development level to software development level models.

With this objective in mind the work focuses on improving the continuity of information interchange for architectural designs from system development level (Automotive SPICE [26] ENG.3 respectively ISO 26262 [10] 4-7 System design) to software development level (Automotive SPICE ENG.5 respectively ISO 26262 6-7 SW architectural design). More specifically, the approach is based on the enhancement of a model-driven system engineering framework with software-architecture design capabilities. The model-transformation framework automatically generates software architectures in Matlab/Simulink described via high level control system models in SysML format. The goal is, on the one hand, to support a consistent and traceable refinement from the early concept phase to software implementation. On the other hand, the bidirectional update function of the transformation framework enables facilitation in gaining mutual benefits for the basic software and the application software development from the coexistence of information for them both within the central database.

The document is organized as follows: Section II presents an overview of related approaches as well as model-based development and integrated tool chains. In Section III a description of the proposed bridging approach for the refinement of the model-based system engineering model to software development is provided. An application and evaluation of the approach is presented in Section IV. Finally, this work is concluded in Section V with an overview of the approach.

II. Related Work

Model-based systems and software development as well as tool integration aim at moving the development steps involved closer together and thus improving the consistency of information over the expertise and domain boundaries. Pretschner's roadmap [19] highlights the benefits of such a seamless model-based development tool-chain for automotive software engineering. Model-based development is also claimed to be the best approach to managing the large amount of information and the complexity of the modern embedded systems involved by Broy et al. [4]. Their paper illustrates why seamless solutions have not been achieved so far and mentions concepts and theories for model-based development of embedded software systems. Additionally they make reference to commonly used solutions and problems arising with inadequate tool-chain support (e.g. redundancy, inconsistency and lack of automation). Nevertheless, the challenge of enabling a seamless integration of models into model-chains is still an open issue [20], [21], [27] 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 due to process and tooling gaps.

The work of Holtmann et al. [9] highlights process and tooling gaps between different modeling aspects of a model-based development process. Giese et al. [8] address issues of correct bi-directional transfer between system design models and software engineering models. The authors propose a model synchronization approach consisting of tool adapters between SysML models and software engineering models in AUTOSAR representation.

Dealing with this gap between system architecture and software architecture, especially while considering component-based approaches such as UML and SysML for system architecture description and AUTOSAR for SW architecture description, is one of the most important topics in this entire issue. Two common variants in the automotive domain are the usage of SysML [3], [8], [11], [14], [17] or X-MAN [12] approaches for architectural description and AUTOSAR for software system description. Boldt [3] 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 approach of bridging the gap between model-based system engineering and software engineering models based on EAST-ADL2 architecture description language and a complementary AUTOSAR representation is also very common in the automotive software development domain [5], [16], [25]. EAST-ADL represents an architecture description language using AUTOSAR elements to represent the software implementation layer of embedded systems [2]. More recently the MAENAD Project[1] is also focusing this approach.

Kawahara et al. [11] propose an extension of SysML which enables description of continuous time behavior. Their tool integration base on Eclipse and couples SysML and Matlab/Simulink via API.

[1] http://maenad.eu/

Farkas et al. [7] describe in their paper an integrative approach for Embedded Software Design with UML and Simulink. Their presented approach aims in a stepwise migration towards model-based development and enables the cooperative usage of MATLAB/Simulink & UML for functional specification and code generation. The focus of this work is on the combination of source codes generated by different model-based tools, rather than the interchange of data between the different model representations.

SysML and model-based development (MBD) as the backbone for development of complex safety critical systems is also seen as a key success factor by Lovric et. al [13]. The paper evaluates key success factors of MBD in comparison to legacy development processes in the field of safety-critical automotive systems.

Tool support for automotive engineering development is still organized as a patchwork of heterogeneous tools and formalisms [2]. On the one hand, general-purpose modeling languages (such as UML or SysML) provide modeling power suitable for capturing system wide constraints and behavior, but are lacking in synthesizability. On the other hand, special-purpose modeling languages (such as C, Assembler, Matlab, Simulink, ASCET) are optimized for fine granular design, but are less efficient in high-level design.

The issue of improving these interactions, especially those which deal with cross-domains affairs (such as the architectural design refinement from system development level to software development level), thus requires a comprehensive understanding of related processes, methods, and tools. The work of Sechser [24] describes the experiences gained when combining two different process worlds in the automotive domain.

