Dipl.- Ing. Georg Macher, BSc

Framework for the Integrated Model-Based Development of Dependable Automotive Systems and Software

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Eugen Brenner

Advisor
Dipl.-Ing. Dr.techn. Christian Kreiner

Institute of Technical Informatics

Graz, November 2015

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date

Signature

Abstract

Embedded systems are already integrated into our everyday life and play a central role in all domains including automotive, aerospace, healthcare, and industry. In recent years, the complexity of embedded automotive systems has grown significantly due to the replacement of traditional mechanical systems with electro-mechanical systems. These systems enable the deployment of more advanced control strategies and assistant functions but also triggered the need for more computing power and a paradigm change in development. With the deployment of multi-core systems for the automotive domain and the increasing impact of software on embedded systems' functionality, the automotive domain faces an enormous increase in complexity and a growing gap between the technologies and the required level of expertise. Furthermore, extra-functional product attributes are currently either unique selling point opportunities or show stoppers. Nevertheless, these system features require a fine grasp of commonalities and cross-domain knowledge and pose additional challenges for the development of embedded automotive systems. These interdisciplinary fields, their mutual correlations, and high amount of cross-domain expertise require adequate methodologies and tools support to be integrated in the existing development process landscape. The classical 'separation of concern' approach of the automotive industry has to be reconsidered and an approach found which provokes seamless cross-fertilization across tool, team, and domain skills. Model-based development exemplifies a system under development in a more structured way, enabling different levels of abstraction and incorporates the views of different stakeholders. Thus, development models serve as a central source of information for all parties involved and are geared towards collaboration across tool and domain boundaries. The current model-based development approaches are frequently prevented from tapping into their full potential due to problems arising from inadequate tool-chain support and missing methodical support.

Therefore, this thesis aims to establish a model-based development approach to tackle the issues of modern embedded automotive systems and accelerate the transfer of know-how across the engineering domains involved. The presented approach focuses on the process, method, and tool layer with a special focus on dependable multi-core system development. With regards to the process layer, industrial processes are analyzed and adaptations or extensions are suggested. The focus of the method layer is the exploitation of commonalities and concurrent methods for dependability attributes and cross-domain-enrichment of methods. The intention regarding the tool layer is to merge heterogeneous tools required for multi-core system development.

Kurzfassung

Eingebettete Systeme sind in vielen Bereichen des täglichen Lebens integriert und spielen auch eine zentrale Rolle in Industriezweigen, wie Luftfahrt, Medizintechnik, Energieversorgung und im Automobilbereich. In der Automobilindustrie sind eingebettete Systeme in den letzten Jahren verstärkt aufgetreten. Dies hängt vor allem mit dem Austausch herkömmlicher mechanischer Systeme durch eingebettete elektro-mechanische Steuerung zusammen. Vor allem solchen Steuerungen ist es zu verdanken, dass der Einsatz komplexerer Regelsysteme und ausgefeilterer Fahrerassistenzsysteme ermöglicht wurde. Solche komplexe Funktionalitäten benötigen jedoch auch mehr Rechenkapazität, was den Einsatz von Mehrkernrechnern auch für sicherheitskritische Funktionen im Automobil nach sich zieht. Dieser Technologiewandel, die enorm gestiegene Komplexität von Software im Automobil und der gleichzeitig gestiegene Wunsch nach gesteigerten nichtfunktionalen Eigenschaften (erhöhte Zuverlässigkeit, Verfügbarkeit und Sicherheit) fordern tiefgreifendere Kenntnisse in verschiedensten Wissensgebieten des Automobilbaus und das Überblicken höherer Systemkomplexität. Interdisziplinäres Wissen, Kenntnisse von wechselseitigen Einwirkungen sowie das hohe Maß an technischem Verständnis benötigen die Entwicklung und Bereitstellung adäquater Methoden, Prozessbeschreibungen und Werkzeugunterstützung für die Produktentwicklung. Klassische Ansätze, wie das bisher gängige Aufteilen der Entwicklung in einzelne Teilbereiche, müssen neu überdacht und durch übergreifende Ansätze ersetzt werden. Modellbasierte Entwicklung scheint zur Zeit der geeignetste Ansatz zu sein, um das zu entwickelnde Produkt strukturiert beschreiben zu können, verschiedene Denkmuster und Betrachtungswinkel zu vereinigen und eine gemeinsame Basis für die Entwicklung zu ermöglichen.

Aus diesem Grund beschäftigt sich diese Doktorarbeit mit modellbasierter Entwicklung integrierter Systeme im Automobilbereich und dem Ausschöpfen des Potentials dieses Ansatzes, um den Herausforderungen der Entwicklung auf diesem Sektor gerecht zu werden und einen besseren Wissensaustausch zwischen den involvierten Entwicklern zu ermöglichen. Um das zu gewährleisten, wird die Thematik auf drei Ebenen behandelt: Die Prozessebene betreffend werden industrietypische Prozesse analysiert und Anpassungen oder Erweiterungen vorgeschlagen. Die Methodikebene ist von der Analyse domänenspezifische und domänenübergreifende Vorgehensweisen bestimmt, Gemeinsamkeiten und passende Methodik Kombinationen werden herausgearbeitet. Für die Werkzeugebene, respektive den Machbarkeitsaspekt, wird ein automatisierter Datenaustausch zwischen den, für den Entwicklungsprozess notwendigen, Werkzeugen eingeführt.

Acknowledgments

First and foremost I would like to thank my family and girlfriend for their lifelong support and for their tolerance and encouragement during my studies. Their motivation and cheering up during tough and stressful periods as well as their knocking me out of the skies when I needed it made it possible to finish my PhD study. Thank you very much!

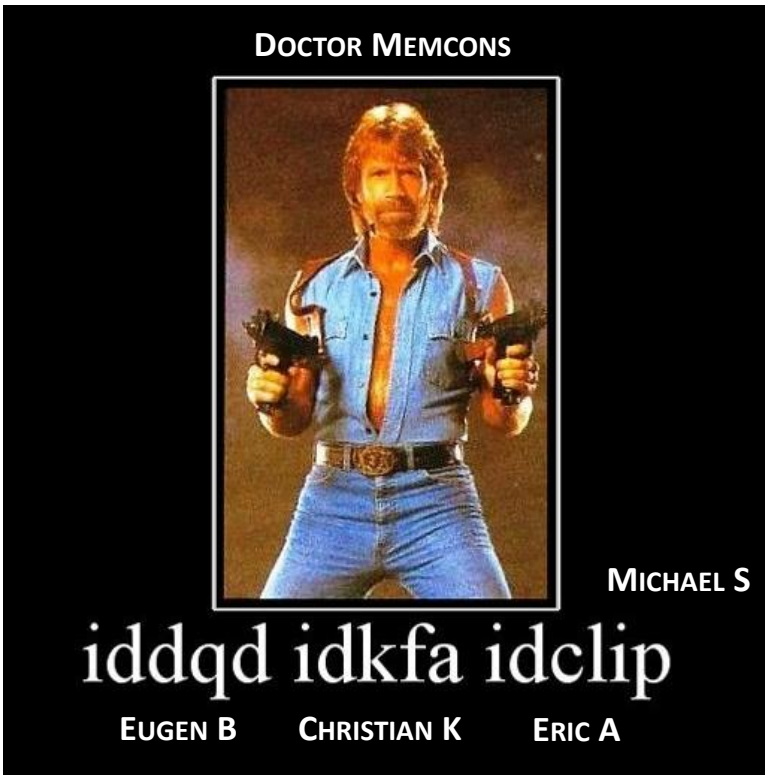
This thesis has been conducted as part of the MEMCONS project in cooperation with supporting project partners, AVL List GmbH, Virtual Vehicle Research Center, and Graz University of Technology. I would like to thank Christian Kreiner, Eric Armengaud and Michael Stolz for managing the cooperation of these partners and keeping me out of managing duties and focused on work. I also want to thank the cooperation partners AVL List GmbH and Virtual Vehicle Research Center for making the project possible and providing me with relevant feedback, real-world use-cases and knowledge of industrial needs. Additionally, I would like to thank my MEMCONS student team for their contributions and the convivial meetings with many fruitful outcomes.

Special thanks to my supervisor at Graz University of Technology, Prof. Eugen Brenner. He made the official and organizational part of my PhD study very easy, gave me a lot of freedom regarding research topics and helped me elaborate the scientific points of my thesis. Also in this context special thanks to Christian Kreiner, who convinced me to start this PhD, turned out to be 'Master Yoda' as my mentor and introduced me to very specific domain topics and granted me access to a selected work group. This provided a good working environment to conduct this doctoral thesis and left me a lot of open space to be creative and innovative. Furthermore, I would like to thank my colleagues at the different involved partner companies for the fruitful collaborations, qualified comments, supporting contributions, numerous coffee meetings and an atmosphere of friendship.

Finally, I want to thank my aunt Sr. Anna and her colleagues for proofreading this work and I want to dedicate the following pages to those who added a special contribution to this thesis.

to my family

to my project facilitators



to my comrades

Executive Summary

In the late 1970s self-contained embedded systems called Electronic Control Units (ECUs) entered production vehicles. Since then, such computer systems have been integrated into almost every aspect of a car: control throttle, transmission, brakes, passenger climate, and infotainment. Currently available premium cars implement more than 100 electronic control units (ECU) with close to 1 Gigabyte software code. Additionally, the amount of software in such embedded systems has grown to approximately 100 million lines of code(LoC) (in comparison the F-35 Joint Strike Fighter will require about 5.7 million LoC and the Boeing 787 Dreamliner about 6.5 million LoC).

Embedded automotive systems are estimated to be responsible for 80 % of product innovation in the past decade and are responsible for 25% of current vehicle costs. Today's information society strongly supports the inclusion of inter-system communication (Car2X) in the automotive domain. Consequently the boundaries between application domains are disappearing even faster than previously due to the replacement of traditional mechanical systems. These factors cause multiple cross-domain collaborations and interactions in the face of the challenge to master the increased complexity.

Current innovation challenges in the domain are related to novel computing paradigms (such as multi-core systems), system interaction and cooperation with the Web (such as Car2Environment), and extra-functional system dependability constraints (such as safety, IT security, and reliability). Hence, the automotive industry is facing a growing gap between technology and the required level of expertise to make best use of it. In this context this doctoral thesis is founded in cooperation with the industrial partner AVL List GmbH and its Powertrain Engineering branch. The thesis is part of the '**M**odel-based **EM**bedded **CON**trol **S**ystems' (MEMCONS) project which tackles a number of research challenges in the environment of embedded control systems development, such as:

- Optimal software calibration
- Framework for thermal management in hybrid vehicles
- Coordination of dynamic requests in hybridized powertrains
- Automotive Software Refactoring
- Realtime software on multi-core platforms
- Safety analysis

The doctoral thesis is aimed at improving the model-based development of embedded automotive multi-core systems from initial system design to software implementation in order to support a comprehensive dependability development. For this aim three major challenges are focused on:

1. multi-core system development
2. extra-functional system attributes development
3. life-cycle wide model-based development

The introduction of multi-core computing platforms provided more computing resources and additional interfaces to answer the needs of new automotive control strategies with respect to computing performance and connectivity. At the same time, the parallel execution and resulting resources and timing conflicts require a paradigm change to the embedded software. The automotive industry is confronted with the central question of how to migrate, optimize, and validate a given application (or set of applications) on a given computing platform with a given operating system. Due to the timing- and safety-criticality of automotive application, lessons learned from other domains (such as consumer electronics) cannot be adopted directly. Therefore, a knowledge transfer is required to identify the application requirements (both functional and extra-functional), perform a mapping to the SW and HW architecture, and grasp the possible impacts of concurrency and HW specifics on the software applications.

Dependability is a superordinate concept regrouping different system attributes such as reliability, safety, security, or availability and other extra-functional requirements for modern embedded systems and requires coordinated development along the entire development life-cycle. In the automotive industry, dependability engineering is currently moving to be the main focus; extra-functional features are either 'show stoppers' or 'unique selling points' for embedded automotive system development. These different attributes might lead to different targets and thus there is a strong need for unified methods to manage these different attributes. Therefore, the focus is set on addressing all these system attributes in combination (not independent from each other) and across different skill teams along the entire life-cycle with the same emphasis. Indeed, a common analysis method delivering consistent dependability targets across the different attributes is the basis for performing consistent dependability engineering during the entire product development.

Model-based development alleviates the issue of inherent complexity and the most promising approach of interdisciplinary development; but in the case of embedded system development, seamless cooperation between the involved domains and development experts is a core challenge. In model-based designs, a model of the application is used to represent complex designs at higher levels of abstraction; showing only specific pieces of information for the purpose of understanding and analyzing large designs. The model

forms the basis for further activities, such as verification, code generation, or model transformation. Tool support for automotive engineering development is still organized as a patchwork of heterogeneous tools and formalisms.

To address these matters, this doctoral thesis focuses on improving the comprehensive dependability argumentation of automotive multi-core system development. The strategy of this doctoral thesis' approach is to tackle the previously mentioned three major challenges on tool, method, and process layer (see Figure 1); layers which have been identified as crucial for industrialization of engineering approaches.

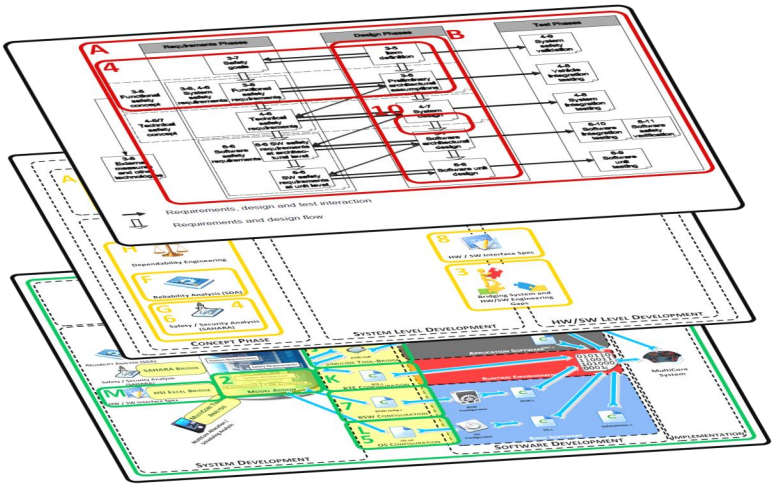


Figure 1: Layers for Industrialization of Engineering Approaches

Figure 2 shows a depiction of this thesis' concept in GSN notation. The major contributions for success are listed on the contribution layer of Figure 2 and supported via the related publications. The main goals of this doctoral thesis are on the one hand, to reveal challenges which appear when developing a dependable multi-core system, especially in the automotive domain and the development of strategies for the migration to multicore systems. On the other hand, the development of such systems needs to be supported by an adequate development framework.

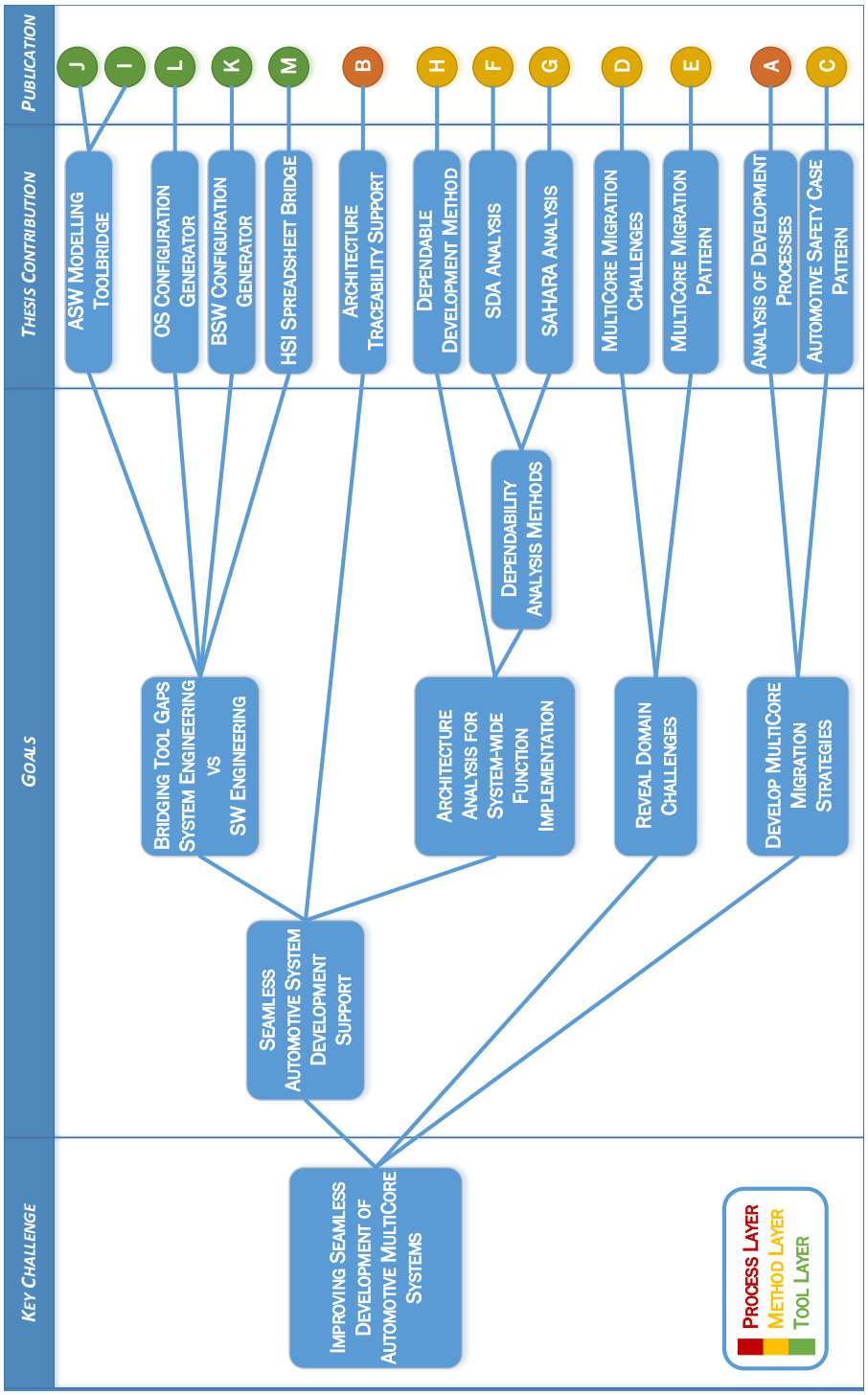


Figure 2: Overview of the Doctoral Thesis Concept Strategy

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-Digital Converter
ADD	Automotive Data Directory
API	Application Programming Interface
ARTOP	AUTOSAR Tool Platform
ARXML	AUTOSAR XML Format
ASIC	Application-Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level
ASM	Asymmetric Multiprocessing
ASW	Application-Software
Automotive SPICE	Automotive Variant of Software Process Improvement and Capability Determination
AUTOSAR	Automotive Open Systems Architecture
AVB	Audio Video Bridging
AVL-PTE	AVL Powertrain Engineering Branch
BMS	Battery Management System
BSW	Basic-Software
CAN	Controller Area Network
CAN-FD	CAN with flexible data-rate
CCU	Central Control Unit
CMMI	Capability Maturity Model Integration
CPU	Central Processing Unit

CTQ	critical-to-quality
DRL	Deterioration Resistance Level
EA	Enterprise Architect
EAST-ADL	Electronics Architecture and Software Technology Architecture Description Language
ECU	Electronic Control Unit
EMF	Eclipse Modeling framework
EU	European Union
EV	Electric Vehicle
FHA	Functional Hazard Assessment
FMEA	Failure Mode and Effects Analysis
FSC	Functional Safety Concept
FSR	Functional Safety Requirement
FTA	Fault Tree Analysis
GSN	Goal Structure Notation
GSS	Global System Structure
HARA	Hazard Analysis and Risk Assessment
HSI	Hardware-Software-Interface
HV	High Voltage
HW	Hardware
ICC	Implementation Conformance Class
IOC	Inter-OS-Application Communication
LIN	Local Interconnect Network
LoC	Lines of Code
MARTE	Modeling and Analysis of Real Time and Embedded systems

MBD	Model-Based Development
MDD	Model-Driven Development
MEMCONS	Modelbased EMbedded CONtrol Systems
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
MPU	Memory Protection Unit
NUMA	Non-Uniform Memory Architecture
OCL	Object Constraint Language
OEM	Original Equipment Manufacturer
OIL	OSEK Interpretation Language
OS	Operating System
OSEK	Offene Systeme für die Elektronik im Kraftfahrzeug
PWM	Puls Width Modulation
RAMS	Reliability, Availability, Maintainability, and Safety
RTE	Runtime Environment
RTOS	Real Time Operating System
RTP	Reference Technology-Platform
SAE	Society of Automotive Engineers
SAHARA	Security-Aware Hazard Analysis and Risk Assessment
SDA	Service Deterioration Analysis
SecL	Security Level
SEooC	Safety Element out of Context
SeSaMo	Security and Safety Modeling
SG	Safety Goal
SIMD	Single Instruction Multiple Data

SISD	Single Instruction Single Data
SME	Small and Medium-sized Enterprise
SMP	Symmetric Multiprocessing
SoC	State-of-Charge
SoH	State-of-Health
SoP	Start of Production
SPI	Serial Peripheral Interface
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
SuD	System under Development
SW	Software
SysML	System Modeling Language
UMA	Uniform Memory Architecture
UML	Unified Modeling Language
VFB	Virtual Function Bus
WCET	Worst Case Execution Time

1 Introduction

In 2012 12.1 million people worked directly or indirectly in the automotive sector (7.6% of EU employment) and produced 14.6 million vehicles in Europe [25]. With a 41.5 billion € investment per year the automotive sector is the EU's number one industry. Currently available premium cars implement more than 100 electronic control units (ECU) with close to 1 Gigabyte software code [24]. These systems are responsible for 25% of vehicle costs and an added value of between 40% and 75% [69].

Embedded automotive systems are estimated to be responsible for 80% of product innovation in the past 10 years. Today's information society strongly supports inter-system communication (Car2X) and also expects it in the automotive domain. Consequently, the boundaries between application domains are disappearing even faster than previously due to the replacement of traditional mechanical systems. At the same time, multi-core and many-core computing platforms are becoming available for safety-critical real-time applications. These factors call for multiple cross-domain collaborations and interactions in the face of the challenge to master the increased complexity.

Hence, the automotive industry is facing a growing gap between the technology and the level of expertise required to make best use of them. The computing platforms are becoming more and more sophisticated with concurrent computing capabilities, larger embedded memories as well as an increasing number of integrated peripherals. Additionally, the functional integration of the control strategies (e.g., transmission with combustion engine and e-drive) further raises the complexity of the resulting application.

1.1 Embedded Automotive Systems

In the late 1970s self-contained embedded systems called Engine Control Units (ECUs) entered production vehicles. Since then, such computer systems have been integrated into almost every aspect of car equipment: control throttle, transmission, brakes, passenger climate, and infotainment. This caused the term ECU to be generalized to Electronic Control Units. Additionally, the amount of software in such embedded systems has grown to close to 100 million lines of code(LoC)[1]. In comparison the F-35 Joint Strike Fighter will require about 5.7 million LoC and the Boeing 787 Dreamliner about 6.5 million LoC.

[1] http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

These ECUs are used for energy saving and emission lowering purposes, passengers comfort and safety systems. Automotive embedded systems are connected with several in-vehicle networks (such as CAN or FlexRay), have to conform to high reliability and safety requirements, strict real-time constraints and need to withstand severe environmental conditions and production cost restrictions. Current innovation challenges in the domain are related to novel computing paradigms (such as multi-core systems), system interaction and cooperation with the Web (such as Car2Environment), and extra-functional system dependability constraints (such as safety, IT security, and reliability). These features require a fine grasp of cross-domain knowledge and pose additional challenges for the development. Additionally the classical 'separation of concern' approach of the automotive industry has to be reconsidered and an approach found which provokes seamless cross-fertilization across tool, team, and domain skills.

1.2 Thesis Background

This thesis is carried out in cooperation with the industrial partner AVL List GmbH and its Powertrain Engineering branch. AVL List is the world's largest independent company for the development of powertrain systems with internal combustion engines as well as instrumentation and test systems. AVL Powertrain Engineering (PTE) is an expert partner of the global automotive and mobility industry for the development of innovative powertrain systems. In this function AVL PTE offers various engineering services, training and coaching. The organizational independence of AVL allows the company to adapt to customer needs more easily and to offer project-specific tailoring and scaling of development processes and project-dependent variations of the tool landscape.

Due to these matters cross-domain expertise and an adaptable process-, method-, and tool-landscape approach is required even more. To that aim this thesis is geared towards the extension of existing model-based development approaches to enable a seamless model-based development (MBD) of automotive multi-core systems from initial development phase to final software implementation in context of ISO 26262 [40]

1.3 Problem Statement

This doctoral thesis aims at improving the model-based development of embedded automotive multi-core systems from initial system design to software implementation in order to support comprehensive dependability development. For this purpose a fundamental factor is to gain a good understanding of the complications that appear in this context and a comprehensive overview of related work done in the domain.

1.3.1 Seamless Model-based Development

One of the most relevant trends in embedded software engineering is the move towards more abstraction and thus the ability to better manage complexity throughout the development [24]. In model-based designs, a model of the application is used to represent complex designs at higher levels of abstraction; showing only specific pieces of information for the purpose of understanding and analyzing large designs. The model forms the basis for further activities, such as verification, code generation, or model transformation. Model-based development approaches support design space exploration and are less error prone and more efficient, in spite of the constant growth in complexity of embedded systems [64]. Model-based development alleviates the issue of inherent complexity and is the most promising approach to interdisciplinary development [20]; but in the case of embedded system development, seamless cooperation between the domains involved and development experts is a core challenge. The challenge of enabling a seamless integration of models into model-chains is still an open issue [76, 56, 59]. Often, different specialized models for specific aspects are used at different development stages at varying abstraction levels. Traceability between these different models is commonly established via manual linking due to process and tooling gaps. Various different specialized models are used for specific aspects at different development stages with varying abstraction levels and traceability between these different models is established via manual linking. Tool support for automotive engineering development is still organized as a patchwork of heterogeneous tools and formalisms [15]. The most important topic that needs to be dealt with is probably the gap between system architecture and software architecture. 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 lack in synthesizability. On the other hand, special-purpose modeling languages (such as C, Assembler, Matlab, Simulink, ASCET) are optimized for fine granular design and are less efficient in high-level design.

1.3.2 Safety-Critical Multi-Core Systems

The introduction of multi-core computing platforms aims at providing more computing resources and additional interfaces to answer the needs of new automotive control strategies with respect to computing performances and connectivity (e.g. connected vehicle, hybrid powertrains). At the same time, the parallel execution, resulting resources and timing conflicts require a paradigm change to the embedded software. So far concurrency and HW specifics (shared resources, interference between independent tasks) could be omitted from function developers by low level software. With modern multi-core systems the automotive industry is facing a growing gap between technology and the required level of expertise to make best use of them. The complexity of these computing platforms is very high, the related user guides are comprised of several

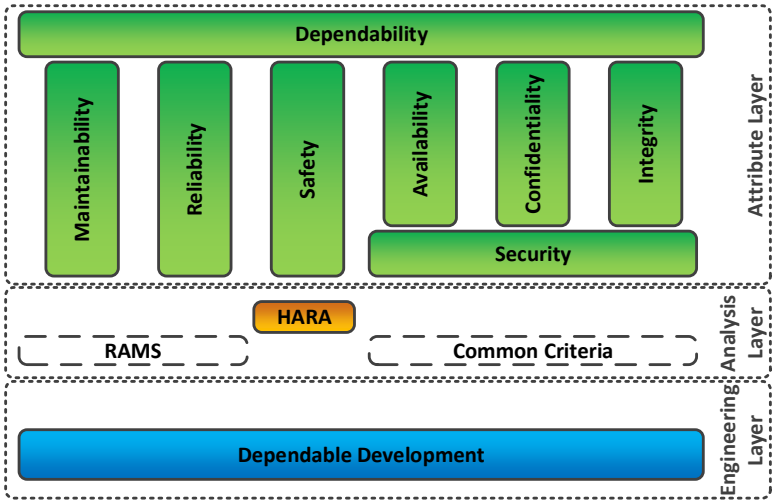


Figure 1.1: General Overview of Dependability Attributes and Analysis Methods

thousands of pages. In the context of automotive operating systems, the AUTOSAR [8] approach is following a similar trend by standardizing several tens of basic software (BSW) modules in several tens of thousands of specification pages. The automotive industry is confronted with the central question of how to migrate, optimize, and validate a given application (or set of applications) on a given computing platform with a given operating system. A knowledge transfer is required to identify the application requirements (both functional and extra-functional), perform a mapping to the SW and HW architecture, and grasp the possible impacts of concurrency and HW specifics on the software applications. Moreover, due to the timing- and safety-criticality of automotive application, lessons learned from other domains (such as consumer electronics) cannot be adopted directly and straightforwardly.

1.3.3 Comprehensive Dependability

The aim of comprehensive dependability is to provide a seamless argumentation that the developed system has achieved a certain level of maturity and can be justifiably trusted. Dependability is a superordinate concept regrouping different system attributes such as reliability, safety, security, or availability and other extra-functional requirements for modern embedded systems. These different attributes, however, might lead to different targets. Furthermore, the non-unified methods required to manage these different attributes might lead to inconsistencies, which are identified in late phases of development. Therefore, the focus is set on the need to address all these system attributes in combination (not independent from each other) and across different skill teams with the same

emphasis. In the automotive industry, dependability engineering is currently shifting its main focus from mainly mechanical reliability towards functional safety (ISO 26262 [40]) and security of the control system. While the objective is still to provide a convincing argumentation that the system can justifiably be trusted [12]; the hazards that need to be considered as well as the development methods, must be adapted accordingly.

Hence, dependability is seen according to [13] and [12], as an integrating concept that encompasses several different attributes. Figure 1.1 provides an overview of the attributes (aspects) of dependability, the analysis layer for each attribute, and a common dependable development block indicating the fact that each aspect needs to be addressed within a consistent engineering framework. Indeed, a common analysis method delivering consistent dependability targets over the different attributes is the basis to perform consistent dependability engineering during the entire product development. As can be seen in the depiction of Figure 1.1 although an analysis method solely for safety features exist, other attributes may only be analyzed by approaches not developed for the needs of the automotive domain. The common engineering basis for all dependability aspects raises the requirement for a combination of the different analysis methods and targets, thus a mutual understanding of focuses and language concepts. Dependability engineering in the automotive domain, while relying on a strong and long-term body of experience, is still an emerging trend. Although, some approaches exist for addressing safety, security and reliability in early development stages, the methods are often performed independently and cross-dependencies and mutual impacts are often not considered. Therefore, (a) a mapping of safety, security, and service oriented engineering terms, (b) a combination of the different analysis methods and targets, and (c) regulations of the mutual dependencies and the required information handover between the different dependability aspect analysis has to be found.

1.4 Publication Statistics

In the course of this doctoral thesis scientific essays have been published in several highly rated domain-specific conferences and journals. Most notably (in chronological order):

- 7th European Congress Embedded Real Time Software and Systems, Toulouse, France, 2014.
- Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France, 2015.
- SAE World Congress 2015, Detroit, Michigan (USA), 2015.
- SAE International Journal on Passenger Cars - Electronics and Electronical Systems, 2015.
- 45th Annual International Conference on Dependable Systems and Networks (DSN), Rio de Janeiro, Brazil, 2015.
- 13th IEEE International Conference on Industrial Informatics (INDIN), Cambridge, United Kingdom, 2015.

- 34th International Conference Computer Safety, Reliability, and Security (SAFECOMP 2015), Delft, The Netherlands, 2015.
- 22nd European Conference Systems, Software and Services Process Improvement (EuroSPI 2015), Ankara, Turkey, 2015.
- 8th European Congress Embedded Real Time Software and Systems, Toulouse, France, 2016.
- **under-review** SAE World Congress 2016, Detroit, Michigan (USA), 2016.
- **under-review** 53rd Design Automation Conference 2016, Austin, Texas (USA), 2016.

The paper *'Integration of Heterogeneous Tools to a Seamless Automotive Toolchain'* published at the 22nd European Conference Systems, Software and Services Process Improvement (EuroSPI 2015) received the Best Paper Award. Additionally with 21 accepted and 2 pending principal author publications this doctoral thesis is the actual record holder in terms of most accepted principal author publications of a PhD student at Institute of Technical Informatics, Graz University of Technology.

1.5 Thesis Organization

The structure of this dissertation is organized as follows. Chapter 2 provides an overview of related work done in the automotive domain and approaches dealing with open issues stated in the problem statement (Section 1.3). The approach developed for this thesis is described in more detail in chapter 3 and applied in chapter 4 for an automotive use-case. This chapter provides an overview of a reduced SW prototype of an automotive battery management system (BMS) for which the tools and methods defined during the thesis have been applied for development. The aim of this chapter is to provide evidence of the bidirectional traceability features (a.k.a. 'Vertikaler Schnitt') of the MBD toolchain with focus on functional safety and multi-core constraints. The use-case is only illustrative material, does not represent either an exhaustive nor a commercially sensitive project, and shall only provide evidence for the benefits of the established approach.

2 Related Work

This chapter reviews the works related to the thesis problem statement (1.3) in the given context (1.2). The first section 2.1 reviews research projects in the automotive domain, their focus and goals, and their shortcomings within the given work context. Further sections concentrate on model-based development (2.2) and safety-critical system development (2.3) in particular. The aim of this chapter is not to be exhaustive, but rather to categorize existing solutions, and to analyze challenges that still exist within the domain.

2.1 Research Project in the Automotive Domain

The following projects have been performed with relation to the automotive domain or have an automotive use-case. Additionally, these projects have, due to their timing and similarity of research focus, an effect on the work of this doctoral thesis.

2.1.1 CESAR Project

The CESAR project[1] tries to harmonize the data structures by defining a common meta-model (CMM). CESAR aims at achieving a common understanding and an agreement on concepts for interoperability and tool integration for safety-critical embedded systems development between industrial and academic players in the automotive, aerospace, automation and rail domains. Moreover CESAR aims at a flexible approach where both, solutions from vendors and SME as well as open-source solutions can compete in offering best-in-class solutions for particular stages of these processes. The main focus of the proposed tool-chains in CESAR is related to systems and safety engineering. The introduced multi-domain approach, European cross-sectoral standard reference technology platform (RTP), provides meta-models and methods for this purpose.

Distinctive Feature of the Thesis For less abstract development phases the RTP needs to be more specific and refined to tighter couple inter-operations between different tools. The multi-domain and common meta-model basis is not concrete enough to support the features specifically required for the aims of this work. Furthermore, the integration of analysis of the extra-functional aspects into an integrated design for the entire system focuses solely on safety aspects and disregards other extra-functional aspects.

[1] http://www.cesarproject.eu/

2.1.2 SPES_XT

The project SPES_XT[2] focuses, among other factors, on methodology and integration of development tools within a seamless tool-chain and the deployment of software over different control units. The prototypical tool-chain implementation is based on the eclipse modeling framework model (EMF) and the CESAR reference platform (RTP). The SPES MF takes an artifact-oriented view on the development of embedded software and bridges the gap between system and software engineering processes. This means the modeling framework defines structures, content and concepts used in artifacts rather than prescribing processes and methods. The related basic activities supported by the SPES MF framework conform the system engineering process standard ISO/IEC 15288 and the software engineering process standard ISO/IEC 12207. Consequently, the project does not specifically focus on the automotive domain.

Distinctive Feature of the Thesis Other than the SPES_XT project, this thesis is solely focused on the automotive domain and therefore the evolved methodology and tool implementation is more specialized to the needs of the automotive domain. The topics of extra-functional system wide features (e.g. safety and security) and automotive multi-core systems are evaluated in more detailed in this project.

2.1.3 AMALTHEA project

The AMALTHEA project[3] began after the thesis related MEMCONS project and the focus is on development of an open source development platform for engineering embedded multi- and many-core software systems with common data models and interfaces. Specifics of the project include support for multi-core systems combined with AUTOSAR compatibility and product-line engineering. The resulting tool platform is distributed under an Eclipse public license.

Distinctive Feature of the Thesis Although the project began after the related MEMCONS project, the analog project goals highlight the domain's need for such a seamless model-based development approach. In distinction to the AMALTHEA project this work focuses on the application of the approach within a specific process- and tool-landscape of the industrial partner, also the applicability of the approach for non-AUTOSAR tools, and additionally considers the evaluation and support of other extra-functional features (such as security).

[2] http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html
[3] http://amalthea-project.org/

2.1.4 parMERASA Project

The parMERASA[4] project focuses on the parallelization of hard real-time programs in avionics, automotive, and construction machinery. The goal of the project's approach is to facilitate the use of multi-core processors in hard real-time systems. To meet this goal, the project has to overcome the problems of existing timing analysis approaches, which only cover sequential program execution. Therefore, the innovations created specifically for the project include, among others, parallelization techniques for safety-critical applications, timing analyzable parallel design patterns, and verification and profiling tools extensions. Furthermore, a software engineering approach which is developed to ease sequential to parallel program transformation and the development of verification and profiling tools.

Distinctive Feature of the Thesis Other than the parMERASA project this doctoral thesis will not focus mainly on parallelization techniques for safety-critical applications. The engineering approach developed in this thesis focuses on the seamless integration of tools, methods, and processes to support the development of safety-critical multi-core systems in the automotive domain.

2.1.5 Safe Project

The SAFE project[5] objective is to enhance methods for defining safety goals and define development processes that comply with the new ISO26262 standard for functional safety in automotive electrical and electronic systems. Traceability of safety requirements through the whole lifecycle has been identified as an essential need of the project. Therefore, the project defines a meta-model and tool platform to enhance the collaboration of automotive companies involved in the development process. The focus of this project it to extend AUTOSAR architectural models in order to support ISO 26262 product development at concept phase (part 3 of ISO 26262).

Distinctive Feature of the Thesis The difference with this project is that the focus is not on collaboration between automotive companies or on the extension of the AUTOSAR architectural model. In contrast, this thesis focuses on the improvement of the information interchange continuity of architectural designs from system development level to software development level and an approach that seamlessly combines the development tools involved. This merges the heterogeneous tools required for development of automotive safety-critical multi-core systems to support seamless information interchange across tool boundaries.

[4] http://www.parmerasa.eu/
[5] http://safe-project.eu/

2.1.6 Evita Project

The objective of EVITA[6] is to design, verify, and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data is protected from compromise. Vehicle-to-vehicle and vehicle-to-infrastructure communication have been identified as a means for decreasing the number of fatal traffic accidents. While these functions are promoted as a start to a new era of traffic safety, additional security requirements need to be considered to prevent attacks on these systems. Therefore, the objective of the EVITA project is to design, verify, and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data is protected against compromise when transferred inside a vehicle.

Distinctive Feature of the Thesis While vehicle-to-X communication is not part of this thesis, the described security function development and security related requirement modeling using UML affect the evaluation- and support-concepts of other extra-functional features.

2.1.7 Maenad Project

The Model-based analysis and engineering of novel architectures for dependable electric vehicles (MAENAD) project[7] focuses on the extension of EAST-ADL with advanced capabilities to facilitate development of dependable, efficient and affordable electric vehicles (EV). To support the automotive safety standard ISO 26262 and model-based prediction of dependability and performance attributes of EV, MAENAD aims to exploit and further develop the present state of the art in model-based design, assessment and optimization technologies. In addition, the MAENAD project proposes an overall design methodology for EV.

Distinctive Feature of the Thesis In difference to the MEANAD project, the work presented here focuses on automated techniques for transferring information between special purpose software development tools and the seamless integration of tools, methods, and processes to support the development of safety-critical multi-core systems in the automotive domain.

2.1.8 SeSaMo Project

The SeSaMo project[8] addresses the problems that arise with the convergence of safety and security in embedded systems at architectural level. The project claims the ab-

[6] http://www.evita-project.org/
[7] http://www.maenad.eu/
[8] http://sesamo-project.eu/

sence of a rigorous theoretical and practical understanding of safety and security feature interaction as the root cause for problems and develops a component-oriented design methodology based upon model-driven technology which jointly addresses safety and security aspects and their mutual relations in multiple domains (e.g., avionics, transportation, industry control). Key elements of the SeSaMo approach that are presented are an overall design methodology and tool-chain utilizing the constructive elements and integrated analysis procedures to ensure that safety and security are intrinsic characteristics of the system. The SeSaMo methodology constructs structured assurance cases for communicating and building confidence in the safety and security properties of a system. The model-based development technology of SeSaMo is based on a SysML profile.

Distinctive Feature of the Thesis This doctoral thesis work goes beyond the objectives of the SeSaMo project and does not focus on the ability to model safety and security features and interaction but on the unification of supporting the development of safety-critical multi-core system and comprehensive dependability.

2.2 Model-based Development

One of the most relevant trends in embedded software engineering is the move towards more abstraction and thus the ability to better manage complexity throughout the development[24]. Therefore, Model-based development is a very important trend in the design of embedded system. In model-based designs, a model of the application is used to represent complex designs at higher levels of abstraction; showing only dedicated pieces of information for the purpose of understanding and analyzing large designs. In Pretschner's roadmap[55] the authors highlight the benefits of a seamless model-based development tool-chain for automotive software engineering. The authors also claim model-based development the best approach for managing the large amount of information and complexity of modern embedded systems with safety constraints. The three fundamental ingredients needed to achieve seamless model-based development are identified as: (a) a comprehensive modeling theory as a semantic domain for formal definition of models, (b) an integrated architectural model that describes the detailed structure and a process to develop such a model, and (c) an integrated model engineering environment which guarantees seamless tool support.

Nevertheless, the challenge of enabling a seamless integration of models into model-chains is still an open issue [76, 56, 59]. 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. [34] 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 and traceability between these different models is established via manually linking. The Automotive SPICE[75] process reference model demands properties such as traceability and defines development stages as well as resulting artifacts, but does not specify how the realization of these properties or artifacts can be achieved.

The most important topic to deal with is the gap between system architecture and software architecture - especially whilst considering component-based approaches such as UML and SysML for system architecture description and AUTOSAR for SW architecture description[34]. The authors also highlight the crucial importance of powerful tool support for model-driven software engineering. Boldt[17] 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 into company process flows.

The tool support for automotive engineering development is still organized as a patchwork of heterogeneous tools and formalisms[17]. On the one hand, general-purpose modeling languages (such as UML or SysML) provide modeling power suitable to capture system wide constraints and behavior, but lack in synthesizability. On the other hand, special-purpose modeling languages (such as C, Assembler, Matlab, Simulink, ASCET) are optimized for fine granular design and are less efficient in high-level design. Therefore, the EAST-ADL approach represents an architecture description language using AUTOSAR elements to represent the software implementation layer of embedded systems[17].

To manage the complexity of the embedded software functions and ensure the fulfillment of the rigorous extra-functional requirements, model-driven development (MDD) is one of the most promising approaches that has emerged over the last decade. In this context there are tow main strategies that can be followed [54].

Domain Specific Language MBD In the first variant designers create a new domain specific language (DSL) for modeling the domain of interest. DSL approaches are optimally suited for the problem at hand, but the main drawback of this approach is the additional efforts required to obtain an integrated and consistent tool-chain due to the non-standardized modeling specification and missing interoperability infrastructure.

General Purpose Meta-Model MBD The other main variant relies on the extension of existing general purpose meta-models (such as UML meta-model) for the specific domain of interest. In the context of UML this mechanism is called a profile. Each extension and adaptation of the domain specifics is formally captured by a stereotype which can be associated with properties and constraints. With this approach various semi-automatic tools can access the information captured by the profile and existing meta-models can be reused and specialized. Therefore, this approach enables the possibility of selectively

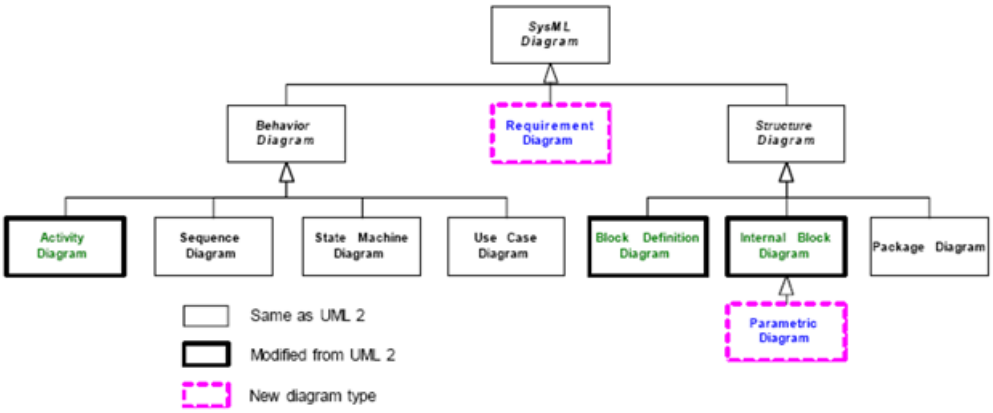


Figure 2.1: Overview of SysML Diagram Types [32]

choosing the most beneficial items from what is available (cherry-picking). Excerpts and brief descriptions of the most well-known representatives of the group of UML variants are given in the next few paragraphs.

2.2.1 System Modeling Language (SysML)

The SysML profile is not limited to software-centric systems development and addresses a broader concept of system architectures and interactions with the environment (system and system-of-system development). Figure 2.1 shows an overview of available diagram types in SysML framework. SysML improvements compared to UML are:

- more flexible semantic and new diagram types
- the inclusion of requirements engineering
- easier to learn and apply due to smaller language base
- variants of allocations
- constructs to support models, views, and viewpoints

The work of Quadri and Sadovykh [56] presents a model-driven engineering approach which aims to develop novel model-driven techniques and new tools supporting design, validation, and simulation. These authors defined profiles using a subset of UML and SysML for their approach and mentioned the usage of effective design tools and methodologies as crucial in order to be capable of managing complex real-time embedded systems.