III. Model-Transformation Bridge Approach

This paragraph gives a brief overview of the underlying framework and related preliminary work which supports the proposed approach. The presented framework focuses on improving the continuity of information interchange from system development level to software development level. The basic concept behind this framework is to have a consistent information repository as central source of information, to store all information of all the engineering disciplines involved for embedded automotive system development in a structured way [15].

The methodical support of system architectural design and refinement of this design to software design often fell short of the mark. To handle this situation the AUTOSAR methodology [1] provides standardized and clearly defined interfaces between different software components and development tools and also provides such tools for easing this process of architectural design refinement. Nevertheless, the enormously complex AUTOSAR model requires a high amount of preliminary work and projects with limited resources often struggle to achieve adequate quality within budget (such as time or manpower) using this approach. This approach thus arises out of common AUTOSAR based approaches and forces a direct model transformation from SysML representation to Matlab/Simulink. The reason for making the decision of not fostering an AUTOSAR approach is based on the one hand on

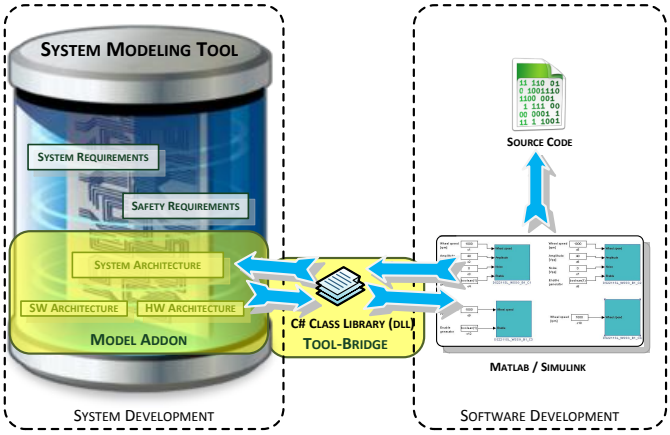

Fig. 1. Portrayal of the Bridging Approach Transferring System Development Artifacts to SW Development Phase

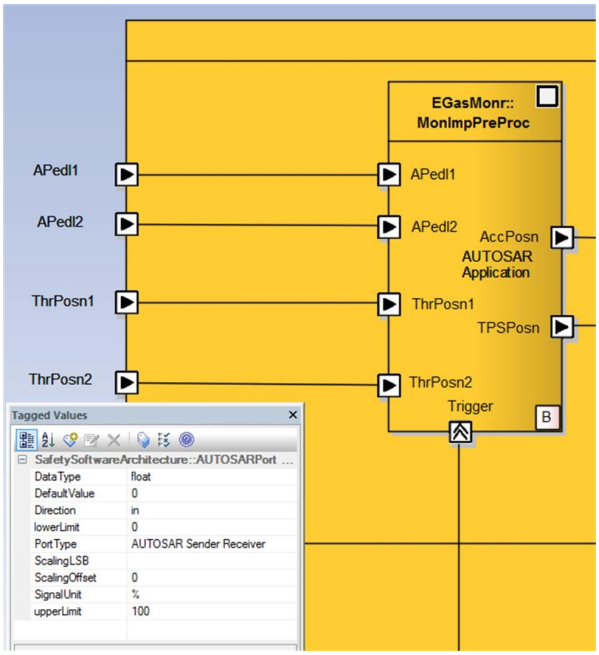

Fig. 2. Screenshot of the SW Architecture Representation within the System Development Tool and Representation of the Interface Information

focusing not only AUTOSAR but also rather on generally Matlab/Simulink based automotive software development. On the other hand, experiences we made with our previous approach [14] confirm the problem mentioned by Rodriguez et al. [21]. Not all tools fully support the whole AUTOSAR standard, because of its complexity, which leads to several mutual incompatibilities and interoperability problems. The presented MDB model has been developed using profiles which use a subset of the SysML language to define a SW architecture model particularly tailored to automotive SW engineering in context of ISO 26262. In the following paragraphs we describe the additional model enhancements to support software development and modeling of complex software architectures for function software development. The contribution presented in this work supports automatic generation of software architectures, interface definition, timing setting, and auto-routing of signals in Matlab/Simulink based on SysML representation.

Figure 1 shows an overview of this approach and the imbedded bridging of abstract system development and concrete software development models. More specifically, our contribution consists of the following parts:

- *SW modeling framework*: Enhancement of a SysML profile for the definition of SW component interfaces and SW architecture composition. Required for consistent SW system description, see Figure 1 – model addon.
- *SW architecture exporter*: Exporter to generate the designed SW architecture in Matlab/Simulink for further development of SW functions, see Figure 1 – tool bridge.
- *SW architecture importer*: Importer to integrate refined SW architecture and interfaces from the software development tool (to support round-trip engineering), see Figure 1 – tool bridge.

This proposed approach closes the gap, also mentioned by Giese et al. [8], Holtmann et al. [9], and Sandmann and Seibt [22], between system-level development at abstract UML-like representations and software-level development modeling tools (e.g. Matlab/Simulink or Targetlink). The bridging supports consistency of information transfer between system engineering tools and software engineering tools and minimizes redundant manual information exchange between these tools. This contributes to simplifing seamless safety argumentation according to ISO 26262 [10] for the system developed. The 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 can be seen in Figure 1, the lack of supporting tools for information transfer between system development tools and software development tools can be dispelled by our approach. The implementation of the bridge based on versatile C# class libraries (dll) and Matlab COM Automation Server ensures tool in-dependence of the general-purpose UML modeling tool (such as Enterprise Architect or Artisan Studio) and version in-dependence of Matlab/Simulink through API command implementation. This makes the method especially attractive for projects and companies with limited resources (either in manpower or finances). Small projects or start-up companies in particular often struggle with the problem of setting up their development processes so as to achieve adequate quality.

A. Software Modeling Framework

The first part of the approach is a specific SysML modeling framework which enables the possibility of designing software architectures in an AUTOSAR aligned manner within a system development tool. The profile enables an explicit definition of AUTOSAR components, component interfaces, connections between interfaces and makes the SysML representation more manageable for the needs of the design of an automotive software architecture. Furthermore, it opens up the possibility for defining software architecture and ensures establishment of communication between architecture artifacts with interface specifications (e.g. upper limits, initial values, formulas). Special basic software and hardware abstraction modules are

TABLE I. SW ARCHITECTURE IMPORTER INDICATORS OF TYPE OF CHANGE

Indicator	Type of Change
A	model artifact added
AC	interface connection added
D	model artifact deleted
DC	interface connection deleted
U	model artifact updated
UC	interface connection updated

assigned to establish links to the underlying basic software and hardware abstraction layers. Moreover, these SW modeling artifacts can be linked to the system model artifacts and requirements in such a manner that traceable links can be established more easily. This has further benefits in terms of constraints checking, reuse, and reporting generation (e.g. for safety case generation). Figure 2 shows an example of software architecture artifacts and interface information represented in Enterprise Architect. Furthermore, this integrated definition of system artifacts and software module in one tool supports the work of safety engineers by adding values and visual labels for safety-relevant software modules.

In addition to standard VFB AUTOSAR profiles the profile features assignment and graphical representation of ASIL to dedicated signals and modules and provides specification of runnables with timing constraints (such as WCET), ASIL, and priority. This additional information enables mapping of tasks to a specific core and establishment of a valid scheduling in a later development phase. Further benefits result in terms of constraints checking and traceability of development decisions.

B. SW Architecture Exporter

The second part of the approach is the SW architecture exporter. The implementation of the exporter is based on Matlab COM Automation Server and generates models through API command implementation, which ensures tool version-independence. The export functionality enables the export of software architecture, component containers, and their interconnections designed in SysML to the software development tool Matlab/Simulink. The SW architecture artifacts to be transferred can be selected by user input and the corresponding Matlab/Simulink model is generated by a background task. As can be seen in Figure 3 the user is able to select the SW artifacts for exporting, the desired model representation in Matlab/Simulink (either TargetLink or Simulink representation), and the exporting mode (m-file based, API based, or as ARXML file). The export mode variants also enable exporting if Matlab/Simulink is not available (m-file based) or an AUTOSAR based SW development toolchain is used (ARXML file based). Listing 1 shows some excerpts of the automatically generated Matlab API commands. As can be seen in this listing, each model artifact, parameter, and connection is transferred to Matlab/Simulink, where the blocks are arranged and sized in a correct manner. Besides this, unique links to the EA representation and assigned safety-criticality marking of the artifact (Listing 1 line 3 and 8) are established.