Giese et al. [30] 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.

2.2.2 MARTE System Profile

The work of Quadri et al. [56] presents a model-driven engineering approach using a subset of UML profiles MARTE and SysML used to present a case study of a collision avoidance system. The MARTE system profile is designed to support real-time embedded system development and unify various existing approaches of the real-time system community. MARTE structures a set of sub-profiles for design and analysis support and the expression of timing characteristics. It provides facilities to annotate models with information required to perform specific performance and schedulability analysis. The main benefits of this profile are:

- a common way of modeling both hardware and software aspects of real-time systems
- enabling interoperability between development tools
- models that may be used to make quantitative predictions on timing

2.2.3 EAST ADL

The EAST-ADL UML profile is a language implementation that provides an architecture description for electronic architectures and software technologies for the automotive domain. The EAST-ADL2 profile ensures:

- focus on the description of the function
- support of layered development processes
- usage of domain-related vocabulary

Figure 2.2 depicts the implementation layers of EAST-ADL2 and their relation to the AUTOSAR concept. Chen et al. [21] describe how EAST-ADL supports safety requirements, faults, failures, hazards, and safety constraints in the context of ISO26262. Their approach, project SPEEDS, aims at providing support for modeling and analysis of complex embedded systems through usage of formal analysis tools. Also, more recently the MAENAD Project has focussed on design methodologies for electric vehicles based on EAST-ADL2 language.

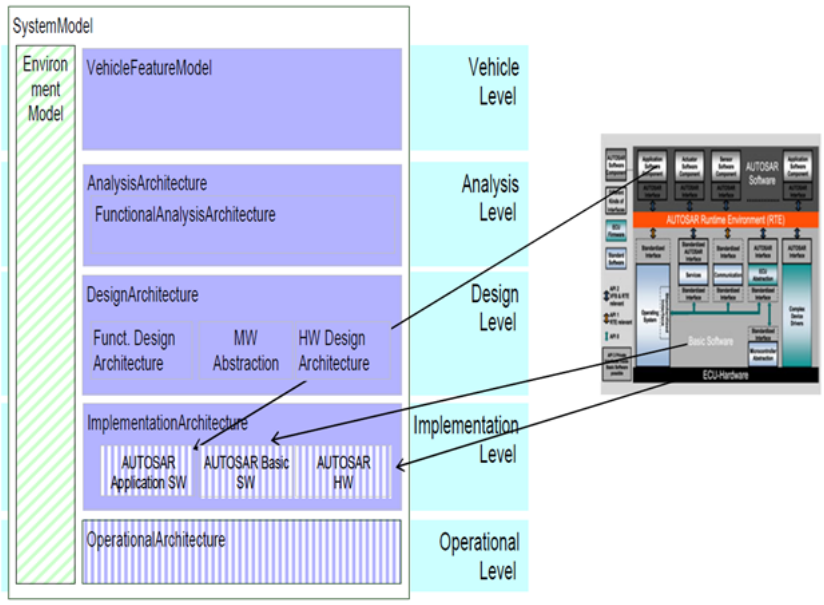


Figure 2.2: Overview of EAST-ADL2 Abstraction Layers and their Relation to AUTOSAR [22]

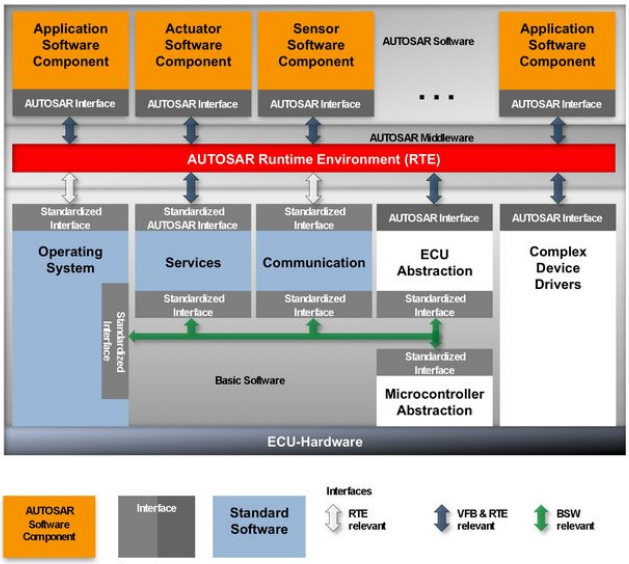


Figure 2.3: Overview of the AUTOSAR ECU SW Architecture[8]

2.2.4 AUTOSAR Meta-Model

AUTOSAR [8] is an acronym for AUTomotive Open System ARchitecture and stands for a partnership of OEM, tiers, tool- , and microcontroller developers. This initiative focuses on an open system architecture for the needs of automotive software engineering. This system architecture aims to simplify embedded software development, exchange of software components, and hardware independent software development. AUTOSAR defines a software architecture, a common format for interfaces, and a methodology for software development to ensure scalability, exchangeability of SW components, and reuse of SW parts. The first release of AUTOSAR was published 2005 and since then further improved to release 4.2. With release 4.0 in 2011 features of multi-core controllers and safety related software issues have been taken into account. Nevertheless, the focus of the AUTOSAR UML profile is set on software development, layered software development approaches, and ensuring the independence of the application software from hardware until the late stages of development. The description of hardware and more abstract layers of system development as well as the separation of different views is not the focus of this approach, as can be seen in Figure 2.2. Figure 2.3 shows the AUTOSAR common software infrastructure for automotive systems based on standardized interfaces for the different layers to ensure modularity, scalability, transferability and re-usability of approach.

The AUTOSAR consortium [8] and their AUTOSAR methodology was founded to provide standardized and clearly defined interfaces between different software components. The AUTOSAR approach features three different classes of implementation (ICC - implementation conformance class). AUTOSAR ICC1 approach clearly saves time in terms of not requiring additional familiarization with usually very complex and time-consuming AUTOSAR tools, compared to the full AUTOSAR approach (ICC3). The ICC1 approach does not take advantage of the AUTOSAR benefits of the full AUTOSAR tool-chain supporting tools for RTE configuration and communication interfaces, but standardized component interfaces for exchange of data between ASW and BSW and therefore features the separation of application specific and hardware specific software parts (like native non-AUTOSAR development). The ICC1 approach 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 work of Scheickl et al. [65] presents model-based software engineering to ensure the fulfillment of timing constraints for AUTOSAR systems. The work states that appropriate tools for specification of AUTOSAR timing requirements are still missing and proposes a set of tools to support seamless timing specification, analysis and verification.

In [44], the authors describe a framework for a seamless configuration process for the development of automotive embedded software. The framework is also based on

AUTOSAR, which defines architecture, methodology, and application interfaces. The steps through this configuration process are established by a system configuration and ECU configuration.

Boldt [17] 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 into company process flows.

The work of Kum et al. [43] describes an approach for an AUTOSAR migration of existing automotive software. Generally, automotive embedded application software is closely coupled with the underlying hardware and basic software and the existing interfaces between these layers are not clearly defined and standardized. The authors highlight the benefits of separating the application software and the basic software and exhibit a way towards configuration instead of coding embedded software manually. The manual generation of software includes time consuming and error-prone tasks. Therefore the automatic generation of automotive embedded software and the according configuration of the embedded systems improve the quality as well as the re-usability.

A work depicting the influence of the AUTOSAR methodology on software development tool-chains is presented in [77]. This tool framework, named ARTOP (AUTOSAR Tool Platform), enables a seamless tool-chain from requirements to implementation. ARTOP is an infrastructure platform that provides features for the development of tools used for the configuration of AUTOSAR systems. These features are base functionalities that are required by different AUTOSAR tool implementations.

Nevertheless, non-AUTOSAR tools and system developments are still very common and in terms of safety-critical development and multi-core system development only more recent AUTOSAR releases (since $R4.0$) support solutions.

2.3 Safety-Critical Systems

Safety standards, such as the road vehicles – functional safety norm ISO 26262 [40] and its basic norm IEC 61508 [37] present requirements and guidance for safety-critical system development. A guide to the functional safety of automotive systems according to ISO 26262 can be found in [29, 16] or in the SafEUr functional safety manager training [63]. The AUTOSAR development cooperation also focuses on safety in the technical safety concept report [9] produced by this group.

The work of Gashi et al. [28] focuses on redundancy and diversity and their effects on the safety and security of embedded systems. This work is part of SeSaMo (Security and Safety Modeling for Embedded Systems) project, which focuses on synergies and trade-offs between security and safety through concrete use-cases.

Born et al. [18] give a general overview of ISO26262 and evaluate a document-centric development approach versus the model-based development approach preferred by ISO26262. The work claims document-centric approaches are fundamentally flawed, because this ap-

proach usually distributes information over multiple documents and reuse is only done via copy-and-paste techniques. The document-centric approach is difficult to manage and understand and the required traceability may only be established through manual cross-referencing of all documents. In contrast to this, model-based development stores all information in a central model, non redundantly, and supports generation and import of documents based upon defined templates. The authors claim that model-based approaches take longer time to introduce and also some additional effort for settling of the tool-chain, but this approach has some major benefits for ISO26262 compliant development.

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. [46]. The integration of SysML models for the development of the ECU safety concept ensures efficient design changes, and immediate awareness of functional safety needs. The paper evaluates key success factors of MBD in comparison to legacy development processes in the field of safety-critical automotive systems.

The work of Ebert [23] highlights three key components of sustainable safety engineering in automotive systems: (a) system-oriented development, (b) safety methods closely coupled to engineering, and (c) process maturity. Ebert further mentions functional safety needs and that these are to be seen as a critical product liability issue with all the consequences this implies and also that engineers need to understand the safety needs at all levels of the development process.

In the issue of improving processes or workflows, especially those which deal with cross-domains affairs (such as the traceability of architectural designs from system development level to software development level), a comprehensive understanding of related processes, methods, and tools is required. The work of Sechser [70] describes experiences gained when combining two different process worlds in the automotive domain. The author mentions, among other points, the need for a common language and process architecture.

Liggesmeyer and Trapp[45] analyzed trends in embedded software engineering and concluded that the increasing complexity of functional and especially extra-functional requirements (such as safety, security, or dependability demands) of embedded systems call for new software development approaches. Embedded software is rarely a standalone product without interaction with the physical world (such as the majority of IT applications) and must therefore consider environmental conditions as well as related mechanics, electrics, and electronics.

2.4 Dependability Attributes

In addition to DO-178C [71] for aerospace software safety, ARP4754 [62] gives guidance for system level development and defines steps for adequate refinement and implemen-

tation of requirements. Safety assessment techniques, such as failure mode and effects analysis (FMEA), and functional hazard assessment (FHA) among others, are specified by ARP4761 [61]. Security concerns in the avionic domain are tackled e.g. by common criteria [74] approach and ED202 [26] specification.

Reliability and availability standards mainly originate from railways and the armaments industry. DIN EN 50126 [6] focuses on specification and demonstration of reliability, availability, maintainability, and safety (RAMS) of the railway system. In 1980 the US Department of Defense defined a standard reliability program for systems and equipment development and production (MIL-STD-785B [2]). Additionally, the military handbooks 338B [5] and 781A [4] assist with guidelines for electronic reliability design and reliability test methods, plans and environment for engineering. Nevertheless, most standards and guidelines, like the military handbook 217F [3] and the technical report TR 62380 [35] rely on reliability prediction of electronic equipment based on mathematical reliability models of the system components. Only a few works focus on quantification of dependability features (other than safety or security) in early stages of the development process.

Most reliability measures and work focus on estimation of probabilities and stochastic processes. All of this work requires detailed design information of the SuD and is therefore not applicable for an early design phase evaluation. Nevertheless, the process improvement techniques of Six Sigma [36], [73] aims at improving the quality of process outputs by identifying and removing the causes of defects (errors). The Six Sigma approach uses a set of quality management methods, including statistical methods. One of the Six Sigma methods, CTQ trees (critical-to-quality trees) are the key measurable characteristics of a product or process and are used to decompose broad customer requirements into more easily quantified elements. These elements are then converted to measurable terms, this approach is also the basis for Service Deterioration Analysis [49] described later in this section.

Some recent publications in the automotive domain also focus on security in automotive systems. On the one hand, the work of Schmidt et al. [66] presents a security analysis approach to identify and prioritize security issues, but only provides an analysis approach for networked connectivity. The work of Ward et al. [78], on the other hand, also mentions a risk assessment method for security risk in the automotive domain termed threat analysis and risk assessment, based on the HARA.

The works of Roth et al. [60] and Steiner et al. [72] also deal with safety and security analysis, but focus on state/event fault trees for modeling of the system under development. Schmittner et al. [67] presents a failure mode and failure effect model for safety and security cause-effect analysis. This work categorizes threats by using the STRIDE threat model, but focusing on IEC60812 conform FMEA.

Finally, the STRIDE threat model approach [50] developed by the Microsoft Corporation can be used to expose security design flaws. This approach uses a technique termed threat modeling. With this approach the system design is reviewed in a methodical way,

which makes it applicable for integration into the HARA approach. Threat models, like the STRIDE approach, may often not prove that a given design is secure, but they help to learn from mistakes and avoid repeating them, which is another commonality with HARA in the safety domain.

There are approaches addressing safety, security, and reliability in early development stages. However, these methods are often performed independently and cross-dependencies and mutual impacts are often not considered. The most commonly used analysis methods are hazard analysis and risk assessment (HARA), failure mode and effects analysis (FMEA), and fault tree analysis (FTA) in different variations for safety, security, or reliability/availability. FMEA [38] [1] and FTA [39] variation are common for all system features. Nevertheless in the automotive domain a method is standardized for initial design assessment exclusively for safety analysis (HARA [40]).

3 Proposed Solution

The survey of Albers and Zingel[7] mentions the advantages of using formal models to specify a complex technical system as manifold (e.g. fewer inconsistencies, less redundancies and concurrently, the basis for clear communication and sustainable documentation). The authors claim that models can help to force systemic thinking, but also mention that trans-disciplinary system architecture interaction is still an unsolved issue in industrial practice. Moreover, existing organizational structures are frequently incompatible with trans-disciplinary systems engineering and their literature review shows that the change from traditional document-based towards a model-based development approach has still not taken place. Insufficient usability of modeling tools, missing modeling methods or guidelines, a high learning effort, and insufficient model2model-transformation further hinder the model-based development approaches.

As mentioned in the related work section (2), the approach of extending the existing general purpose meta-models (such as UML meta-model) for the specific domain of interest inherits the benefit of various semi-automatic tool being able to access the information captured by the profile. It also means that existing meta-models can be reused and specialized. Therefore, such an approach enables the possibility of selectively choosing the most beneficial items from what is available and ensuring a lean meta-model and explicit representation of the required modeling artifacts. To fully profit from MBD benefits, a methodology has been specifically established for automotive powertrain development at system level in the environment of AVL-PTE.

This doctoral thesis approach is based on this AVL-PTE MBD environment and extends its applicability to the needs of safety-related embedded software development to achieve a holistic system description which is capable including structural and functional aspects of one system. Moreover, the approach is geared towards improving the comprehensive dependability argumentation of automotive multi-core systems based on model-based development. For the purpose of improving the comprehensive dependability argumentation of automotive multi-core system development the strategy of this doctoral thesis is to tackle individual sub-problems. Figure 3.1 shows a depiction of this concept in GSN notation.

The main goal of this doctoral thesis is thus partitioned into three sub-goals. First, challenges which appear when developing a dependable multi-core system especially in automotive domain to be have to be revealed and analyzed (see Figure 3.1 - goal 'Reveal Domain Challenges'). Second, the strategies for the migration to multi-core systems have

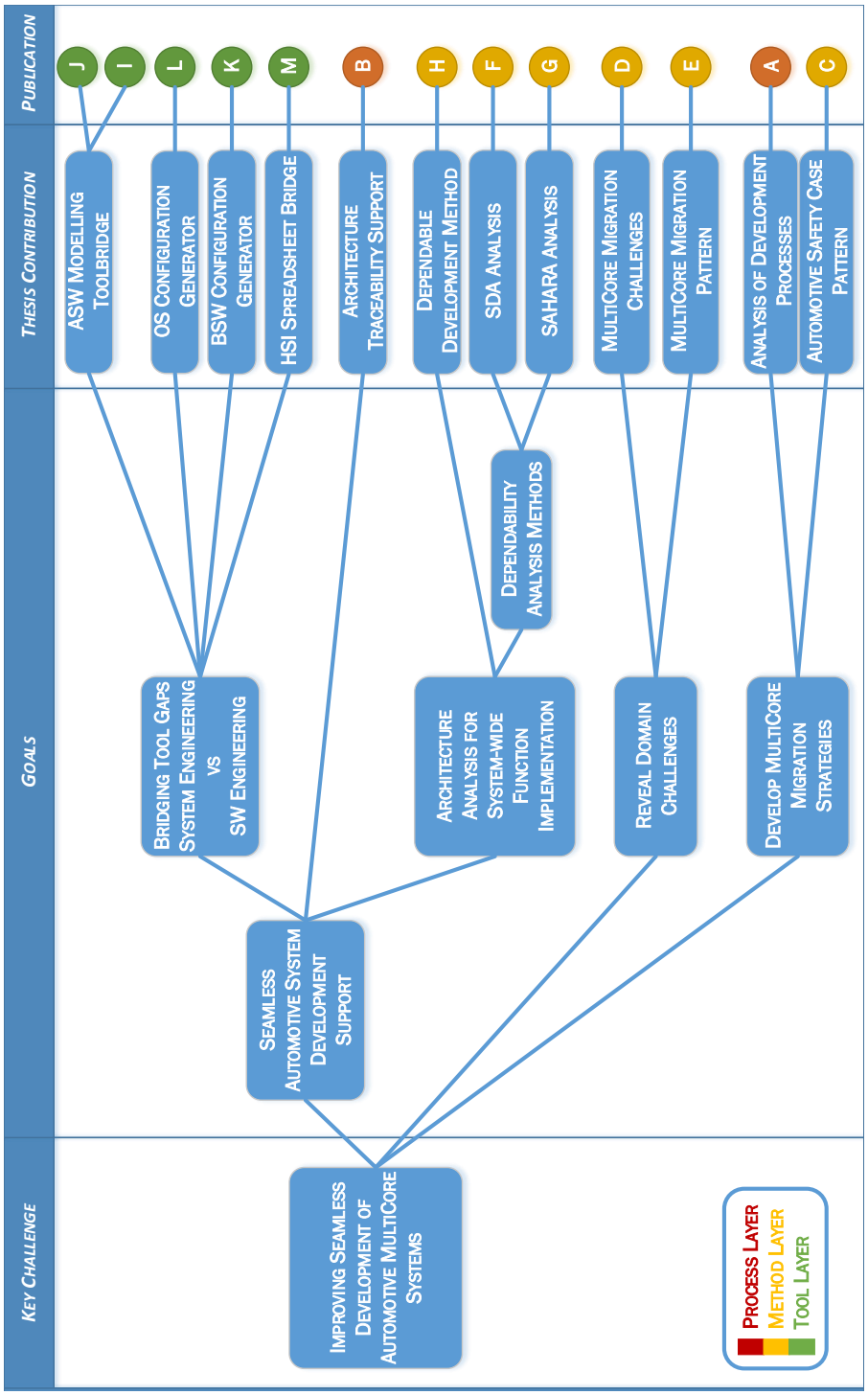


Figure 3.1: Overview of the Doctoral Thesis Concept Strategy

to be developed (see Figure 3.1 - goal 'Develop MultiCore Migration Strategies'). Finally, the development of such a system needs to be supported by an adequate development framework (see Figure 3.1 - goal 'Seamless Automotive System Development Support').

This goal is further supported via two sub-goals, the adequate analysis of the architecture design for system-wide function implementation (with its sub-goal existence of dependability analysis methods) and a merging of the heterogeneous development tools (see Figure 3.1 - goals 'Architecture Analysis for System-wide Function Implementation', 'Dependability Analysis Methods', and 'Bridging Tool Gaps System Engineering vs SW Engineering').

The major contributions for success are listed on the contribution layer of Figure 3.1 and supported via the related publications (more details regarding publications related to this thesis can be found in Section 6). The contributions itemized in the contribution layer contribute to the issues encountered in the presented approach in the tool, method, and process layer. In the following sections the contributions of this doctoral thesis approach are described and differentiated by their layer of contribution (tool, method, and process layer); these layers have been identified by [7, 23] as crucial for the industrialization of engineering approaches.

3.1 Process Layer related Contribution

The automotive domain's process landscape is already very mature and well defined. Automotive processes have to comply to either Automotive SPICE[75] or CMMI assessment models. Domain specific standards, such as ISO 26262[40] or AUTOSAR[8], provide guidance and well-defined safety-lifecycles

Additionally, numerous best practices and company specific proceedings are established and in place.

Hence, this doctoral thesis does not aim to redefine or introduce completely new process structures, but analyze the given process landscape and suggest adaptations for improvement in terms of multi-core system and system-wide feature development.

3.1.1 Analysis of Development Processes

In the publication **Paper A** (depicted as publication A in Figure 3.1 - publication layer) patterns of concurrent workflows in embedded system development have been identified. These patterns can be used to identify dependencies and consequences of parallel workflows and thus foster seamless MDD paradigm along the development life cycle.

During the model-driven development of embedded systems it is often found that numerous model types from the same system under development exist spread over different stages of the development process and different development tools in use. In these cases special attention needs to be paid to keep the dependent models consistent. Each model

transformation generates potential sources for ambiguous mapping and often dependencies between the models are hard to identify.

The intention of this work is to identify and update dependencies between concurrent models and thereby set up the foundation for tool integration into a holistic tool-chain. In the case of solely uni-directional dependability and independent workflows (e.g. generation of interim status model-based documentation), the Forward Update Dependability Relation pattern shall be applied. If feedback of information from one model to another is required after some refinement activity(e.g. the updating of the SW architecture model with SW function models), the Backward Update Dependability Relation pattern shall be applied. Finally, the Bidirectional Update Dependability Relation pattern shall be used if a full concurrent development of mutually dependent models is required (e.g. concurrent SW and HW development). The identification of these relationships serve as the basis for a bridging of the ascetic MDD tools to a seamless tool-chain.

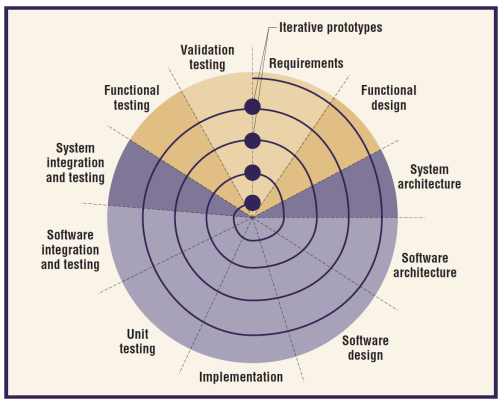


Figure 3.2: Iterative Model-Driven Development Process for Embedded Systems [45]

Figure 3.2 shows all phases of such an MDD development life cycle. As can be seen in this figure, model-driven development iteratively updates the model over time. This process lets developers produce partial implementations after each iteration. The figure also indicates a potential pitfall of such an iterative model-driven development approach. Frequently, manifold model types of the same system under development exist across different stages of the development process and different development tools in use. In the case of such a concurrent development of dependent development models special attention needs to be paid to keep the dependent models consistent. Each transformation step implies potential sources for ambiguous mapping and a common model as a single source of information is rather unusual or too complex for application. For the course of this document this process is stretched over time, which leads to a corkscrew like representation of the three identified patterns in Figure 3.3.

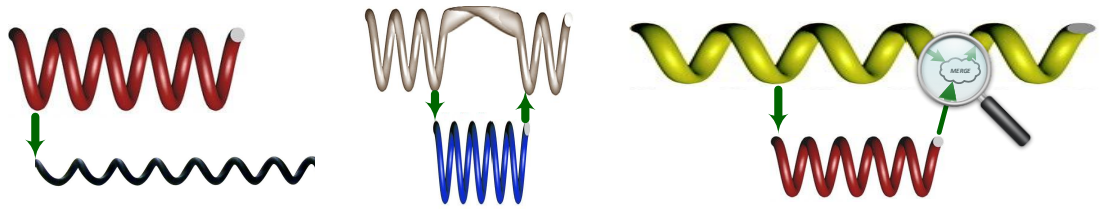


Figure 3.3: Depiction of: (a) Forward Update Dependability Relation, (b) Backward Update Dependability Relation, and (c) Bidirectional Update Dependability Relation

3.1.2 Architecture Traceability Support

The aim of this contribution and the publication **Paper B** (depicted as publication B in Figure 3.1 - publication layer) is to improve the information interchange continuity of architectural designs from system development level to software development level. Consequently, this work focuses on the analysis of information interchange of architectural designs within the focus of Automotive SPICE [75] ENG.3/ ENG.5 respectively ISO 26262 [40] 4-7/6-7. Furthermore, special focus is given to safety-critical multi-core system development, due to the higher complexity of such systems and limited availability of supporting methods and tools. The aim of this work is to analyze the constraints of merging the heterogeneous tools required for development of automotive safety-critical multi-core systems especially during the transition phase from system to SW development.

3.2 Method Layer related Contribution

The methodical support of system architectural design and refinement of this design to software design has often fallen short of the mark[7]. To cope with new domain challenges (such as holistic dependability argumentation), novel technologies (such as automotive multi-core systems), and to gain benefits from tools and setting up toolchains the fundamental concepts and methods need to be scrutinized. This section therefore involves contributions related to analysis concepts, development patterns, and procedures.

3.2.1 Automotive Safety Case Pattern

The safety standard ISO 26262 [40] has been established to provide guidance during the development of safety-critical systems. It provides a well-defined safety lifecycle based on hazard identification and mitigation, and defines a long list of work-products to be

generated. One of these required work-products is the safety case. The challenge of the safety case is to provide evidence of consistency of the different work-products during product development. Thus, the establishment of such a safety case could be a tedious task and could result in extra workload with only limited additional benefit.

Besides this, safety cases give information about implicit domain knowledge and convincingly and concisely argue that a system meets its safety requirements in a coherent and reproducible way. Therefore, the analysis of patterns for safety case generation and the publication **Paper C** (depicted as publication C in Figure 3.1 - publication layer) identifies the main claims which must be satisfied to ensure system safety and the fundamental concepts which must be methodically supported. These concepts are:

Process & Product Pattern

First, establish a development process according to the domain best-practices and requirements of e.g. Automotive SPICE. Second, provide evidence of application of these certified processes in a correct manner. This separation of concerns enables easier reuse of process claims and independent certification of production processes (e.g. audits of the company process structure).

Traceability Pattern

Establish traces between each development artifact to explicitly indicate dependencies and relations. These traces can be useful in practice, due to reducing the need of inter-tool traces (e.g. reviews of consistency check protocols) and highlighting of the impact of changes made to a specific artifact.

Functional Breakdown Pattern

Further, argumentation of overall system safety for a safety case can be done via the argumentation of safety of each of the involved sub-functionalities. Thereby responsibility for safety argumentation is holistic and cannot be handed over to solely one specific sub-function domain. This implies an introduction of an additional system viewpoint (functional view illustrating vehicle-wide functions and their decomposition), which simplifies conception and development of system-wide features (such as safety).

3.2.2 Multi-Core Migration Challenges

Moore's law[51], states the doubling of computer capacity every 2 years. Nevertheless, the current development trend for computing platforms has moved from increasing the frequency of single cores to increasing the parallelism (increasing the number of cores on the same die). Multi-core and many-core technologies have a strong potential to further support the different technology domains, but simultaneously they present new

challenges. The complexity of these computing platforms is very high, the related user guides are comprised of several thousands of pages. Hence, the automotive industry is facing a growing gap between the technologies and level of expertise required to make best use of them. Therefore, the automotive industry is confronted with the central question of how to migrate, optimize, and validate a given application (or set of applications) on a given computing platform with a given operating system.

This contribution and the publication **Paper D** (depicted as publication D in Figure 3.1 - publication layer) are (1) to provide a state-of-practice survey on multi-core CPUs and operating systems for the automotive domain, and (2) based on this survey to provide guidelines for the migration of legacy SW. Finally the related challenges and opportunities for the development of high-integrity control systems on multi-cores, as a platform for dependable systems are discussed.

In particular, the following topics are addressed: architecture of the parallel computer, memory architecture, concept of processing functions, synchronization protocols, and scheduling policies. All these factors influence the achievable speedup, efficiency and dependability features (e.g. safety, reliability) of the multi-core system.

Typical side-effects which need to be focused on when migrating to multi-core systems are: (a) starvation - when a process is perpetually denied necessary resources, (b) deadlocks - when two or more processes are each waiting for the other to finish, (c) livelocks - similar to a deadlock, except that the state of the processes is in constant change in relation to one another, not progressing, (d) race conditions - output is dependent on the sequence or timing of other uncontrollable events, (e) priority inversion - high priority task is indirectly preempted by a medium priority task, (f) freedom from interference - failures of one process cannot influence other processes, (g) timing correctness - correctness of an operation depends not only upon its logical correctness, but also upon its computation time, and (h) execution order correctness - correctness of the execution sequence of cascaded tasks. This brief definition may make the effects of the following multi-core aspects more evident.

Parallel Computer Architectures

According to Flynn's taxonomy [27, 52] 3 types of multi-core computer architectures exist (fourth variant 'SISD - single instruction single data' implies only one core): *Single Instruction Multiple Data (SIMD)* - this architecture can access multiple data memory locations at once and execute specific single operations in parallel. Beneficial for an application that operates with a large volume of data (such as GPU). *Multiple Instruction Single Data (MISD)* - Multiple cores are capable of handling different instructions simultaneously for a single data stream. Exists only in theory and is not commercially available. *Multiple Instruction Multiple Data (MIMD)* - One of the most complex and most frequently used modern hardware architectures. Multiple processing elements can handle multiple independent and concurrent instructions on multiple independent data.

MIMD (and single-core SISD) architecture is only appropriate for the automotive domain. SIMD, as used for graphic processing (e.g. GPU), is not currently available for powertrain controls.

Multi-Core Suitable Memory Architectures

Two memory architectural versions are commonly used for multi-core systems, uniform memory architecture (UMA) and non-uniform memory architecture (NUMA). UMA divides the memory into blocks of unique data and allows unified access of each core to different blocks of the memory. NUMA is designed for concurrently running processors, it provides separate memory space for each core and therefore unlimited access to this local memory. Shared memory is realized by the moving of shared data between local processor memories, this increases the efforts involved, but prevents all cores from starvation. Automotive multi-core system hardware is currently based on UMA, but the NUMA approach is an improvement in terms of safety and freedom from interference and can be realized using dedicated additional hardware features (memory protection unit), if available.

Multiprocessing Models

Basically two types of processing models are common. Asymmetric Multiprocessing (ASM) which can also be used for non-multi-core operating systems (OS). For this approach one core becomes the master, responsible for OS and the other(s) slave core(s) solely runs user level applications triggered by the master OS. The Symmetric Multiprocessing (SMP) approach features OS and user level approaches on each core. All CPUs are interconnected via a bus or crossbar with access to global memory and peripherals, which requires appropriate synchronization mechanisms.

The ASM approach is frequently used as a first step introduction into multi-core systems. SMP is an intuitive approach when combining the functionalities of two separated control units into one multi-core. Therefore, both approaches are common in the automotive domain and appropriate mechanisms must be considered and applied.

Multi-Core Synchronization

Synchronization features help to synchronize tasks with shared resources and avoid deadlocks, starvation, and priority inheritance of parallel computing platforms. Synchronization primitives are features of real-time operating systems and are the building blocks of synchronization in complex systems. For this reason, they are hard to use in complex software systems, frequently make code less readable and often get lost during programming (e.g. unreleased locks, or interferences via busy waiting).

The most popular and commonly used synchronization primitives in consumer electronics multi-core systems are:

- Mutex - a special type of variable, which controls access to shared resources (can either be locked or unlocked).
- Semaphore - a special variable, used to record the number of available units of a particular resource and one of the most powerful means of synchronization.
- Event - used for synchronization of branch execution and whenever a task requires information from another task.
- Monitor - is a higher level synchronization primitive, using Mutex and semaphores to achieve synchronization. A monitor allows threads to have both mutual exclusion and the ability to wait (block) for a specific condition to become true, but it is not supported by every OS or programming language.
- Critical Section - mechanism to avoid race conditions by granting access to shared resources one task at a time.

Scheduling Policies

Scheduling plays a central role by defining the ordering of the tasks to be executed. There are two principal directions: global and partitioned schedulers. A global scheduler stores all tasks in a single queue based on their priorities and can schedule a task on any available core of the system; task migration is allowed. The partitioned scheduler by contrast, assigns tasks to cores and features several queues, depending on various task attributes (such as shared resource of task groups). Partitioned scheduling is more common in the automotive domain, although the applied scheduling policy strongly depends on the use-case. For safety-related systems hard real-time requirements are of crucial importance, even in worst-case scenarios.

3.2.3 Multi-Core Migration Pattern

For the migration of legacy software to multi-core systems the aspects of (a) migration preparation steps, (b) guidelines for migration, and (c) performance analysis after migration needs to be scrutinized. The publication **Paper E** (depicted as publication E in Figure 3.1 - publication layer) provides such guidelines for the migration of legacy SW.

Migration Preparation: Selection of the CPU and of the BSW

The first step is the identification of the proper CPU and of the proper BSW / operating system. For the automotive domain there are only a few, but very different, multi-core hardware implementations available for automotive applications. Besides the obvious criteria (e.g. availability of supporting/debugging tools, clock rates, and experiences), possible disadvantages and architectural benefits (e.g. HW implementation of MPU, peripheral bus guardians, lockstep mode cores, and HW safety features) are the contributory factors in the selection of the multi-core hardware.

Knowledge of the application is important here for understanding the required interfaces to the real world (sensors / actuators) and possible supports by the CPU peripherals. Such integration might have an important impact on the performance, e.g., when down-sampling and filtering is performed directly in HW and no longer by a low-level SW driver (requiring high-frequency interrupts). Definition of the overall architecture taking into account the peripherals embedded in HW is thus a key aspect for improving performance. Another aspect is the level of dependability (safety, security, reliability...) of the application and the respective integrity level of the computing platform (CPU and peripherals). Again, the CPUs provide mechanisms implemented in HW (e.g. lock-step, memory protection) to shift part of the integrity issues from low-level SW to HW, thus saving computing resources for the application. Here again, refinement of the architecture taking into account the mechanisms embedded within the CPU is a key aspect for improving performance.

The next important step is the selection of an adequate automotive operating system. Although only few operating system variants exist for the automotive domain, the selection of the OS and its applicability for different multiprocessing models, scheduling policies, and implemented synchronization primitives is crucial. Also memory separation and protection (spatial freedom from interference) of individual tasks must be ensured and peripheral resources must be accessed with care. Peripherals, like other shared resources, are potential pitfalls and one of the biggest bottlenecks of multi-core systems. Important aspects for the choice of the BSW and operating aspects are:

- *Maturity target for the application*: The targeted development might well have a different maturity level (e.g., demonstrator, mature prototype, SoP) and therefore make different requirements on the BSW layers in respect to cost, flexibility, openness, and maturity. Open source solutions might be preferred to rapidly make the first steps in a flexible prototyping context, while of-the-shelf components solutions might be the better choice to rely on stable functionalities compliant to a given standard.
- *Functionalities and CPU support*: The correct migration of the BSW / OS for a specific CPU and for a specific compiler shall be available. Sometimes new CPU generations are not (fully) supported, or that a specific BSW layer has not been migrated or that specific HW mechanisms from the CPU are not accessible for the BSW.
- *Deployment and industrial acceptance of the BSW / OS*: The automotive industry is split into a complex ecosystem comprising of car manufacturers, Tier 1 / 2 suppliers, and technology suppliers. It is unlikely that car manufacturers will develop the entire software on their own. On the contrary, the car manufacturer or Tier 1 supplier is responsible for system integration and must therefore make a decision about a computing platform and BSW / OS. The technology supplier needs to follow this choice and typically also ensure the compatibility of its application SW for different computing platforms.

Migration of Legacy Software to Multi-Core Systems

The process of customizing existing applications for multi-core systems is a key issue in multi-core development and a tough and enormously important task within multi-

core system migration. Although, parallelization of source code takes a back seat in the automotive domain compared to the merging of multiple functions in one multi-core system. Automatic supporting tools, such as parallelizing compilers or automatic schedule generators, are in early development stages and still frequently inadequate in parallelizing codes [14, 57]. The following generic steps thus turned out to be best practice for parallelizing software [52, 19]:

1. *Identification of parallelism*: As a first step, the type of parallelism (data and/or task parallelism) must be determined to select the adequate computing architecture.
2. *Profiling of existing application*: This step identifies the single-core/ best possible/ current performance of particular functions or the whole application to get reference values.
3. *System decomposition*: The biggest problems of parallel programming, this step shall break down the system into as independent as possible parts. There is no simple and straightforward way available to do this, due to the creative and multi-constraint nature of software architecture development. In the case of intersection of parallel tasks, appropriate synchronization must be applied.
4. *Identification of connections*: In this step shared memories, task execution orders, and task synchronization efforts shall be determined to evaluate these intersections in later development phases and ensure that these facts do not get omitted.
5. *Detailing synchronizations*: If decomposition identified independent tasks and connection between dependent tasks are also identified, this step should determine the ranking of task dependencies (e.g. which task gets shared resource first, how to react on resource limitation).

Merging of the functionalities into one multi-core system and making use of additional hardware features are the most common drivers for multi-core migration in the automotive context. Allocating tasks and cores efficiently and safely is the main challenge in the context of real-time multi-core systems [31].

1. Shared variables need to be identified and only accessed at the start and end time of the runnable.
2. A sequential structure diagram and data flow diagram of runnables to determine data consistency must be generated.
3. HW resources access shall be identified and only be done via basic software modules (ASW access to HW resources must be avoided).
4. Core mapping of the task must be done with a focus on minimizing inter-core communication and optimizing the distribution of workloads.

The last step of the above mentioned task to core allocation varies from case to case, depending on the main optimization constraints (e.g. minimizing inter-core communication, workload balancing, separation of code) and will not be reasonably feasible without an optimization tool for complex systems. Optimization tools are based on the graph-cut problem solution and are likely to provide various solutions depending on the main optimization parameters and on the software functionalities themselves. It is thus likely

that every SW function update will lead to different optimized task allocations, resulting in different program flows and finally having negative side-effects for the certification of safety-critical software.

Post-Migration: Performance Analysis

Even when all the previously mentioned restrictions have been taken into account, some side-effects or design shortcomings might still exist. As a result of this situation a migration analysis and alignment of simulated and tested results is required. Also the true performance increase and execution times of the overall system compared with the expected performance and timings should be correlated. Here too debug, instrumentation, and calibration tool supports are of major importance. In contrast to single-core systems, multi-core systems inherit new timing effects (such as delays through resource conflicts, IOC overheads) which must be analyzed and harmonized with the timing models; an evaluation of meeting the safety-critical timings at corner cases must also be done.

3.2.4 Service Deterioration Analysis

Dependable systems rely on mature quality management and development methods such as requirements / systems engineering and system analyses. In the automotive domain analysis methods for safety and security attributes at early development phases are well known and partially mandatory by domain standards. Nevertheless, approaches for analysis of serviceability attributes (the combination of reliability and maintainability) at early development phases are not yet available. System dependability features have mutual impacts, similarities, and interdisciplinary values in common and a considerable overlap among existing methods.

Therefore, the automotive industry is confronted with the central question of how to optimize and quantify dependability features (such as safety, security, and reliability) of a system under development (SuD) already at an early development phase. This contribution (published as **Paper F**; depicted as publication F in Figure 3.1 - publication layer) presents a novel approach to quantify the impact of individual system parts to the overall system serviceability at early development phases. For this approach, the inductive analysis method HARA (hazard analysis and risk assessment) is also used to enable the quantification of dependability features (such as reliability and maintainability) of the SuD. The service deterioration analysis (SDA) approach gives further information about the deterioration resistance level (DRL) required for a certain system reliability/availability.

A key concept of the HARA approach is defining the automotive safety integrity level (ASILs) [40]. The assigned ASIL determines the criticality of the SuD and defines requirements and measures to be applied to the rest of the system's lifecycle. For the purpose of determining the SuDs ASIL, possible hazards have to be identified, which

have the potential to put the system in a hazardous state. Afterwards, these hazards are quantified according to their potential harm severity (S), probability of exposure (E), and the controllability of the resulting hazardous event (C). The final step formulates high level safety requirements known as safety goals.

In analogy to this, it is of high importance for systems serviceability to have a clue of the contribution of each individual system component. Therefore, the serviceability features are quantified according to the deterioration impact (I) on the system's dependability, the component's repair aggravation (A), and the operation profile (O).

The deterioration impact (I) relates to the component's impact on the dependability of the system. Table 3.1 classifies the **deterioration impact (I)** and gives some examples of impacts of outage of the component. Either an outage of the component does not impair the system function at all (level 0), or it minimally reduces the functionality (level 1). Level 2 indicates maximum impacts which could lead to reputation losses.

The component's repair aggravation factor (A) is related to the component's capability and ease of a repair. Table 3.3 mentions some examples for the different repair aggravation levels. The repair aggravation factor is determined aligned with the definition of inspection frequency and reliability life cycle degradation control of military handbook 338B [5]. Therefore, the determination of the A factor of a specific component is simply done by adding of the components complexity ($high \rightarrow 1, low \rightarrow 0$), accessibility for maintenance ($hard \rightarrow 1, easy \rightarrow 0$), and diagnostic capability ($complex \rightarrow 1, manageable \rightarrow 0$) together.

The operation profile (O) values range from level 0, which means that the component is operated in its intended/normal environment. Level 1 is intended for usage in unplanned and harsh environments, while level 2 indicates misuse and operation out of operation limits. Table 3.2 outlines some examples of operation profiles.

These three values categorize the components deterioration resistance level (DRL). A higher repair aggravation or deterioration impact level result in a higher DRL level, whilst DRL level is raised the less the component is misused. Therefore the DRL can be calculated via the equation 3.1.

$$DRL = \begin{cases} 0 & \text{if } I + A - O < 2 \text{ or } I = 0 \\ 1 & \text{if } I + A - O = 2 \\ 2 & \text{if } I + A - O = 3 \\ 3 & \text{if } I + A - O = 4 \\ 4 & \text{if } I + A - O = 5 \end{cases} \tag{3.1}$$

3.2.5 Security-Aware Hazard and Risk Analysis Method

As mentioned in the previous section, dependable systems rely on the maturity of quality management and development methods, furthermore, system dependability features have

Table 3.1: Deterioration Impact (I) Classification - Classification of 'I' Value of Impact of Outage of the Component

Level	Deterioration Impact	Example
0	no impact	no impairment of function
1	minor impact	reduced functionality, self-healing temporal impairment of functions, expendable part repair
2	major impact, reputation compromised	stop in service, callbacks required, non- expendable part repair

Table 3.2: Operation Profile (O) Classification - Classification of 'O' Value of Intended Harshness of the Environment of the Component

Level	Operation Profile	Example
0	normal / intended environment	daily usage, typical application
1	unplanned / harsh environment	usage at normal operation limits, corner-cases
2	misuse / out of limits	misappropriation, vandalism

Table 3.3: Repair Aggravation (A) Classification - Classification of 'A' Value of Capability and Ease of Repair of the Component

Level	Repair Aggravation	Example
0	easy repair	can be performed by end-user
1	moderate repair	can be performed at any workshop (trained skills required)
2	difficult repair	needs to be performed at the production center (specialized skills required)
3	serious repair	no repair possible (repair action not foreseen, product no longer useable)

mutual impacts, similarities, and interdisciplinary values in common. Because modern automotive systems are increasingly interlaced, it is no longer acceptable to assume that safety systems are immune to security risks and vice versa. Although safety and security are two seemingly contradictory system features and have been treated separately in the past; they have now become more important due to increasing knowledge of their mutual impacts, similarities, and interdisciplinary values. Security is defined, according to [12], as the concurrent existence of availability for authorized users only, confidentiality, and integrity with improper meaning of unauthorized.

This contribution (published as **Paper G**; depicted as publication G in Figure 3.1 - publication layer) presents a combined approach of the automotive HARA (hazard analysis and risk assessment) with the security domain STRIDE concept, and outlines the impacts of security issues on safety concepts at system level. The fundamental concept of the SDA is to classify the probability of security threats, which can be used to determine the appropriate number of countermeasures that need to be considered.

The contribution presents a framework for the security-aware identification of safety hazards. SAHARA (Security-Aware Hazard Analysis and Risk Assessment) is an expansion of the inductive analysis method called hazard analysis and risk assessment (HARA), and encompasses threats from the STRIDE Threat Model [50].

Threat modeling using STRIDE can be seen as the security equivalent to HARA. STRIDE is an acronym for spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges. The key concept of this threat modeling approach is the analysis of each system component for susceptibility of threats and mitigation of all threats to each component in order to argue that a system is secure.

Figure 3.4 shows the conceptual overview of the novel SAHARA method. As can be seen in the figure, an ISO 26262 conforming HARA analysis (right part of the overview figure) can be performed in a conventional manner. Besides this, attack vectors of the system can be modeled using the STRIDE approach independently (left part of Figure 3.4) by specialists of the security domain. The two-stage SAHARA method then combines the outcome of this security analysis with the outcomes of the safety analysis. Therefore, a key concept of the HARA approach, the definition of automotive safety integrity level (ASILs) is applied to the STRIDE analysis outcomes. Threats are quantified aligned with ASIL analysis, according to the resources (R), know-how (K) required to exert the threat, and the threats criticality (T). The second stage is the hand-over of information of security threats that may lead to a violation of safety goals for further safety analysis. This improves completeness of safety analysis in terms of hazardous events initiated due to security attacks, related to the ISO 26262 requirement of analysis of 'foreseeable misuse'. The first step of the SAHARA approach to combining security and safety analysis is to quantify the STRIDE security threats of the SuD in an analog manner as done for safety hazards in the HARA approach.