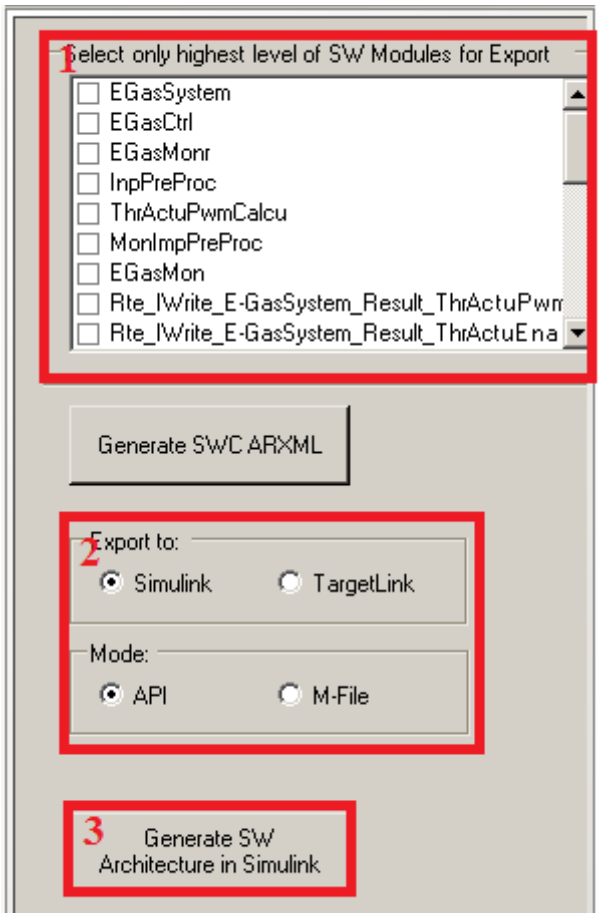

Fig. 3. Screenshot of the SW Architecture Exporter GUI

Listing 1. Excerpts of Matlab API Commands

```
1  addpath(genpath('C:\EGasSystem'))
2  add_block('Simulink/Ports & Subsystems/Model','EGasSystem/
       EGasCtrl')
3  set_param('EGasSystem/EGasCtrl','ModelNameDialog','EGasCtrl'
       ,'Description','EA_ObjectID@1969;ASIL@QM')
4  set_param('EGasSystem/EGasCtrl','Position',[250 50 550 250])
   .
5  :
6  add_block('Simulink/Ports & Subsystems/In1','EGasSystem/
       APedl2')
7  set_param('EGasSystem/APedl2','Position',[50 200 80 215])
8  set_param('EGasSystem/APedl2','Outmin','0','Outmax','5','
       OutDataTypeStr','single','Description','
       EA_ObjectID@1966;ASIL@B');
   .
9  :
10 add_line('EGasSystem','APedl1/1','EGasMonr/1','AUTOROUTING',
       'ON')
   .
11 :
12 save_system('EGasSystem')
13 close_system('EGasSystem')
14 cd ..
15 cd C:\EGasSystem
```

C. SW Architecture Importer

The last part of the approach is the import functionality add-on for the system development tool, which in combination with the export function, enables bidirectional updates of software architecture representations in the system development tool and the software modules in Matlab/Simulink. The

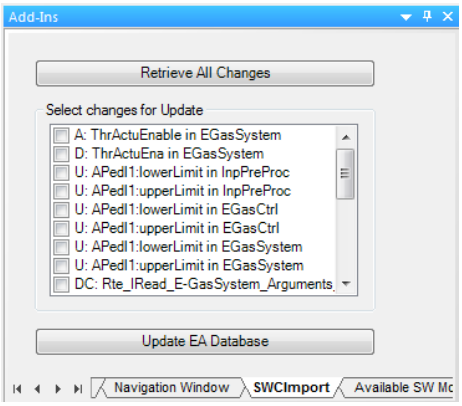

Fig. 4. SW Architecture Importer User Interface

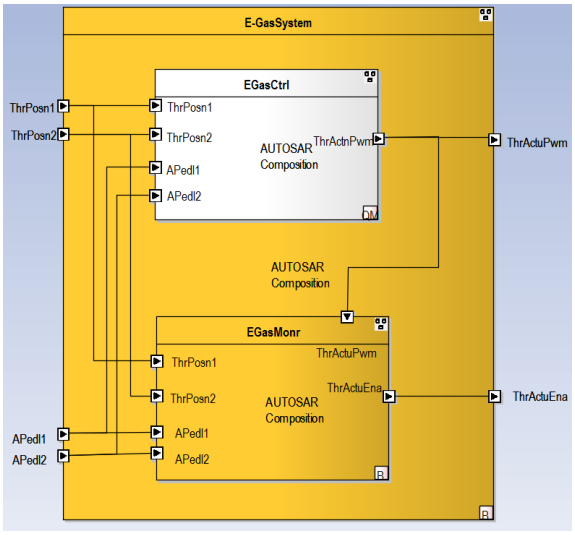

Fig. 5. Top-Level Representation of Demonstration Use-Case in Enterprise Architect

importer analyzes the Matlab/Simulink model representation and identifies the unique links to the EA representation (shown in Listing 1 line 3 and 8). Thereby new and modified model artifacts can be differentiated and changes made in the software development tool can be kept consistent within the system development model representation. This ensures consistency between the models, enables importing of newly available software modules from Matlab/Simulink, and therefore guarantees consistency of information across tool boundaries. Figure 4 shows the user interface within the system development tool. As can be seen in this figure, modifications between the two models are identified and a selective update of the SysML representation can be triggered by the user. Furthermore, a highlighting of the type of change can also be depicted. Table I shows the different change type indicators and types of changes.

IV. Application of the Proposed Approach

This section demonstrates the introduced approach by an automotive embedded system use-case. To provide a comparison and highlighting of the improvements of our approach we use the 3 layer monitoring concept [28] as an evaluation use-case. This elementary use-case is well-known in the automotive domain, but is nevertheless representative. Moreover, this elementary use-case is illustrative material, which is also used for internal training purposes with students and engineers. The disclosed and commercially non-sensitivity use-case is not intended to be exhaustive, nor to be representative of leading-edge technology.

The definition of the software architecture is usually performed by a software system architect within the software development tool (Matlab/Simulink). With our approach this work package is included in the system development tool (depicted in Figure 5). This does not hamper the work of the software system architect, but it enables constraint checking features and helps to improve system maturity in terms of consistency, completeness, and correctness of the development artifacts. Besides this, the change offers a significant benefit for the development of safety-critical software in terms of traceability, replicability of design decisions and it unambiguously visualizes dependencies while putting visual emphasis on view-dependent constraints (such as graphical safety-criticality highlighting of SW modules in Figure 5).

The 3 layer monitoring concept use-case presented consists of 7 SW modules with 34 interfaces and 30 signal connections. Hereby the SW module representations contain 3 configurable attributes per element and the SW interfaces 34 attributes per element. The use-case thus sums up to a total count of 41 model artifacts with 361 configuration parameters and 30 relations between the elements. This elementary example already indicates that the number of model elements and relations between the model elements already becomes confusing. A manual transformation of the information represented within the models would already be cumbersome, error-prone, and would involve a great amount of additional work to ensure consistency between the two models.

The presented approach in this work checks the information and model artifacts for point-to-point consistency of interface configurations before automatically transferring the model representation via 212 lines of auto-generated Matlab API code, which provides evidences and ensures the completeness of the model transformation. The presented SW architecture importer functionality enables round-trip engineering and bi-directional updates of both models and therefore supports evidence for the consistency of both models.

In terms of safety-critical development and reuse the features of the approach presents are crucial to transfer information between separated tools and link supporting safety-relevant information. Moreover, the approach eliminates the need for manual information reworking without adequate tool support, ensuring reproducibility, and traceability argumentation.

V. Conclusion

The challenge with modern embedded automotive systems is to master the increased complexity of these systems and ensure consistency of the development along the entire product life cycle. Automotive standards, such as ISO 26262 safety standard provide a process framework which requires efficient

and consistent product development and tool support. Nevertheless, various heterogeneous development tools in use are hampering the efficiency and consistency of information flows.

This work thus focuses on improving the continuity of information interchange of architectural designs from system development level (Automotive SPICE ENG.3 respectively ISO 26262 4-7 System design) to software development level (Automotive SPICE ENG.5 respectively ISO 26262 6-7 SW architectural design). For this purpose, an approach to seamlessly combine model-based development tools on system level (such as Enterprise Architect) and on SW development level (such as Matlab/Simulink) has been proposed.

The applicability of the approach has been demonstrated utilizing an elementary automotive use-case, the 3 layer monitoring concept, which is an illustrative material and does not represent either an exhaustive or a commercially sensitive project. The main benefits of the presented approach are: improved consistency and traceability from the initial design at the system level down to the software implementation, as well as, a reduction of cumbersome and error-prone manual work along the system development path.

ACKNOWLEDGMENTS

This work is partially supported by the EMC^2 and the $MEMCONS$ projects.

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement nr 621429 (project EMC^2) and 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

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

[2] H. Blom, H. Loenn, F. Hagl, Y. Papadopoulos, M.-O. Reiser, C.-J. Sjoestedt, D. Chen, and R. Kolagari. EAST-ADL - An Architecture Description Language for Automotive Software-intensive Systems. White Paper 2.1.12, 2013.