Table 3.4 classifies the **required resources - 'R'** to threaten the SuDs security and gives some examples of tools required to successfully exert the security threat. Level 0

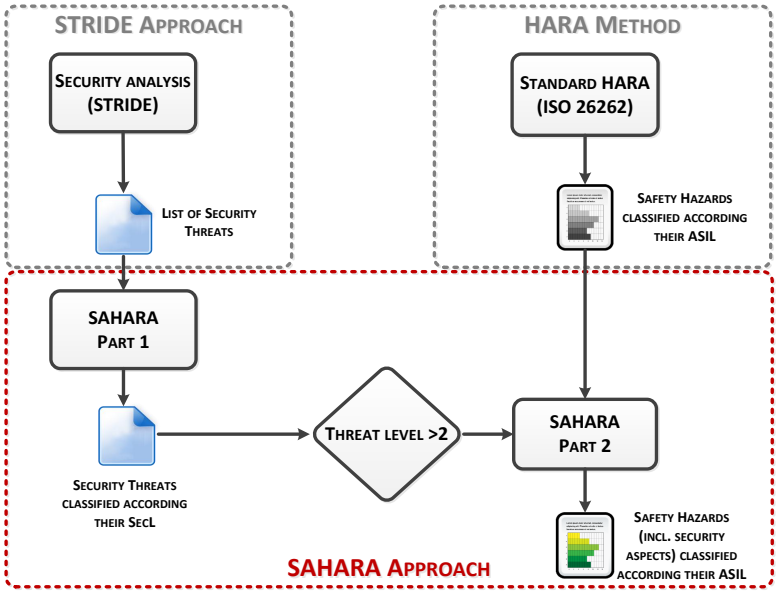


Figure 3.4: Conceptual Overview of the SAHARA Method

covers threats not requiring any tools at all or an everyday commodity, available even in unprepared situations. Level 1 tools can be found in any average household, while availability of level 2 tools is more limited (such as special workshops). Tools assigned to level 3 are advanced tools whose accessibility is very limited and are not wide-spread.

Table 3.5 does the same classification for the **required know-how - 'K'**. Here level 0 requires no prior knowledge at all (the equivalent of black-box approach). Level 1 covers persons with technical skills and basic understanding of internals, representing the equivalent of gray-box approaches, while level 2 is tantamount to white-box approaches and represents persons with focused interests and domain knowledge.

An overview of the **criticality of a security threat - 'T'** is given in Table 3.6. Level 0 indicates in this case a security irrelevant impact, such as raw data which can be visualized but whose meaning cannot be determined. The threat impact of level 1 threats is limited to annoying, maybe reduced, availability of services, but does not imply any damage of goods or manipulation of data or services; such threats belong to level 2. Level 3 threats imply privacy intrusion or impacts on human life (quality of life) as well as possible impacts on safety features.

Table 3.4: Required Resource 'R' Classification - Determination of 'R' Value for required Resources to Exert Threat

Level	Required Resource	Example
0	no additional tool or everyday commodity	randomly using the user interface, strip fuse, key, coin, mobile phone
1	standard tool	screwdriver, multi-meter, multi-tool
2	simple tool	corrugated-head screwdriver, CAN sniffer, oscilloscope
3	advanced tools	debugger, flashing tools, bus communication simulators

Table 3.5: Required Know-How 'K' Classification - Determination of 'K' Value for Required Know-how to Exert Threat

Level	Required Know-How	Example
0	no prior knowledge (black-box approach)	average driver, unknown internals
1	technical knowledge (gray-box approach)	technician, basic understanding of internals
2	domain knowledge (white-box approach)	person with technical training and focused interests, internals disclosed

Table 3.6: Threat Criticality 'T' Classification - Determination of 'T' Value of Threat Criticality

Level	Threat Criticality	Example
0	no security impact	no security relevant impact
1	moderate security relevance	annoying manipulation, partial reduced availability of service
2	high security relevance	damage of goods, invoice manipulation, non availability of service, possible privacy intrusion
3	high security and possible safety relevance	maximum security impact and life-threatening abuse possible

$$SecL = \begin{cases} 0 & \text{if } T = 0 \\ > 0 & \text{if } T = 3 \\ 4 & \text{if } 5 - K - R + T \geq 7 \\ 3 & \text{if } 5 - K - R + T = 6 \\ 2 & \text{if } 5 - K - R + T = 5 \\ 1 & \text{if } 5 - K - R + T = 4 \end{cases} \tag{3.2}$$

These three factors determine the resulting security level (SecL). The SecL determination is based on the ASIL determination approach and is calculated according to (3.2).

The quantification of the threats impact, on the one hand, determines whether the threat is also safety-related (threat level 3) or not (all others). This information is handed over to the safety analysis method in the second stage of the SAHARA approach. On the other hand, this quantification enables the possibility of determining limits of resources spent to prevent the SuD from a specific threat (risk management for security threats). After this quantification these threats may then be adequately reduced or prevented by appropriate design and countermeasures.

In the case of safety-related security threats, the threat can be analyzed and resulting hazards evaluated according their controllability, exposure, and severity. This improves, the completeness of the required situation analysis of the HARA approach by implying factors of reasonably foreseeable misuse (security threats) in a more structured way.

3.2.6 System Dependability Analysis Methods

Dependability is a superordinate concept regrouping different system attributes such as reliability, safety, security, or availability and non-functional requirements for modern embedded systems. These different attributes, however, might lead to different targets. Furthermore, the non-unified methods to manage these different attributes might lead to inconsistencies, which are identified in late development phases. The aim of this contribution is to present a combined approach for system dependability analysis to be applied in early development phases. This approach regroups state-of-the-art methods for safety, security, and reliability analysis, thus enabling consistent dependability target identification across the three attributes. This, in turn, is a pre-requisite for consistent dependability engineering along the development lifecycle.

In the publication **Paper H** (depicted as publication H in Figure 3.1 - publication layer) the experiences of this combined dependability system analysis method are discussed based on an automotive application. This paper presents a combined approach for analysis of dependability features in early design phases of an embedded automotive system.

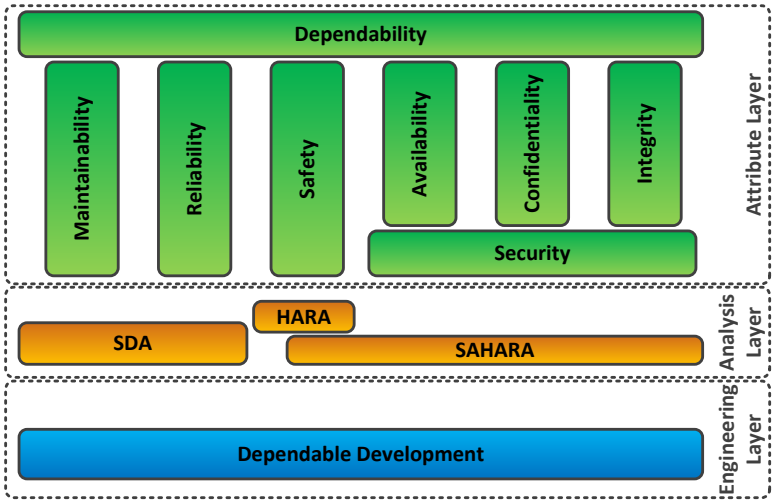


Figure 3.5: General Overview of Dependability Attributes and Analysis Methods

In the automotive industry, dependability engineering is currently moving its center of gravity from mainly mechanical reliability towards functional safety and security of the control system. While the target is still to provide a convincing argument that the system can justifiably be trusted [12], the hazards to consider as well as the development methods must be adapted accordingly. Hence, dependability is seen according to [13] and [12], as an integrating concept that encompasses numerous different attributes. Figure 3.5 provides an overview of the attributes (aspects) of dependability, the analysis methods available for each attribute, and a common dependable development block indicating the fact that each aspect needs to be addressed within a consistent engineering framework. Indeed, a common analysis method delivering consistent dependability targets over the different attributes is the basis to performing consistent dependability engineering during the entire product development. It can also be seen that security is a composition of the attributes of confidentiality, integrity, and availability. Security is the combination of confidentiality, the prevention of unauthorized information disclosure and amendment or deletion of information, plus availability and also the prevention of unauthorized withholding of information [12].

The common engineering basis for all dependability aspects raises the requirement for a combination of the different analysis methods and targets, thus a mutual understanding of focuses and language concepts. Table 3.7 shows a mapping of safety, security, and service oriented engineering terms.

Combining the different dependability feature analysis methods and dependability targets is of high importance. The SAHARA approach already implies an identification of security threats having a possible impact on safety and an information exchange

Table 3.7: Mapping of Safety, Security, and Service Oriented Engineering Terms

		Safety Engineering	**Security Engineering**	**Service-Oriented Engineering**
Analysis Subjects	risk	hazard	threat	warranty claim, unplanned maintenance
	system inherent deficiency	malfunction	vulnerability	service loss or degradation
	external enabling condition	hazardous situation	attack	(mis)usage profile
Analysis Categories	impact analysis	severity	threat criticality	reputation loss, deterioration impact
	external risk control analysis	controllability	attacker skills, know-how	repair efforts, repair aggravation
	occurrence analysis	exposure	point of attack, attack resources	operation condition spectrum
Analysis Results	design goal	safety goal	security target	dependability target
	design goal criticality	ASIL	SecL	DRL

between the security and safety domain. Nevertheless, the approach described in this paper relies on the combination of the outcomes (targets and classifications) of HARA, SAHARA, and SDA to raise the level of completeness of the analysis and consistency between mutual dependencies. Figure 3.6 shows an overview of the described approach and highlights the distinctive features of the presented approach (broad red arrows).

The mutual impact of serviceability analysis considerations and safety considerations (see Figure 3.6 - arrow I) exists between safe states and reliability targets of the system. A tradeoff between higher availability and higher safety of the system impacts the design of safe states (e.g. a system shutdown in the case of uncertainty of the actual system state can be a good option from a safety or cost point of view, but has negative effects on system availability). The targets and target classifications found in SDA, therefore have an impact on the design of safe states (e.g. fail-silent vs. fail operational). On the other hand, higher safety levels (ASIL) require higher reliability of the related components. This affects the design and quality requirements of components which might have not been in focus of SDA. Formulating these mutual dependencies between serviceability and safety leads to two regulations:

$DRL \geq 3 \rightarrow$ component related safe states need to be reviewed, eventual adaptation of degradation concept required

$ASIL > QM \rightarrow$ specific component reliability required, eventual adaptation of DRL of neglected component required

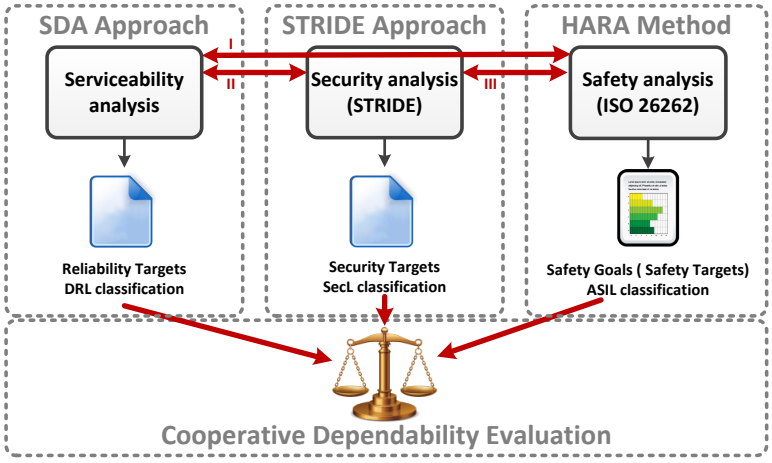


Figure 3.6: Overview of the Described Approach with Distinctive Features Highlighting

Preventing unauthorized access to control interfaces by security password affects the system security positively, beside this, it reduces the availability and controllability of the whole system (see Figure 3.6 - arrow II). Requiring authentication of each system component positively affects the overall security, but simultaneously increases the maintenance effort and time requirement, which further impacts system availability. On the other hand, easing the maintenance burden by reducing security authentication discloses attack vectors that were probably not considered during security analysis (e.g. the simple replacement of the encryption chip). The regulation of the mutual dependencies between serviceability and security leads to the following:

$DRL \geq 3 \rightarrow$ component related security targets need to be reviewed

$SecL \geq 3 \rightarrow$ serviceability might be affected, review required to determine impact on operational readiness

The third mutual impact (see Figure 3.6 - arrow III) exists between safety and security. Safety and security features frequently appear to be in total contradiction to the overall system features. An example of this contradiction can be shown by the electrical steering column lock system. In the security context the system locks the steering column when in doubt, because this doubt area might be the result of an attack. From the safety perspective, however, it might not be the best approach to lock the steering column as a fallback, since the issue involved might well be an occurrence directly before a high speed corner turn. Mutual dependencies between safety and security can be prescribed as:

$ASIL > QM \rightarrow$ safe state need to be reviewed for possible deactivation of security

$T = 3 \rightarrow$ safety might be affected, a review is required to determine impact on operational readiness

The regulation of these mutual dependencies, the required information handover, and the mapping of the different engineering domain terms allow a cooperative dependability evaluation by cross-domain expert teams and provide traceable measures for early design decisions.

3.2.7 Seamless Modeling Approach

The modeling support of system architectural design and refinement of this design to software is also a contributing part to this thesis and publication [48]. The AUTOSAR methodology [8] 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 in budget (such as time or manpower) with this approach. The model presented in this thesis has thus emerged from full AUTOSAR based approaches, SysML, or EAST-ADL approaches and focuses forcefully on a lean MBD model to gather all required information in a central MBD database as a single-source of information concept. This database inherits all information from the involved engineering disciplines (system, software, and safety) in a structured way, and allows different engineers to do their job in their specific manner. The presented model has been developed using profiles which use a subset of the SysML language to define a system model particularly tailored to automotive engineering and safety engineering in context of ISO 26262. This contribution focus on the additional model enhancements to also support software development of basis software functions (HW driver), operating system configuration and task allocation, and modeling of complex software architectures for function software development.

The main benefit of this proposed approach contributes to closing the gap between system-level development at abstract UML-like representations and software-level development. Furthermore, the model minimizes redundant manual information exchange between tools and contributes to simplifying seamless safety argumentation according to ISO 26262 for the developed system. The following sections describe the key contribution parts of the model in more detail.

Application Software Model Part

The first part is a specific UML model enabling software architecture design in AUTOSAR like representation within a state-of-the-art system development tool. A specific SysML profile is used to limit the SysML possibilities, to the needs of software architecture devel-

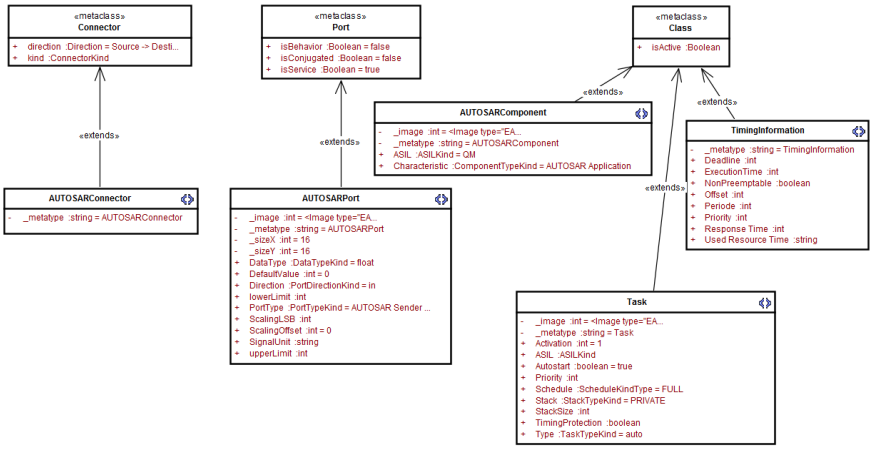


Figure 3.7: Shows the Representation of Application SW Artifacts.

opment of safety-critical systems and enable software architecture design in AUTOSAR like representation within the system development tool.

This profile makes the SysML representation more manageable for the needs of the design of an automotive software architecture by taking advantage of an AUTOSAR aligned VFB abstraction layer. In addition to the AUTOSAR VFB abstraction layer [11], the profile enables an explicit definition of 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). Hence, the SW architecture representation within EA can be linked to system development artifacts and traces to requirements can be easily established.

Figure 3.7 shows the representation profile of application software architecture artifacts. As can be seen in the depiction, all artifacts required to model the SW architecture are represented and inherit the required information as tagged values. Table 3.8 sums up the artifacts and their inherited information.

Hardware and Basic Software Model Part

Special basic software (BSW) and hardware 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 very late in the development phase and therefore enables application software developers and basic software developers to work in parallel. The hardware profile,

Table 3.8: Highlights the Application SW Development Artifacts and their Attributes.

Artifact	Configurable Attribute
AUTOSAR composition modules	2
TASKS (AUTOSAR runnables)	9
Port definitions	9
Timing information	8

depicted in Figure 3.8, allows representation of hardware resources (such as ADC, CAN), calculation engines (core), and connected peripherals which interact with the software. This further enables the establishment of software and hardware dependencies and a hardware-software interface (HSI), as required by ISO 26262. Software signals of BSW modules can be linked to HW port pins via dedicated mappings. On the one hand this enables the modeling and mapping of HW specifics and SW signals. On the other hand, this mapping establishes traceable links to port pin configurations. Finally, these HW dependencies can be used to interlink scheduling and task allocation analysis tools for analysis and optimization of resource utilization.

Temporal Analysis Model Part

The ability to generate model links between software signals and HW resources besides the possibility of specifying timing constraints (see Figure 3.7) of software modules enables the ability to use the model to interlink scheduling and task allocation analysis tools. This enables the analysis and optimization of resource utilization, which is especially important for multi-core system development.

3.3 Tool Layer related Contribution

This section describes the tool concept for establishment of a seamless model-based development tool-chain for development of safety-critical automotive systems. The add-ons for the system development tool presented here are the results of the experiences gained during the project. The add-ons aim to extend the system engineering tools field of application to a comprehensive tool-chain for the whole development cycle of safety-critical and multi-core automotive systems. Figure 3.9 shows the conceptual overview of the approach. The yellow highlighting marks the introduced extensions.

Extension needed for the existing approach are:

- an AUTOSAR aligned profile for software architecture development and extraction to SW development tools

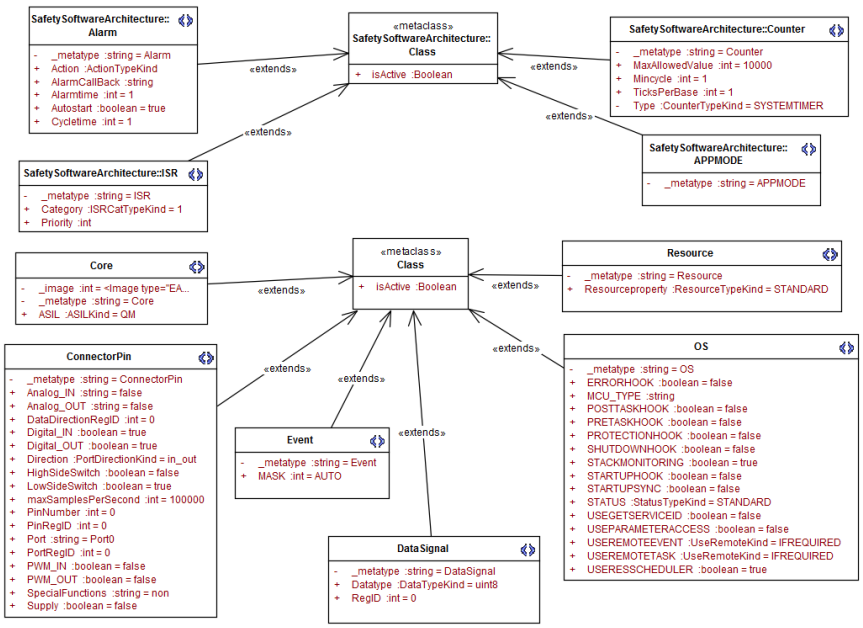


Figure 3.8: Shows the Representation of Basic SW Artifacts and HW Representations for Optimization of Resource Utilization.

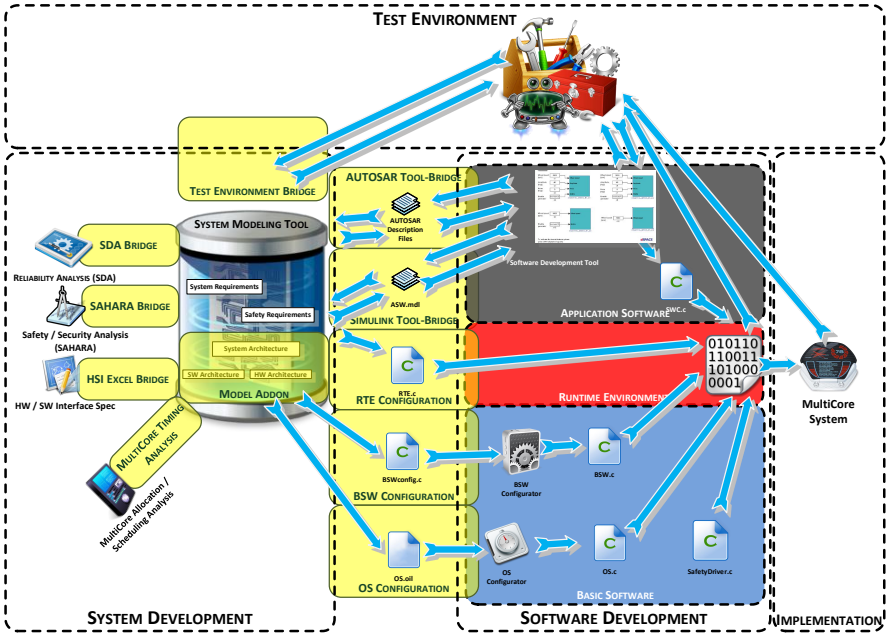


Figure 3.9: Overview of Tool Layer related Contributions

- a model representation of the hardware in use for safety analysis
- hardware-software interface capabilities for HSI definition according ISO 26262
- extractors to automatically generate ECU BSW and RTE configurations from existing artifacts / information at system development level
- interconnection support for multi-core scheduling configuration and validation tools

These extensions close the gap between system level development at abstract UML-like representations and software level development modeling tools (such as Matlab Simulink / Targetlink). By closing this gap a seamless tool-chain from initial requirements coming from an application life cycle management tool, through definition of safety concepts and software architectures in a model-based development environment, to final decisions in code implementation in compliance with ISO26262 is available (see Figure 3.10). Figure 3.10 illustrates the improvement due to this approach in terms of tools in use and in relation to the ISO 26262 SW development process. The gray arrow in the upper left corner and the reduced quantity of tools at the upper part of the 'Design Phase' originate from the related preliminary AVL-PTE safety system modeling approach. The arrows depicted in red and the extension required for the lower part of the figure are part of this thesis contribution to extend the tool to an information backbone for the whole development cycle of an embedded automotive multi-core system. The following sections focus in particular on one specific part of this approach and describe their principles. Moving to a more formal (and automated) component-based approach enables the definition of a framework (e.g. guidelines, automated checks) to identify and correct (and thus minimize) the number of errors and lessen their impact. The main benefit of this enhancement is an improved consistency and traceability from the initial requirements at the system level down to the single software components, as well as a reduction of cumbersome and error-prone manual work-flows along the system development path. Furthermore these improvements include the concept change from a document-centric approach to a seamless model-based approach with a single information backbone for all engineering disciplines involved.

3.3.1 Application Software Modeling Toolbridge

A specific profile (described in Section 3.2.7) enables the possibility of designing software architectures in an AUTOSAR aligned way within a system development tool. The 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 enables the possibility of 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 assigned to establish links to the underlying basic

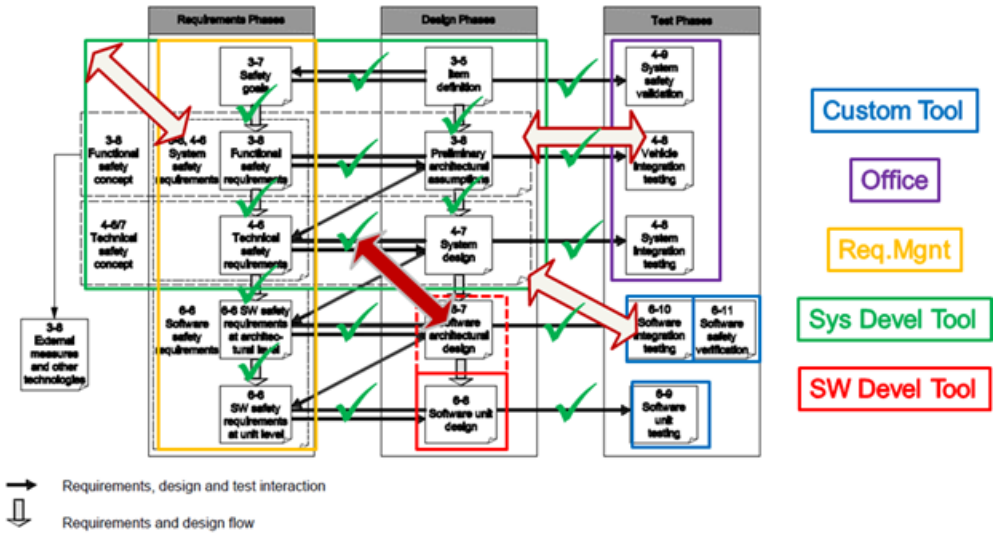


Figure 3.10: ISO 26262 SW Development Process [40] and Tool Mapping

software and hardware abstraction layers. In addition to standard VFB AUTOSAR profiles the profile features assignment and graphical representation of ASIL to dedicated signals and modules and provides a specification for runnables with timing constraints (such as WCET), ASIL, priority, and required stack sizes. 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. Via a dedicated exporter the software architecture and related information can be exported in ARXML AUTOSAR format (see Figure 3.11), which many software engineering tools are able to import, process, and re-export by default.

The exporter generates an AUTOSAR conform software component description file enriched with system and safety development artifact traces. Information that is not importable by default AUTOSAR import functions of third-party tools are still available for the user of this particular tool. In addition, a re-import of changes in the ARXML file, after implementation of software function code, can be performed. This ensures consistency between system development artifacts and changes done in the software development tool (**Paper I** depicted as publication I in Figure 3.1 - publication layer).

A variation of this approach(**Paper J** depicted as publication J in Figure 3.1 - publication layer) is an exporter, which is able to export the software architecture, component containers, and their interconnections designed in SysML directly to the software development tool Matlab/Simulink and thus, enabling the information to be handed over to a special purpose tool (model-driven software engineering tools) for detailing of the

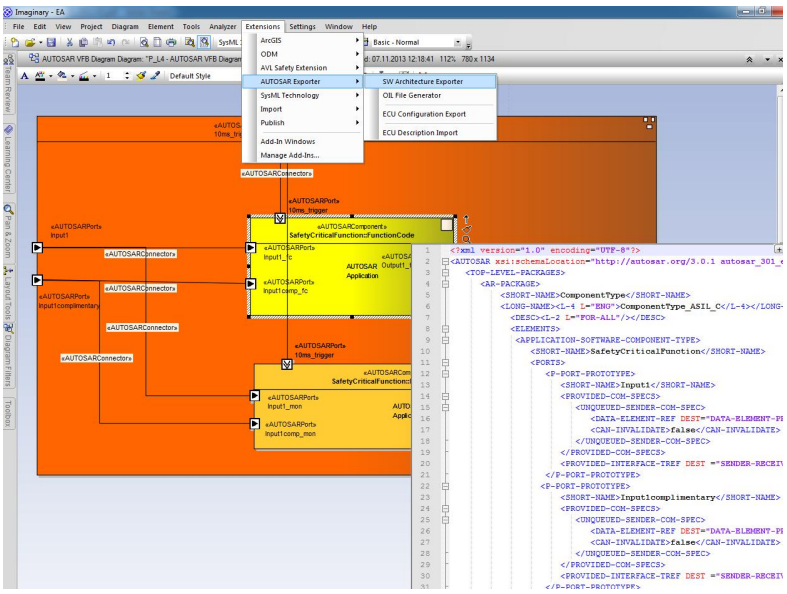


Figure 3.11: Screenshot of the SW Architecture Representation within the System Development Tool and Extension of Bridging Approach

SW architecture and SW modules. The related import functionality, in combination with the export function, enables the bidirectional update of software architecture representations. On the one hand, this ensures consistency between system development artifacts and changes done in the software development tool. On the other hand, the import functionality enables reuse of available software modules, guarantees consistency of information across tool boundaries, and shares information more precisely and less ambiguously. The API based variant has been evaluated versus .m file based approach and direct .mdl file manipulation. The main factors of this evaluation are briefly summarized in Table 3.9. As can be seen in the table, most factors favor the API based approach with its major drawback of requiring a Matlab/Simulink installation.

Table 3.9: Summary of Evaluation Factors of Simulink Tool Bridge Variants

Factor	**API based**	**Script based**	**Direct model access**
Complexity	+	+	−
Automation	+		+
Matlab installation required	yes	no	no
Data amount	+	+	−
Round-trip Engineering	easier	easier	tough

3.3.2 Basic Software Configuration Generator

The ASW/BSW interface generator and BSW configuration generator (related publication **Paper K** depicted as publication K in Figure 3.1 - publication layer) generates .c and .h files defining SW/SW interfaces between application software signals and basic software signals based on modeled HSI artifacts. In addition, this generation eliminates the need of manual SW/SW interface generation without adequate syntax and semantic support and ensures reproducibility and traceability of these configurations. Figure 3.12 shows the conceptual overview of generated files. The.c and .h files on application software level are generated via a model-based software engineering tool, such as Matlab/Simulink. The files at the basic software level are usually provided by the hardware vendor. While the files mentioned in SW/SW interface layer are generated by our approach. These generated files are designed as a two step approach. First step of the interfacing approach (interface.c and interface.h) establishes the interface between ASW and BSW based on AUTOSAR RTE calls. The second step (AVLIL_BSWa.c and AVLIL_BSWa.h) maps this AUTOSAR RTE based calls to the HW specific implementation of basic SW drivers. The basic software configuration generator is also part of the dll- based tool, which generates BSW driver specific *_cfg.c files. These files configure the vendor specific low level driver (basic software driver) of the HW device according to the HSI specifications. The mapping of HSI specifications to low level driver configuration is hardware and low level driver implementation specific and needs to be done once per HW device and supported low level driver package.

3.3.3 OS Configuration Generator

The OS Configuration bridge (**Paper L** depicted as publication L in Figure 3.1 - publication layer) overcomes the existing gap between the model-driven system engineering tool and software engineering tools for automotive real-time operating systems (RTOS). The approach makes use of existing high level control system information in SysML format to generate the configuration of automotive real-time operating systems in a standardized OSEK Implementation Language file format (OIL files)[53]. Information of the control system (such as control strategies) can thus be mapped to a configuration at software level (e.g., required interfaces to other SW components, allocation to a CPU respectively to a task). The goal of this tool is to support a consistent and traceable refinement, from the early concept phase to individual configurations of the RTOS. The approach minimizes redundant manual information exchange between tools and also takes IS0 26262 requirements (especially traceability) and constraints into account.

The OS configuration tool comprises of the following aspects: An extractor which automatically generates OIL files from existing information at system development level. This ensures consistency of the specification and implementation for the RTOS. The importer supports round-trip engineering by re-importation of information updates from

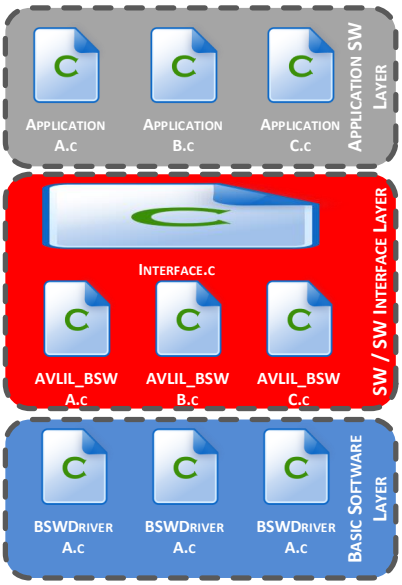


Figure 3.12: Overview of Interface Files Generated by the BSW Configuration and Interface Generator

OIL files. Automotive OS do not have dynamic scheduling parts, in that all OS settings are static and can be specified during the development phase. With the introduced approach for HW representation integration of tools supporting the specification of task priority, duration, the mapping of tasks to cores, task activation policies, and a support for specification of resources, alarms, and interrupts is possible. Additional information originating from system development level, e.g., required resources, deadlines, and ASILs derived from safety concepts complete this description. Therefore, an export/import functionality based on standardized exchange format (OSEK OIL files) is introduced to establish and validate scheduling and configuration of the real-time operating system in a way that supports round-trip engineering. The graphical representation of this approach is already shown in Figure 3.8.

3.3.4 Hardware Software Interface Definition Toolbridge

The hardware-software interface (HSI) definition document is the last development artifact of the system development phase and the starting point for parallel development of hardware and software. HSI definition requires mutual domain knowledge of hardware and software and is to be the product of a collective workshop of hardware, software, and system experts. The document is used to agree on topics relevant to both, hardware and software development, and acts as the link between different levels of development.

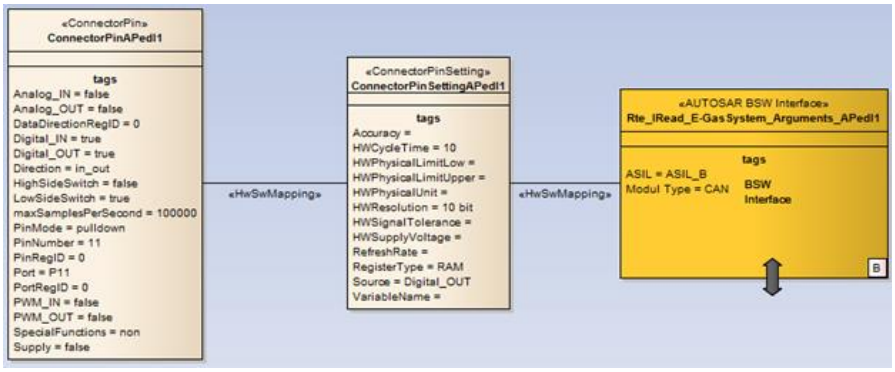


Figure 3.13: Graphical Representation of Hardware Software Interface

Insufficient definition of the HSI can cause several additional iteration cycles and communication issues between development teams. Consistency and evidence of correct implementation of HSI can be challenging in the case of concurrent HW and SW development and cross-dependencies of asynchronous update intervals. Therefore, this tool approach for ISO 26262 aligned hardware-software interface definition (**Paper M** depicted as publication M in Figure 3.1 - publication layer) enables a practicable and intuitive way of engineering HSI definitions in a spreadsheet tool (Excel) and transforms them to a reusable and version able representation in the MDB tool. With this approach, the spreadsheet document and system model can be bidirectionally aligned via program-specific APIs. This, on one hand, enables a practicable, tool-independent, and intuitive way of engineering HSI definitions with spreadsheet tools and transforms the generated information to a reusable and version-able representation in the MDB tool. On the other hand, this approach unifies the project-dependent process for HSI definitions across the variety of different projects and contributing partners without requiring exactly the same development tools or processes in place. Thirdly, the machine- and human-readable notation of spreadsheets ensures a cost- and time-saving alternative to usually complex special-purpose tools. Figure 3.13 illustrates the graphical model representation of the HSI.

3.3.5 Multi-Core Scheduling Tool Support

As mentioned by several related works [33, 10, 41] additional constraints occur during development of multi-core systems. The initiation of software development in an AUTOSAR-aligned way supports this development with the virtual functional bus (VFB) abstraction level (see [11] for further details). But several issues of multi-core systems cannot be addressed via AUTOSAR, e.g. correct timing and resource-efficient scheduling. These issues can usually be solved using third-party tools, which inter operate via

either AUTOSAR ARXML files or proprietary interchange formats. These formats and supporting information from the model need to be extracted via dedicated MBD tool plugins and transferred via API calls. Again a solution for re-importation of the generated output to update the system model is supported. The introduced extension for HW representation, AUTOSAR aligned SW architecture, and ECU BSW configurations support multi-core systems with specific tags and values for these constraints. Furthermore, in accordance with Richter et al. [58], this approach supports an early resource estimation and early timing estimation, which both can be kept up to date due to round-trip engineering support of the specific MBD tool extensions. Richter et al. claim wrong or non-estimation of resource consumption and missing SW architecture considerations as the main risk for failure in launching multi-core systems. The approach therefore scales for single-core as well as multi-core systems. Solely some of the extensions might be optional for single-core systems, but mandatory for multi-core systems. Table 3.10 shows a brief summarization of these mandatory extensions for multi-core systems:

Table 3.10: Summarization of Extensions required for Multi-core Support

Extension	**Relation to multi-core system development**
Software architecture	enables HW independent (core independent) development of ASW
	ASW task can be mapped to cores for early scheduling and resource estimation
	task hold timing constraints
Hardware representation	enables linking of tasks and resources
	supports automatic generation of multiple OS configurations
HSI definition	enables configuration of BSW driver
	enables automatic configuration of safety HW features (such as MPU, access privileges)

This timing bridge integrates a mutual information exchange with the Symta Vision's SymTA/S tool. SymTA/S and TraceAnalyzer are well-known in the automotive and other embedded real-time industries. The tools can be used for early system analysis, the affect of increasing number of software functions analysis, for calculation of computing power and communication bandwidth needs, and improving the system performance or cost optimization. SymTA/S is a model-based timing analysis, optimization and verification tool for analysis of (a) Worst-Case Scheduling, (b) optimization of distribution of functions, and (c) analysis of different distribution scenarios. TraceAnalyzer analyses the timing of actual traces obtained from other 3rd party probes such as Vector CAN-analyzer, Lauterbach TRACE32 or Gliwa T1. These tools support OSEK, AUTOSAR,

and multi-core ECUs, as well as CAN (including CAN-FD), LIN, FlexRay, and Ethernet (standard, AVB) bus analysis. SymTA/S and TraceAnalyzer are Eclipse-based tools which also support scriptings, remote access, import / export, and report capabilities which enable their integration into the proposed tool-chain.

3.3.6 Test Tool Integration

Enterprise Architect supports the generation of documentation according to various standards and templates, once a generic document generator template is defined. This feature can be used to automatically generate test specifications for the provided use-case diagrams. Furthermore, as for integration of special-purpose tools, MDG technology extensions can be used here to export test signal vectors for stimulation of the software models and thus ensure test coverage of all typical use-case scenarios. AVLab has been recently introduced as a company-wide software test environment. It supports model-based design in MATLAB/Simulink from model development to code generation over model testing. Supported features of the AVLab environment are: (a) interfaces to ADD, Integrity, TargetLink, EmbeddedCoder, AVL Concerto, SimDiff Tools, (b) the support of the PTE control processes, and (c) traceability features of object under test and ADD configurations. This means, the AVLab environment integrates interfaces to Integrity (life-cycle management tool), ADD (calibration parameter versioning tool),TargetLink, and AVL Concerto (data analysis tool) which support the PTE process landscape. Therefore, interconnecting AVLab with the model-based development tool-chain features traceability to the AVL test environment. More specifically traceable links to the object under test (SW code), its calibration parameters, test cases, and test reports can be established and made available for convincing argumentation of maturity of the developed system.

3.3.7 Constraint Checker

Another feature in high demand for ensuring consistency and correctness of complex multi-core systems development is to have a consistent model as the central source of information. To maintain consistency, correctness, and completeness of models, constraint checking features of the MDB tool-chain are substantial. This MBD tool add-on implements constraint checking abilities based on the OCL framework of Krallinger[42] and an interface for constraint definition within the EA tool.

4 Use-Case Application

In the context of the automotive domain it is very common to have some basic structure that serves to organize/handle the technical complexity of the system under development. Within AVL PTE a Global System Structure (GSS) has been defined and an aligned "Level Structure" for Requirements, Specifications, Design, and Integration Steps & Testing has been set up. This GSS level structure is depicted in Figure 4.1 and also used for this doctoral thesis use-case.

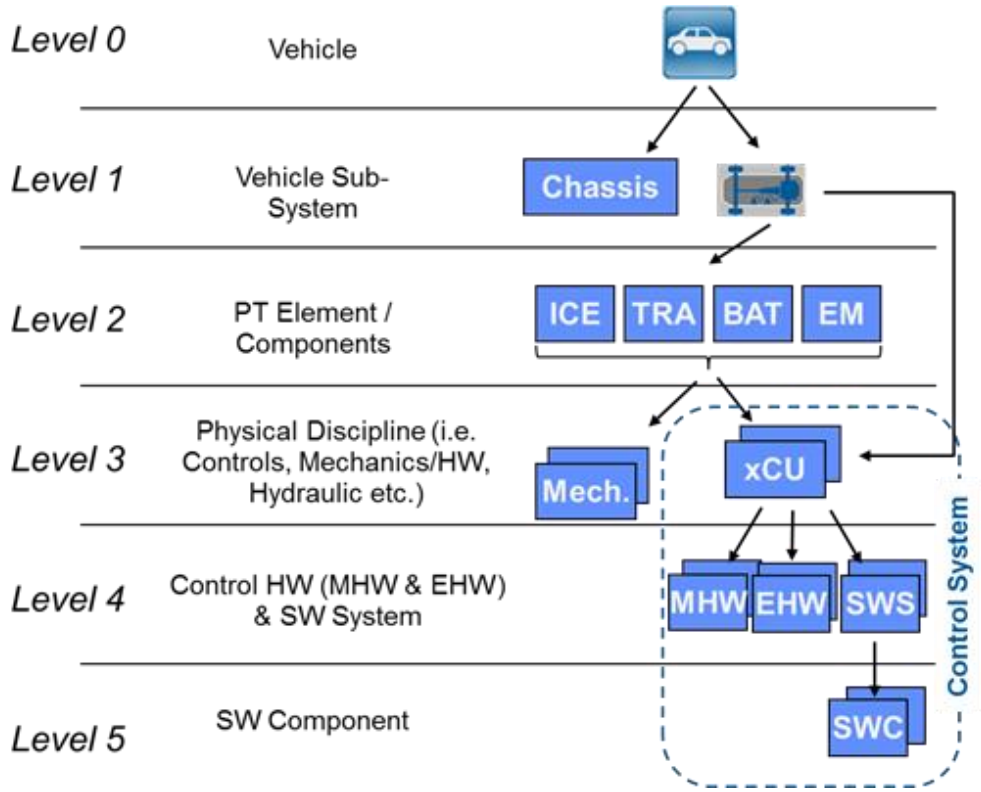


Figure 4.1: AVL PTE Global System Structure (GSS) and Level Structure ©AVL

The purpose of the level approach is to have a classification framework to assign requirements/designs according to the level they belong to. On the individual level, the

requirements/designs are used as the basis to derive further requirements/designs at the next down-stream level (in general this would be following level, however, the project scope may require to jump across certain levels). In other words: the requirements/designs are developed following a top-down cascading approach where on every level the requirements/designs become more detailed.

For the evaluation of the proposed approach an automotive battery management system (BMS) has been chosen as a use-case. This use-case is based on the INCOBAT project[1] and represents solely illustrative material, reduced for publication and is not intended to be exhaustive or representative of leading-edge technology. The technology-specific details have thus been abstracted for commercial sensitivity and the analysis results presented are not intended to be exhaustive. Although, all layers of the AVL PTE global system structure are processed by the use-case, the focus is set on the three most concrete layers (level 3, level 4, and level 5). Figure 4.2 shows the schematic diagram of a BMS. The BMS main interfaces are:

- Voltage sensors for individual cell voltages
- Voltage sensors for the overall battery voltage
- Current sensors for the overall battery current
- Temperature sensors to estimate the cell temperatures
- Interlock circuit ensuring electric shock protection
- Contactor control connecting and disconnecting the battery from other systems

To follow the ISO 26262 SW development process, first the item definition is done and system boundaries are defined. The second section focuses on the hazard analysis and risk assessment (HARA), which enlists possible hazards, hazardous situations and derives ASILs and safety goals. The usual HARA, required by ISO 26262, is in this case extended by a combined approach for dependable system development. Therefore not only a HARA, but also a SDA[49] and SAHARA[47] of the BMS use-case is done. In the third section the functional safety concept is briefly described. The fourth section focuses on the model representation of the use-case at system level and the final section highlights the model representation of the implementation level (SW development level) of the use-case.

4.1 Item Definition

ISO 26262 [40] defines an item as

> *system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied.*

[1] http://www.incobat-project.eu/

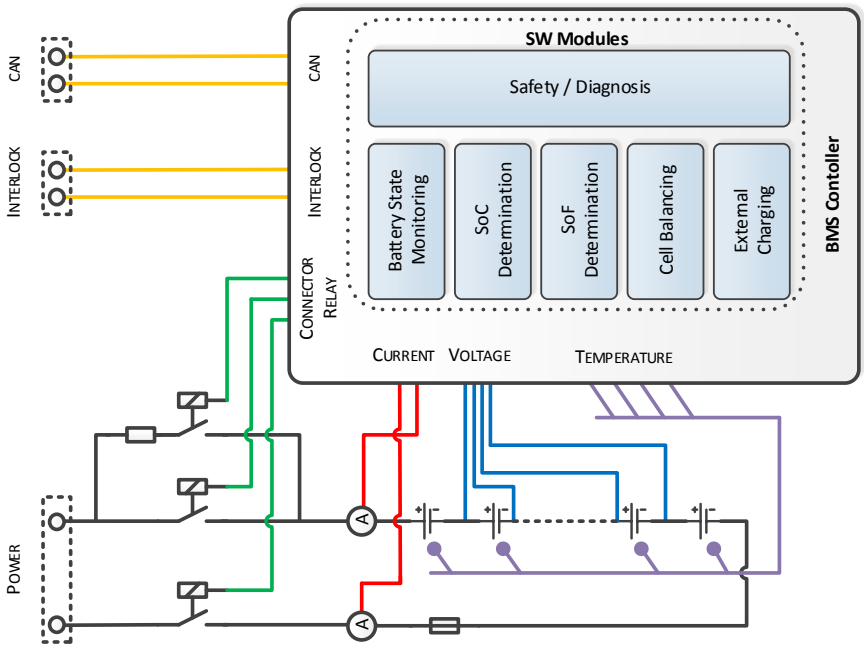


Figure 4.2: Schematic Diagram of a BMS

A System is further defined as

> *set of elements that relates at least a sensor, a controller and an actuator with one another.*

4.1.1 Provided Functions

For the use-case this means the item considered is a set of systems, called a HV battery. The HV battery consists of (a) a main battery module and (b) a variable number of battery packs satellite modules. The set of systems therefore implements the following vehicle level functions:

- Store provided electric energy
- Provide electric energy
- Control energy provided by external charger
- Request energy from range extender
- Control thermal conditions of battery cells

4.1.2 Elements of the Item

The battery pack satellites are connected to the main module and provide temperature, voltage, and current information for each individual cell. This information is gathered by the main battery module and serves as a base for the overall battery state estimation, state of charge (SoC), state of health (SoH), and power limitation calculation. Communication between BMS main module and satellite modules is carried out via a quasi SPI communication bus, which can be equipped with additional satellite modules during operation (hot-plugable). Control of the air cooling is done via PWM control signals and digital feedback signals. All other communication with connected systems is done via CAN.
The HV battery can be simplified using the elements mentioned in Table 4.1.

Table 4.1: Elements of the HV Battery

Sensor	**Controller**	**Actuator**
Cell voltage sensor Cell current sensor Cell temperature sensor	Satellite controller	Cell balancing switch
Output voltage sensor Output current sensor HV Interlock loop signal Ambient temperature sensor	Main controller	HV+ Relay HV- Relay Pre-Charge Relay Blower control

4.1.3 Intended Use and Assumptions

The BMS is intended for use within a serial-hybrid, on-road, electric passenger car, which goes in line with the intended field of application of ISO 26262. The BMS is designed according to the Safety Element out of Context (SEooC) approach of ISO 26262 [40, 68] with the following assumptions:

A_01 BMS is intended for usage in a passenger car with serial hybrid powertrain

A_02 BMS is air cooled by blower, which is directly connected and controlled via the BMS

A_03 BMS is the only supplier of HV interlock loop signal within the car

A_04 recuperative braking is not the only brake system within the car

A_05 recuperative brakes only support 10 % of overall brake-force

A_06 external charger is controlled by BMS via CAN signals

A_07 range extender can be requested by BMS via CAN signals

A_08 external insulation monitoring is supported and communicates with BMS via CAN

A_09 external discharge electronic is supported

A_10 battery is located outside the passenger compartment, battery outgassing takes place outside the passenger compartment

A_11 fire isolation compartmentalize passenger compartment from battery for at least 10 *s*

Figure 4.3 depicts the BMS as specified in the item definition. The left graphic shows the BMS and its connected electric systems, the right graphic shows the array of systems of which the BMS consists.

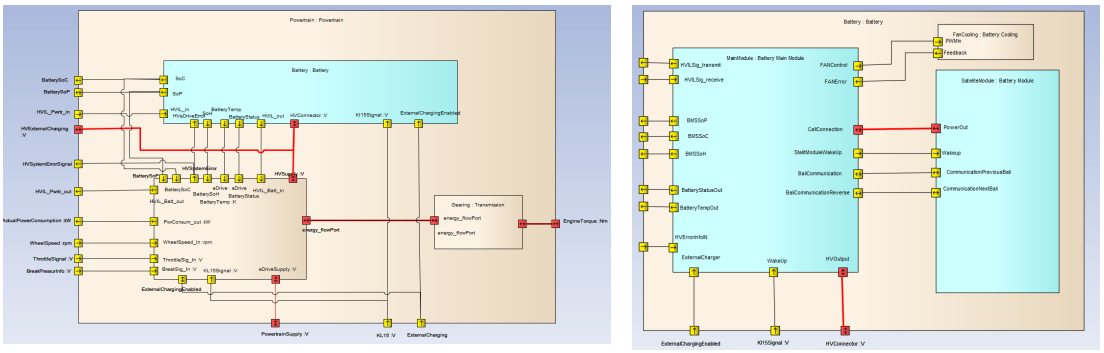


Figure 4.3: Screenshot of BMS Item and System Boundaries within the System Development Tool (EA)

4.2 Combined Analysis for Dependable System Development

This evaluation includes an ISO 26262 [40] aligned HARA safety analysis, a security analysis based on the SAHARA approach, and a serviceability analysis based on the SDA approach. An excerpt of the SAHARA analysis of the BMS use-case is shown in Figure 4.4. The excerpt highlights (a) the threat level classification 'T' triggering further analysis of the threat for safety impact and (b) a security hazard aiming at denial of service of the HV fuse. The HARA of the BMS use-case covers 52 hazardous situations, quantifies the respective ASIL and assigns safety goals fully in line with the ISO 26262 standard. Additionally, 37 security threats have been identified using the SAHARA approach, 18 of these security threats have been classified as having possible impacts on safety concepts. Furthermore, 63 service deterioration scenarios have been analyzed using the SDA approach.

Figure 4.5 shows an excerpt of the SDA for normal operation modes of the BMS use- case and also highlights the HV fuse data. The overlaid excerpt shows the impact of the security countermeasure against the threat 'replace fuse with non current limiting element'. As can be seen in the overlay, using corrugated-head screws for the fuse cover decreases the security risk of 'replace fuse with non current limiting element', but increases the repair aggravation value of the HV fuse, which also results in a higher DRL.

Security Risk description			Security Risk classification				Security Risk related Safety goal description					
Security Hazard ID	STRIDE Function	attacker generated malfunction	Required Resources 'R'	Required Know-How 'K'	Threat Level 'T'	Resulting SecL	Severity 'S'	Exposure 'E'	Controlability 'C'	Resulting ASIL	Safety Goal	Safe State
SH_4	Spoofing	extending safe operation areas of HV battery (temp, cell current, cell voltages)	1	1	3	3	3	4	2	ASIL C	Estimate correct cell status information	assume worst case, No torque provided, driver warning
SH_5	Denial of service	communication with charger jammed	2	2	3	1	3	2	3	ASIL B	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_6	Denial of service	immobilizing of driving functionality	1	0	2	3	0	0	0	QM	--- no safety impact --	--- no safety impact --
SH_7	Denial of service	emergency kill switch without function	1	2	3	2	3	2	3	ASIL B	Manual main switch off must be possible	Disconnect HV battery, driver warning
SH_8	Denial of service	replace fuse with non current limiting element	1	1	3	3	3	4	3	ASIL D	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_9	Spoofing	HV system ready without ensured overall system safety	2	2	3	1	3	2	3	ASIL B	Detect short circuit and isolation faults, Prevent from electric shock	Disconnect HV battery, driver warning
SH_10	Tampering	extending safe operation areas of HV battery (temp, cell current, cell voltages)	3	3	3	0	3	4	3	ASIL D	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_11	Spoofing	extending safe operation areas of HV battery (temp, cell current, cell voltages)	1	2	3	2	3	4	2	ASIL C	Correct amount of power shall be maintained	No torque provided, driver warning
SH_12	Denial of service	bypass HVIL	1	2	3	2				ASIL A	Detect short circuit and isolation faults, Prevent from electric shock	Disconnect HV battery, driver warning

Figure 4.4: Excerpt of the Application of the SAHARA Analysis of the BMS Use-Case

			Deterioration Impact 'I'		Repair Aggravation 'A'				Operation Profile 'O'		DRL
ID	Part	System Service	Effect on System Service	Deterioration Impact 'I'	Part Complexity	Part Accessibility	Diagnostic Capability	Resulting 'A'	Operation Profile 'O'	Operation Profile Description	DRL
8	HV Fuse	provide electric energy	required for system service	2	0	1	0	1	0	normal operation	2
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	1	0	1	0	normal operation	2
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4
8	HV Fuse	provide electric energy	required for system service	2	0	0	0	0	0	normal operation	1
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	0	0	0	0	normal operation	1
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4

Figure 4.5: Excerpt of the SDA Application of the BMS Use-Case

As a result of this analysis the following safety goals have be determined:

$SG1$ Correct amount of power shall be provided **ASIL C**

$SG2$ Battery outgassing and fire shall be prevented **ASIL D**

SG2 can be split into three safety goals, if favored for further development steps:

$SG2.1$ Battery over-charging shall be prevented **ASIL D**

$SG2.2$ Battery over-current shall be prevented **ASIL D**

$SG2.3$ Battery over-temperature shall be prevented **ASIL D**

4.3 Functional Safety Concept

Based on these safety goals adequate functional safety requirements are determined and defined in the functional safety concept (FSC). These requirements still apply at system level and are not yet broken down into hardware or software requirements or imply any technical definition of the solution. For each safety goal at least one functional safety requirement is required.

4.4 MBD Representation of Use-Case at System Level

As mentioned previously, for this use-case the vehicle structure element hierarchy and the vehicle level architecture on higher levels of abstraction solely represents a possible use-case for the battery management system which helps to serve as a common knowledge base for the stakeholders to determine the sources and sinks for information of the BMS and the electronic and mechanical interaction partners. Furthermore, the AVL model-representation for product development at concept phase (ISO 26262-3[40]) is already established and not the focus of this doctoral thesis. Due to the fact that no assumptions, requirements, or design specifications are done at these abstraction levels.

Product development at system level (ISO 26262-4[40]) serves as a starting point for this use-case application and starts on level 2 of the AVL PTE GSS (depicted in Figure 4.1).

4.4.1 L2 - Powertrain Element Level

On this layer the ISO 26262 item definition for the BMS system is done. Therefore, the use-case battery item and its interfaces to the environment are defined (see Figure 4.6) and the dependability analyses have been performed; an excerpt of the model representation of the HARA can be seen in Figure 4.7. Safety goals and their assigned ASIL are determined by a systematic evaluation of hazardous events. This analysis is based on the item's functional behavior and determines ASIL for the system by considering the estimation of the impact factors severity, probability of exposure, and controllability of the hazardous events.

Table 4.2: Excerpt of BMS Functional Safety Concept

SG #	**Safety Goal**	**FSR #**	**Functional Safety Requirement**	**ASIL**
1	Correct amount of power shall be provided	FSR1	Communication with HV system must be ensured	ASIL C
		FSR2	If no correct power request received battery shall be disconnected from HV system	
2.1	Battery over-charging shall be prevented	FSR3	Voltage of each cell shall be monitored, required charging conditions calculated and communicated to chargers	ASIL D
		FSR4	If overcharging condition detected, current must be interrupted within 500 ms	
2.2	Battery over-current shall be prevented	FSR4	If overcharging condition detected, current must be interrupted within 500 ms	ASIL D
		FSR5	Battery currents shall be measured and limits shall be communicated	
2.3	Battery over-temperature shall be prevented	FSR6	If over-temperature condition detected, battery must be disconnected from HV system within 200 ms	ASIL D
		FSR7	Battery temperature gradient shall be monitored and counteracted via cooling	
		FSR8	Battery temperature shall be measured	

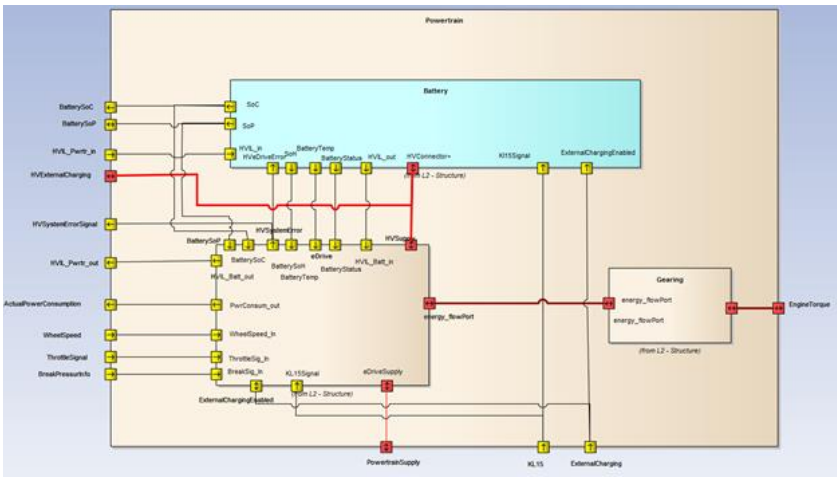


Figure 4.6: L2 Architecture of Powertrain and BMS Use-Case

4.4.2 L3 - Control System Level

This level constitutes the most concrete layer of system development. Therefore the structure elements of the main and satellite modules are composed of hardware and software elements. Figure 4.8 depicts the individual structures. As can be seen in Figure 4.8 the CCU structure consists of several HW, basic software and application software elements, which is caused due to the fact that actually the whole BMS strategy is implemented by the CCU. While the number of structure elements of the battery satellite modules is quite limited and neglected for this use-case application. This results due to the fact that the satellite modules consist solely of an ASIC (BALI chip), its implemented SW functionality and the peripherals required to operate properly. A depiction of the BMS architecture is given in Figure 4.9. This figure shows the connections between the main module (CCU) and the various satellite modules.

4.5 MDB Representation of Use-Case at Implementation Level

In abstraction level four (L4 - HW / SW System Level depicted also in Figure 4.1) software and hardware architecture design takes place. This layer is not included in classical system development approaches but defines the start of parallel software and hardware development (ISO 26262 part 5 and part 6). This abstraction level is therefore mostly only present in special purpose SW development tools or HW design tools rather than SysML based system development tools. With the presented modeling approach this tool breach and semantic gap can be bridged and SW architecture definition can be represented by the MBD tool. To support this the elements on this level are represented using the model approach and meta-models presented in Section 3.2.7. The demonstration of the contribution related to these layers is done by a more reduced software architecture than the INCOBAT SW architecture. Therefore, for further comparison and highlighting of the improvements of the thesis approach the 3 layer monitoring concept [79] is used as an evaluation use-case (depicted in Figure 4.10). This elementary use-case is an

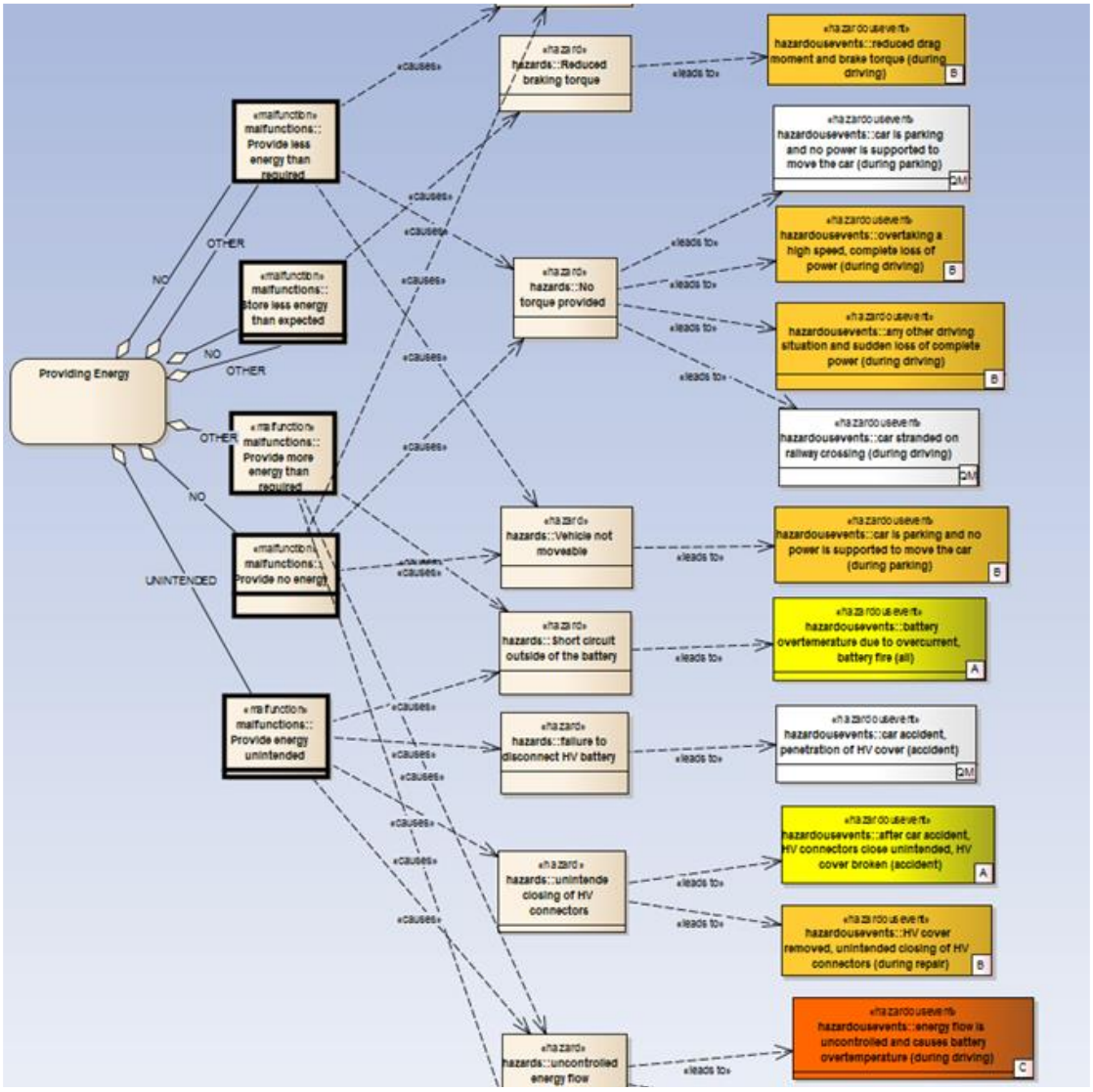


Figure 4.7: Excerpt of the HARA of the BMS Use-Case

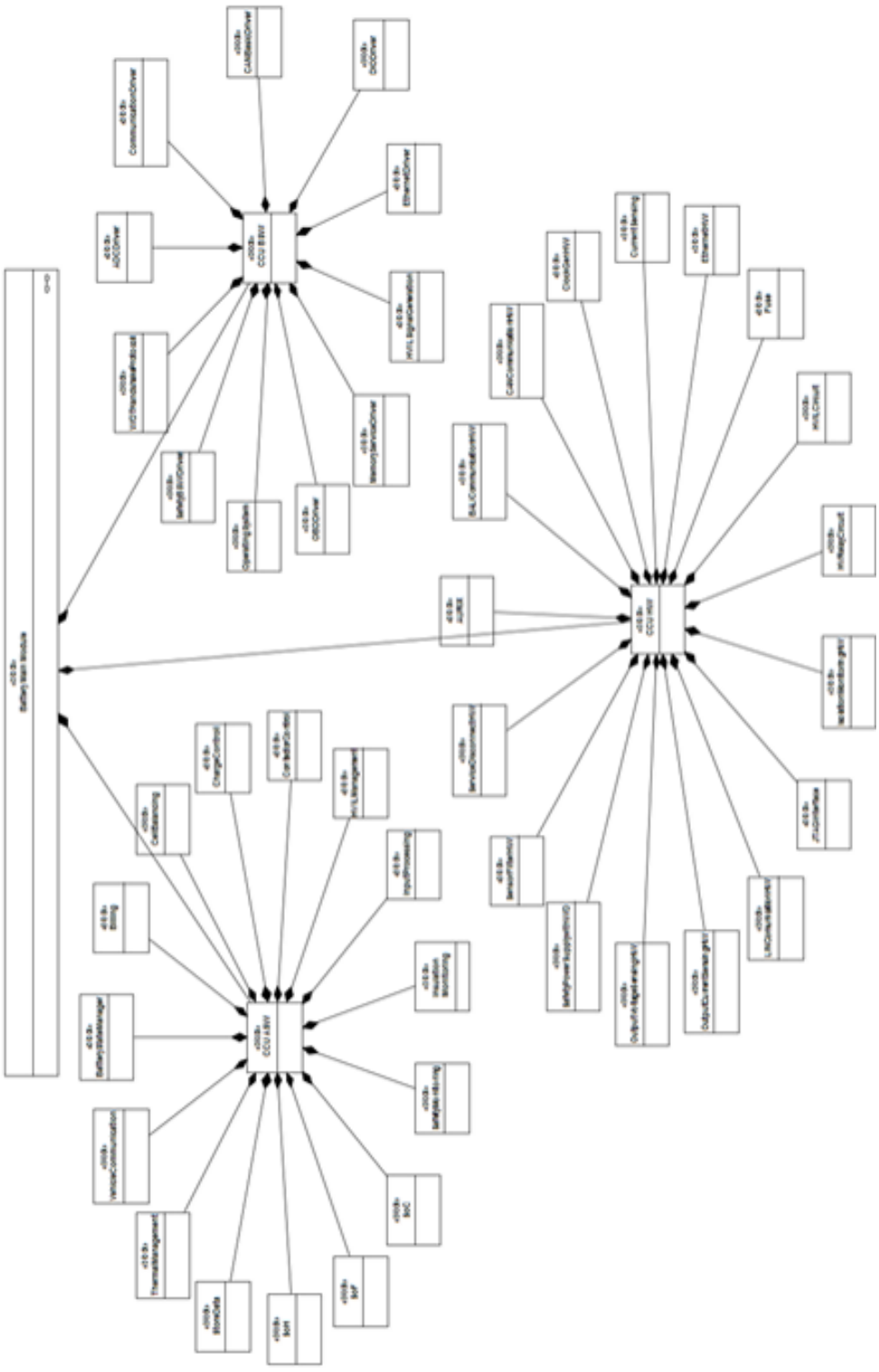


Figure 4.8: HW and SW Elements of the CCU of the Use-Case

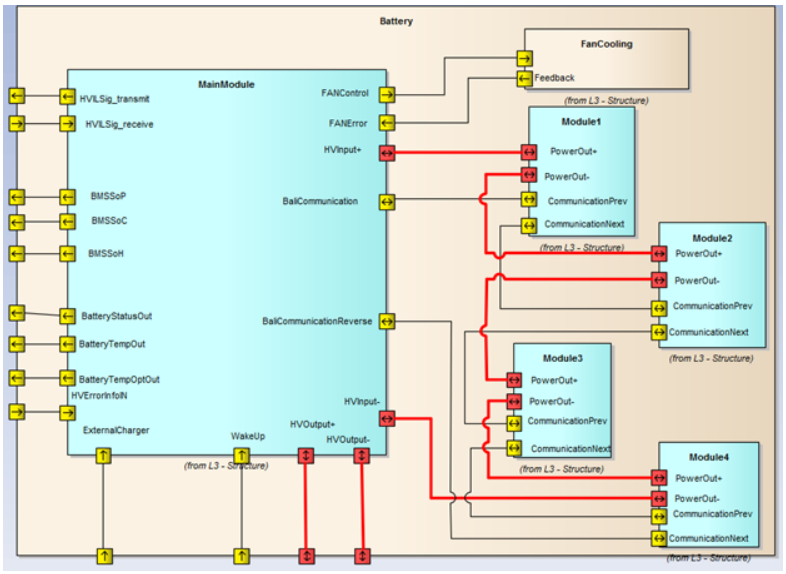


Figure 4.9: CCU Architecture (L3 Architecture)

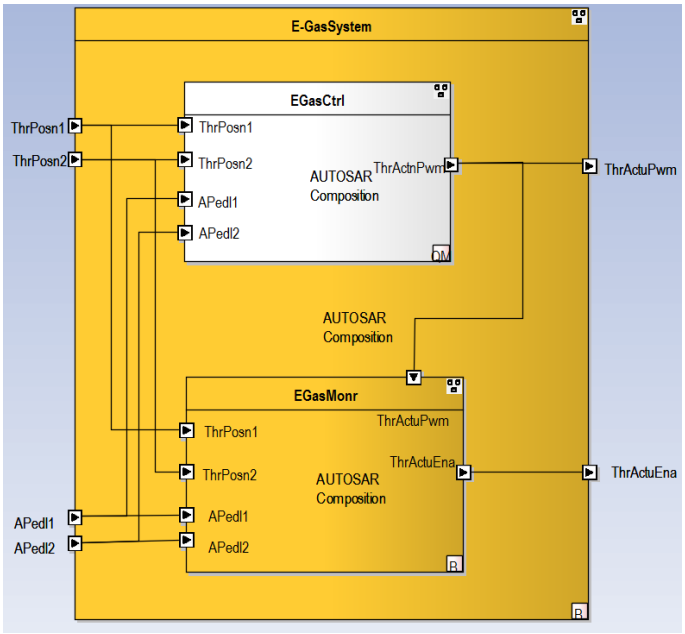


Figure 4.10: Top-Level Representation of SW Demonstration Use-Case in Enterprise Architect

illustrative material but nevertheless representative, due to the fact that several safety-related functions are based on this concept and its disclosed nature and commercial non-sensitivity.

4.5.1 Application Software Layer

As mentioned earlier in this section, this definition of the software architecture is usually done 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 4.10). This does not hamper the work of the software system architect but 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 development of safety-critical software in terms of traceability, replicability of design decisions, and unambiguously visualizes dependencies and puts visual emphasis on view-dependent constraints (such as graphical safety-critical highlighting of SW modules in Figure 4.10).

The presented 3 layer monitoring concept use-case consists of 7 ASW modules with 36 interfaces and 29 signal connections. Hereby the SW module representations contain 3 configurable attributes per element and the SW interface representations contain 10 attributes per element. Therefore, the use-case totals 41 ASW model artifacts with 381 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 has already become confusing. A manual transformation of the information represented within the models would already be cumbersome, error-prone, and would inherit lots of additional work to ensure consistency between the two models. A more complete overview of the use-case is given in Table 4.3.

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 (Listing 4.1 shows an excerpt of the Matlab API commands), which provides evidence and ensures completeness of the model transformation. The mentioned importer functionalities enable round-trip engineering and bi-directional updates of both models and therefore supports evidence for consistency of both models.