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

[4] M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu. Seamless Model-based Development: from Isolated Tool to Integrated Model Engineering Environments. *IEEE Magazin*, 2008.

[5] D. Chen, R. Johansson, H. Loenn, Y. Papadopoulos, A. Sandberg, F. Toerner, and M. Toerngren. Modelling Support for Design of Safety-Critical Automotive Embedded Systems. In *SAFECOMP 2008*, pages 72 – 85, 2008.

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

[7] T. Farkas, C. Neumann, and A. Hinnerichs. An Integrative Approach for Embedded Software Design with UML and Simulink. In *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, volume 2, pages 516–521, July 2009.

[8] H. Giese, S. Hildebrandt, and S. Neumann. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *LNCS 5765*, pages pp. 555 –579, 2010.

[9] J. Holtmann, J. Meyer, and M. Meyer. A Seamless Model-Based Development Process for Automotive Systems, 2011.

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

[11] R. Kawahara, D. Dotan, T. Sakairi, K. Ono, A. Kirshin, H. Nakamura, S. Hirose, and H. Ishikawa. Verification of embedded system's specification using collaborative simulation of SysML and Simulink models. In *Proceedings of Second International Conference on Model Based Systems Engineering*, pages 21 – 28, March 2009.

[12] K.-K. Lau, P. Tepan, C. Tran, S. Saudrais, and B. Tchakaloff. A Holistic (Component-based) Approach to AUTOSAR Designs. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 203–207, Sept 2013.

[13] T. Lovric, M. Schneider-Scheyer, and S. Sarkic. SysML as Backbone for Engineering and Safety - Practical Experience with TRW Braking ECU. In *SAE Technical Paper*. SAE International, 04 2014.

[14] G. Macher, E. Armengaud, and C. Kreiner. Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures. In *12. Workshop Automotive Software Engineering (ASE)*, Lecture Notes in Informatics, pages 2145–2156, 2014.

[15] G. Macher, E. Armengaud, and C. 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 –263, 2014.

[16] R. Mader, G. Griessnig, A. Eric, L. Andrea, K. Christian, Q. Bourrouilh, C. Steger, and R. Weiss. A Bridge from System to Software Development for Safety-Critical Automotive Embedded Systems. *38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 75 –79, 2012.

[17] J. Meyer. *Eine durchgaengige modellbasierte Entwicklungsmethodik fuer die automobile Steuergeraeteentwicklung unter Einbeziehung des AUTOSAR Standards*. PhD thesis, Universitaet Paderborn, Fakultaet fuer Elektrotechnik, Informatik und Mathematik, July 2014.

[18] A. Petrissans, S. Krawczyk, L. Veronesi, G. Cattaneo, N. Feeney, and C. Meunier. Design of Future Embedded Systems Toward System of Systems - Trends and Challenges. European Commission, May 2012.

[19] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner. Software Engineering for Automotive Systems: A Roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 55–71, Washington, DC, USA, 2007. IEEE Computer Society.

[20] I. R. Quadri and A. Sadovykh. MADES: A SysML/MARTE high level methodology for real-time and embedded systems, 2011.

[21] E. Rodriguez-Priego, F. Garcia-Izquierdo, and A. Rubio. Modeling Issues: A Survival Guide for a Non-expert Modeler. *Models2010*, 2:361–375, 2010.

[22] G. Sandmann and M. 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.

[23] G. Scuro. Automotive industry: Innovation driven by electronics. http://embedded-computing.com/articles/automotive-industry-innovation-driven-electronics/, 2012.

[24] B. Sechser. The marriage of two process worlds. *Software Process: Improvement and Practice*, 14(6):349–354, 2009.

[25] C.-J. Sjoestedt, J. Shi, M. Toerngren, D. Servat, D. Chen, V. Ahlsten, and H. Loenn. Mapping Simulink to UML in the Design of Embedded Systems: Investigating Scenarios and Structural and Behavioral Mapping. In *OMER 4 Post Workshop Proceedings*, April 2008.

[26] The SPICE User Group. Automotive SPICE Process Assessment Model. Technical report, 2007.

[27] J. Thyssen, D. Ratiu, W. Schwitzer, E. Harhurin, M. Feilkas, T. U. Muenchen, and E. Thaden. A system for seamless abstraction layers for model-based development of embedded software. In *Software Engineering Workshops*, pages 137–148, 2010.

[28] T. Zurawka and J. Schaeuffele. Method for checking the safety and reliability of a software-based electronic system, January 2007.