Listing 4.1: 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')
```

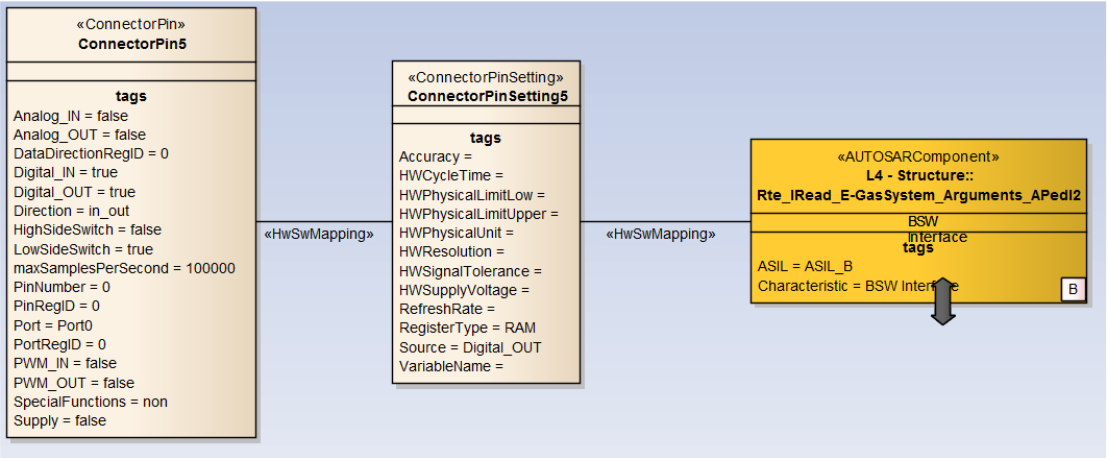


Figure 4.11: Screenshot of the BSW and HW Pin Representation within the System Development Tool

4.5.2 Basic Software and HW Abstraction Layer

Other key aspects for ISO 26262 aligned development are correct, consistent, and complete interface descriptions. Furthermore, evidence of correct implementation of these interfaces is required. This can be challenging in the case of concurrent HW and SW development and cross-dependencies of asynchronous update intervals. Therefore, the presented approach provides a single source of HSI definitions, available as a graphical HSI model and spreadsheet and automatically generates code configurations.

The definition of the 6 HW/SW interfaces with 10 parameters for each SW signal and 13 parameters for each HW pin totals 138 parameter configurations within the HSI spreadsheet template or in the MDB tool, which can be used to generate ASW/BSW interfaces and BSW configurations. Figure 4.11 depicts a HSI artifact mapping via BSW module and HW pin configuration within the system development tool. This leads to a file architecture as depicted in Figure 4.12. With the use of the approach 8 additional interfacing files with 481 lines of code (LoC) source and 288 LoC configuration have been generated.

Regarding the SW/SW Interfaces on the ASW layer, consistency checks for the 36 interfaces ensure point-to-point consistency of the signal routing. For 10 definable features per signal, this adds up to 360 definitions, which are automatically checked for consistency with this approach.

These interfacing files generate an interface layer (similar to AUTOSAR RTE) without relying on full AUTOSAR tooling support. In terms of safety-critical development this automatically generated interface layer supports traceability links between BSW configurations and HSI information and eliminates the need for manual interface source code rework.

4.5.3 Operating System Configuration

Lastly, the configuration of the operating system can also be generated and linked to the modeling artifacts via OIL configuration file generation and import. Figure 4.13 shows the graphical

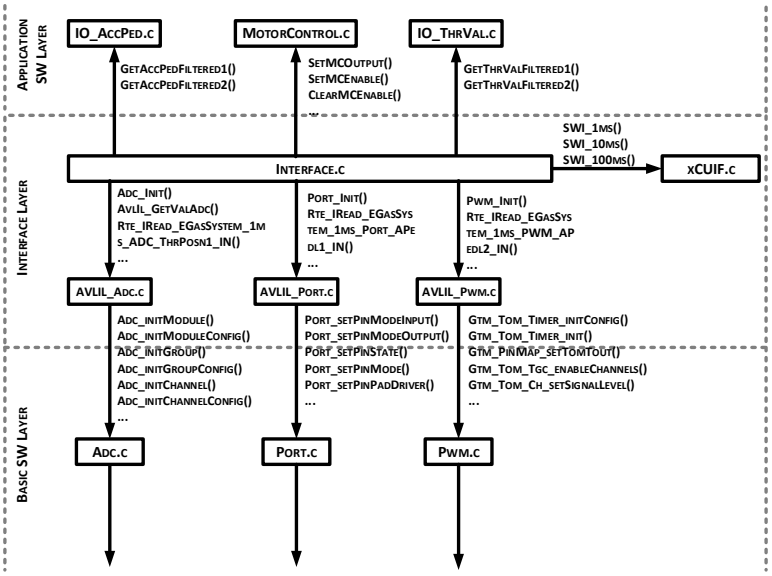


Figure 4.12: Excerpt of Generated Files for the BMS Use-Case

Table 4.3: Overview of the Evaluation Use-Case SW Architecture

Object type	**Element-count**	**Configurable Attributes per Element**
ASW Modules	7	3
BSW Modules	6	3
ASW Module Interfaces	36	10
ASW/BSW Interfaces	6	-
HW/SW Interfaces	6	13
CPU	3	2
OS	1	15
APPMODE	2	1
TASKS	6	9
COUNTER	1	5
ALARMS	6	6

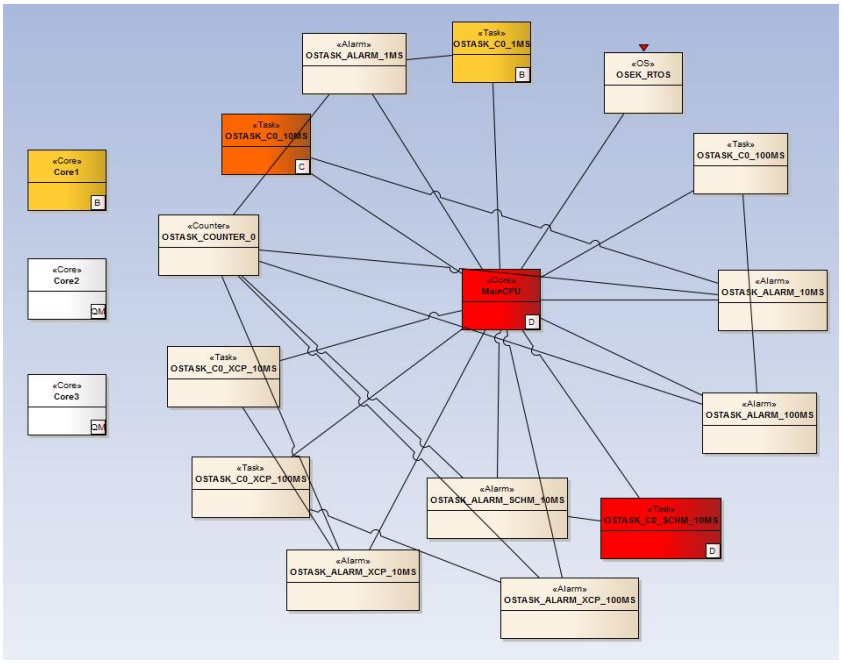


Figure 4.13: Modeling Artifacts representing the OS Configurations of the Use-Case

representation of the model artifacts required for configuration of the OS. This amounts to a total of 20 OIL objects and 46 relations between the elements for this use-case. Again, the number of relations between the elements increase quickly and become confusing with more complex use-cases. Nevertheless, this issue can be reduced by the model-based development approach by hiding specific relations, as also done in Figure 4.13. It can also be argued that this approach does not reduce the workload or significantly speed up the generation of OIL files, due to the high number of relations that need to be established. However, the approach provides guidance to minimize configuration failures. Additionally, it supports round-trip engineering features, which split workloads among different development phases and thus simplifies reuse. In terms of safety-critical development and reuse, these are crucial additional features. Furthermore, the approach reduces the need for manual generation of OIL files without adequate syntax and semantic checking support, ensuring reproducibility, and traceability argumentation.

5 Conclusion and Future Work

This chapter concludes this doctoral thesis by briefly summarizing the contributions and sketching potential future work and research. The dominant character of modern embedded automotive system development has been identified as the challenge of mastering the increased complexity and ensuring consistency of the development of system-wide features along the entire product life cycle. Automotive standards provide a process framework which requires efficient and consistent product development and tool support. Nevertheless, using various heterogeneous development tools hampers the efficiency and consistency of information flows. Additionally, a fine grasp of commonalities and cross-domain knowledge required for development of system-wide features promotes the growing gap between technology and required level of expertise.

5.1 Summary and Conclusion

This doctoral thesis approach is based on an MBD environment deployed at the industrial partner AVL and extends its applicability to the needs of development of embedded software for dependable (safety-critical, security-relevant, and/or high-reliable) multi-core systems to achieve a holistic system description. Thus, the approach aims at improving the comprehensive dependability argumentation of automotive multi-core systems based on model-based development.

For this aim the contributions of this doctoral thesis approach add values on tool, method, and process layer which have been identified as crucial for the industrialization of engineering approaches.

The main goals of this doctoral thesis are: (a) identification of challenges which appear when developing a dependable multi-core system especially in the automotive domain, (b) revealing strategies for the migration to multicore systems, and (c) supporting such development by an adequate development framework. Additionally, adequate analysis of the architecture design for system-wide function implementation has been mined and a merging of the heterogeneous development tools for embedded system engineering and SW engineering has been demonstrated by an automotive battery management system use-case.

5.2 Future Work

> Experience is never limited, and it is never complete... (Henry James)

This doctoral thesis contributes to state-of-the-art dependable multi-core system development in the automotive domain, it does not claim to be exhaustive or 'the holistic solution'. Therefore, there is space for future work and further improvements and the following paragraphs propose ideas and directions for future work with respect to the major contributions.

Dependability Analysis Methods

SDA and SAHARA approaches (related publications: **Paper F** and **Paper G**) provide a fundamental approach to quantify system-wide attributes other than safety (e.g. the combination of reliability and maintainability or security) at early development phases. Nevertheless, system dependability features have mutual impacts, similarities, and interdisciplinary values in common and there is a considerable overlap. Therefore, a further analysis of the presented methods for improvements in terms of applicability and enhancement of additional approaches for other development phases and more detailed design is recommended.

Cross-fertilization of Domain Knowledge

A combined approach for analysis of dependability features in early design phases of an embedded automotive system is proposed in publication **Paper H**. The objective of this work was to combine domain knowledge and provoke interdisciplinary development a cooperative dependability evaluation and a language base enabling dependable system development. Further efforts geared towards the cross-fertilization of domains and cooperative dependable development are suggested.

Multi-Core Migration

There is a current trend of replacing traditional mechanical systems with embedded systems and applying multi-core systems for safety-critical applications. This enables the deployment of more advanced control strategies and raises new challenges. Evidence of correctness of the different applications, both in the time domain and value domain has to be guaranteed.

These challenges received special attention within this work. However, additional research on resolving these challenges in a holistic manner in order to enable legacy SW to run on a multi-core platform and new control strategies to be deployed successfully can be considered. Hence, the challenge of efficiently combining expertise between multi-core computing platforms, lessons learned in other domains, and the automotive domain is focused by several EU founded projects.

Integration of Development Tools

This thesis focuses on development tools involved in the development of automotive systems from the system development level down to the software development. For this purpose, an approach which seamlessly combines the heterogeneous development tools involved has been proposed and a prototypical tool-bridge implementation has been made. Further improvements to the prototypical implementation as well as the usability and extensibility of the tool-bridge are considered. Moreover, research regarding novel multi-core related tools and timing analyzers is highly recommended. Additionally, tools involved in hardware development, as well as, validation and verification tools have not been focused on by this thesis.

6 Publications

Figure 3.1 show a depiction of this concept in GSN notation.

This doctoral thesis base on 13 main publications by the thesis author, which describe individual parts of the presented approach in more details. As mentioned the contributions of this doctoral thesis approach add values on tool, method, and process layer. Publications **Paper A** and **Paper B** are related to the process layer; Publications **Paper C** to **Paper H** contribute on method layer, whereas Publications **Paper I** to **Paper M** are focusing tool implementations on tool layer. Figure 6.1 illustrates the mapping of the publications to the different layers.

Paper A: G. Macher and C. Kreiner. *Model Transformation and Synchronization Process Patterns.* 20th European Conference on Pattern Languages of Programs -EuroPLoP '15, Irsee (Germany), July 08 - 12, 2015. **in press**

Paper B: G. Macher, E. Armengaud, and C. Kreiner. *Integration of Heterogeneous Tools to a Seamless Automotive Toolchain.* 22nd European Conference Systems, Software and Services Process Improvement - EuroSPI 2015, Ankara (Turkey), September 30 - October 2, 2015. **best paper award**

Paper C: G. Macher, H. Sporer, and C. Kreiner. *Automotive Safety Case Pattern.* 19th European Conference on Pattern Languages of Programs - EuroPLoP '14, Irsee (Germany), July 09 - 13, 2014.

Paper D: G. Macher, A. Hoeller, E. Armengaud, and C. Kreiner. *Automotive Embedded Software: Migration Challenges to Multi-Core Computing Platforms.* 13th IEEE International Conference on Industrial Informatics, Cambridge (United Kingdom), July 22 - 24, 2015.

Paper E: G. Macher, A. Hoeller, E. Armengaud, and C. Kreiner. *Pattern Catalog for Multi-Core Migration of Embedded Automotive Systems.* 20th European Conference on Pattern Languages of Programs - EuroPLoP '15, Irsee (Germany), July 08 - 12, 2015. **in press**

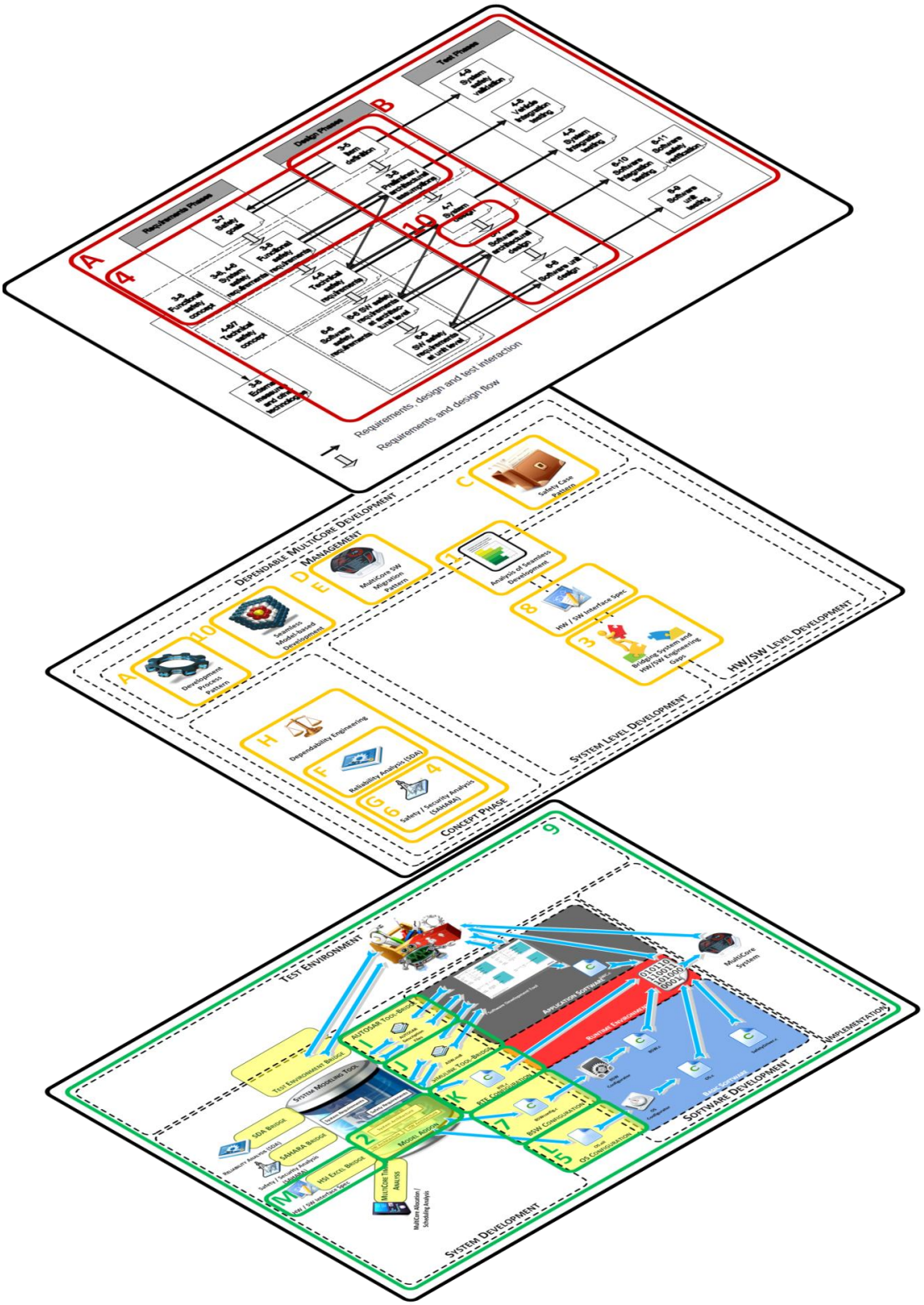


Figure 6.1: Mapping of Publication and Layers of Contribution

Paper F: G. Macher, A. Hoeller, H. Sporer, E. Armengaud, and C. Kreiner. *Service Deterioration Analysis (SDA): An Early Development Phase Reliability Analysis Method.* 45th Annual International Conference on Dependable Systems and Networks (DSN) - RADIANCE Workshop, Rio de Janeiro (Brazil), June 22 - 24, 2015.

Paper G: G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner. *SAHARA: A security-aware hazard and risk analysis method.* 18th Design, Automation Test in Europe Conference - DATE '15,Grenoble (France), March 09 - 13, 2015.

Paper H: G. Macher, A. Hoeller, H. Sporer, E. Armengaud, and C. Kreiner. *A Comprehensive Safety, Security, and Serviceability Assessment Method.* 34th International Conference Computer Safety, Reliability, and Security - SAFECOMP 2015, Delft (The Netherlands), September 23-25, 2015.

Paper I: G. Macher, E. Armengaud, and C. Kreiner. *Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures.* 12. Workshop Automotive Software Engineering (ASE), Stuttgart (Germany), September 22 -26, 2014.

Paper J: G. Macher, H. Sporer, E. Armengaud, E. Brenner, and C. Kreiner. *A Seamless Model-Transformation between System and Software Development Tools.* 8th European Congress Embedded Real Time Software and Systems - ERTS 2016, Toulouse (France), January 27 - 29, 2016. **in press**

Paper K: G. Macher, R. Obendrauf, E. Armengaud, E. Brenner, and C. Kreiner. *RTE Generation and BSW Configuration Tool-Extension for Embedded Automotive Systems.* 8th European Congress Embedded Real Time Software and Systems - ERTS 2016, Toulouse (France), January 27 - 29, 2016. **in press**

Paper L: G. Macher, M. Atas, E. Armengaud, and C. Kreiner. *A Model-Based Configuration Approach for Automotive Real-Time Operating Systems.* SAE International Journal on Passenger Cars - Electronics and Electronical Systems, 2015.

Paper M: G. Macher, H. Sporer, E. Armengaud, and C. Kreiner. *A Versatile Approach for ISO26262 compliant Hardware-Software Interface Definition with Model-based Development.* SAE World Congress & Exhibition 2015, Detroit (Michigan, USA), April 21 -23, 2015.

Related publications not included in this Thesis (only principal authorship papers)

1. G. Macher. *Seamless Model-Based Safety Engineering from Requirement to Implementation.* In Proceedings of Doctoral Symposium co-located with 17th International Conference on Model Driven Engineering Languages and Systems (2014), Valencia, Spain, September 30, 2014.
2. G. Macher, E. Armengaud, E. Brenner, and C. Kreiner. *An Automotive Engineering Model to Improve the Architectural Design Interchange Continuity.* In SAE Technical Paper, pages 001 - 007, 2016. **under review**
3. 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.
4. G. Macher, E. Armengaud, and C. Kreiner. *A Practical Approach to Classification of Safety and Security Risks.* In EuroSPI 2015 Industrial Proceedings, pages 10.1 - 10.10. WHITEBOX, Denmark, 2015.
5. G. Macher, M. Atas, E. Armengaud, and C. Kreiner. *Automotive Real-time Operating Systems: A Model-Based Configuration Approach.* In ACM SIGBED Review Special Interest Group on Embedded Systems, volume 1291 of CEUR Workshop Proceedings. Association for Computing Machinery. Special Interest Group on Embedded, 2014.
6. G. Macher, A. Hoeller, H. Sporer, E. Armengaud, and C. Kreiner. *A Combined Safety-Hazards and Security-Threat Analysis Method for Automotive Systems.* In Proceedings of SAFECOMP 2015 Workshops ASSURE, DECSoS, ISSE, ReSA4CI, and SASSUR, volume LNCS 9338 of Lecture Notes in Computer Science, pages 237 - 250. Springer International Publishing AG, 2015.
7. G. Macher, R. Obendrauf, E. Armengaud, and C. Kreiner. *Automated Generation of Basic Software Configuration of Embedded Systems.* In Proceedings of the 2015 Research in Adaptive and Convergent Systems (RACS 2015), pages 461 - 465, 2015.
8. G. Macher, H. Sporer, E. Armengaud, E. Brenner, and C. Kreiner. *Using Modelbased Development for ISO26262 aligned HSI Definition.* In EDCC Conference Proceedings, 2015.
9. G. Macher, M. Stolz, E. Armengaud, and C. Kreiner. *An Automotive Model-Based System, Safety, and Software Engineering Tool Approach.* In 8th Grazer Symposium Virtuelles Fahrzeug, 2015.
10. G. Macher, M. Stolz, E. Armengaud, and C. Kreiner. *Filling the Gap between Automotive Systems, Safety, and Software Engineering.* e&i Journal, 132(3), pages 142 - 148, June 2015.
11. G. Macher, A. Hoeller, E. Armengaud, E. Brenner, and C. Kreiner. *Combined Safety and Security Development of Automotive Systems.* In Design Autmoation Conference (DAC'2016) Proceedings, pages 001 - 006, 2016. **under review**

Model Transformation and Synchronization Process Patterns

Georg MACHER, Graz University of Technology
Christian KREINER, Graz University of Technology

Embedded systems are already integrated into our everyday life and play a central role in all domains including automotive, aerospace, healthcare or industry. The complexity of embedded systems and software has grown significantly in recent years. Software's impact on embedded system's functionality, has led to an enormous increase of SW complexity, while reduction of innovation cycles and growing demand for extra-functional requirements.
Supporting cooperation between the involved domain and SW development experts to combine their expertise is a core challenge in embedded software development. Nevertheless, today, a lack of tool support and integration makes it impossible to cover the complete development life cycle using model-driven development (MDD) paradigms.
This paper identifies patterns of concurrent workflows in embedded system development which can be used to identify dependencies and consequences of concurrency of workflows and thus, highlight the basic problem and provide know-how how to overcome these issues and foster MDD along the development life cycle.

Categories and Subject Descriptors: H.5.0 [**Information Interfaces and Presentation**]: General—; H.1.1 [**Models and Principles**]: System and Information Theory—; K.2.2 [**Computers and Society**]: Social Issues—

General Terms: Model Transformation and Synchronization

Additional Key Words and Phrases: embedded systems, model-driven development, concurrent workflow pattern.

ACM Reference Format:

Macher G. and Kreiner C., 2015. Model Transformation and Synchronization Process Patterns. jn 0, 0, Article 0 (0), 11 pages.

1. INTRODUCTION

Embedded systems are already integrated into our everyday life and play a central role in all domains including automotive, aerospace, healthcare or industry. In 2010, the embedded systems market accounted for almost 852 billion dollar, and is expected to reach 1.5 trillion by 2015 (assuming an annual growth rate of 12%) [Petrissans et al. 2012]. As an example, in the automotive industry, embedded systems components are responsible for 25% of vehicle costs, while the added value from electronics components range between 40% for traditional vehicle up to 75% for electrics and hybrid vehicles [Scuro 2012]. Current premium cars imply more than 70 electronic control units (ECU) with close to 1 Gigabyte software code [Ebert and Jones 2009] implemented.

This trend is also strongly supported by the ongoing replacement of traditional mechanical systems with modern embedded systems. This enables the deployment of more advanced control strategies, providing added values for the customer and more environment friendly vehicles. At the same time, automotive multi-core computing platforms enable the deployment of more advanced control strategies and a higher degree of integration, which leads to cost savings by reducing the number of ECUs. On the contrary, the higher degree of integration and the safety-criticality of the control application increases the system's complexity and raises new challenges. Safety standards such as

EuroPloP '15, July 08 - 12, 2015, Kaufbeuren, Germany

ACM 978-1-4503-3847-9/15/07 ...$15.00.
http://dx.doi.org/10.1145/2855321.2855347

ISO 26262 [ISO - International Organization for Standardization 2011] 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. The key challenge in this context is to provide evidence of consistency during product development among the different work-products.

Model-driven development (MDD) is alleviating the issue of inherent complexity and the most promising approach of interdisciplinary development [Broy et al. 2008], but in case of embedded system development seamless cooperation between the involved domains and development experts is a core challenge. Today, a lack of tool support and integration makes it impossible to cover the complete development life cycle using MDD.

During model-driven development of embedded systems frequently manifold model types of the same system under development exist spread over different stages of the development process and different development tools in use. In these cases special attention need to be paid to keep the dependent models consistent. Each model transformation implies potential sources for ambiguous mapping and often dependencies between the models are hard to identify.

This paper identifies patterns of concurrent workflows in embedded system development which can be used to identify dependencies and consequences of parallel workflows. Thus, a highlighting of the basic problem and providing of know-how how to overcome these issues as well as encouragement of MDD along the development life cycle is done.

In the course of this document, a description of the state of the art and related works is given in Section 2. The paper presents, according to our knowledge, three previously undocumented but common dependability patterns in concurrent embedded system development workflow FORWARD UPDATE DEPENDENCY RELATION (Section 4), BACKWARD UPDATE DEPENDENCY RELATION (Section 5) and BIDIRECTIONAL UPDATE DEPENDENCY RELATION (Section 6).

The intention of the three patterns presented in this work is to identify update dependencies between concurrent models and thereby set up the base for tool integration into a holistic tool-chain. The patterns do not need to be applied in a specific order, but are related to another. This is due to the fact that either one or the other shall be applied in case of concurrent development of dependent development models. In case of solely uni-directional dependability and independent workflows (e.g. generation of interim status model-based documentations), the FORWARD UPDATE DEPENDENCY RELATION pattern shall be applied. If feedback of information from one model to another is required after some refinement activity (e.g. updating of SW architecture model with SW function models), the BACKWARD UPDATE DEPENDENCY RELATION pattern shall be applied. Finally, the BIDIRECTIONAL UPDATE DEPENDENCY RELATION pattern shall be used if a full concurrency development of mutually dependent models is required (e.g. concurrent SW and HW development).

The patterns have been mined during the analysis of the actual model-driven development work-flow at our industrial partner. The aim of this analysis was the exploration of open issues and interface shortcomings of the MDD tools in place to setup a seamless model-driven development tool-chain covering all development phases. During this analysis the three types of relations between concurrent MDD models have been investigated. The identification of these relations serve the basis for a bridging of the ascetic MDD tools to a seamless tool-chain.

2. RELATED WORK

Several existing approaches deal with model-driven development of embedded systems. Nevertheless, only a few number of publications focus on holistic tool-chains and methodologies. Still many system-wide and cross-domain constraints, such as dependability features (safety or security), need to be exchanged manually or without adequate methodical support across tool- and domain borders. Throughout our research we discovered several challenges.

Model-driven systems and software development as well as tool integration are engineering domains and research topics aim at moving the development steps closer together and thus improving the consistency of the system over the expertise and domain boundaries. Although seamless solutions have not been achieved so far due

to problems by using an inadequate tool-chain (e.g. redundancy, inconsistency and lack of automation) [Broy et al. 2008] or the usage of different specialized models for specific aspects at different development stages with varying abstraction levels [Holtmann et al. 2011]. Traceability between these different models is commonly established via manual linking due to a lack of automation and missing guidance.

Two important gaps need to be bridged: First, missing links between system level tools and software development tools. Second, several very specific and non-interacting tools which require manual synchronization and therefore often rely on inconsistent, redundant information need to be aligned to one single source of information.

Therefore, system design models have to be correctly transferred to the software engineering model, and later changes must be kept consistent [Giese et al. 2010]. A major drawback thereby stems from the bidirectional model transformation, each transformation step implies potential sources for ambiguous mapping and model mismatching. Recent projects which focus on model-driven development environments for automotive multi-core systems are the AMALTHEA Project[1], SAFE Project[2], and MAENAD Project[3].

The work of Asplund et. al [Asplund et al. 2012] focus on tool integration in context of the automotive safety standard ISO 26262. The authors mention that tool integration is a complicated issue and guidance regarding tool integration into tool chains are very limited. Nevertheless, their work focuses more on tool qualification issues in ISO 26262 context and does not provide the mentioned guidance for tool integration.

Finally, the work of Raschke et. al [Raschke et al. 2014] bases on the concept that development of systems is a sequence of evolutions, interrupted by revolutions. Therefore the authors build up a catalog of patterns for software modeling. This pattern catalog focuses on the dependencies between software models and software code and their possible relations.

3. SPECIFICATION OF THE PATTERN

The following sections will discuss the patterns in more detailed manner. Therefore we specify the structure [Salingaros 2000; Alexander et al. 1977] in which we describe the three update-dependability patterns:

—Context - Describes the situation in which the pattern can be applied and in which the problem occurs.

—Problem - States the scenario which requires action to be taken and problems the pattern shall solve.

—Forces - Gives additional motivation for usage of the pattern and describes the constraints which shape the specific solution for the problem.

—Solution - Describes the steps and actions to be taken to solve the existing problem in the focus of the given context and specific forces.

—Consequences - The application of the pattern solves the problem, but also results in some consequences, which themselves also include side effects.

—Known Uses - publications which used the pattern

—Example - Characterizes the context and problem scenario by a striking example.

[1] http://www.amalthea-project.org/
[2] http://safe-project.eu/
[3] http://maenad.eu/

3.1 Common Pattern Context

Common for all three patterns is an iterative model-driven development process of an embedded system. Figure 1 shows all phases of such a development life cycle with the involved tooling discontinuities. As can be seen in this figure, model-driven development iteratively updates the model over the time. This process lets developers produce partial implementations after each iteration. The figure highlights necessary tool transitions via the different colored backgrounds of the iteration steps and also indicates a potential pitfall of such an iterative model-driven development approach. Frequently manifold model types of the same system under development exist spread over different stages of the development process and different development tools in use. In case of such a concurrent development of dependent development models special attention need to be paid to keep the dependent models consistent. Each transformation step implies potential sources for ambiguous mapping and a common model as single source of information is rather unusual or too complex for application. For the course of this document this process is stretched over time, which leads to a corkscrew like representation.

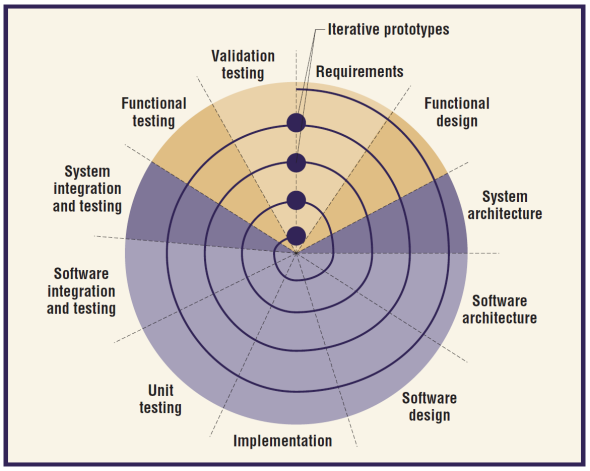


Fig. 1. Iterative Model-Driven Development Process for Embedded Systems [Liggesmeyer and Trapp 2009]

4. FORWARD UPDATE DEPENDENCY RELATION PATTERN

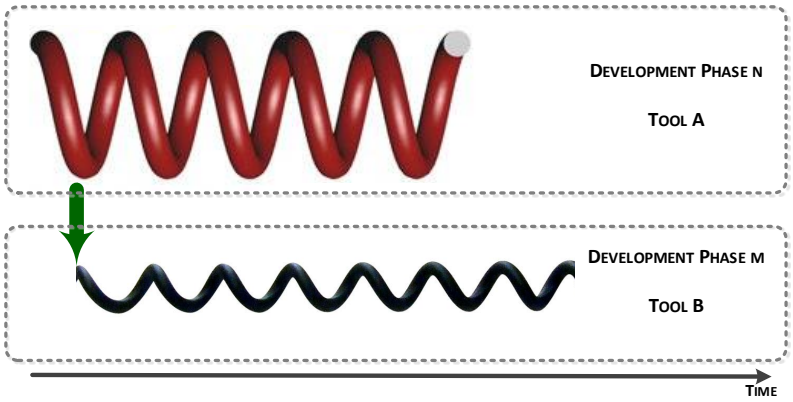


Fig. 2. Forward Update Dependency Relation

4.1 Context

The model-driven development follows an iterative model like depicted in Figure 1. After a certain time period the current model status needs to be tested to verify the current development status and development progress. Meanwhile the development of the model cannot continue until this verification is done.

4.2 Problem

The model-driven development lifecycle requires status and progress measures of current development status of the model. The development of the model cannot be continued until this step has been finished and the ongoing refinement of the model is delayed.

4.3 Forces

—work-product generation from current model status required

—some work-products of the iterative model-driven development can be carried out independently of the rest of the development

—some work-products can be generated from the current information status in parallel without hampering the other development process

—some work-products do not need feedback of generated artifacts

—concurrent development/generation of work-product is independent from main development workflow

—different MDD steps require different skills, which can be carried out concurrently by different teams

4.4 Solution

Enable concurrent model-driven development by generating related models from the original model. The generation of a related model enables concurrent development without hampering the development of the original model.

In such a case, a model is generated and iteratively updated like depicted in Figure 1. At a certain point in time a related model is generated from the current status of the original model (see Figure 2). This related model inherits all information required for the new model, but does not necessarily need to inherit all available information. The

basis for the generation of the related model can be of any type computable by the development tool in use for the new model (such as XMI export, AUTOSAR XML file, or information transfer direct via API). References between the related model and the generation basis information shall be linked traceable for later model analysis.

Further changes and ongoing development of the two concurrent models must not be mutually affecting. Moreover, triggering of additional independent model-driven development may be done at any phase of the development. The pattern is also applicable for parallelization of independent MDD work-products.

4.5 Consequences

+ Enables parallelization of concurrent, independent MDD models.
+ Concurrent development without delaying main development cycle is possible, due availability of related models.
+ Productivity can be increased due to parallelization of models and concurrent working on related models.
+ Establish a basis for attendant trend analysis and validation without hampering development progress (in case of generation of related model for analysis purpose).
- Derived models basis is outdated with next iteration step of the source model.
- Tedious rework is required if creation of derived model is not supported via a dedicated tool.
- Errors corrected in one model will not be corrected in the other model.

4.6 Known Uses

This pattern can be seen as standard way of generating source code from models or generating documentation of project metrics. Further example is a typical top-down approach. A final requirement model may be transferred from a requirement management tool to a system development tool to start the design of the system under development.

4.7 Example

Consider the following example from software domain. Typically after a fixed time period or at a certain milestone the status of the SW development model is handed over to testing department to verify the current status and progress of development. Meanwhile the development of the SW model continues and does not represent the tested status anymore.

5. BACKWARD UPDATE DEPENDENCY RELATION PATTERN

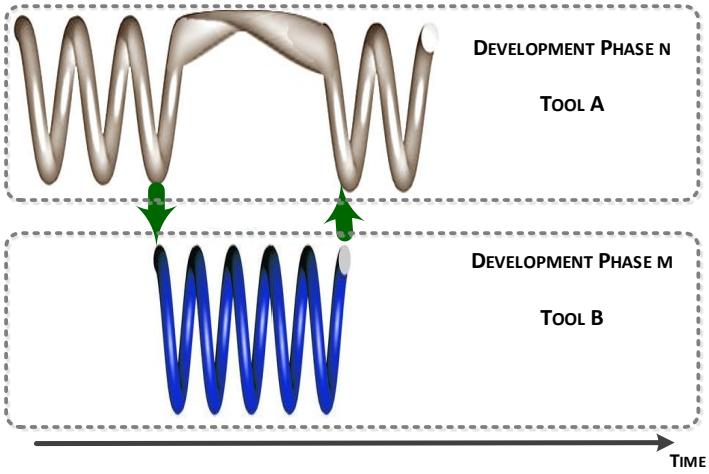


Fig. 3. Backward Update Dependability Relation

5.1 Context

Application of the FORWARD UPDATE DEPENDENCY RELATION pattern leads to the generation of more detailed model artifacts also required in original model or additional enhancements of the original model. During this time the equivalent artifacts of the original model do not change (independent artifacts of the original model may undergo changes).

In other words, the original model is improved by the related model, but the base the related model remained unchanged in the original model.

5.2 Problem

A set of properties related to both models can be refined easier or only in the derived model. Nevertheless, these refinements need to be also updated in the original model.

5.3 Forces

—Rework and enhancement also of generated work-products required.

—Refinement has to be restored into the main development life-cycle.

—Loosely coupled dependencies between main work-product and generated work-product.

—Information refinement required for original model, although not adequately supported.

5.4 Solution

Refine parts of the derived model and keep the related model artifacts in the original model unchanged. After finalization of changes (done in derived model), update the original model with artifact of the derived model.

For this purpose, a related model which is supporting further model refinement more adequately shall be generated by using the FORWARD UPDATE DEPENDENCY RELATION pattern 4. After finalization of iterative refinement of the derived model (depicted in Figure 1), information inherited by the derived model are more accurate than the data of the original model. These updates of the related model have to be handed over back to

the source (original model). Old values of the original model are replaced by the new data and deleting or adding of artifacts must be supported. Therefore, the data basis (data of original model used for generation of derived model) must not change during development-time of the derived model (indicated in Figure 3).

5.5 Consequences

+ Allows feedback of refinements for concurrent work-product (due to update functionality of the original model).
+ Enables quasi- parallel development in case of loosely coupled work-product dependencies, due to the fact that original model can be the origin of numerous independent derived models which feedback their work-products after finalization of development.
+ Enables development of optimal solutions of smaller parts of the over-all work-product, also due to concurrent development on special purpose models.
+ Enables transformation of information between special purpose tools.
- Solely for loosely coupled dependencies, system-wide constraints may not be adequately described by derived special-purpose models.
- Source model data serving as basis for derived model must not change over lifetime of derived model, otherwise inconsistency of data and loss of information may occur.
- Merging of multiple derived models can be complicated.

5.6 Known Uses

This pattern is applicable in case of transferring of information to special purpose tool and tracing back of previously not available additional information (such as transferring of SW function to task scheduling tool and feedback of valid schedule tables). Another example, also related to MDD of SW, is a SW architecture representation in a system level development tool (described in more details in the example section).

5.7 Example

An example from MDD software development domain: A first design of the software architecture and its interface is done in context of system engineering to establish a hardware software interface document and enable concurrent development of HW and SW models. This SW architecture is transferred (e.g. application of FORWARD UPDATE DEPENDENCY RELATION pattern) into a more concrete model for SW development (such as Matlab/Simulink models). During SW development the SW interfaces and the SW architecture itself change in the concrete SW model (Matlab/Simulink model), while in the original model (at system development) the SW architecture artifacts remains unchanged. Nevertheless, these changes (done in Matlab/Simulink model) need to be updated in the original model (system SW architecture representation).

6. BIDIRECTIONAL UPDATE DEPENDENCY RELATION PATTERN

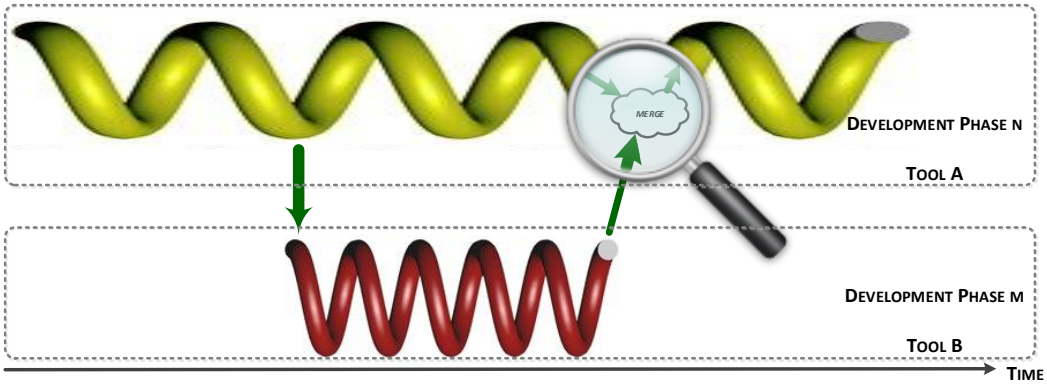


Fig. 4. Bidirectional Update Dependency Relation

6.1 Context

Application of the BACKWARD UPDATE DEPENDENCY RELATION pattern leads to problems if the original model has been evoluted concurrently with the development of the derived model. Updating of model artifacts of the original model with information from the derived model leads to information losses and is not possible.

To put it differently, both models change their common artifacts. Hence, a form of synchronization is required to conjoin the enhancements of both models.

6.2 Problem

Concurrent development processes with mutual impacts on each other must lead to consistent result, regardless of sequential order, and mutual dependencies.

6.3 Forces

—Limited time-to-market and different required domain-expertise require a structuring to ensure concurrent working of departments.

—Constraints affecting the system as a whole are too complex to be managed as a whole.

—Cross-domain constraints cannot be separated and affect independent parts of the model.

6.4 Solution

Make data serving as basis for related models explicit. Change data in a concurrent model and trigger synchronization of other models.

The BIDIRECTIONAL UPDATE DEPENDENCY RELATION pattern shall be used for full concurrency development of mutually dependent models. Incorporate the FORWARD UPDATE DEPENDENCY RELATION pattern and the BACKWARD UPDATE DEPENDENCY RELATION pattern, including the synchronization of both. Figure 4 shows that both models are updated in parallel and change information on which they mutual dependent. As can be seen in the figure, this pattern also requires additional merging strategies to prevent from data corruption or race-conditions.

Therefore, data serving as basis for concurrent models must be highlighted and made explicit. Changes of the data basis must be synchronized with the representation within the other model. Hence, synchronization

techniques, such as *diff* and *merge*, are required. If a merging is not supported at least diff must be available to support the review of changes.

For starting of concurrent development models, the data serving as basis for the models are used to generate the concurrent models and highlighted as shared data. The development of the two concurrent models can then be independent from another, as long as the shared data do not change. In case of a change of the shared data, a merging of the new information of both models is required. Supporting of such a synchronization is not easy and requires an adequate synchronization mechanism. Nevertheless, synchronization of the models is crucial and has to be managed with care.

6.5 Consequences

+ Splitting of work packages allows a separation of responsibilities and concurrent development of individual work-packages.
+ Divided responsibilities support optimized solutions of individual work-packages, due to working in expert groups with special-purpose models.
+ Split of responsibility of overall argumentation to domain experts for their sub-parts (possible due to splitting of models into special-purpose model).
- Additional consolidation of the work-packages and overall system required (especially for system-wide features).
- Individual optimal solutions may not result in the overall optimal solution (aka Vasa syndrome).
- Requires additional merging activities of the work-packages (due to concurrent changing of models and to prevent from information losses).
- Requires tracing of changes and change impact for merging activity (minor changes in one concurrent model may lead to major changes in the related models).

6.6 Known Uses

Classical approach of concurrent hardware and software development of embedded systems. Related patterns are: *Cooperative Application Lifecycle Management* pattern and *Coordinative Application Lifecycle Management* pattern.

6.7 Example

For example: A first design of the software architecture and its interface is done in context of system engineering to establish a hardware software interface document and enable concurrent development of HW and SW models. This SW architecture is transferred (e.g. application of FORWARD UPDATE DEPENDENCY RELATION pattern) into a more concrete model for SW development (such as Matlab/Simulink models). During SW development some SW interfaces change (e.g. signal resolution) in the concrete SW model (Matlab/Simulink model), while also in the original model (at system development) the SW architecture interfaces change due to some miss-alignment at HW level (e.g. signal is remapped to another pin). If BACKWARD UPDATE DEPENDENCY RELATION pattern would be used to update the original model the changes done in concurrency get lost or inconsistency occur.

7. CONCLUSION

To sum up, the three presented patterns present concurrent workflows of the embedded system development domain. The pattern can be used to identify dependencies and consequences of concurrent workflows and thus, enable to cover the whole development life cycle using model-driven development approaches. This paper highlights these dependencies to enable the identification of such demands and overcome tool-breaks with adequate tool-bridging approaches.

Acknowledgments

The authors would like to express their thanks to our shepherd Johannes Koskinen who had a determining influence on the improvement of our paper and untiringly helped to improve the maturity of the paper in several iterations.

REFERENCES

ALEXANDER, C., ISHIKAWA, S., SILVERSTEIN, M., JACOBSON, M., FIKSDAHL-KING, I., AND ANGEL, S. 1977. A Pattern Language. *Oxford University Press, New York*.

ASPLUND, F., BIEHL, M., EL-KHOURY, J., FREDE, D., AND THOERNGREN, M. 2012. Tool Integration, from Tool to Tool Chain with ISO 26262. In *SAE Technical Paper*. SAE International.

BROY, M., FEILKAS, M., HERRMANNSDOERFER, M., MERENDA, S., AND RATIU, D. 2008. Seamless Model-based Development: from Isolated Tool to Integrated Model Engineering Environments. *IEEE Magazin*.

EBERT, C. AND JONES, C. 2009. Embedded Software: Facts, Figures, and Future. *IEEE Computer Society 0018-9162/09*, 42–52.

GIESE, H., HILDEBRANDT, S., AND NEUMANN, S. 2010. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *LNCS 5765*, pp. 555 –579.

HOLTMANN, J., MEYER, J., AND MEYER, M. 2011. A Seamless Model-Based Development Process for Automotive Systems.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. 2011. ISO 26262 Road vehicles Functional Safety Part 1-10.

LIGGESMEYER, P. AND TRAPP, M. 2009. Trends in embedded software engineering. *IEEE Software 26,* 3, 19–25.

PETRISSANS, A., KRAWCZYK, S., VERONESI, L., CATTANEO, G., FEENEY, N., AND MEUNIER, C. 2012. Design of Future Embedded Systems Toward System of Systems - Trends and Challenges. European Commission.

RASCHKE, W., ZILLI, M., LOINIG, J., WEISS, R., STEGER, C., AND KREINER, C. 2014. Patterns of Software Modeling. In *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, R. Meersman, H. Panetto, A. Mishra, R. Valencia-Garcia, L. de Silva, I. Ciuciu, F. Ferri, G. Weichhart, T. Moser, M. Bezzi, and H. Chan, Eds. Lecture Notes in Computer Science Series, vol. 8842. Springer Berlin Heidelberg, 428–437.

SALINGAROS, N. 2000. The Structure of Pattern Languages. *Architectural Research Quarterly 4*, 149–161.

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

EuroPloP '15, July 08 - 12, 2015, Kaufbeuren, Germany
 ACM 978-1-4503-3847-9/15/07 ...$15.00.
http://dx.doi.org/10.1145/2855321.2855347

Integration of Heterogeneous Tools to a Seamless Automotive Toolchain

Georg Macher[1,2], Eric Armengaud[2], and Christian Kreiner[1]

[1] Institute for Technical Informatics
Graz University of Technology
{georg.macher, christian.kreiner}@tugraz.at
[2] AVL List GmbH
{georg.macher, eric.armengaud}@avl.com

Abstract. Modern embedded multi-core system platforms are key innovation drivers in the automotive domain.
The challenge, 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 and automotive SPICE require efficient and consistent product development and tool support. The existing solutions are still frequently insufficient when transforming system models with a higher level of abstraction to more concrete software engineering models.
The aim of this work is to improve the information interchange continuity of architectural designs from system development level to software development level (Automotive SPICE ENG.3 and ENG.5 respectively ISO 26262 4-7 System design and 6-7 SW architectural design). An approach for seamlessly combining the development tools involved is thus proposed. This approach merges the heterogeneous tools required for development of automotive safety-critical multi-core systems to support seamless information interchange across tool boundaries.

Keywords: ISO 26262, automotive SPICE, automotive systems, multi-core, architectural design, traceability.

1 Introduction

Embedded systems are already integrated into our everyday life and play a central role in many vital sectors including automotive, aerospace, healthcare, industry, energy, or consumer electronics. Current premium cars implement more than 90 electronic control units (ECU) with close to 1 Gigabyte software code [9], are responsible for 25% of vehicle costs and an added value between 40% and 75% [27]. 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 the environment, but at the same time the higher degree of integration and criticality of the control application raise new challenges. Today's information society strongly supports inter-system communication (Car2X) also in the automotive domain. Consequently

the boundaries of application domains are disappearing even faster than previously due to the replacement of traditional mechanical systems. At the same time, multi-core and many-core computing platforms are becoming available for safety-critical real-time applications. These factors cause multiple cross-domain collaborations and interactions in the face of the challenge to master the increased complexity and ensure consistency of the development along the entire product life cycle.

Consequently, this work focuses on improving the continuity of information interchange of architectural designs from system development level (Automotive SPICE [29] ENG.3 respectively ISO 26262 [16] 4-7 System design) to software development level (Automotive SPICE ENG.5 respectively ISO 26262 6-7 SW architectural design). We further make a special focus on safety-critical multi-core system development, due to the higher complexity of such systems and limited availability of supporting methods and tools.

The aim of this work is to merge the heterogeneous tools required for development of automotive safety-critical multi-core systems (especially during transition phase from system to SW development) and establish a single source of information concept to ease cross-domain consolidation.

The document is organized as follows: Section 2 presents an overview of related works. In Section 3 a description of the proposed approach and a detailed depiction of the contribution parts is provided. An implementation prototype and a brief evaluation of the approach is presented in Section 4. Finally, this work is concluded in Section 5 with an overview of the approach presented.

2 Related Works

Model-based development in general and the development of embedded automotive systems in particular are engineering domains and research topics aimed at moving the development process to a more automated work-flow, which improves in terms of consistency and tackles the complexity of the development process across expertise and domain boundaries. Recent publications are either related to AUTOSAR [2] methodology based approaches, focus ISO 26262 [16] related methods, model-based development, multi-core system development or deal with Automotive SPICE [29] concept.

2.1 Publications related to Model-based Development

In [18], the authors describe a framework for a seamless configuration process for the development of automotive embedded software. The framework is also based on AUTOSAR, which defines architecture, methodology, and application interfaces. The steps through this configuration process are established by a system configuration and ECU configuration.

An approach for a design of a vehicular code generator for distributed automotive systems is addressed by Jo et al. [17]. The authors mention the increasing complexity of the development of automotive embedded software and systems and the manual generation of software leading to more and more software defects and problems. Thus the authors integrated a run-time environment (RTE) module into the earlier development tool to design and evolve an automated embedded code generator with a predefined generation process.

An important topic to deal with in general terms is the gap between system architecture and software architecture. Broy et al. [6] claim model-based development to be the best approach to manage large complexity of modern embedded systems and provide an overview of basic concepts and theories. The work illustrates why seamless solutions have not been achieved so far, it mentions commonly used solutions and the problems arising from the use of an inadequate tool-chain (e.g. redundancy, inconsistency and lack of automation).

The work of Quadri and Sadovykh [23] presents a model-driven engineering approach aiming to develop novel model-driven techniques and new tools supporting design, validation, and simulation. These authors defined profiles using a subset of UML and SysML for their approach and mentioned the usage of effective design tools and methodologies as crucial to be capable of managing complex real-time embedded systems.

The work of Holtmann et al. [14] 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 and traceability between these different models is commonly established via manual linking.

Chen et. al. [7] present an approach that bridges the gap between model-based systems engineering and the safety process of automotive embedded systems. More recently also the MAENAD Project [3] is focusing on design methodologies for electric vehicles based on EAST-ADL2 language.

Giese et al. [13] 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.

Fabbrini et al. [10] provide an overview of software engineering in the European automotive industry and present tools, techniques, and countermeasures to prevent faults. The authors also highlight the importance of tool integration and model-based development approaches.

Recent projects which focus on model-based development environments for automotive multi-core systems are the AMALTHEA Project [4], SAFE Project [5], and MAENAD Project [6].

[3] http://maenad.eu/
[4] http://www.amalthea-project.org/
[5] http://safe-project.eu/
[6] http://maenad.eu/

The work of Asplund et. al [1] focuses on tool integration in context of ISO 26262. The authors mention that tool integration is a complicated issue and that guidance regarding tool integration into tool chains is very limited.

2.2 Publications related to Safety-critical Development

Safety standards, such as the road vehicles functional safety norm ISO 26262 [16] and its basic norm IEC 61508 [15] present requirements and guidance for safety-critical system development. A guide to the functional safety of automotive systems according to ISO 26262 can be found in [4, 12] or in the SafEUr functional safety manager trainings [25]. The AUTOSAR development cooperation also focuses on safety in the technical safety concept report [3] produced by this group.

The work of Gashi et al. [11] focuses on redundancy and diversity and their effects on the safety and security of embedded systems. This work is part of SeSaMo (Security and Safety Modeling for Embedded Systems) project, which focuses on synergies and trade-offs between security and safety through concrete use-cases.

Born et al. [5] recommend a transition from a document-centric approach to a model-based approach. Their work mentions the problem that organizations already have their own safety processes in place and want to keep their existing document-centric processes and tool landscape, which mostly inherits fundamental flaws in terms of traceability, a key requirement in ISO 26262.

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 [19]. The integration of SysML models for the development of the ECU safety concept ensures efficient design changes, and immediate awareness of functional safety needs. The paper evaluates key success factors of MBD in comparison to legacy development processes in the field of safety-critical automotive systems.

The work of Ebert [8] highlights three key components of sustainable safety engineering in automotive systems: (a) system-oriented development, (b) safety methods closely coupled to engineering, and (c) process maturity. Ebert further mentions functional safety needs and that these are be seen as a critical product liability issue with all the consequences this implies and also that engineers need to understand the safety needs at all levels of the development process.

In the issue of improving processes or workflows especially those which deal with cross-domains affairs (such as the traceability of architectural designs from system development level to software development level), however, a comprehensive understanding of related processes, methods, and tools is required.

The work of Sechser [28] describes experiences gained at combining two different process worlds in the automotive domain. The author mentions, among other points, the need for a common language and process architecture.

The work of Raschke et. al [24] is based on the concept that development of systems is a sequence of evolutions, interrupted by revolutions. The authors build up a catalog of patterns for software modeling to these terms. This pattern

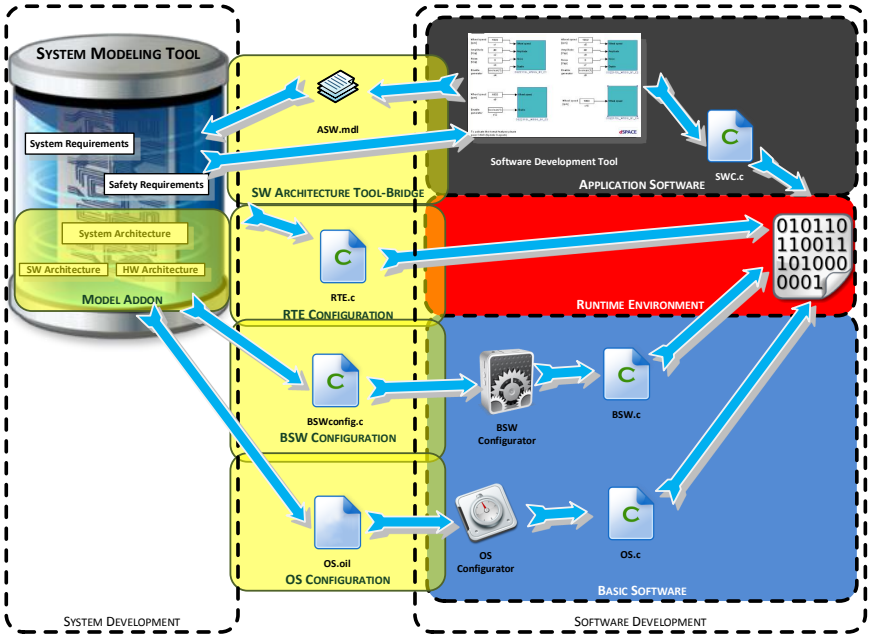


Fig. 1. presents an overview of the approach and highlights the key contribution parts.

catalog focuses on the dependencies between software models and software code and their possible relations.

According to the work of Sechser, the works of Messnarz et. al [20, 21] also showed that the combination of Automotive SPICE and ISO 26262 functional safety processes is possible and worth the effort involved in developing generic best practices and process models.

3 Architectural Design Refinement Approach

While methods and tools for elicitation and definition of requirements along the different process steps (such as Automotive SPICE ENG.2 or ENG.4) are already settled, tool supported and well-known best-practices exist. The methodical support of system architectural design (ENG.3) and refinement of this design to software design (ENG.5), however, often fell short of the mark. To handle this situation the AUTOSAR methodology [2] 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, projects with limited resources in particular (as well as non-AUTOSAR projects) often struggle to achieve adequate quality in budget (such as time or manpower) with this approach. The approach presented in this work has thus emerged from full AUTOSAR based approaches and fo-

cuses forcefully on a central MBD database as a single-source of information concept.

The main benefit of this proposed approach contributes to closing the gap, also mentioned by Giese et al. [13], Holtmann et al. [14], and Sandmann and Seibt [26], between system-level development at abstract UML-like representations and software-level development. This bridging supports consistency of information transfer between system engineering tools and software engineering tools. Furthermore, the approach minimizes redundant manual information exchange between tools and contributes to simplifying seamless safety argumentation according to ISO 26262 for the developed system. The benefits of this development approach are clearly visible in terms of re-engineering cycles, tool changes, and reworking of development artifacts with alternating dependencies, as also mentioned by Broy et al. [6].

Figure 1 depicts the overview of the approach and highlights the key contribution parts. The approach bridges the existing gap between system design and software implementation tools (also for multi-core system development) to guarantee consistency of information and minimizes redundant manual information exchange between tools. The following sections describe the key contribution parts of the approach in more details.

3.1 SW Modeling Framework Addon

The first part of the approach is a modeling framework addon that enables software architecture design in AUTOSAR like representation. This enables the design of an automotive software architecture by taking advantage of an AUTOSAR aligned VFB abstraction layer and an explicit definition of components, component interfaces, and connections between interfaces. This provides the possibility to define software architecture (ENG5.BP1) and ensures proper definition of the communication between the architecture artifacts, including interface specifications (ENG5.BP3) and timing information (ENG5.BP4). In addition this SW architecture representation can be linked to system development artifacts and traces to requirements can be easily established (ENG5.BP2). This brings further benefits in terms of constraints checking, traceability of development decisions (e.g. for safety case generation), and reuse. Figure 2 shows the representation profile of software architecture artifacts.

3.2 HW Modeling Framework Addon

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. The hardware profile, depicted in Figure 3, allows representation of hardware resources (such as ADC, CAN), calculation engines (core), and connected peripherals which interact with the software (ENG5.BP5). This

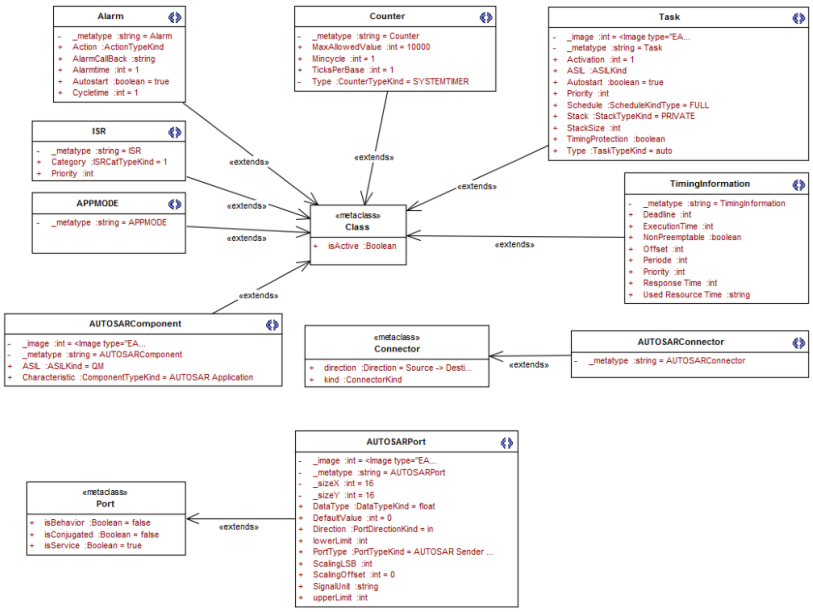


Fig. 2. shows the representation of SW architecture artifacts.

further enables the establishing of software and hardware dependencies and a hardware-software interface (HSI), as required by ISO 26262. Software signals of BSW modules can be linked to HW port pins via dedicated mappings. On the one hand this enables the modeling and mapping of HW specifics and SW signals. On the other hand, this mapping establishes traceable links to port pin configurations (ENG5.BP8).

3.3 Software Architecture Toolbridge

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 the software development tool Matlab/Simulink and thus, enabling the information handover to a special purpose tool (model-driven software engineering tools) for detailing of the SW architecture and SW modules (ENG5.BP6).

The import functionality, in combination with the export function, enables bidirectional update of software architecture representations. On the one hand, this ensures consistency between system development artifacts and changes done in the software development tool (related to ENG6.BP8 and ENG5.BP10). On the other hand, the import functionality enables reuse of available software modules, guarantees consistency of information across tool boundaries, and shares information more precisely and less ambiguously.

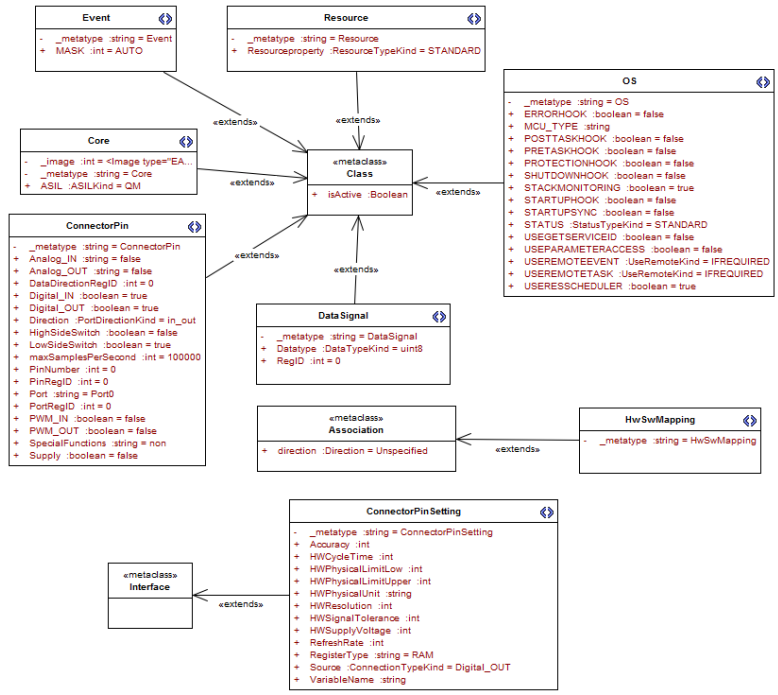


Fig. 3. shows the representation of HW architecture artifacts.

3.4 Runtime Environment Generator

The fourth part of the approach presented is the SW/SW interface generator (RTE generator). This dll- based tool generates *.c* and *.h* files defining SW/SW interfaces between application software signals and basic software signals from the modeled artifacts. The RTE generation eliminates the need of manual SW/SW interface generation without adequate syntax and semantic support and ensures reproducibility and traceability of these configurations (ENG5.BP3).

3.5 Basic Software Configuration Generator

The basic software configuration generator is also part of the dll- based tool, which generates BSW driver specific configuration files. These files configure the basic software driver of the HW device according to the HSI specifications and eliminates the need of manual information rework.

3.6 OS Configuration Generator

The last part of the approach is an exporter capable of exporting the RTOS configuration available from the model to an OIL file [22] and the corresponding

import functionality. The exporter generates OIL files enriched with the available system and safety development artifact traces (such as required ASIL of task implementation). The import functionality enables bidirectional update of the information representation within the database. Most state-of-the-art software development frameworks are able to configure the RTOS according to the specifications within such an OIL file.

4 Prototypical Implementation and Application of the Proposed Approach

This section demonstrates the benefits of the presented approach for development of automotive embedded systems. For this evaluation a prototypical implementation of the approach has been made for Enterprise Architect [7]. An automotive use-case of a central control unit (CCU) of a battery management system (BMS) prototype for (hybrid) electric vehicle has been chosen for evaluation of the approach. This use-case is an illustrative material, reduced for internal training purpose and is not intended to be exhaustive or representing leading-edge technology.

The definition of the software architecture is usually done 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 Enterprise Architect. This does not hamper the work of the software system architect but enables the possibility to also link existing HSI mapping information to the SW architecture and offers a significant benefit in terms of traceability (ENG5.BP9 and ENG5.BP10), replicability of design decisions, and unambiguously visualizes dependencies.

The use-case consists of 10 ASW modules and 7 BSW modules with 19 interface definitions between ASW and BSW, which are transferred via the SW architecture tool-bridge and makes use of the 3 fundamental low level HW functions (digital input/output, analog input/outputs, and PWM outputs). A more complete overview of use-case is given in Table 1.

As can be seen in Table 1, 7 ASW/BSW input interfaces and 12 ASW/BSW output interfaces need to be defined. This definition sums up to more than 30 lines of code (LoC) which can be generated automatically with the runtime environment generator. The actual HW/SW interface, mapping of BSW signals to HW pins, also consist of 19 interfaces and 3 low level driver for this specific SW architecture. This mapping includes 23 settings per mapping and can be used to automatically generate basic software configurations with the help of the basic SW configuration generator and OS configuration generator. This ensures actuality of development artifacts and simplifies tracing of development decisions.

[7] http://www.sparxsystems.com/

Table 1. comprises an overview of the evaluation use-case SW architecture, element counts, and number of configurable attributes per element.

Object type	Element-count	Configurable Attributes per Element
ASW Modules	10	3
BSW Modules	7	3
ASW Module Inputs	54	10
ASW Module Outputs	32	10
ASW/ASW Module Interfaces	48	
ASW/BSW Module Interfaces	19	
HW/SW Interfaces	19	13

5 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 and automotive SPICE provide a process framework which requires efficient and consistent product development and tool support. Nevertheless, various heterogeneous development tools in use hamper 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 the development tools involved has been proposed and a prototypical tool-bridge implementation has been made. The approach presented merges the heterogeneous tools required for development of automotive systems to support seamless interchange of information across tool boundaries and helps to ease cross-domain consolidation via establishing a single source of information concept. The application of the approach presented has been demonstrated utilizing an automotive BMS use-case, which is intended for training purposes for students and engineers and does not represent either an exhaustive or a commercially sensitive project. The main benefits of this 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. Further improvements of the approach include the progress in terms of reproducibility and traceability of design decisions.

Acknowledgments

This work is partially supported by the INCOBAT and the MEMCONS projects.

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement n 608988 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. F. Asplund, M. Biehl, J. El-khoury, D. Frede, and M. Trngren. Tool Integration, from Tool to Tool Chain with ISO 26262. In *SAE Technical Paper*. SAE International, 04 2012.
2. AUTOSAR development cooperation. AUTOSAR AUTomotive Open System ARchitecture, 2009.
3. AUTOSAR Development Cooperation. Technical Safety Concept Status Report. Technical Report Document Version: 1.1.0, Revision 2, AUTOSAR development cooperation, October 2010.
4. K. Boehringer and M. Kroh. Funktionale Sicherheit in der Praxis, July 2013.
5. M. Born, J. Favaro, and O. Kath. Application of ISO DIS 26262 in Practice. *CARS 2010*, 2010.
6. 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.
7. 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.
8. C. Ebert. Functional Safety Industry Best Practices for Introducing and Using ISO 26262. In *SAE Technical Paper*. SAE International, 04 2013.
9. C. Ebert and C. Jones. Embedded Software: Facts, Figures, and Future. *IEEE Computer Society*, 0018-9162/09:42–52, 2009.
10. F. Fabbrini, M. Fusani, G. Lami, and E. Sivera. Software Engineering in the European Automotive Industry: Achievements and Challenges. In *COMPSAC*, pages 1039–1044. IEEE Computer Society, 2008.
11. I. Gashi, A. Povyakalo, L. Strigini, M. Matschnig, T. Hinterstoisser, and B. Fischer. Diversity for Safety and Security in Embedded Systems. In *International Conference on Dependable Systems and Networks*, volume 26, 06 2014.
12. V. Gebhardt, G. Rieger, J. Mottok, and C. Giesselbach. *Funktionale Sicherheit nach ISO 262626 - Ein Praxisleitfaden zur Umsetzung*, volume 1. Auflage. dpunkt.verlag, 2013.

13. H. Giese, S. Hildebrandt, and S. Neumann. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. *LNCS 5765*, pages pp. 555 –579, 2010.
14. J. Holtmann, J. Meyer, and M. Meyer. A Seamless Model-Based Development Process for Automotive Systems, 2011.
15. ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.
16. ISO - International Organization for Standardization. ISO 26262 Road vehicles Functional Safety Part 1-10, 2011.
17. H. C. Jo, S. Piao, and W. Y. Jung. Design of a Vehicular code generator for Distributed Automotive Systems. In *Seventh International Conference on Information Technology*. DGIST, 2010.
18. J.-C. Lee and T.-M. Han. ECU Configuration Framework based on AUTOSAR ECU Configuration Metamodel. 2009.
19. 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.
20. R. Messnarz, F. Knig, and V. O. Bachmann. Experiences with trial assessments combining automotive spice and functional safety standards. In D. Winkler, R. V. O'Connor, and R. Messnarz, editors, *EuroSPI*, volume 301 of *Communications in Computer and Information Science*, pages 266–275. Springer, 2012.
21. R. Messnarz, I. Sokic, S. Habel, F. Knig, and O. Bachmann. Extending Automotive SPICE to Cover Functional Safety Requirements and a Safety Architecture. In R. O'Connor, J. Pries-Heje, and R. Messnarz, editors, *EuroSPI*, volume 172 of *Communications in Computer and Information Science*, pages 298–307. Springer, 2011.
22. OSEK/VDX Steering Committee. OSEK/VDX System Generation OIL: OSEK Implementation Language. http://portal.osek-vdx.org/files/pdf/specs/oil25.pdf, 2004.
23. I. R. Quadri and A. Sadovykh. MADES: A SysML/MARTE high level methodology for real-time and embedded systems, 2011.
24. W. Raschke, M. Zilli, J. Loinig, R. Weiss, C. Steger, and C. Kreiner. Patterns of Software Modeling. In R. Meersman, H. Panetto, A. Mishra, R. Valencia-Garca, A. Soares, I. Ciuciu, F. Ferri, G. Weichhart, T. Moser, M. Bezzi, and H. Chan, editors, *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, volume 8842 of *Lecture Notes in Computer Science*, pages 428–437. Springer Berlin Heidelberg, 2014.
25. SafEUr Training Material Committee. ECQA Certified Functional Safety Manager Training Material. training dossier, April 2013.
26. 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.
27. G. Scuro. Automotive industry: Innovation driven by electronics. http://embedded-computing.com/articles/automotive-industry-innovation-driven-electronics/, 2012.
28. B. Sechser. The marriage of two process worlds. *Software Process: Improvement and Practice*, 14(6):349–354, 2009.
29. The SPICE User Group. Automotive SPICE Process Assessment Model. Technical report, 2007.

Automotive Safety Case Pattern

Georg MACHER, Graz University of Technology
Harald SPORER, Graz University of Technology
Christian KREINER, Graz University of Technology

Automotive embedded systems have become very complex, are strongly integrated, and safety-criticality of these systems raises new challenges. Due to this safety-criticality the ISO 26262 road vehicle safety norm was introduced. Development conforming ISO 26262 requires providing consistency of the safety concept during the entire product lifecycle, and supporting evidences, known as and combined via safety case. A safety case is a collection of development artifacts (e.g. test protocols, interface descriptions, domain experts group meeting protocols, certificates) aiming to convince customers or auditors by arguing that the product is capable safe. Establishment of such a safety case is a tedious task, and practical examples and guidelines are yet rather uncommon due to intellectual property reasons.

This paper presents the application of patterns to generate a ISO 26262 safety case documentation for an industrial case study. The introduced patterns and use case should serve for novices in the area of automotive safety as guidance for construction of safety cases.

Categories and Subject Descriptors: H.1.0 [**Information Systems**]: Models and Principles—*General*; K.6.m [**Management of Computing and Information Systems**]: Miscellaneous—

General Terms: Documentation

Additional Key Words and Phrases: Automotive embedded systems, ISO 26262, pattern application, safety case.

ACM Reference Format:

Macher G., Sporer H., and Kreiner C.,2015. International Conference Proceedings Series (ICPS). jn 2, 3, Article 1 (May 2015), 17 pages.

1. INTRODUCTION

The number of embedded systems in the automotive domain has grown significantly in recent years. Current premium cars imply more than 70 electronic control units (ECU) with close to 1 Gigabyte software code [**?**] implemented.

In 2018 30% of the overall vehicle costs are predicted to stem from vehicle electronics [**?**]. This trend is also strongly supported by the ongoing replacement of traditional mechanical systems with modern embedded systems. This enables the deployment of more advanced control strategies, thus providing new benefits for the customer and environment, such as reduced fuel consumption and better driveability.

At the same time, the higher degree of integration and the safety-criticality of the control application raise new challenges. Hence, the correctness of the applications in both the time domain and the value domain has to be guaranteed. Safety standards such as ISO 26262 [**?**] 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 of these required

EuroPloP '14, July 09 - 13, 2014, Irsee, Germany
 ACM 978-1-4503-3416-7 ...$15.00.
http://dx.doi.org/10.1145/2721956.2721962

work-products is the safety case. The challenge in this context is to provide evidence of consistency of the different work-products during product development.

The ISO 26262 standard mentions three principal elements of which a safety case shall consist of: requirements, arguments, and evidences.

However establishment of such a safety case is a tedious task, and practical examples and guidelines are yet rather uncommon due to intellectual property reasons. Due to this lack of practical available examples many safety cases result in extra workload with only limited additional benefit. This paper presents, according to our knowledge, three previously undocumented but industrial best practices patterns and furthermore, the application of these patterns and their benefits for the specific case study of an automotive battery management system (BMS).

—PROCESS & PRODUCT Pattern - organizes the safety case argumentation into a product and a process specific branch.

—TRACEABILITY Pattern - supports argumentation via requirement and traceability to realization and artifacts.

—FUNCTIONAL BREAKDOWN Pattern - decomposes generic arguments into smaller sub-sets or specific argumentation.

The document is structured as follows:
Section 2 briefly describes the safety case in context of ISO 26262 and gives a very brief overview of ISO 26262 aligned development. Section 4 covers related work on safety cases and patterns for safety case generation. In Section 5 we provide a case study of a battery management system (BMS), which is afterwards used to evaluate the application of pattern for safety cases (Section 5.2). The main point of this work is the description of the three patterns, which is done in Section 3. Finally, Section 6 concludes this work with an overview of what has been presented.

2. BACKGROUND INFORMATION

This section briefly characterizes the basics of an automotive safety case and product development of safety-critical systems in the automotive domain.

2.1 Safety Case in Context of ISO 26262

The role of a safety case is to communicate and explicitly show implicit domain knowledge, to summarize safety argument and the referencing reports capturing the supporting evidence. It convincingly argues in a compact way that a system meets its safety requirements and describes why which decision has been made or not made in a coherent and reproducible way. Thereby a safety case aims to convince customers or auditors by arguing that the product is sufficiently safe.

In ISO 26262 Part 10 [**?**] section $5.3.1$ states the purpose of a safety case as

> '*... to provide a clear, comprehensive and defensible argument, supported by evidence, that an item is free from unreasonable risk when operated in an intended context.*'[**?**]

The relationship between the previously mentioned three elements are shown in Figure 1 and described very general by the standard: (a) Requirements describe the objectives a safety-related system needs to fulfill. (b) Arguments communicate the relationship between objectives and evidences. These (c) Evidences are the convincing production artifacts within a safety case. Both arguments and evidences are crucial elements of the safety case. Arguments without supporting evidences are unfounded and unconvincing. Evidences without arguments are unexplained and unclear. Due to this formalization there is much room for individual interpretation and implementation of a safety case. A standardized format for a safety case is still missing. Therefore, a frequently

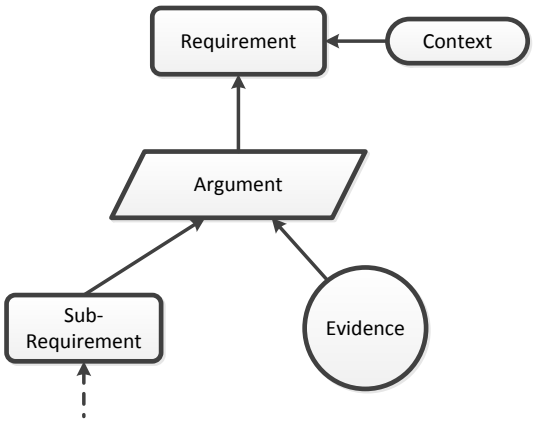


Fig. 1. Key Elements of a Safety Case

observed phenomenon, due to a lack of guidance, is that safety case development results in a lot of additional effort with only limited advantages for the audit. Also the way how a safety case should be documented is not specified by the ISO 26262 standard. In many other industries safety arguments have often been communicated in safety case reports through narrative text. Alternatively the use of a graphical notation is popular in automotive industry. Two visualization styles are recommended for automotive safety cases: (a) Claim-Argument-Evidence Notation and (b) Goal Structure Notation (GSN)[**?**].

The development of a safety case is an incremental activity that has to be integrated within the development process. Therefore, a good approach might be (as also mentioned by the standard) to plan safety cases in incremental steps and preliminary versions (also proposed by [**?**] see Section 4). Therefore, a preliminary version of the safety case can be created after creation of technical safety requirements, an interim version after definition of the system design, and a final version just prior finalization of product.

Nevertheless the safety case is an important development artifact, but it should not result in reproduction of production artifacts. Quite the contrary, it should briefly and precisely guide uninvolved persons through the product development and convince them.

2.2 Product Development of Safety-critical Systems

Product development of a safety-critical system starts with management of functional safety at company level, and is done in parallel to a typical development cycle from initial requirement to final product. Beyond that, ISO 26262 address safety engineering from initial requirement to final decommission of the product. For this paper we focus on development phases only and consider development from concept phase to product release; not considering supporting or management processes, which are also in the focus of ISO 26262. The following section demonstrates a brief overview about the ISO 26262 aligned development of the BMS prototype, and explains domain specific keywords. This section thus is subdivided according to the software safety requirement process structure of ISO 26262 (see [**?**]).Main deliverables to be mentioned are:

	HAZARD IDENTIFICATION		CLASSIFICATION OF HAZARDOUS EVENTS				
	possible malfunction	Situation	Severity	Exposure	Controlability	ASIL	Safety Goal
3.1.	battery fire	traffic jam, tunnel, BMS does not open HV relais	3	1	0	QM	battery fire must be prohibited
3.2.	battery fire	speedway, high speed, tunnel, BMS does not open HV relais	3	2	3	B	battery fire must be prohibited
3.3.	battery fire	normal operation, BMS does not open HV relais	3	4	2	C	battery fire must be prohibited
3.4.	loss of power	high speed, loss of acceleration can cause an accident while overtaking, BMS does not open HV releais	3	1	2	QM	BMS must provide state of charge information (SoC)

Fig. 2. Hazard Analysis and Risk Assessment of BMS (from [**?**])

—Item definition and safety goals - defines the system under development and its high level requirements related to safety

—Functional safety concept (FSC), preliminary architecture, failure mode and effects analysis (FMEA), and fault tree analysis (FTA) -
FSC defines the safety concepts on function point of view, while FMEA and FTA analyze the impact of faults and failures

—Technical safety concept (TSC), system design, and HW/SW Interface (HSI) - final development artifacts of system development and technical implementation specification of safety concepts

2.2.1 *Item Definition and Safety Goals.* As a first step of ISO 26262 related development, the boundaries of the system and its interacting environment must be specified. This definition in the context of ISO 26262 is called item definition. An item is defined by ISO 26262 as a

> *system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied.[**?**]*

A system consists at least of a sensor, a controller, and an actuator. Based on this item definition, possible malfunctions of all described functions need to be identified. This process is followed by an identification of possible hazards for the system and an identification of resulting risks; called hazard analysis and risk assessment (HARA). Domain experts identify hazards and hazardous situations and quantify the involved risks. This results in hazardous events, which are classified according to ISO 26262 in terms of severity, controllability, and exposure. Finally, this results in a specific automotive safety integrity level (ASIL) and a related safety goal. Figure 2 shows a HARA in tabular form, the quantification of risks and the resulting safety goals. Further details of the system analyzed by the HARA in Figure 2 will be given in the case study section 5.

2.2.2 *Functional Safety Concept, Preliminary Architecture, FMEA, and FTA.* The previously posed safety goals lead to functional safety requirements and the functional safety concept (FSC). For each element of the system under development possible internal failures and failure propagation needs to be identified. This error model can be analyzed and evaluated using Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA). FMEA and FTA can further be used to evaluate system design decisions against each other, e.g. for indicators for the most probable cause of a failure and therefore for a robust design.

2.2.3 *Technical Safety Concept, System Design, and HSI.* The refinement of the FSC is called technical safety concept (TSC) and specifies the system design. These requirements are generally divided into hardware-related and software-related requirements and linked to specific parts of the system design. The system design is finalized by the definition of the hardware-software interface (HSI). This mapping provides a basis for parallel development of HW and SW.

3. SPECIFICATION OF THE PATTERN

The following section discusses the patterns more detailed. Therefore we specify the three safety case pattern in the typical structure [**?**; **?**] of:

Context	- situation in which the pattern is usable
Problem	- scenario which requires action
Forces	- motivation for usage of the pattern
Solution	- description of problem solution
Consequences	- description of results and drawbacks
Known Uses	- publications which used the pattern

The application of the patterns will be presented in Section 5.2 for an automotive battery management system.

3.1 PROCESS & PRODUCT Pattern

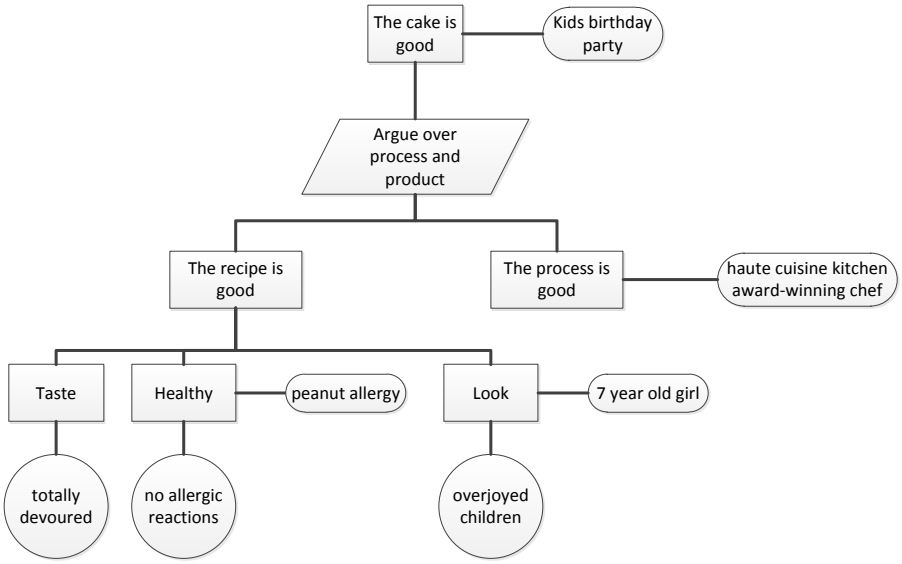


Fig. 3. Depiction of PROCESS & PRODUCT Pattern Approach Example

The intention of the pattern is to provide a top level decomposition for the safety argument of a system. In particular, the pattern identifies two main claims which must be satisfied to show the system safety: product safety, and process safety. This pattern shall be applied at the very beginning of the product development and safety case generation. Therefore, the application of this pattern must be before the item definition and safety goal generation 2.2.1.

3.1.1 *Context.* Safety-critical system development for the automotive domain require a certification of the development process and an argumentation that the certified development process has been correctly applied for the specific product. A certification, such as Automotive SPICE (ISO/IEC 15504 [**?**]), is required to get enlisted as supplier (called tier) of any hardware or software system in the automotive domain. Goal of Automotive SPICE is to provide indicators to compare the maturity of business rival's processes and company standards.

3.1.2 *Problem.* To ensure repeatability of equal product quality, independent of individuals, processes and methods needs to be identified and established. These processes need to be certified, usually via external certification authorities, to ensure comparability and competitiveness. Issues in this context are to distill a process reusable for individual project, certify this processes, and provide evidence of compliance with this regulations for each individual application.

3.1.3 *Forces*

—Companies with no Automotive SPICE certificate will not be considered for supporting engineering activities.

—Certification of the development process is typically a complex, cost intensive, and time consuming process, which should not be done for each project individually.

—The certification process requires external and internal resources (manpower) which leads to nonproductive time periods.

—For development of safety-critical systems furthermore evidence of application of domain expertise at development of the specific product has to be proven.

3.1.4 *Solution.* First, establish a development process according to the domain best-practices and requirements of Automotive SPICE. Second, provide evidence of application of these certified processes in correct manner. Third, organize the argumentation of a safety case into a product branch, specific to the product, and a process branch, product independent. The separation of concerns enables easier reuse of process claims and independent certification of production processes (e.g. audits of the company process structure). Arguments are grouped into topics which relate to best-practices and approaches which can be directly reused for other projects (e.g. a guideline for generation of a domain-specific report) and on the other hand the specific application for the specific product (e.g. the domain-specific report itself).
This separation of arguments is depicted in Figure 3 for an easy example of baking a cake. The goal of baking a good cake is supported by the strategy of having a good process in place (skilled award-winning chef and haute cuisine kitchen), the product-independend (process-specific) part and a tasty recipe (product-specific part).

3.1.5 *Consequences*

\+ Helps during auditing to split into domain specific expert groups; thus enable parallel working ability and speedup of the auditing process.

\+ During development of the product and the safety case same split of domain expert groups and speedup in time can be measured.

\+ Supports the identification of product-independent best practices and methods.

\+ Separation of product-dependent and product-independent parts enables reuse of generic parts for other product developments.

\- Separation of arguments according to this approach requires additional work and can be challenging in terms of determination of product independence.

\- Distillation of multi-purpose product-independent argumentation may result in arguments lacking in substance.

3.1.6 *Known Uses.* This pattern can be seen as best practice and de-facto standard way of preparing safety cases in the automotive domain [**?**], [**?**], [**?**], [**?**]. Although this pattern is not mentioned in the GSN Pattern catalog [**?**], it is presented for a preliminary safety case by Kelly [**?**] and Birch et. al. [**?**]. Also the case study of Ridderhof et. al. [**?**] is structured in a product and process specific part.

3.2 TRACEABILITY Pattern

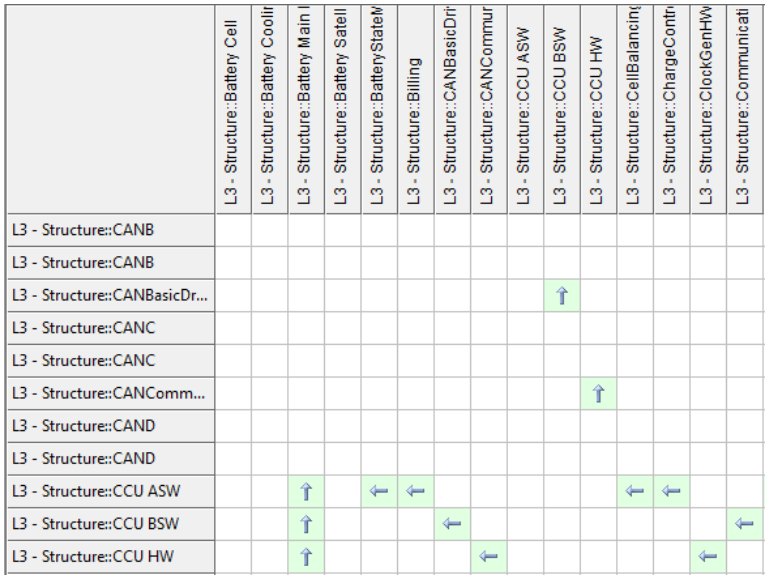


Fig. 4. Depiction of TRACEABILITY Pattern Approach in Form of a Traceability Matrix

The pattern describes an approach to address the demands of ISO 26262 needs of establishing traceability from origin to realization and proof of implementation of all decision made during development process. For generation of safety cases it is essential to be able to trace decisions and their impacts at each stage of the safety case generation process 2.2.

3.2.1 *Context.* Safety-related system development requires cross-domain knowledge and requirements originate from several expert groups and iteration cycles. Requirements frequently originate from non-experts and their impact is often not easy to determine.

3.2.2 *Problem.* According ISO 26262, full traceability of development artifacts of a safety-critical system needs to be proven. Decision-making procedure of development artifacts needs to be documented to ensure reproducibility. Requirements influences and meaning are often unclear, their validity changes during the development period, and minor requirement changes may have huge impacts on safety strategies and the safety case. Moreover, the complexity of system requires links between different development artifacts to ensure consistent and complete implementation of required features.

3.2.3 *Forces*

—Various types of development artifacts need to be traced.
—Requirements are frequently ambiguous and originate from non-experts.
—Validity of requirements change during development phases.
—Each type of artifacts needs to be version controlled.
—Artifact dependencies need to be explicitly linked.
—Evidence of origin, validation, and realization of each artifact in a correct, complete, and consistent way needs to be provided.

3.2.4 *Solution.* First, assign unique IDs and validity states for each development artifact and requirement. Second, establish traces between each artifact to explicitly indicate dependencies and relations. Most intuitive way of establishing traces and relation is to collect al IDs in a matrix and flag relations within the matrix. Life-cycle management tools and version control systems shall be used to keep trace of development process, iteration cycles, and artifact dependencies and ensure consistency and correctness of such a traceability matrix.

Figure 4 shows a traceability matrix of different development artifacts within a life-cycle management tool. Traces are indicated via arrows in this figure and can have directions and different types of relation. These traces can be useful in practice, due to reducing the need of inter-tool traces (e.g. reviews of consistency check protocols) and highlighting of the impact of changes done at a specific artifact.

3.2.5 *Consequences*

+ Gives good evidence that all requirements have a defined origin, have been validated, satisfied, and realized.
+ Comprehensible impacts of requirement changes.
+ Traceability across all development documents in both directions can be shown in a structured and repeatable way.
+ Constraints checker can be applied to check for missing links and auto generate reports.
+ Explicit illustration of dependencies.
+ Convincingly argumentation of a reliable development of the system.
- Plenty of dependencies hardly manageable without adequate tool support.
- Lot of additional effort needed to establish traces.

3.2.6 *Known Uses.* Armengaud [**?**] and Ridderhof et. al. [**?**] mainly base their work on this approach. Providing traceability from requirements to artifacts tent to be the most intuitive and popular approach for safety case development.

3.3 FUNCTIONAL BREAKDOWN Pattern

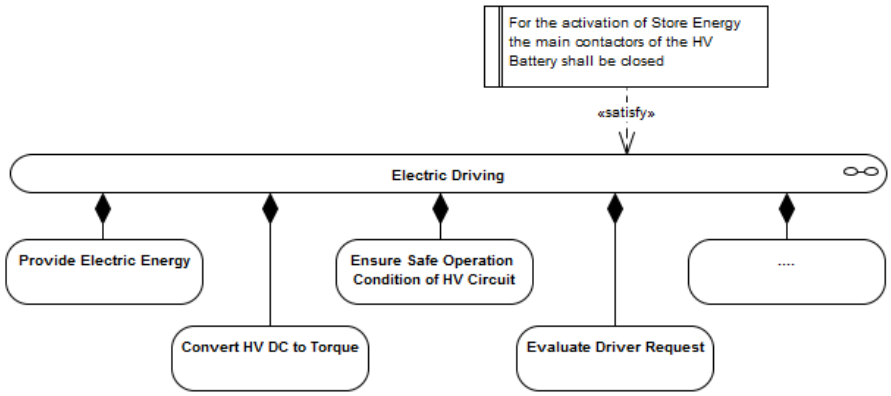


Fig. 5. Depiction of FUNCTIONAL BREAKDOWN Pattern Approach

The intent of this pattern is to structure arguments to support a safety goal by decomposition into smaller sub-goals. This structuring concerns the decomposition of overall system functionalities and involves the introduction of an additional system viewpoint. The FUNCTIONAL BREAKDOWN pattern is applicable for item definition 2.2.1, development of safety concepts and safety architectures 2.2.2. Other than the DIVIDE AND CONQUER pattern, the FUNCTIONAL BREAKDOWN pattern does not solely focus on decomposition of systems to sub-systems, but rather focuses on the decomposition of functions. This also considers cross-domain concerns and relations which can not be tackled with DIVIDE AND CONQUER patterns on system level.

3.3.1 *Context.* The development of highly integrated multi-domain features of an automotive system requires several departments of several domains to cooperatively work in parallel. A decomposition of the overall system is indispensable to manage the huge system size and complexity. Without adequate breakdown the number of required domain expertises within one team would yield to unmanageable effort. Nevertheless, system-wide features (such as safety) require different approaches than naif fragmentation into smaller technical implementation units.

3.3.2 *Problem.* Safety-critical systems are not manageable as a whole in terms of required domain-expertise, diversity, workload, and time-to-market. But several constraints (e.g. safety-criticality) affect the whole system and need to be considered by all involved parties. Therefore, fragmentation into technical implementation units is insufficient. A breakup of functions on vehicle-level is required to get a clear conception of causal relations regarding safety.

3.3.3 *Forces*

—Limited time-to-marked and different required domain-expertise require a structuring to ensure parallel working of departments.

—Safety constraints affect the system as a whole but are too complex to be managed as a whole.

—Evidences need to be provided within several domains (e.g. evidence of correct HW selection, evidence of correct SW behavior).

3.3.4 *Solution.* Breaking a system into distinct sub-functions at vehicle view of abstraction that overlap in their purpose as little as possible and assign specific teams for each sub-function. Solutions for sub-problems are easier to determine due to reduced complexity and dependencies. Further, argumentation of overall system safety for a safety case can be done via argumentation of safety of each involved sub-functionalities. Thereby responsibility for safety argumentation is passed over to experts of the specific sub-function domain. This implies an introduction of an additional system viewpoint (functional view illustrating vehicle-wide functions and their decomposition), which simplifies conception and development of system-wide features (such as safety). Figure 5 shows a high level application of the pattern for a vehicle. Safety argumentation for the powertrain can be separated according the involved powertrain elements (eDrive, Transmission, Battery, and Engine) and cross-domain / cross-element arguments (such as high-voltage safety of passenger) can be argued for each element individually.

3.3.5 *Consequences*

+ Resizing of work packages allows a separation of responsibilities and parallel development of individual sub-features.
+ Divided responsibilities ensure optimized solutions of individual work packages.
+ Increasing number of measures for management.
+ Split of responsibility of overall argumentation to domain experts for their sub-parts.
- Additional consolidation of the sub-systems and overall system constraints required.
- Individual optimal sub-solutions may not result in the overall optimal solution.
- Requires additional merging activities of the sub-parts.
- Introduction of an additional view on the system.

3.3.6 *Known Uses.* The work of Kelly [**?**] focus on a modular approach to safety case construction for modular constructed safety critical and safety related systems.

4. RELATED WORK

This section briefly mentions some related work on safety cases, on their supported information, and on notations of a safety case.

The essential work of Kelly [**?**] depicts the nature of a safety case as to refer to and pull together, potentially many other pieces of information. Safety case report should present the key conclusions and findings that convince the reader that the system is acceptably safe to operate in its intended design context. Kelly describes the relationship between requirements, arguments, and evidences which has also been taken over into the ISO 26262 standard. Kelly also claims that well-structured approaches to express safety arguments in text form can be effective, but recommends the Goal-Structure-Notation (GSN). Furthermore, Kelly proposes the safety case development lifecycle, as stated by defense standards and also recommended by ISO 26262, in three versions:

(a) Preliminary Safety Case - after definition and review of system requirements,

(b) Interim Safety Case - after initial system design, and

(c) Operational Safety Case - prior to in-service; including complete evidence of satisfaction of system requirements.

The work of Holloway [**?**] presents five possible text-based notations for representing safety cases. But as mentioned by Kelly, drawbacks of text representation can be: (a) an unclear and poorly structured English, (b) ambiguous structuring of text and sentences, and (c) multiple cross-references in the text which disrupt the flow of the main argument. This work therefore also highlights the positive effects of pattern and GSN approaches for development of safety cases.

Palin and Habli [**?**] proposed some patterns for automotive safety case in a very early draft phase of ISO 26262. After final release, the standard has changed in a way that the proposed usage sounds slightly odd, although the patterns might still be usable. Due to popularity of GSN in the automotive domain, the patterns available at the GSN Working Group homepage [**?**] are more applicable.

The works of Ridderhof et. al. [**?**] and Armengaud [**?**] prove evidence of their safety case from tracing system safety requirements to the development artifacts by which they are realized. Ridderhof achieves this with the help of a commercial tool collecting all information artifacts of different involved tools and represents the safety case in GSN. Armengaud uses a special solution for compilation of the safety case based on automated extraction of information from existing work-products and also takes test campaign into account. Both solutions support automated checks of different criteria (e.g. completeness and consistency of argumentation), which strongly reduces the manual efforts for compilation of a safety case. Ridderhof's and Armengaud's works do also make implicit use of the three in this paper introduced pattern.

The integration of a safety strategy to model driven development with SysML is main goal of Hause and Thom [**?**]. Although the work dates from the period before the introduction of ISO 26262 and solely refers to a very abstract safety case example it focuses on still valid procedures. Separation of concerns is the process of breaking a system into distinct features that overlap as little as possible, which is the basis of modular system design. However cross-cutting concerns, like safety, traceability, or timing need to be addressed in all subsystems. Their approach makes use of an integrated database and ergonomic profiling to support all disciplines involved in development. They make use of SysML model-driven development and discipline specific SysML profiles to gather all information within a single database. Furthermore, they integrated a GSN representation possibility in the model-driven development tool to have access to all development artifacts and compile the system's safety case with actual development artifacts.

5. CASE STUDY

This section inherits a description of the use case and the application of the previously introduced patterns for establishment of an automotive safety case of the use case.

5.1 Description of the Case Study

To demonstrate the use of patterns for creation of an automotive safety case we use an industrial case study of a battery management system. Battery management systems (BMS) are control systems inside of high-voltage battery systems used to power electric or hybrid vehicles. The BMS consists of several input sensors, sensing e.g. cell voltages, cell temperatures, output current, output voltage, and actuators, the battery main contactors. Furthermore, the BMS also includes an electronical control system and may impact the safety of driver and pedestrians (e.g. unintended acceleration, overheating of battery). Therefore, these systems are safety-critical electronical devices which need do be developed according legislative regimentation and industry standards to be used for passenger cars. The specific case study is a prototype battery and BMS aiming to improve life-time and better utilization of energy storage capabilities of battery cells via innovative control strategies.

Figure 6 depicts the general structure, main hardware components, and software modules of the high-voltage battery with BMS. The illustration shows the main features of a BMS, also summarized in Table I

Table I. Main Features of the BMS

Power contactors	- connect the HV battery with the rest of the vehicle
Interlock (yellow connection)	- signal preventing the battery of harming the operator when tripped
CAN (yellow connection)	- automotive communication interface between vehicle and battery
Relay (green connection)	- main contactor and output unit of the BMS
Temperature sensors (purple connection)	- feedback of actual cell temperatures
Voltage sensors (blue connection)	- feedback of actual cell voltages
Current sensors (red connection)	- feedback of actual electric power consumption
Fuse	- protective circuit breaker in case of fault
Cells	- energy storage which needs to be kept in defined operating conditions
BMS controller	- monitoring unit to keep battery cells within defined operating conditions

Furthermore, the main software modules of the BMS control strategy are shown in Figure 6. These software modules estimate the charging level of the battery (State of Charge, SoC), battery performance level (State of Fitness, SoF), and the current battery state. Beside this basic features also cell balancing functionality, to ensure uniformity of all cells, and supervising functionality of external chargers are mandatory. The ‘Safety / Diagnosis’ module ensures additional error detection and safety in case of failures.

From the safety case point of view the system as a whole (including SW and HW modules) needs to be analyzed and additional safety features might be considered for critical parts. Some high level requirements for safety relevant parts are in Table II.

Table II. High Level Requirements for Safety Relevant Parts of the BMS

Relay	switching status shall be monitored by BMS
Interlock	signal shall be generated by BMS and disconnection of battery needs to be progressed if trapped
CAN	status of vehicle needs to be OK before closing of power connection
	sensor signals shall be validated and over-temperature has to be prevented
Sensors	over/under voltage of cell needs to be prevented
	power consumption measures (current or voltage) must not exceed defined maximum levels
Cells	energy cells needs to be kept in defined operating conditions
	SW module malfunctions need to be detected and prevented
BMS controller	controller failures need to be detected and battery disconnected
	maximum power limits must be provided to ensure safe operation of battery cells

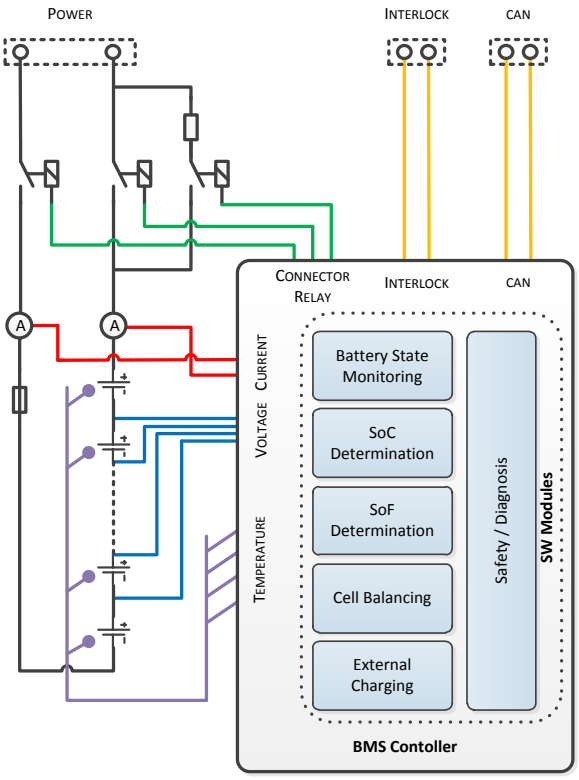


Fig. 6. BMS Structure

5.2 Application of the Patterns for Development of an Automotive Safety Case

The following section discusses the application of patterns for the use case of the battery management system safety case. We focus on the three pattern presented in Section 3.

To start with the development of an automotive safety-critical system and thus an safety case the PROCESS & PRODUCT pattern shall be applied initially before starting the development of the SUD.

Figure 7 depicts the application of PROCESS & PRODUCT pattern for the highest level of the case study's safety case. This illustration is done in a typically development engineers point of view.

As can be seen in upper part of the figure, the intention is to argue the development of a BMS which is developed dependable safe, in context of ISO 26262. This argumentation is based on two pillars. First, the development processes are dependable safe (right pillar in Figure 7), which means, the processes are according automotive best practices. The argumentation is based on the evidence of an Automotive SPICE certificate [**?**]. This means, that processes used for the development of the BMS comply with the automotive standard for software development.

The second (left) pillar of the argumentation focuses on the product itself. In this case the argument for a dependable safe BMS is stated upon the strategy that the environment is known and defined, and the fact that an adequate risk management hast been performed.

Here the argumentation is not based on the implausible evidence that the system under development is safe in any conditions and nothing will ever fail, but rather on the opposite. The argumentation for a dependable safe BMS is based on systematic identification, assessment, and prioritization of risks (see 2.2.1).

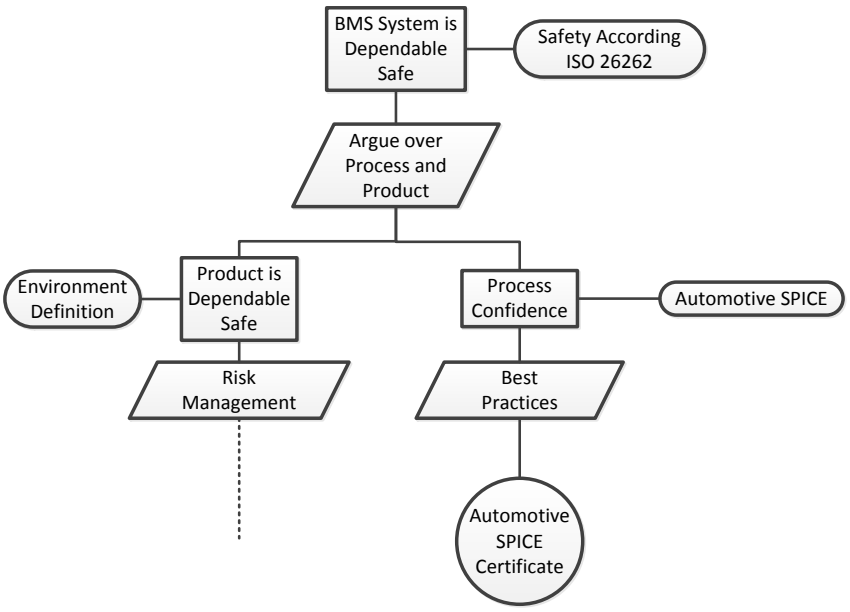


Fig. 7. PROCESS & PRODUCT Pattern Application in Goal Structure Notation

The depicted application is solely one specific use-case for this pattern, Figure 3 previously illustrates the usage of the PROCESS & PRODUCT pattern for another level, by differencing work-product templates and work-products itself (e.g. requirement templates for the process of requirement engineering and requirement documents itself). With the utilization of the PROCESS & PRODUCT pattern a first important step towards a meaningful safety case is done. As a sideline Figure 7 already mentions a safety argumentation based on an risk management (in automotive cases HARA, see 2.2.1).

Figure 8 shows the graphical representation of an excerpt from a HARA. Furthermore, the application of TRACEABILITY pattern can be seen within this picture.

The TRACEABILITY pattern can be applied for any development artifact to ensure correct, complete, and consistent refinement. Figure 8 shows the graphical representation of traces within the model-based development tool Enterprise Architect (EA). The traces depicted within this figure represent analysis done during initial risk management process (called Hazard Analysis and Risk Assessment (HARA) in terms of ISO 26262). Here an analysis of a specific function of the BMS (such as store energy) is done. Possible malfunctions and caused hazards are identified and prioritized according to specific hazardous events, their probability of occurring, criticality, and controllability. This analysis leads to safety goals, which represent high level requirements which have to be meet to be dependable safe. Such a tracing can further be established from requirements to specific implementation artifacts, automatically verified and used to convincingly argue a safety case. In this case the depiction shows the usage of a proprietary tool addon for Enterprise Architect for safety engineers.

The application of the previously mentioned two pattern led, on one hand, to the approach of separating product- specific and product- independent artifacts (PROCESS & PRODUCT pattern) and therefore enable reuse of these product-independent artifacts. On the other hand, the TRACEABILITY pattern afforded an opportunity to link sundry artifacts dependencies according the decision-making process during development to generate an easily

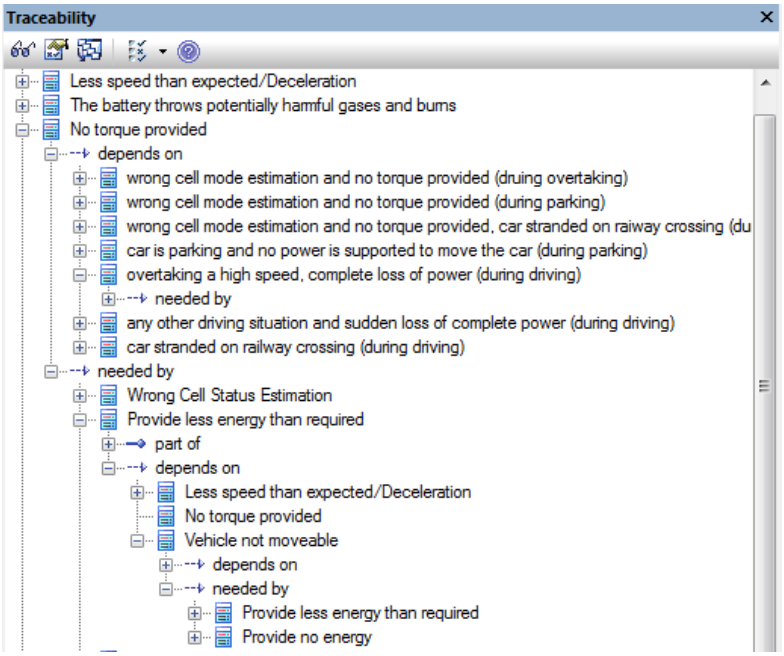


Fig. 8. Tree-View Representation of HARA Traces within Model-based Development Tool

understandable safety case for the specific product.

The intent of the third, FUNCTIONAL BREAKDOWN, pattern is to structure the arguments to support a safety goal by decomposition into smaller sub-functionalities. For an automotive BMS and its complexity a decomposition of the overall system is indispensable.

During design of system, software, or hardware architecture the overall function needs to be subdivided into smaller sub-systems to enable parallel development and re-size work packages to manageable sizes, this technical segmentation although is not sufficient to argue a safety-case. For our case study a functional breakdown of one BMS safety goal into several independent BMS sub-functionalities has been carried out. Figure 9 shows the application of the FUNCTIONAL BREAKDOWN pattern for a specific safety goal of the BMS. This segmentation can be continued to a sufficient stage of sub-functions. In case of having a convincing argumentation for the safety of each individual sub-function and freedom from mutual interference, safety of the overall system can be proven. These sub-functions can further be mapped to multiple involved technical implementation (TRACEABILITY pattern) and therefore ensure consideration of system-wide safety feature across domain boundaries.

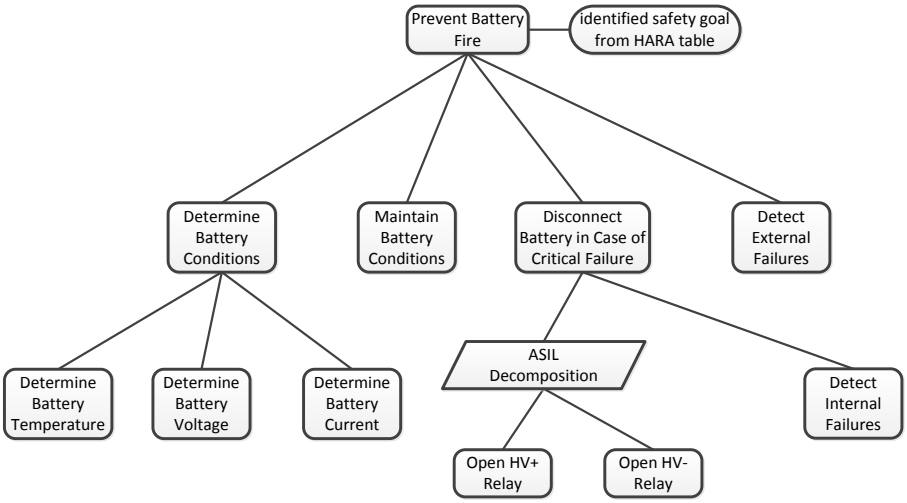


Fig. 9. Application of FUNCTIONAL BREAKDOWN Pattern for a Specific Safety Goal of the BMS Case Study

This safety view ensures consistent argumentation across domain boundaries and prevents from cross-domain design pitfalls, such as:

—Having a watchdog hardware, but not used by software.

—Providing alive counters, but receiver ignoring it.

—Having redundant HW sensors, but relying solely on ‘Sens_xxx_1’.

6. CONCLUSION

In summary, the three presented patterns are some of the most commonly applied approaches for construction of an automotive safety case. Although these patterns are used implicit already before, an explicit representation within the safety case, e.g. in Goal-Structure-Notation, of these patterns will facilitate the construction of a convincing report. Among these pattern, several other pattern might be applied to produce a convincingly safety case argumentation. Nevertheless, the presented patterns shall serve as a basis for this domain and encourage other domain experts to collect their knowledge in such a re-useable form.

This paper aims in motivating that novices in the area of automotive safety case construction to start their work rather with the search for appropriated patterns than a classical "‘give-it-a-try"’ approach. A further objective of this paper is to initiate a collection of pattern approaches for automotive safety case generation and encourage domain experts to document their approaches in form of patterns-. Our future plans for continuing with the collection of automotive safety case generation pattern are focusing not only on system development level but on software safety cases and hardware- software interface approaches.

Acknowledgments

The authors would like to express our thanks to our shepherd Veli-Pekka Eloranta who had a determining influence on the improvement of our paper and untiringly helped to improve the maturity of the paper in several iterations. Furthermore me would thank Christopher Preschern and our co-author Christian Kreiner who called our attention to pattern approaches and the EuroPLOP conference.

REFERENCES

ALEXANDER, C., ISHIKAWA, S., SILVERSTEIN, M., JACOBSON, M., FIKSDAHL-KING, I., AND ANGEL, S. 1977. A Pattern Language. *Oxford University Press, New York*.

ARMENGAUD, E. 2014. Automated Safety Case Compilation for Product-based Argumentation. In *ERTS2014 Conference Proceeding* .

BIRCH, J., RIVETT, R., HABLI, I., BRADSHAW, B., BOTHAM, J., HIGHAM, D., JESTY, P., MONKHOUSE, H., AND PALIN, R. 2013. Safety Cases and Their Role in ISO 26262 Functional Safety Assessment. In *SAFECOMP*. 154–165.

EBERT, C. AND JONES, C. 2009. Embedded Software: Facts, Figures, and Future. *IEEE Computer Society 0018-9162/09*, 42–52.

HAUSE, M. C. AND THOM, F. 2008. An Integrated MDA Approach with SysML and UML. In *Hause2008*. 249–254.

HILBRICH, R., REINIER VAN KAMPENHOUT, J., AND GOLTZ, H.-J. 2012. Modellbasierte Generierung statischer Schedules fuer sicherheitskritische, eingebettete Systeme mit Multicore-Prozessoren und harten Echtzeitanforderungen. *Informatik aktuell*, 29 – 38.

HOLLOWAY, C. M. 2008. Safety Case Notations: Alternatives for the Non-Graphically Inclined? In *Third IET Systems Safety Conference*. The Institution of Engineering and Technology, NEC, Birmingham, UK.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. 2011. ISO 26262 Road vehicles Functional Safety Part 1-10.

KELLY, T. 2004. A Systematic Approach to Safety Case Management.

KELLY, T. P. 2001. Concepts and Principles of Compositional Safety Case Construction. *Contract Research Report for QinetiQ COMSA/2001/1/1*.

ORIGIN CONSULTING YORK LIMITED. 2011a. GSN Community Standard Version 1. online.

ORIGIN CONSULTING YORK LIMITED. 2011b. GSN Pattern. online.

PALIN, R. AND HABLI, I. 2010. Assurance of Automotive Safety - A Safety Case Approach. In *SAFECOMP*. 82–96.

RIDDERHOF, W., GROSS, H.-G., AND DOERR, H. 2007. Establishing Evidence for Safety Cases in Automotive Systems - A Case Study. In *SAFECOMP*. Number SAE 2013-01-1415. 507–513.

SAFEUR TRAINING MATERIAL COMMITTEE. 2013. ECQA Certified Functional Safety Manager Training Material. training dossier.

SALINGAROS, N. 2000. The Structure of Pattern Languages. *Architectural Research Quarterly 4*, 149–161.

THE SPICE USER GROUP. Automotive SPICE.

EuroPloP '14, July 09 - 13, 2014, Irsee, Germany

ACM 978-1-4503-3416-7 ...$15.00.
http://dx.doi.org/10.1145/2721956.2721962

Automotive Embedded Software: Migration Challenges to Multi-Core Computing Platforms

Georg Macher*†, Andrea Höller*, Eric Armengaud† and Christian Kreiner*

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

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

***Abstract*—The introduction of multi-core computing platforms aims at providing more computing resources and additional interfaces to answer the needs of new automotive control strategies with respect to computing performances and connectivity (e.g. connected vehicle, hybrid powertrains). At the same time, the parallel execution and resulting resources and timing conflicts require a paradigm change for the embedded software. Consequently, efficient migration of legacy software on multi-core platform, while guaranteeing at least the same level of integrity and performance as for single cores, is a significant challenge. The contributions of this paper are (1) to provide a state-of-practice survey on multi-core CPUs and operating systems for the automotive domain, and (2) based on this survey to provide guidelines for the migration of legacy SW. Finally the related challenges and opportunities for the development of high-integrity control systems on multi-cores, as platform for dependable systems are discussed.**

***Keywords*—automotive, multi-core computing platform, functional safety.**

I. Introduction

Embedded systems are already integrated in our everyday lives and play a central role in many vital sectors including automotive, aerospace, healthcare or industry. The complexity of embedded systems has grown significantly in the automotive domain in recent years. In the automotive industry, embedded systems components are responsible for 25% of vehicle costs, while the added value from electronics components ranges between 40% for traditional vehicle up to 75% for electrics and hybrid vehicles [40].

Moore's law [30], stating the doubling of the computer capacity every 2 years, is still a strong enabler for this fast function increase and at the same time cost-per function decrease. The current development trend for computing platforms has moved from increasing the frequency of single cores to increasing the parallelism (increasing the number of cores on the same die). Multi-core and many-core technologies have a strong potential to further support the different technology domains, but simultaneously they present new challenges.

Hence, the automotive industry is facing a growing gap between the technologies and required level of expertise to make best use of them. The computing platforms are becoming more and more high-performance with concurrent computing capabilities, larger embedded memories as well as an increasing number of integrated peripherals. Low level mechanisms (e.g. memory protection, diagnostics) typically provided by the basic software or operating system are now being moved into the CPU. The complexity of these computing platforms is very high, the related user guides are comprised of several thousands of pages. In the context of automotive operating systems, the AUTOSAR [1] approach is following a similar trend by standardizing several tens of basic software (BSW) modules in several tens of thousands of specification pages. Similarly for the application software (ASW, e.g., control strategy for hybrid powertrains), the complexity is already very high and is still growing by the introduction of new applications such as advanced driver assistance systems (ADAS) or predictive energy management strategies. Additionally, the functional integration of the control strategies (e.g., transmission with combustion engine and e-drive) further raises the complexity of the resulting application.

The automotive industry is confronted with the central question of how to migrate, optimize, and validate a given application (or set of applications) on a given computing platform with a given operating system. A know-how transfer is required to take over the role of control system integrator and identify the application requirements (both functional and non-functional) and perform a mapping to the SW and HW architecture. The quality of this mapping has a direct impact on the performance of the control system, and thus of the entire mechatronic system.

The contributions of this paper are (a) the presentation of a state-of-practice survey on multicore computing platforms and operating systems for the automotive domain, and (b) the discussing of migration scenarios for legacy application software from single to multi-core computing platforms. The proposed scenarios shall support the reuse of ASW on new platforms while minimizing the migration efforts and risks. The paper is organized as follow: Section II discusses the fundamentals of parallel computing and presents the solutions available for the automotive domain. In Section III the migration scenarios are discussed, while Section IV specifically focuses on the dependability aspects. Finally, Section V concludes this work.

II. Multi-core in the Automotive Domain

The aim of this section is to discuss the fundamentals of parallel computing and provide a state-of-practice on the available solutions in the automotive domain. Figure 1 shows the overview map of the addressed topics and is intended to provide relations of this sections structure to the basic multi-core blocks. More especially, the following topics are addressed: architecture of the parallel computer (II-A, related to

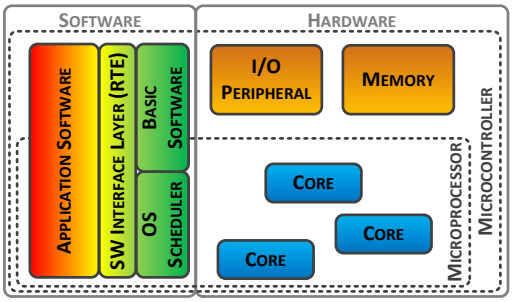


Fig. 1. Overview Map of Multi-Core Fundamentals

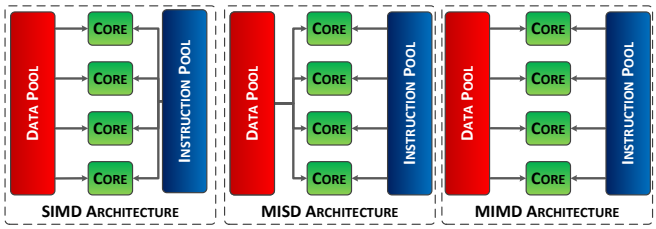


Fig. 2. Parallel Computer Architectures according Flynn's Taxonomy

the microprocessor cores), memory architecture (II-B, related to the microcontroller memory), concept of processing functions (II-C, related to the operating system and scheduler), synchronization protocols (II-D, related to OS and basic software), and scheduling policies (II-E, related to the operating system scheduler). All these factors influence the achievable speedup, efficiency and dependability features (e.g. safety, reliability) of the multi-core system. Note that most publications focus on specific aspect of multi-core development. The following publications are suggested [6], [15], [28], [36] in order to obtain a global overview. In addition to these publications, AUTOSAR's recent releases also take multi-core systems into account and present several approaches and operating system primitives to enable parallel processing in the automotive domain [2].

Typical side-effects which need to be focused when migrating to multi-core systems are: (a) starvation - when a process is perpetually denied necessary resources, (b) deadlocks - when two or more processes are each waiting for the other to finish, (c) livelocks - similar to a deadlock, except that the state of the processes is in constantly change in relation to one another, not progressing, (d) race conditions - output is dependent on the sequence or timing of other uncontrollable events, (e) priority inversion - high priority task is indirectly preempted by a medium priority task, (f) freedom from interference - failures of one process can-not influence other processes, (g) timing correctness - correctness of an operation depends not only upon its logical correctness, but also upon its computation time, and (h) execution order correctness - correctness of the execution sequence of cascaded tasks. This brief definition may make the effects of the following multi-core aspects more evident.

A. Parallel Computer Architectures

According to Flynn's taxonomy [12], [31] 3 types of multi-core computer architectures exits (fourth variant 'SISD - single instruction single data' implies only one core), see Figure 2: *Single Instruction Multiple Data (SIMD)* - this architecture can access multiple data memory locations at once and execute specific single operations in parallel. Beneficial for an application that operates with a large volume of data (such as GPU). *Multiple Instruction Single Data (MISD)* - Multiple cores are capable of handling different instructions at simultaneously for a single data stream. Exists only in theory and is not commercially available. *Multiple Instruction Multiple Data (MIMD)* - One of the most complex and most frequently used modern hardware architectures. Multiple processing elements can handle multiple independent and concurrent instructions on multiple independent data. MIMD (and single-core SISD) architecture only is appropriate for the automotive domain. SIMD, as used for graphic processing (e.g. GPU), is not currently available for powertrain controls.

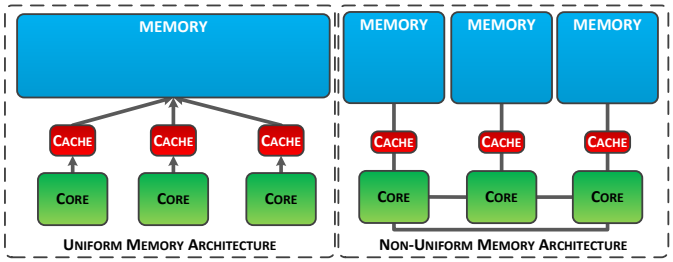


Fig. 3. Comparison of Multi-Core Memory Architectures

B. Multi-core Suitable Memory Architectures

Two memory architectural versions are commonly used for multi-core systems, uniform memory architecture (UMA) and non-uniform memory architecture (NUMA), see Figure 3. UMA divides the memory into blocks of unique data and allows unified access of each core to different blocks of the memory. NUMA is designed for concurrently running processors, it provides separate memory spaced for each core and therefore unlimited access to this local memory. Shared memory is realized by moving of shared data between local processor memories, this increases the efforts involved, but prevents all cores from starvation.

Automotive multi-core system hardware is currently based on UMA, but the NUMA approach is an improvement in terms of safety and freedom from interference and can be realized using dedicated additional hardware features (memory protection unit), if available.

C. Multiprocessing Models

Basically two types of processing models are common, see Figure 4. Asymmetric Multiprocessing (ASM) which can also be used for non-multi-core operating systems (OS). For this approach one core becomes the master, responsible for OS and the other(s) slave core(s) solely run user level applications triggered by the master OS. The Symmetric Multiprocessing (SMP) approach features OS and user level approaches on each core. All CPUs are interconnected via a bus or crossbar with access to global memory and peripherals, which requires appropriate synchronization mechanisms (see II-D).

The ASM approach is frequently used as a first step introduction into multi-core systems. Computationally intensive functions can be outsourced to the additional core and impact of multi-core constraints can be reduced. ASM is

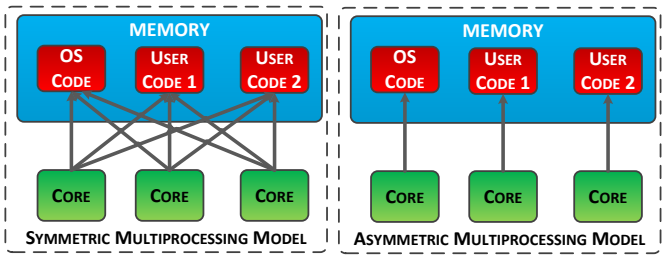


Fig. 4. Comparison of Asymmetric Multiprocessing and Symmetric Multiprocessing

also the preferred approach for AUTOSAR basic software [2], this eliminates conflicts on accessing peripherals. SMP is an intuitive approach when combining the functionalities of two separated control units into one multi-core. Therefore, both approaches are common in the automotive domain and appropriate mechanisms must be considered and applied.

D. Multi-core Synchronization

Synchronization features help to synchronize tasks with shared resources and avoid deadlocks, starvation, priority inheritance of parallel computing platforms. Synchronization primitives are features of real-time operating systems and are the building blocks of synchronization in complex systems. For this reason, they are hard to use in complex software systems, frequently make code less readable and often get lost during programming (e.g. not released locks, or interferences via busy waiting).

The most popular and commonly used synchronization primitives in consumer electronics multi-core systems are:

- Mutex - a special type of variable, which controls access to shared resources (can either be locked or unlocked).
- Semaphore - a special variable, used to record the number of available units of a particular resource and one of the most powerful means of synchronization.
- Event - used for synchronization of branch execution and whenever a task requires information from another task.
- Monitor - is a higher level synchronization primitive, using Mutex and semaphores to achieve synchronization. A monitor allows threads to have both mutual exclusion and the ability to wait (block) for a specific condition to become true, but it is not supported by every OS or programming language.
- Critical Section - mechanism to avoid race conditions by granting access to shared resources only one task at a time.

More details and further explanations on synchronization and synchronization primitives can be found in [41]. A more detailed overview of the challenges arising due parallel execution of tasks from consumer electronic point of view can be found in [34].

E. Scheduling Policies

Scheduling plays a central role by defining the ordering of the tasks to be executed. There are two principal directions there: global and partitioned schedulers. A global scheduler stores all tasks in a single queue based on their priorities and can schedule task on any available core of the system; task migration is allowed. The partitioned scheduler by contrast assigns tasks to cores and features several queues, depending on various task attributes (such as shared resource of task groups). Partitioned scheduling is more common in the automotive domain, although the applied scheduling policy strongly depends on the use-case. Various scheduling algorithms are used for the automotive domain and safety-critical applications (see [10], [11], [17], [24], [27], [41] for more details on scheduling and applications in the automotive domain).

For safety-related systems hard real-time requirements are of crucial importance, even in worst-case scenarios. One way to generate static schedules for safety-critical multi-core systems for the avionics domain is mentioned by Hilbrich et al. [16]. Timing correctness in safety-related automotive SW is also the aim of the work of Zalman et al. [42]. The paper presents principles of worst-case execution time (WCET) analysis with a commercially available WCET toolkit. The main topic of Scheidemann et al. [38] is also related to multi-core processors and AUTOSAR. A balanced graph-cut problem approach is used to resolve the allocation of software components (SWCs) to the cores. The work of Nemati et al. [32] represents a partitioning algorithm to efficiently distribute tasks along different cores. With this approach safety-critical functionality has a dedicated core, while non-safety tasks can run concurrently to improve performance.

F. Industrial and Open Sources Solutions for the Automotive Domain

The market for automotive computing platforms is relatively small in comparison to consumer electronics. The harsh environmental constraints as well as cost pressures makes a technology hand-over from consumer electronics to automotive electronics (e.g., multi-core CPU) challenging. Multi-core controller for automotive safety-critical applications are frequently delayed lockstep systems (computing the same instructions and data on two cores), but only rarely real parallel computing units.

Table I outlines currently available automotive multi-core systems and their key features (related to safety-critical application and core functionality).

The availability of automotive operating systems supporting multi-core systems is also very limited. Table II presents an overview of currently available automotive operating systems supporting automotive multi-core systems, but does not claim to be a complete or exhaustive list. Nevertheless, not all of the RTOS variants mentioned may be appropriate for the specific application or multi-core HW in use.

III. Migration Strategy Concept for Legacy Software to Multi-Core Systems

The motivation for migrating legacy software to multi-core computing platforms is to maximize the reuse (and therefore reducing costs) of ASW and control strategies already validated on previous (single core) computing platforms. While the AUTOSAR methodology targets independence between ASW and underlying computing platforms, still different aspects such as overall SW architecture and proper use of embedded peripherals still need to be considered to improve the overall performances (HW/SW co-design). In the following, the aspects of (a) migration preparation steps, (b) guidelines for migration, and (c) performances analysis after migration are

TABLE I. OVERVIEW OF AUTOMOTIVE MULTI-CORE SYSTEMS AND THEIR KEY FEATURES

Vendor	Controller Family	Multi-core Features	Distinctive Features
Freescale	MPC574xP	2x e200z4 in delayed lock step (up to 200 MHz),embedded floating point unit, 32-channel eDMA in delayed lockstep	cores and DMA controller in delayed lock step, end-to-end error correction coding, duplicated periphery, start-up self-test, voltage regulator redundancy and supply monitoring, fault collection unit
Renesas	RH850/F1H	2x RH850 (up to 120MHz), floating point unit, 32-channel DMA	clock monitor for PLL, internal oscillator and main oscillator, 4 channel data CRC, low voltage indicator, core voltage monitors, error collection coding for code flash, data flash, local RAM, and retention RAM, HW security module, memory protection unit, peripheral and processor guards
Texas Instruments	Hercules TMS570	2 x AMR Cortex R in delayed lock step (up to 300 MHz), floating point unit	separate clock tree, CPU logic built-in self-test, memory protection, memory self-test, physical CPU diversity
Infineon	AURIX	3x TriCore (up to 300 MHz), 2 TriCore with additional lockstep core	diverse lockstep architecture, emulation device chip for multicore debugging, tracing and calibration, optional extended temperature range, redundant and diverse timer modules, access permission system, safety management unit, DMA
STMicroelectronics	SPC5	2x e200z4d (up to 120MHz)	multiple clock generation, memory protection unit with ECC, AES-128 cryptographic service engine, 32 channel eDMA, optimized peripheral set for safety applications
Spansion	Traveo MB9D560	2x ARM Cortex R5 (up to 200 MHz)	multiple timers, multiple ADC, IPCU core communication, CRC generators, exclusive access memory

TABLE II. OVERVIEW OF AUTOMOTIVE MULTI-CORE OPERATING SYSTEMS AND THEIR KEY FEATURES

Vendor	RTOS Name	License Model	Key Features
ERIKA Enterprise	ERIKA OS [a]	open source for non-commercial use	ASIL D certified, OSEK/VDX based, AUTOSAR aligned, high gear community
COMASSO	COMASSO [b]	open source for community	AUTOSAR, high gear automotive community
Elektrobit	EB tresos AutoCore OS [c]	COTS	AUTOSAR 4.0, ASIL D certified, market-leading product, exhaustive tool and integration support
Vector	MICROSAR OS [d]	COTS	AUTOSAR 3.x, ASIL D certified, market-leading product, exhaustive tool and integration support
QNX	QNX Neutrino [e]	COTS	ASIL D and EAL 4+certified
ETAS	RTA-OS [f]	COTS	proof-of-concept prototype, AUTOSAR 3.0
ENEA	Enea OSE [g]	COTS	POSIX compatible
EUROS	EUROSmp [h]	COTS	currently no certification
Mentor Graphics	Nucleus RTOS [i]	free evaluation	AMP only, Mentor Hypervisor approach, no certification available

[a] http://erika.tuxfamily.org/drupal/
[b] https://www.comasso.org/
[c] https://automotive.elektrobit.com/products/ecu/eb-tresos/
[d] http://vector.com/vi_microsar_os_en.html
[e] http://www.qnx.com/products/certified_os/automotive-safety.html
[f] http://www.etas.com/en/products/rta_osek.php
[g] http://www.enea.com/solutions/rtos/ose/
[h] http://www.euros-embedded.com/v3/index.php/de/produkte/echtzeitbetriebssysteme/eurosmp
[i] http://www.mentor.com/embedded-software/nucleus/amp

discussed under the perspective of the application owner (SW integrator).

A. Migration Preparation: Selection of the CPU and of the BSW

The first step is the identification of the proper CPU and of the proper BSW / operating system. As indicated in Table I there are few, but very manifold, multi-core hardware implementations available for automotive applications. Besides the obvious criteria (e.g. availability of supporting/debugging tools, clock rates, and experiences), possible disadvantages and architectural benefits (e.g. HW implementation of MPU, peripheral bus guardians, lockstep mode cores, and HW safety features) are the contributory factors in the selection of the multi-core hardware.

Knowledge of the application is important here for understanding the required interfaces to the real world (sensors / actuators) and possible supports by the CPU peripherals. Such integration might have an important impact on the performances, e.g., when down-sampling and filtering is performed directly in HW and no longer by a low-level SW driver (requiring high-frequency interrupts). Definition of the overall architecture taking into account the peripherals embedded in HW is thus a key aspect for improving performances. Another aspect is the level of dependability (safety, security, reliability...) of the application and the respective integrity level of the computing platform (CPU and peripherals). Again, the CPUs are providing mechanisms implemented in HW (e.g. lock-step, memory protection) to shift part of the integrity issues from low-level SW to HW, thus saving computing resources for the application. Here again, refinement of the architecture taking into account the mechanisms embedded within the CPU is a key aspect for improving performances.

The next important step is the selection of an adequate automotive operating system. Although only few operating system variants exist for the automotive domain (see Table II for an overview), the selection of the OS and its applicability for different multiprocessing models (II-C), scheduling policies (II-E), and implemented synchronization primitives (II-D) is crucial. Also memory separation and protection (spatial freedom from interference) of individual tasks must be ensured and peripheral resources must be accessed with care. Peripherals, like other shared resources, are potential pitfalls and one of the biggest bottlenecks of multi-core systems. Important aspect for the choice of the BSW and operating aspects are:

- *Maturity target for the application*: The targeted development might well have a different maturity level (e.g., demonstrator, mature prototype, SOP) and therefore make different requirements on the BSW layers in respect to cost, flexibility, openness, and maturity. Open sources solutions might be preferred for rapidly making the first steps in a flexible prototyping context, while components-of-the-shelf solutions might be the better choice to rely on stable functionalities compliant to a given standard.
- *Functionalities and CPU support*: The correct migration of the BSW / OS for a specific CPU and for a specific compiler shall be available. It can occur that new CPU generations are not (fully) supported, that specific BSW layer has not been migrated or that specific HW mechanisms from the CPU are not accessible for the BSW.
- *Deployment and industrial acceptance of the BSW / OS*: The automotive industry is split into a complex ecosystem comprising car manufacturers, Tier 1 / 2 suppliers, and technology suppliers. It is unlikely that car manufacturers will develop the entire software on their own. On the contrary, the car manufacturer or Tier 1 supplier is responsible for system integration and must therefore take a decision about a computing platform and BSW / OS. The technology supplier needs to follow this choice and typically also ensure the compatibility of its application SW for different computing platforms.

In summary, selection of CPU and BSW as well as a profound understanding of their functionalities and maturity are crucial aspects for the proper integration of the application SW and for optimizing the entire control system.

B. Migration of Legacy Software to Multi-Core Systems

The process of customizing existing applications for multi-core systems is a key issue in multi-core development and a tough and enormously important task of multi-core system migration. Although, parallelization of source code takes a back seat in automotive domain compared to the merging of multiple functions in one multi-core system. Automatic supporting tools, such as parallelizing compilers or automatic schedule generators, are in early development stages and still frequently inadequate in parallelizing codes [4], [35]. The following generic steps thus turned out to be best practice for parallelizing software [6], [31]:

1) *Identification of parallelism*: As first step, the type of parallelism (data and/or task parallelism) must be determined to select the adequate computing architecture (see II-A).
2) *Profiling of existing application*: This step identify the single-core/ best possible/ current performance of particular functions or the whole application to get reference values.
3) *System decomposition*: The biggest problems of parallel programming, this step shall brake down the system into as independent as possible parts. For this no simple and straightforward way is available, due to the creative and multi-constraint nature of software architecture development. In case of intersection of parallel tasks, appropriate synchronization (see II-D) must be applied.
4) *Identification of connections*: In this step shared memories, task execution orders, and task synchronization efforts shall be determined to evaluate these intersections in later development phases and ensure that these facts do not get omitted.
5) *Detailing synchronizations*: If decomposition identified independent tasks and connection between dependent tasks are also identified, this step should determine the ranking of tasks dependencies (e.g. which task gets shared resource first, how to react on resource limitation).

As previously mentioned, merging of the functionalities into one multi-core system and making use of additional hardware features are the most common drivers for multi-core migration in the automotive context. Allocating tasks and cores efficiently and safely is the main challenge in context of real-time multi-core systems [15].

1) Shared variables need to be identified and only accessed at starting and ending time of the runnable.
2) Sequential structure diagram and data flow diagram of runnables to determine data consistency must be generated.
3) HW resources accesses shall be identified and only be done via basic software modules (ASW access to HW resources must be avoided).
4) Core mapping of task must be done with a focus on minimizing inter-core communication and optimizing the distribution of workloads.

The last step of the above mentioned task to core allocation varies from case to case, depending on the main optimization

constraints (e.g. minimizing inter-core communication, workload balancing, separation of code) and will not be reasonably feasible without an optimization tool for complex systems. Optimization tools are based on the graph-cut problem solution and are likely to provide various solutions depending on the main optimization parameters and on the software functionalities themselves. It is thus likely that every SW function update will lead to different optimized task allocations, resulting in different program flows and finally having negative side-effects for the certification of safety-critical software.

There are many toolsets available for parallelizing serial software codes in the consumer electronic domain [31]:

- *CriticalBlues Prism tool*[1]: Provides analysis and an exploration and verification environment for embedded software development using multicore architectures. The tool works for many microprocessor chips including x86, PowerPC (PPC), MIPS, ARM, and the Android operating system.
- *Vector Fabrics Pareon tool*[2]: For parallelizing serial software for Intel x86 and the ARM Cortex-A multicore embedded platforms.
- *OpenMP*[3]: Provides compiler directives, library functions, and environment variables that can be used for parallelizing serial applications. Big disadvantage, it does not assist with race condition and synchronization bugs.
- *Clean C*[4]: Is an eclipse based plug-in automatically converting C code from a single core microprocessor to a multicore microprocessor. The tool requires compliance with 29 Clean C programming rules otherwise the result is most likely non-operational.

In general, the applicability of these tool suites must be carefully scrutinized. Other approaches focus on parallelization of Matlab/Simulink models, as with Jahr et. al [20], [21] and Cha et. al [7]. Lastly, also parallelization approaches on higher level of abstraction, based on UML activity diagram with extension [15], [22], [29] exist. However, these approaches require a detailed model of the data flow and program flow dependencies and therefore require quite excessive UML modeling activities.

Absolutely independently running tasks on different cores tend to be exceptions. Inter-core communication and synchronization is not always avoidable, and should therefore be carefully considered when linking execution chains running on different cores. The use of semaphores and locks, together with split program execution over several cores is not necessarily deterministic and can create data race conditions [28]. The safest way to implement inter-core communication is by using shared memory regions with only one producer possible. OSEK OS standard base task synchronization on *WaitEvent()* service does not have an associated time-out and needs to be monitored via alarms. Nevertheless, OSEK OS is intentionally developed for single-core systems.

By contrast, OSEK's successor AUTOSAR specifies a multi-core SpinlockType mechanism (for inter-core task synchronization) and Inter OSApplication Communication (IOC), which allows communication between different applications on different cores.

[1]http://www.criticalblue.com
[2]http://www.vectorfabrics.com
[3]http://www.opnemp.org
[4]http://www.imec.be/CleanC

C. Post-Migration: Performance Analysis

Even when all the previously mentioned restrictions have been taken into account, some side-effects or design shortcomings might still exist. As a result of this situation a migration analysis and alignment of simulated and tested results is required. Also the true performance increase and execution times of the overall system compared with the expected performance and timings should be correlated. Here too debug, instrumentation, and calibration tool supports are of major importance. In contrast to single-core systems, multi-core systems inherit new timing effects (such as delays through resource conflicts, IOC overheads) which must be analyzed and harmonized with the timing models; an evaluation of meeting the safety-critical timings at corner cases must also be done. Several techniques and a list of supporting tools to identify concurrency issues can be found in [34]. The taxonomy of Fernandez et. al [9] presents state-of-the-art techniques to analyze timing impacts of resource contention. Several tools are applicable for timing analysis:

- *Precision Pro*: Tool prototype for generation of hard real-time schedule tables.
- *SymTA/S*[5]: Tool for scheduling analysis, architecture optimization and timing verification of multi-core ECUs and distributed embedded systems (E2E communication).
- *RTaW*[6]: Simulation and timing analysis tools aiming in automating design, verification, and configuration of embedded networks and ECUs.
- Inchron[7]: Design and test tools for model-based real-time simulation and analysis.

IV. Approaches for Ensuring the Integrity of Multi-Core Computing Platforms

According to [3], the original definition of dependability is the ability to deliver service that can justifiably be trusted. In the context of the automotive domain some dependability aspects are predominant such as e.g., reliability (life time and reputation) and safety (absence of catastrophic consequences). The 3-layer monitoring concept [28], [43] is meanwhile state-of-practice; it consists of software functionality (layer 1) implementing the normal functions, function monitoring (layer 2) preventing the physical system to move to an unsafe state, and device monitoring (layer 3) ensuring integrity of the control unit and typically integrating a handshake with an external chip (alive message).

It is probable that the most important characteristic of multi-core computing platforms is the parallel execution of instructions, which can have a serious impact on the determinism of the application. Hence parallel execution can lead to race conditions and resource conflicts between the tasks, thus creating deadlock or starvation. The performances and more especially the timing requirements (in case of hard real-time systems) can no longer be guaranteed, thus having an impact on the dependability of the system. A core aspect is therefore to guarantee freedom from interference, defined in [19] as the *absence of cascading failures between two or more elements that could lead to the violation of a safety requirement*. Proper encapsulation of the tasks must be provided.

Traditional methodologies are often not applicable without further ado [39] and too conservative SW migration could even

[5]https://www.symtavision.com/products/symtas-traceanalyzer/
[6]http://www.realtimeatwork.com/
[7]http://www.inchron.com/

lead to a performance decline of the multi-core system. The challenges involved in this context are clearly manifold and one common solution for all of these challenges is not yet available. Some publications thus focus on the general applicability of multi-core systems for safety-critical applications [23], [35]. The work of Gashi et al. [13] focuses on redundancy and diversity, and their effects on the safety and security of embedded systems. This work is part of the SeSaMo (Security and Safety Modeling for Embedded Systems) project, which focuses on synergies and trade-offs between security and safety through concrete use-cases. As indicated in this work and mentioned in the earlier section, a proper selection of the multi-core hardware based upon architecture constraints is crucial. According to Pollack's Rule [8] the increase of available performance (roughly proportional to the square root of increase of complexity) should on the other hand not be overrated.

Safety standards, such as the road vehicles functional safety norm ISO 26262 [19] and its basic norm IEC 61508 [18] present requirements and guidance for safety-critical system development. An important concept well suitable for multi-core computing platforms is decomposition of the control path as two independent channels. Due to the redundancy, the integrity of each path can be lowered (e.g., decomposing an ASIL D system into two independent ASIL B components). To support this independence, evidence for the freedom from interference of these channels is required. A guide to functional safety of automotive systems according to ISO 26262 can be found in [5], [14] or in the SafEUr functional safety manager trainings [37].

Freedom from interference is the most crucial feature needed to be ensured for safety-critical systems. This can be argued via: (a) freedom from temporal interference (ensuring meeting timing constraints), (b) freedom from data interference (preventing from data corruption), and (c) freedom from communication interference (ensuring end-to-end communication within time limits). Temporal autonomy must be ensured by the real-time operating system (RTOS). Asymmetric Multi-Processing model is recommended for safety-critical context to ensure task or thread locking to a specific core. The main requirements for safety-critical applications operating systems are: (a) guaranteeing mutual exclusion over critical data sections, (b) ensuring freedom from deadlock, livelock, and starvation, (c) prevention of uncontrolled priority inversion, and (d) ensuring freedom from interference. Ensuring data consistency and prevention from data corruption must be provided either via dedicated HW features (such as MPU) or via synchronization primitives supported by the RTOS. A common approach is to store safety-related data in diverse representation on a multiple basis to ensure consistency of these data and make use of error-correcting codes (ECC). Freedom from interference of communication and peripherals also needs to be ensured via RTOS and basic software. Peripherals, like other shared resources, are potential pitfalls for freedom from interference argumentation and one of the biggest bottlenecks of multi-core systems. Also exclusive usage for peripherals needs to be ensured via synchronization primitives, which could also lead to obscured dependencies. A frequently applied approach in the safety-related context is to assign all peripheral access to one core, which acts as an IO master. The AUTOSAR multi-core approach [2] also recommends allocation of BSW functionality solely on one master core. This implicitly prevents the parallel cores from accessing the same microcontroller hardware resource simultaneously [15].

A promising approach for the development of dependable systems is the time-triggered paradigm [25], [26], [33]. An event-triggered architecture is characterized by the fact that all system activities are initiated by an event and consequently react to its environment (an operation is started as soon as the event is received, regardless the current processing status). On the contrary, in the time-triggered architecture, every action is derived solely from the progression of real-time and thus follows the progression of its environment (an operation is started at a pre-defined starting point and processes the information that has collected since the last computation; conflicts about processing resources are avoided per construction). The time-triggered paradigm provides the following attributes, important for the efficient development of dependable systems:

1) Stability of prior certification, means that validated services do not refute by integration.
2) Constructive integration, in a way that additional services do not affect already integrated timing behavior.
3) Robustness against malicious temporal control signal interfaces.
4) Timing and data dependencies resolvable at system design time, thus simplifying the inter-task synchronization and avoiding race conditions.
5) Deterministic task execution with guaranteed worst-case timing.

The time-triggered paradigm, while not fully industrialized for the automotive domain yet, still provides important concepts for the design of high integrity multi-core platforms as a basis for dependable applications.

V. Conclusion

Multi-core computing platforms as well as respective basis software and operating systems are providing good opportunities to improve existing automotive control strategies or even becoming the enabler for new functionalities (e.g., ADAS). At the same time, the introduction of concurrent execution by different cores is leading to a paradigm change in the software. Topics such as resource conflicts, race conditions, task synchronizations, guaranteed execution time, and freedom from interference are becoming more and more relevant. These challenges, however, need to be resolved in a holistic manner in order to enable legacy SW to run on a multi-core platform and new control strategies to be deployed successfully. Hence, during SW integration on multi-core the computing platform, BSW and operating systems must be configured or even tailored according to the specific needs from the control strategy. The final challenge is the ability to efficiently combine expertise between multi-core computing platforms, operating systems and control strategies. Hence, only with this combination can the real performances of the resulting automotive embedded system be achieved. Based on a state-of-practice survey in the automotive domain, this paper provides guidance for the migration of legacy SW from single core to multi-core platforms.

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 nb 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] AUTOSAR Development Cooperation. Guide to Multi-Core Systems. online, 2013.

[3] A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and its Threats - A Taxonomy. In R. Jacquart, editor, *IFIP Congress Topical Sessions*, pages 91–120. Kluwer, 2004.

[4] V. Bendiuga. Multi-Core Pattern. Master's thesis, School of Innovation, Design and Engineering Malardalen University Sweden, December 2012.

[5] K. Boehringer and M. Kroh. Funktionale Sicherheit in der Praxis, July 2013.

[6] B. Brecht and S. Luys. 201121, March 2011.

[7] M. Cha, S. K. Kim, and K. H. Kim. An Automatic Parallelization Scheme for Simulink-based Real-Time Multicore Systems Systems. In *Software Technology*, volume 5 of *ASTL*, pages 215 – 217. Sience & Engineering Research Support Society, 2012.

[8] J. Circello. Rationale for Multicore Architectures in Auto Apps. Webinar, June 2011.

[9] G. Fernandez, J. Abella, E. Quiñones, C. Rochange, T. Vardanega, and F. J. Cazorla. Contention in Multicore Hardware Shared Resources: Understanding of the State of the Art. In H. Falk, editor, *14th International Workshop on Worst-Case Execution Time Analysis*, volume 39 of *OpenAccess Series in Informatics (OASIcs)*, pages 31–42, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[10] C. Ficek, N. Feiertag, and K. Richter. Applying the AUTOSAR timing protection to build safe and efficient ISO 26262 mixed-criticality systems, 2012.

[11] C. Ficek, K. Richter, and N. Feiertag. Schedule Design to Guarantee Freedom of Interference in Mixed Criticality Systems. *SAE Int. Journal Passeng. Cars - Electron. Electr. Syst.*, 5:46–54, 04 2012.

[12] M. J. Flynn. Some Computer Organizations and Their Effectiveness. *IEEE Trans. Comput.*, 21(9):948–960, September 1972.

[13] I. Gashi, A. Povyakalo, L. Strigini, M. Matschnig, T. Hinterstoisser, and B. Fischer. Diversity for Safety and Security in Embedded Systems. In *International Conference on Dependable Systems and Networks*, volume 26, 06 2014.

[14] V. Gebhardt, G. Rieger, J. Mottok, and C. Giesselbach. *Funktionale Sicherheit nach ISO 262626 - Ein Praxisleitfaden zur Umsetzung*, volume 1. Auflage. dpunkt.verlag, 2013.

[15] J. Han, J. S. Park, M. Deubzer, J. Harnisch, and P. Leteinturier. Efficient Multi-Core Software Design Space Exploration for Hybrid Control Unit Integration. In *SAE Technical Paper*. SAE International, 04 2014.

[16] R. Hilbrich and H.-J. Goltz. Model-based Generation of Static Schedules for Safety Critical Multi-Core Systems in the Avionics Domain. In *WMSE11*, 2011.

[17] R. Hilbrich, J. Reinier van Kampenhout, and H.-J. Goltz. Modellbasierte Generierung statischer Schedules fuer sicherheitskritische, eingebettete Systeme mit Multicore-Prozessoren und harten Echtzeitanforderungen. *Informatik aktuell*, pages 29 – 38, 2012.

[18] ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.

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

[20] R. Jahr, M. Frieb, M. Gerdes, and T. Ungerer. Model-based Parallelization and Optimization of an Industrial Control Code. *Tagungsband des Dagstuhl-Workshops*, page 63, 2014.

[21] R. Jahr, M. Frieb, M. Gerdes, T. Ungerer, A. Hugl, and H. Regler. Paving the Way for Multi-cores in Industrial Hard Real-time Control Applications. In *9th IEEE International Symposium on Industrial Embedded Systems (SIES), WiP-Session*, 2014.

[22] R. Jahr, M. Gerdes, and T. Ungerer. On Efficient and Effective Model-based Parallelization of Hard Real-Time Applications. *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme IX*, pages 50–59, 2013.

[23] S. Jena and M. Srinivas. On the Suitability of Multi-Core Processing for Embedded Automotive Systems. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on*, pages 315–322, Oct 2012.

[24] F. Kluge, C. Yu, J. Mische, S. Uhrig, and T. Ungerer. Implementing AUTOSAR Scheduling and Resource Management on an Embedded SMT Processor. In *12th International Workshop on Software & Compilers for Embedded Systems*, pages 33 –42, 2009.

[25] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1997.

[26] H. Kopetz and G. Bauer. The Time-Triggered Architecture. In *PROCEEDINGS OF THE IEEE*, pages 112–126, 2003.

[27] E. Lalo and M. Deubzer. Effects of Task Priority Assignment in Embedded Multicore Real-Time Systems. *Embedded Systems Engineering*, June 2014.

[28] P. Leteinturier, S. Brewerton, and K. Scheibert. MultiCore Benefits & Challenges for Automotive Applications. In *SAE Technical Paper*. SAE International, 04 2008.

[29] C.-S. Lin, C.-H. Lu, S.-W. Lin, Y.-R. Chen, and P.-A. Hsiung. VERTAF/Multi-Core: A SysML-Based Application Framework for Multi-Core Embedded Software Development. *Journal on Computer Science and Technology*, 26(3):448–462, 2011.

[30] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114 – 117, April 1965.

[31] B. Moyer. *Real World Multicore Embedded Systems*. Expert guide. Newnes, Newton, MA, USA, 1st edition, 2013.

[32] F. Nemati, M. Behnam, and T. Nolte. Efficiently Migrating Real-Time Systems to Multi-Cores. In *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2009, September 22-25, 2008, Palma de Mallorca, Spain*, pages 1–8, 2009.

[33] R. Obermaisser, C. El Salloum, B. Huber, and H. Kopetz. The time-triggered System-on-a-Chip architecture . In *IEEE International Symposium on Industrial Electronics, 2008. ISIE 2008*, pages 1941–1947, 2008.

[34] R. V. Patil and B. George. Tools And Techniques To Identify Concurrency Issues. online, June 2008.

[35] F. Reichenbach and A. Wold. Multi-core Technology – Next Evolution Step in Safety Critical Systems for Industrial Applications? In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 339–346, Sept 2010.

[36] K. Richter and S. Schliecker. Sicher auf Multi-Core umsteigen. *HANSER automotive*, 10:38–42, 10 2013.

[37] SafEUr Training Material Committee. ECQA Certified Functional Safety Manager Training Material. training dossier, April 2013.

[38] K. Scheidemann, M. Knapp, and C. Stellwag. Load Balancing in AUTOSAR-Multicore-Systemen. *WEKA Fachmedien GmbH*, April 2010.

[39] D. Schneider, E. Armengaud, and E. Schoitsch. Towards Trust Assurance and Certification in Cyber-Physical Systems. In *SAFECOMP Workshops*, pages 180–191, 2011.

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

[41] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007.

[42] R. Zalman, A. Griessing, and P. Emberson. Timing Correctness in Safety-Related Automotive Software. In *SAE Technical Paper*. SAE International, 04 2011.

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

Pattern Catalog for MultiCore Migration of Embedded Automotive Systems

Georg MACHER, Graz University of Technology
Andrea HÖLLER, Graz University of Technology
Eric ARMENGAUD, AVL List GmbH
Christian KREINER, Graz University of Technology

Embedded systems are already integrated into our everyday life and play a central role in all domains including automotive, aerospace, healthcare or industry. The complexity of embedded systems and software has grown significantly in recent years. For the automotive industry, as an example, embedded systems components are responsible for 25% of vehicle costs, while the added value from electronic components range between 40% for traditional vehicle up to 75% for electrics and hybrid vehicles. Driven by the ongoing challenge of reducing cost and simultaneously replacing safety-critical mechanical systems with more advanced embedded system, multi-core systems are also increasingly relevant for safety-critical embedded systems. However, when migrating safety-critical applications to multi-core systems special attention should be paid to preserve determinism, and assure certifiability of the system.
Aim of the paper is to present a pattern catalog to provide a migration strategy for legacy software developed for safety-critical embedded single-core systems to parallel computing multi-core platforms. The pattern catalog shall help system integrators and software developers in the automotive domain to migrate existing safety-critical software to more advanced multi-core platforms. Furthermore, this paper highlight aspects and demands which influence the migration of safety-critical systems to multi-core computing platforms.

Categories and Subject Descriptors: H.5.0 [**Information Interfaces and Presentation**]: General—; H.1.1 [**Models and Principles**]: System and Information Theory—; K.2.2 [**Computers and Society**]: Social Issues—

General Terms: Embedded Automotive Multi-Core

Additional Key Words and Phrases: embedded systems, multi-core, migration strategy.

ACM Reference Format:

Macher G., Höller A., Armengaud E., and Kreiner C., 2015. Pattern Catalog for MultiCore Migration of Embedded Automotive Systems. jn 0, 0, Article 0 (0), 11 pages.

1. INTRODUCTION

Moore's law [Moore 1965], stating the doubling of the computer capacity every 2 years, is still a strong enabler for this fast function increase and at the same time cost-per function decrease. The current development trend for computing platforms has moved from increasing the frequency of single cores to increasing the parallelism (increasing the number of cores on the same die). Multi-core and many-core technologies have strong potential to further support the different technology domains, but at the same time present new challenges.

Hence, the automotive industry is facing a growing gap between the technologies and required level of expertise to make best use of them. The computing platforms are becoming more and more high-performance with concurrent

EuroPloP '15, July 08 - 12, 2015, Kaufbeuren, Germany

ACM 978-1-4503-3847-9/15/07 ...$15.00.
http://dx.doi.org/10.1145/2855321.2855346

computing capabilities, larger embedded memories as well as increasing number of integrated peripherals. The complexity of these computing platforms is very high, the related user guides is made of several of thousands of pages. Regarding automotive software, approaches are following a similar trend by standardizing software modules in several tens of thousands pages of specification.

The automotive industry is confronted to the central question how to migrate, optimize and validate a given application (or set of applications) on a given computing platform with a given operating system. A know-how transfer is required to take over the role of control system integrator and identify the application requirements (both functional and non-functional) and perform a mapping to the SW and HW architecture. The quality of this mapping has a direct impact on the performance of the control system, and thus of the entire mechatronic system.

Contributions of this paper is a pattern catalog intended for migration scenarios for legacy application software from single to multi-core computing platforms. The presented pattern catalog shall support the reuse of legacy software on new multi-core platforms while minimizing the migration efforts and risks. The pattern catalog has been mined in industrial projects done in the automotive domain. The paper is organized as follows: In Section 2 the migration pattern catalog is briefly described and the two key patterns are discussed in Section 3 and 4. Section 5 additionally discusses the fundamentals of parallel computing and solutions available for the automotive domain. Finally, Section 6 concludes this work.

2. EMBEDDED SYSTEM MULTI-CORE MIGRATION PATTERN LANGUAGE

The first section gives a brief overview of the pattern catalog itself. This overview is followed by a brief presentation of the patterns included in the pattern catalog in pattlet form and the description of the pattern description structure of the two main patterns. The two most essential patterns of the pattern catalog (also highlighted in Figure 1) are then described in more details in Sections 3 and 4 of this document. The other patterns will be part of future publications.

2.1 Overview of the Pattern Language

In addition to embedded systems also for safety-critical applications the demand for multi-core systems is increasing. For multi-core hardware also the knowledge transfer from consumer electronic multi-core systems to embedded real-time multi-core systems is applicable [Moyer 2013]. But for safety-critical systems it is of utmost importance to analyze possible certification and design pitfalls that multi-core might inherit. Multi-core systems are parallelizing execution of instructions, which can jeopardize safety-critical application's most important characteristic: determinism. Therefore rigorous mitigation strategy for using multi-core systems in safety-critical context is required [Brecht and Luys 2011].

This pattern catalog is intended for migration of safety-critical systems to multi-core platforms. Parallelizing of such functionalities is usually not easy, but requires proving of freedom from interference with other functions. Additionally, the computing performances of single-core platforms reached their limits and special hardware functionalities are frequently only available with modern multi-core systems, thus a sufficient migration and parallelization of SW functionalities is not only desired but demanded.

Therefore, a 6 steps approach for migration of existing software from single-core to multi-core systems helps to migrate in a more structured way and to avoid common migration errors and pitfalls. Figure 1 shows the necessary migration steps and the related patterns of the *Embedded System Multi-Core Migration* pattern catalog. The individual patterns are described briefly (in pattlet format) in the following sections. While the two most essential patterns will be described in more details in the following sections of this work. With the application of this pattern catalog the following assets and drawbacks are implied:

+ Potential multi-core migration pitfalls may be avoided.
+ Determinism and worst-case timing requirements can be proven.
+ Special multi-core HW features can be used more optimal.
+ May help to overcome inhibition level for migration of system.

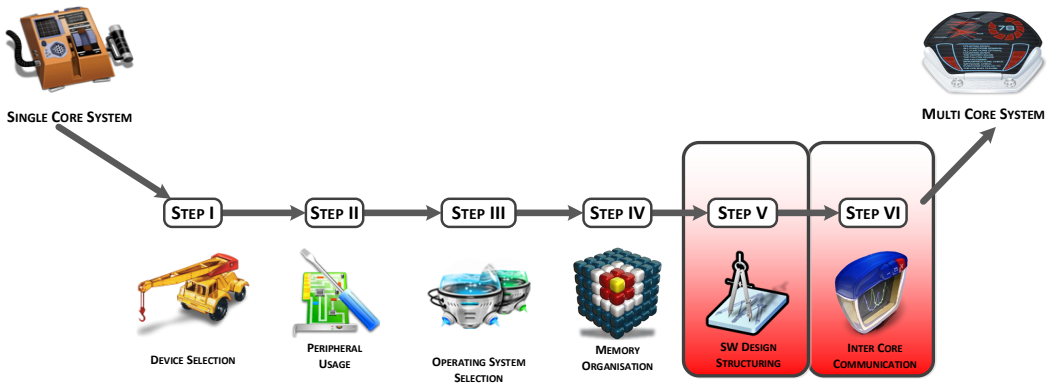


Fig. 1. Overview of the Migration Steps of the Pattern Catalog

- Requires additional efforts for migration of legacy code.

2.2 Presentation of the Patterns included in the Pattern Catalog

The presented pattern catalog consists of 6 patterns for migration from single-core to multi-core systems. The two most essential patterns will be described in more details in the following sections of this work, while the other patterns are described briefly in this section in pattlet format. These patterns will be part of future publications.

Pattern	Problem	Solution
Device Selection	Single-core embedded system controller reach their computing power limitations, while demands for more complex software functions still increases	Therefore, the integration of multiple single-core function controller into one multi-core systems is required. Proper multi-core device shall be selected, based upon architecture and processing constraints.
Peripheral Usage	Each function of an embedded system requires peripherals to interact with the environment, but multi-core systems have only limited number of physical pins.	Peripheral interfaces must be shared between cores in a proper way to support all function interfaces mutually independent and within correct timing limits.
Operating System Selection	Standard automotive real-time operating systems (RTOS) do not adequately support multi-core features and lead to known parallel processing problems	Proper multi-core RTOS must be chosen to avoid parallel processing problems (such as deadlocks, livelocks, starvation, race conditions, non-freedom from interference).
Memory Organization	Multi-core systems imply different available memory sections, which affect execution time and hamper correct and optimized data exchange between parallel executions.	Proper memory organization and access management of memory has to be ensured. Also memory separation and protection (spatial freedom from interference) of safety-critical tasks must be ensured.

2.3 Specification of the Pattern Structure

We specify the following two pattern in more details in the typical structure of [Salingaros 2000; Alexander et al. 1977]:

Context - Describes the situation in which the pattern can be applied and in which the problem occurs.

Problem - States the scenario which requires action to be taken and problems the pattern shall solve.

Forces - Gives additional motivation for usage of the pattern and describes the constraints which shape the specific solution for the problem.

Solution - Describes the steps and actions to be taken to solve the existing problem in the focus of the given context and specific forces.

Consequences - The application of the pattern solves the problem, but also results in some consequences, which themselves also include side effects.

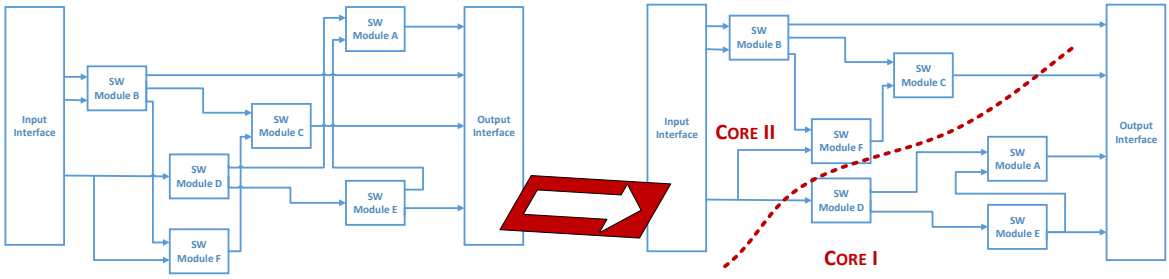


Fig. 2. Depiction of the SW Architectural Design Structuring Pattern

3. *SW ARCHITECTURAL DESIGN STRUCTURING* PATTERN

3.1 Context

Usually SW architectures are originally designed for single-core systems and not optimized for multi-core parallel execution features. Therefore, such a SW architecture must be optimized for multi-core systems. Allocating tasks on cores efficiently and safe is the main challenge in context of safety-critical multi-core systems. Multicore platform enables concurrent execution of tasks, which implies also that the resource management may be corrupted due parallel executed tasks and the execution order of tasks may vary. Such problems may be dormant issues on single-core systems, but manifest when combined with multi-core system technology. The task allocation problem is a NP-hard problem and manual definition of correct real-time scheduling for multi-core systems is not applicable.

3.2 Problem

Using multi-core systems for single-core SW architecture execution may not lead to expected speedup and can introduce parallel task execution problems (e.g. deadlocks, starvation, and race conditions).

3.3 Forces

—Whole SW system cannot be allocated to single-core - when single core SW systems are migrated to new (multi-core) systems this is usually triggered due to the fact that SW architecture does not fit within the system limits.

—Mapping of tasks to cores does not result in expected speedup - frequently speedup is expected to be equivalent to number of cores, which is a very rough estimation and cannot be met due to required communication and synchronization efforts.

—Maximizing of speedup is required to enable execution of more tasks.

—Only correct execution order of tasks ensures correct computation of output - resulting due to data dependencies and time-out requirements for data.

—Task execution order may vary due to parallel task execution - independently running cores may start up with varying delays and time jitters can occur during execution.

—Speedup of multi-core systems only achievable due to parallelization - usually individual cores of the multi-core systems operate at lower frequency than single-cores and only gain speedup due to parallel execution

—Shared resources hamper parallel execution - shared resources must only be accessed by a single core at time, which prevents parallel access and implies delay times and synchronization.

—Miss fitting parallelization may lead to negative speedup due to reduced computation power of a single MC core compared to single-core systems.

3.4 Solution

Analyze and optimize the SW architecture for parallel computing execution. The SW architecture must be partitioned to support correct execution order of tasks on parallel cores.

Hence, collect all SW tasks and split up the tasks in groups which can be executed in parallel:

(1) Identify shared variables and access them only at starting and ending time of the task.
(2) Split tasks into task groups to enable parallel execution.
(3) Protect concurrent access to shared resources via dedicated synchronization primitives.
(4) Assign groups of tasks to different cores of the multi-core system.

For further readings, a description for the generation of static schedules for safety-critical multi-core systems is mentioned by Hilbrich et al. [Hilbrich and Goltz 2011] and by the work of Zalman et al. [Zalman et al. 2011]. Furthermore, the work of Nemati et al. [Nemati et al. 2009] represents a partitioning algorithm to efficiently distribute tasks along different cores.

3.5 Consequences

+ increases speedup due to parallel execution of tasks
+ optimization for application of multi-core features
+ work-load balance between cores
- tedious task, requires adequate tool support
- additional efforts (cost/time/skills) for redesign of SW architecture required
- more complicated timing analysis due to parallel execution
- parallelization activity is complex and requires special know-how and tool support
- errors are harder to find in parallelized code
- correct task execution harder to test due to parallelization
- timing errors harder to find

4. *INTER-CORE COMMUNICATION AND SYNCHRONIZATION* PATTERN

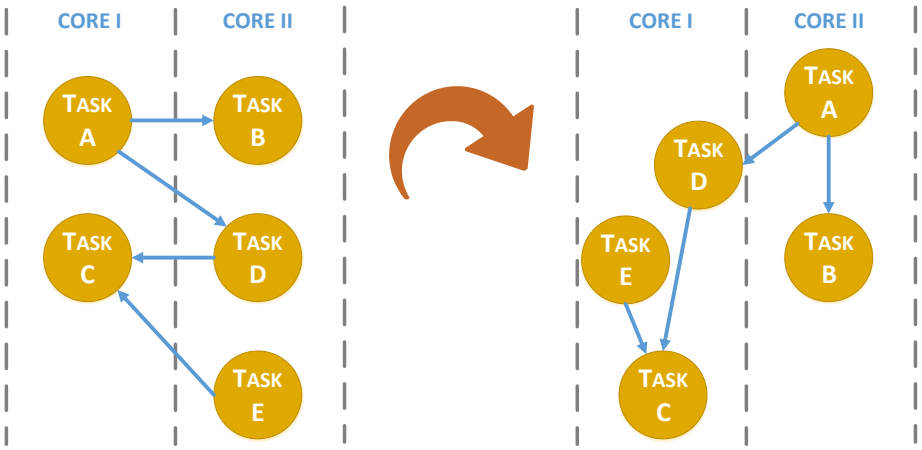


Fig. 3. Depiction of the Inter- Core Communication and Synchronization Pattern

4.1 Context

Absolutely independently running tasks on different cores are rarely. Excessive parallelization of SW may lead to immoderate synchronization time overheads; inter-core communication and synchronization decreases the overall system performance.

4.2 Problem

Parallel execution requires exorbitant amount of time for synchronization of cores and reduces parallelization speedup.

4.3 Forces

—core cycle-time is not fully synchronized - time jitter and varying startup delays can occur between cores.

—need for synchronization of data and communication between cores

—non-interacting tasks rather uncommon in automotive applications

—independently running cores must be blocked to wait for longest execution to finish - if synchronization of independent tasks is required the longest execution time of all parallel tasks is required and therefore other tasks are required to wait.

—SW functions may be spread over different cores for execution - due to the fact that the SW functions complexity may be off the resource limits of a single core.

4.4 Solution

Therefore, decrease inter-core communication efforts by clever allocation of related functions onto the same core.

Analyze data dependencies and information exchange between SW tasks. Afterwards split up the tasks in groups which interchange information only between tasks of same group. Thus minimize communication between groups of tasks. As second step, allocate these groups to dedicated cores for execution.

Inter-core communication and synchronization is not always avoidable, and should therefore be carefully considered when linking execution chains running on different cores.

4.5 Consequences

+ minimizes the synchronization points - due to minimizing inter-core communication
+ enabling more independently parallel execution with less synchronization points - due to reduced synchronization points and less inter-core communication
+ reduction of time consuming data exchange across core boundaries - this ensures further speedup and better parallelism of SW execution
+ higher overall speedup due to less communication efforts
- application specific mapping - changes in application code also requires reanalysis of inter-core communication and might lead to new task to core assignments
- different workloads on cores - due to minimization of inter-core communication workload balancing may not be perfect.
- non-optimal parallelization of SW - inter-core communication and reduction efforts may hinder parallelization of software

5. RELATED WORK

Several existing approaches deal with safety-critical systems and multi-core platforms. Obviously the challenges in this context are manifold and a common solution for all challenges is not yet available. Therefore, some publications focus on the general applicability of multi-core systems for safety-critical applications [Jena and Srinivas 2012; Reichenbach and Wold 2010].

Numerous publications related to each of the following sections exist, even solely focusing the automotive domain. Therefore, the following paragraphs solely present a brief overview and do not claim to be complete or exhaustive.

5.1 Scheduling and Timing Analysis

For safety-related systems hard real-time requirements are of crucial importance (also for worst-case scenarios). One way to generate static schedules for safety-critical multi-core systems for the avionics domain is mentioned by Hilbrich et al. [Hilbrich and Goltz 2011]. The authors state that generating static schedules for simple single-core systems is possible manually, but novel approaches are needed as generating schedulings for multi-core systems increases in complexity. Furthermore, a scheduling tool called PRECISION PRO, capable of processing models presented in a textual notation is presented.

Timing correctness in safety-related automotive SW is also the aim of the work of Zalman et al. [Zalman et al. 2011]. The paper presents principles of WCET analysis with a commercially available WCET toolkit.

Lakshamanan et. al [Lakshmanan et al. 2011] describe the timing uncertainties introduced by standard test-and-set spinlock mechanisms of AUTOSAR OS. The authors present an associated analysis used to bind the worst-case waiting times for accessing shared resources.

5.2 Resource Sharing

The AUTOSAR functional definition is based on the concept of single-threaded processors. Kluge et al. [Kluge et al. 2009] discuss the implementation of an AUTOSAR operating system interface on a simultaneous multi-threaded processor and propose some extensions to the AUTOSAR specification. The approach specially considers problems of synchronization and resource management and introduces a task filtering solution. This work mainly supports predictable timing for the task with the highest priority and is therefore not applicable for automotive mixed-critical systems.

5.3 SW to Core Allocation and SW Parallelization

The main topic of Scheidemann et al. [Scheidemann et al. 2010] is also related to multi-core processors and AUTOSAR. A balanced graph-cut problem approach is used to resolve the allocation of software components (SWCs) to the cores. Nevertheless, the problem of an un-defined execution order (no guarantied data flow between runnables of one particular task), due to parallel execution on different cores remains with this approach.

The work of Nemati et al. [Nemati et al. 2009] represents a partitioning algorithm to efficiently distribute tasks along different cores. With this approach safety-critical functionality have a dedicated core, while non-safety tasks can run concurrently to improve performance. The algorithm identifies task constrains (e.g. dependencies, timing attributes, resource sharing) and partitions the SW based on the bin-packing problem, which is known to be NP-hard. It also considers the approaches rate-monotonic scheduling, scheduling by criticality, and the combination of both (so-called hybrid approach) for multi-core systems. With the presented approach in this work global resources potentially lead to high blocking times, and influence task execution on other cores (endangerment of freedom from interference of tasks).

5.4 Communication

Niklas et al. [Niklas et al. 2012] deal with another issue of safety-related software development for multi-core systems. Their work deals with safe end-to-end (E2E) communication, AUTOSAR 4.0, and a set of mechanisms that are used to identify and detect communication errors.
However, there is no discussion with respect to the seamless description of timing requirements from system level to software components presented.

The work of Devika and Syama [Devika and Syama 2013] present an overview of AUTOSAR multi-core operating system implementations and communication primitives to enable parallel processing in context of the automotive domain.

6. CONCLUSION

Multi-core computing platforms as well as respective basis software and operating systems provide good opportunities to improve existing automotive control strategies or even be the enabler for new functionalities (e.g., ADAS). The challenge, however, is the ability to combine expertise between computing platforms, operating systems and specific application SW. Hence, only with this combination the real performances of the resulting automotive embedded system can be used. Based on a state-of-practice survey in the automotive domain, this paper provides guidance for the migration of legacy SW from single core to multi-core platforms. Future publications including the pattern briefly mentioned in pattlet format and work related to multi-core migration problems and strategies are also planned.

Acknowledgments

The authors would like to express our thanks to our shepherd Ville Reijonen who had a determining influence on the improvement of our paper and untiringly helped to improve the maturity of the paper in several iterations. The telephone conferences together with our shepherd helped to evolve the maturity of the pattern in a fast and straightforward way. Furthermore we would like to thank Christopher Preschern and our co-author Christian Kreiner who called our attention to pattern approaches and the EuroPLoP conference.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n# 608988 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).

Finally, 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

ALEXANDER, C., ISHIKAWA, S., SILVERSTEIN, M., JACOBSON, M., FIKSDAHL-KING, I., AND ANGEL, S. 1977. A Pattern Language. *Oxford University Press, New York*.

BRECHT, B. AND LUYS, S. 2011. Application of multi-core CPUs in a safety-critical context. Tech. rep., Barco Defense and Aerospace. March.

DEVIKA, K. AND SYAMA, R. 2013. An Overview of AUTOSAR Multicore Operating System Implementation. In *International Journal of Innovative Research in Science, Engineering and Technology*. Vol. Vol. 2. IJIRSET, 3163 –3169.

HILBRICH, R. AND GOLTZ, H.-J. 2011. Model-based Generation of Static Schedules for Safety Critical Multi-Core Systems in the Avionics Domain. In *WMSE11*.

JENA, S. AND SRINIVAS, M. 2012. On the Suitability of Multi-Core Processing for Embedded Automotive Systems. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on*. 315–322.

KLUGE, F., YU, C., MISCHE, J., UHRIG, S., AND UNGERER, T. 2009. Implementing AUTOSAR Scheduling and Resource Management on an Embedded SMT Processor. In *12th International Workshop on Software & Compilers for Embedded Systems*. 33 –42.

LAKSHMANAN, K. S., BHATIA, G., AND RAJKUMAR, R. 2011. AUTOSAR Extensions for Predictable Task Synchronization in Multi-Core ECUs. In *SAE Technical Paper*. SAE International.

MOORE, G. E. 1965. Cramming more components onto integrated circuits. *Electronics 38,* 8, 114 – 117.

MOYER, B. 2013. *Real World Multicore Embedded Systems* 1st Ed. Expert guide. Newnes, Newton, MA, USA.

NEMATI, F., BEHNAM, M., AND NOLTE, T. 2009. Efficiently Migrating Real-Time Systems to Multi-Cores. In *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2009, September 22-25, 2008, Palma de Mallorca, Spain*. 1–8.

NIKLAS, M., VOGET, S., AND MOTTOK, J. 2012. Safety-relevant development by adaptation of standardized safety concepts in AUTOSAR 4.0.

REICHENBACH, F. AND WOLD, A. 2010. Multi-core Technology – Next Evolution Step in Safety Critical Systems for Industrial Applications? In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*. 339–346.

SALINGAROS, N. 2000. The Structure of Pattern Languages. *Architectural Research Quarterly 4*, 149–161.

SCHEIDEMANN, K., KNAPP, M., AND STELLWAG, C. 2010. Load Balancing in AUTOSAR-Multicore-Systemen. *WEKA Fachmedien GmbH*.

ZALMAN, R., GRIESSING, A., AND EMBERSON, P. 2011. Timing Correctness in Safety-Related Automotive Software. In *SAE Technical Paper*. SAE International.

EuroPloP '15, July 08 - 12, 2015, Kaufbeuren, Germany

ACM 978-1-4503-3847-9/15/07 ...$15.00.
http://dx.doi.org/10.1145/2855321.2855346

Service Deterioration Analysis (SDA): An Early Development Phase Dependability Analysis Method

Georg Macher*†, Andrea Höller*, Harald Sporer*, Eric Armengaud† and Christian Kreiner*

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

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

***Abstract*—Dependability is a superordinate concept regrouping different system attributes such as reliability, safety, security, or availability and a key selling point of modern embedded systems. Dependable systems rely on mature quality management and development methods such as requirements / systems engineering and system analyses. In the automotive domain analysis methods for safety and security attributes at early development phases are well known and partially mandatory by domain standards. Nevertheless, approaches for analysis of serviceability attributes (the combination of reliability and maintainability) at early development phases are not yet available.**

Aim of the paper is to present a novel analysis method to quantify the impact of individual system parts on the overall system serviceability at early development phases. This approach bases on the concepts of state-of-the-art methods for safety and security analysis and extends their scope of application to serviceability feature quantification, thus enables consistent identification of system dependability target attributes. This, in turn, is a pre-requisite for ensuring a certain level of system dependability from start of development. In the second part of the document the application of the novel approach is demonstrated on an automotive training example of a battery management system.

Keywords*—HARA, automotive, system analysis, reliability quantification.

I. Introduction

Embedded systems are already integrated into our everyday life and play a central role in all domains including automotive, aerospace, healthcare, industry, energy, or consumer electronics. Moore's law [17], stating the doubling of the computer capacity every 2 years, is still a strong enabler for this fast function increase and at the same time cost-per function decrease. In 2015, the embedded systems market is expected to reach 1.5 trillion (assuming an annual growth rate of 12%) [18]. For the automotive industry, embedded systems components are responsible for 25% of vehicle costs, while the added value from electronics components range up to 75% for electrics and hybrid vehicles [22]. Hence, the automotive industry is facing growing complexity of functions and technologies, at the same time development cost and time should be decreased and dependability of the novel systems must be increased.

Therefore, the automotive industry is confronted to the central question how to optimize and quantify dependability features (such as safety, security, and reliability) of a system under development (SuD) already at early development phase. A know-how transfer is required to take over methods or extend the scope of application of state-of-the-art methods to evaluate system features early in the development life-cycle. Future automotive systems will require appropriate systematic approaches to further support front-loading of dependable system engineering. System dependability features have mutual impacts, similarities, and interdisciplinary values in common and a considerable overlap among existing methods. Besides this, standards, such as ISO 26262 [14] in safety and Common Criteria [15] in security domain, have been established to provide guidance during the development of dependable systems. System dependability attributes have a major impact on product development and product release as well as for company brand reputation. Therefore, dependability features of a system may either be 'show stoppers' or 'unique selling points'. For this document we define dependability according to [7] as an integrating concept that encompassing the attributes:

a safety: absence of catastrophic consequences on the users and environment.

b security: the concurrent existence of availability for authorized users only, confidentiality, and integrity with improper meaning of unauthorized.

c serviceability: the combination of reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs)

This paper presents a novel approach to quantify the impact of individual system parts to the overall system serviceability at early development phases. We employed the inductive analysis method HARA (hazard analysis and risk assessment) to also enable the quantification of dependability features (such as reliability and maintainability) of the SuD. The service deterioration analysis (SDA) approach gives further information about the deterioration resistance level (DRL) required for a certain system reliability/availability.

In the course of this document, a description of the state-of-the-art analysis techniques and related works is given in Section II. In Section III a description of the introduced analysis methods is provided. Section IV assesses the SDA

approach based on an automotive training example of a battery management system (BMS). Section V concludes this work.

II. Related Work

Dependability and security are superordinate concepts regrouping different system attributes such as reliability, safety, or availability. Therefore systems dependability and security are challenging research domains inheriting continuous development and growing importance. While innovative functionalities of embedded systems are the basic argument for development costs, dependability features can serve as unique selling points or distinctive features of the system under development (SuD). Dependable systems rely on mature quality management and development methods such as requirements / systems engineering, system analyses (e.g., FMEA), design and validation plans. For the automotive domain legacy (e.g., emission), liability (e.g., safety and security), and reputation (e.g., reliability and availability) aspects have been identified as key system attributes.

Safety standards, such as the road vehicles functional safety norm ISO 26262 [14] and its basic norm IEC 61508 [11], exist in the automotive domain, additional several safety and security norms and guidelines have been established in the avionic domain. In addition to DO-178C [24] for aerospace software safety, ARP4754 [20] gives guidance for system level development and defines steps for adequate refinement and implementation of requirements. Safety assessment techniques, such as failure mode and effects analysis (FMEA) and functional hazard assessment (FHA), among others, are specified by ARP4761 [19].

Reliability and availability standards mainly originate from railway and military industry. DIN EN 50126 [6] focuses on specification and demonstration of reliability, availability, maintainability, and safety (RAMS) of the railway system. In 1980 the US Department of Defense defined a standard reliability program for systems and equipment development and production (MIL-STD-785B [2]). Additionally, the military handbooks 338B [5] and 781A [4] assist with guidelines for electronic reliability design and reliability test methods, plans, and environment for engineering. Nevertheless, most standards and guidelines, like the military handbook 217F [3] and the technical report TR 62380 [9] rely on reliability prediction of electronic equipment based on mathematical reliability models of the system components. Only a few works focus on quantification of dependability features (other than safety or security) in early stages of the development process.

Most reliability measures and works focus on estimation of probabilities and stochastic processes. These works require detailed design information of the SuD and are therefore not applicable for an early design phase evaluation. Nevertheless, the process improvement techniques of Six Sigma [10], [25] aim in improving the quality of process outputs by identifying and removing the causes of defects (errors). Six Sigma approach uses a set of quality management methods, including statistical methods. One of the Six Sigma methods CTQ trees (critical-to-quality trees) are the key measurable characteristics of a product or process and are used to decompose broad customer requirements into more easily quantified elements. These elements are then converted to measurable terms, this approach is also the basis for Service Deterioration Analysis described in this document.

We also proposed an approach of a security-informed hazard analysis and describe an assessment process determining the impact of security attacks on safety features [16]. This work presented a combined approach of the automotive HARA (hazard analysis and risk assessment) with the security domain STRIDE. Therefore, the presented approach also employs the concept of the inductive analysis method HARA and extends their scope of application.

A threat analysis framework for critical infrastructures is proposed by Simion et. al [23]. This framework bases on the procedure of (1) identification and definition of threat attributes and (2) usage of attributes to characterize the threat potential.

Some recent publications in automotive domain also focus on security in automotive systems. On one hand, the work of Schmidt et. al [21] presents a security analysis approach to identify and prioritize security issues, but solely provides analysis approach for networked connectivity. The work of Ward et. al [26], on the other hand, also mentions a risk assessment method for security risk in the automotive domain called threat analysis and risk assessment, also based on the HARA.

To recap, there are approaches addressing safety and security in early development stages, but only few considering reliability. However, several of these methods base on the concept of the inductive analysis method HARA.

III. Proposed Approach

Each phase of the development process has a number of analysis methods for the different system features (e.g. safety or security) in place. But focusing solely on early system level development stages, a lack of analysis methods for serviceability, the combination of reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs), exists. Common analysis methods in the automotive domain are either inductive (bottom-up) or deductive (top-down) approaches. Fault tree analysis (FTA) [13] is a deductive failure analysis which analyses undesired state of a system using Boolean logic to combine a series of lower-level events. In contrary, the failure mode and effects analysis (FMEA) [12], [1] is an inductive analytical method which is performed at either the functional or piece-part level. FMECA extends FMEA by including a criticality analysis, which is used to chart the probability of failure modes against the severity of their consequences.

Nevertheless, for initial design assessment solely for safety analysis a method is standardized (HARA [14]). An adaptation of this early stage analysis method for security is called security-aware hazard analysis and risk assessment (SAHARA) presented by [16]. Hazard analysis and risk assessment (HARA) [14] is used to determine the safety goals for individual system components to avoid unreasonable risks. HARA therefore highlights the criticality of individual system components on system's safety at early design phases, even if detailed design of the system component itself is not known. SAHARA is the equivalent analysis for security measures.

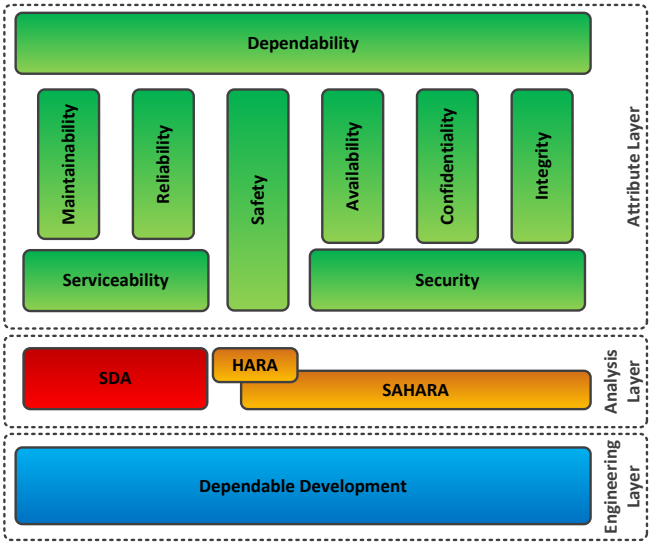


Fig. 1. General Overview of Dependability Attributes and Analysis Methods

Our novel service deterioration analysis (SDA) method bases on the concepts of HARA and SAHARA and quantifies the deterioration resistance level (DRL) of a system component required for a certain system serviceability. Intended for early design phases when detailed design of the system component itself is not known.

A. Terminology

The definitions of dependability attributes vary in different state-of-the-art standards and engineering domains. In this document, dependability is seen according to [8] and [7], as an integrating concept that encompasses different attributes.

Fig. 1 provides an overview of the attributes (aspects) of dependability, the analysis methods for each attribute, and a common dependable development block indicating the fact, that each aspect needs to be addressed within a consistent engineering framework. It can also be seen that security is a composition of the attributes of confidentiality, integrity, and availability. Security is the combination of confidentiality, the prevention of unauthorized information disclosure and amendment or deletion of information, and availability, the prevention of unauthorized withholding of information [7].

Furthermore, we use the collective term serviceability to describe the combination of reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs), two influencing factors of availability [8]. As indicated in Fig. 1 dedicated analysis methods for each dependability attributes at early development phase are in place, except for serviceability, which will be quantified by the proposed approach.

B. Problem Statement

Individual system components may have major or minor impact on systems dependability (e.g. safety, security, or reliability). In the concept phase it is of utmost importance to already identify these impacts and start system components development with appropriate focus. This helps to prevent from late detection of design errors and support front-loading of dependability engineering. While safety and security engineering have analysis methods in place, such an early development phase analysis method for system serviceability is not yet available. Estimating components reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs) at early phases is hard without detailed component design. This makes front-loading of dependability engineering steps hardly possible.

TABLE I. DETERIORATION IMPACT (I) CLASSIFICATION - CLASSIFICATION OF 'I' VALUE OF IMPACT OF OUTAGE OF THE COMPONENT

Level	Deterioration Impact	Example
0	no impact	no impairment of function
1	minor impact	reduced functionality, self-healing temporal impairment of functions, expendable part repair
2	major impact, reputation compromised	stop of service, callbacks required, non- expendable part repair

C. Service Deterioration Analysis (SDA) Solution Approach

A key concept of the HARA approach is defining automotive safety integrity level (ASILs) [14]. The assigned ASIL determines the criticality of the SuD and defines requirements and measures to be applied for the rest of the systems lifecycle. For the purpose of determining the SuDs ASIL, possible hazards have to be identified, which have the potential to put the system in a hazardous state. Afterwards, these hazards are quantified according their potential harm severity (S), probability of exposure (E), and the controllability of the resulting hazardous event (C). The final step formulates high level safety requirements known as safety goals. For more details on determining C, E, and S levels and the determination matrix of the resulting ASIL we recommend ISO 26262 part 3 Annex B [14].

In analogy to this, it is of high importance for systems serviceability to have a clue of the contribution of each individual system component. Therefore, we quantify the serviceability features according to the deterioration impact (I) on the system's dependability, the component's repair aggravation (A), and the operation profile (O).

The deterioration impact (I) relates to the components impact on the dependability of the system. Table I classifies the **deterioration impact (I)** and gives some examples of impacts of outage of the component. Either an outage of the component does not impair the system function at all (level 0), or it reduces the functionality in the ballpark (level1). Level 2 indicates maximum impacts which could lead to reputation losses.

The component's repair aggravation factor (A) is related to the components capability and easiness to undergo a repair. Table II mentions some examples for the different repair aggravation levels. The repair aggravation factor is determined aligned with the definition of inspection frequency and reliability life cycle degradation control of military handbook 338B [5]. Therefore, the determination of the A factor of a

TABLE II. REPAIR AGGRAVATION (A) CLASSIFICATION - CLASSIFICATION OF 'A' VALUE OF CAPABILITY AND EASINESS OF A REPAIR OF THE COMPONENT

Level	Repair Aggravation	Example
0	easy repair	can be performed by end-user
1	moderate repair	can be performed at any workshop (trained skills required)
2	difficult repair	need to be performed at the production center (specialized skills required)
3	serious repair	no repair possible (repair action not foreseen, product not useable anymore)

TABLE III. OPERATION PROFILE (O) CLASSIFICATION - CLASSIFICATION OF 'O' VALUE OF INTENDED HARSHNESS OF THE ENVIRONMENT OF THE COMPONENT

Level	Operation Profile	Example
0	normal / intended environment	daily usage, typical application
1	unplanned / harsh environment	usage at normal operation limits, corner-cases
2	misuse / out of limits	misappropriation, vandalism

specific component is simply done by adding of the components complexity ($high \rightarrow 1, low \rightarrow 0$), accessibility for maintenance ($hard \rightarrow 1, easy \rightarrow 0$), and diagnostic capability ($complex \rightarrow 1, manageable \rightarrow 0$) together.

The operation profile (O) values range from level 0, which means that the component is operated in its intended/normal environment. Level 1 is intended for usage in unplanned and harsh environment, while level 2 indicates misuses and operation out of operation limits. Table III outlines some examples of operation profiles.

These three values categorized the components deterioration resistance level (DRL). A higher repair aggravation or deterioration impact level result in a higher DRL level, while DRL level is raised the less the component is misused. Therefore the DRL can be calculated via the equation 1.

$$DRL = \begin{cases} 0 & \text{if } I + A - O < 2 \text{ or } I = 0 \\ 1 & \text{if } I + A - O = 2 \\ 2 & \text{if } I + A - O = 3 \\ 3 & \text{if } I + A - O = 4 \\ 4 & \text{if } I + A - O = 5 \end{cases} \quad (1)$$

It can also be determined via a look-up table, depicted in Table IV, which also base on (1). This quantification helps to adequately assign limits of resources spent to ensure a certain level of serviceability of a specific component of the SuD and puts emphasis on critical system component in terms of system-dependability. Furthermore, the quantification can be used to choose adequate design patterns for the component design (such as 2oo3 redundancy).

TABLE IV. DRL DETERMINATION MATRIX - DETERMINATION OF DRL VIA I, O, AND A VALUES

Operation Profile 'O'	Repair Aggravation 'A'	Deterioration Impact 'I'		
		0	1	2
0	0	0	0	1
	1	0	1	2
	2	0	2	3
	3	0	3	4
1	0	0	0	0
	1	0	0	1
	2	0	1	2
	3	0	2	3
2	0	0	0	0
	1	0	0	0
	2	0	0	1
	3	0	1	2

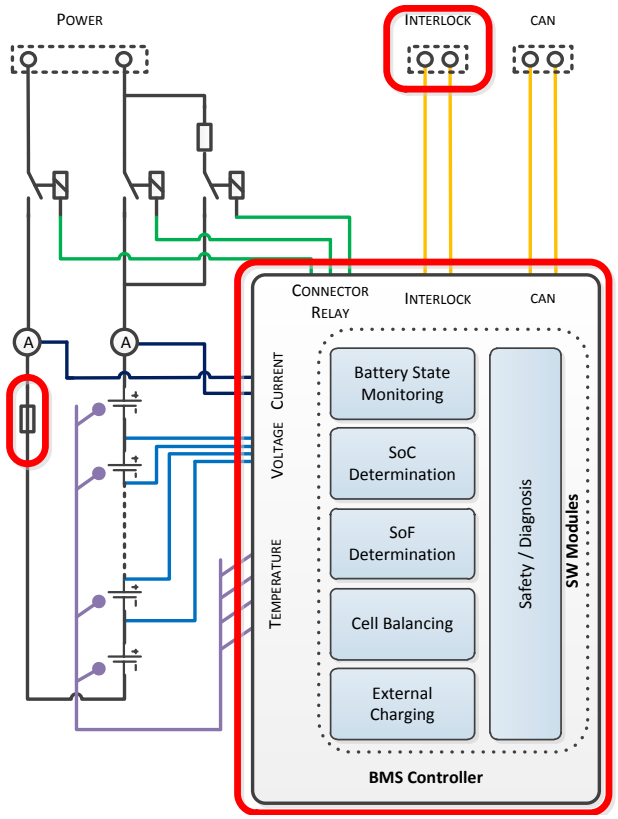


Fig. 2. General BMS Structure with Main HW Components and SW Modules

IV. Application of the Approach

This section describes the application of the previously mentioned approach for an automotive battery management system (BMS). The BMS use-case is an illustrative material, reduced for training purpose of both, students and engineers. Therefore, technology-specific details have been abstracted for commercial sensitivity and presented analysis results are not intended to be exhaustive.

Battery management systems are control systems inside of high-voltage battery systems used to power electric or hybrid vehicles. The BMS consists of several input sensors, sensing e.g., cell voltages, cell temperatures, output current, output voltage, and actuators (the battery main contactors). Fig. 2 depicts the general structure, main hardware components, and software modules of the high-voltage battery with BMS. Furthermore, the three focused components (interlock, BMS controller, and HV fuse) for this paper are highlighted in red boxes. The illustration shows the main features of a BMS:

- Power contactors - connection with vehicle HV system
- Interlock - de-energizing HV system when tripped
- CAN - automotive communication interface
- Relay - main contactor and output unit of the BMS
- Temperature sensors - feedback of actual cell temp
- Voltage sensors - feedback of actual cell voltages
- Current sensors - feedback of actual current flow
- Fuse - protective circuit breaker in case of fault
- Cells - electro-chemical energy storage
- BMS controller - monitoring and control unit

For the scope of this work the focus is set on early design decision evaluation of serviceability of SuD components based on the SDA approach.

Fig. 3 shows an excerpt of the SDA for normal operation modes of the BMS use- case, highlighting the HV fuse data (red box). As can be seen in this figure (column two and three), the components of the system (taken from initial system design) and the system service (high level service provided by the system, also from initial design phase), which might be affected by the component, are listed. First step is the determination of the deterioration impact of the component on the specific system service. Both depicted HV fuse lines (marked with red boxes) indicate the crucial affect of this component on the system service 'store electric energy', due to its connective characteristic within the electrical circuit. This step is followed by the analysis of the repair aggravation of the component. The components complexity, accessibility and diagnostic capability determine the resulting repair aggravation value 'A'. As can be seen in Fig. 3, using for example corrugated-head screws for the fuse cover to prevent it from being opened by average users increases the repair aggravation value of the HV fuse, which also results in a higher DRL (center line marked with red boxes). The prevention from opening the HV fuse cover may be appropriate due to safety considerations, but the fact of thereby increasing the repair aggravation of the HV fuse can be easily overlooked. Finally the determination of operation profile concludes the determination of the DRL.

Furthermore, Fig. 3 also highlights the (a priory anticipated) highest demands of dependability raised for the central control unit 'BMS controller' (highlighted with dashed violet box). This anticipated outcome additionally highlights the crucial affect the 'BMS controller' has for the overall system and therefore on the reputation of system's vendor.

In addition to these depicted examples, the SDA also put emphasis on less highlighted component's constraints at early design phases (such as component location, accessibility, or diagnostic features) and therefore emphasis in early design phases the importance of such considerations. As an example, the main relay connection is crucial (DRL 4), but its DRL value can be reduced if diagnostic features for the component are supported.

V. Conclusion

In conclusion, safety or security analysis of a system under development in early design phases are state-of-the-art in modern automotive development. Dependability engineering aims at providing a convincing and explicit argumentation that the system under development has achieved an appropriate level of maturity. Although, several approaches exist for addressing safety and security in early development stages, other dependability attributes (such as reliability and maintainability) are currently addressed by analysis methods at later stages.

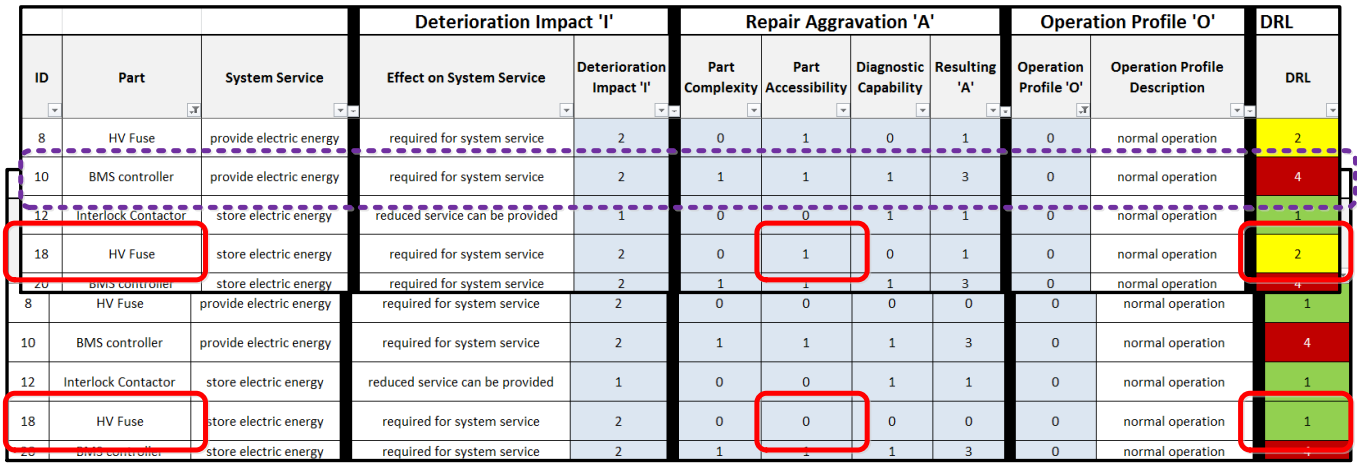
This paper presents a concept for an early development analysis method of system serviceability, called service deterioration analysis (SDA). Serviceability is defined as the combination of reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs). The newly proposed analysis concept implies a quantification scheme to quantify the impact of individual system parts to the overall system dependability. This approach bases on the concepts of state-of-the-art methods for safety and security analysis and enables consistent identification of system dependability target attributes (reliability and maintainability). The presented approach is geared towards high demands of future automotive systems for appropriate systematic front-loading approaches of dependable system engineering.

Moreover, the feasibility and added value of the SDA approach has been demonstrated for a battery management system use-case. While the use-case application does not claim completeness of the analysis (due to confidentiality issues), the benefits of the approach are already evident.

Acknowledgments

This work is partially supported by the INCOBAT and the MEMCONS projects.

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement n 608988 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),



			Deterioration Impact 'I'		Repair Aggravation 'A'				Operation Profile 'O'		DRL
ID	Part	System Service	Effect on System Service	Deterioration Impact 'I'	Part Complexity	Part Accessibility	Diagnostic Capability	Resulting 'A'	Operation Profile 'O'	Operation Profile Description	DRL
8	HV Fuse	provide electric energy	required for system service	2	0	1	0	1	0	normal operation	2
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	1	0	1	0	normal operation	2
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4
8	HV Fuse	provide electric energy	required for system service	2	0	0	0	0	0	normal operation	1
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	0	0	0	0	normal operation	1
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4

Fig. 3. Excerpt of the SDA Application of the BMS Use-Case

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] Military Standard Procedures for Performing a Failure Mode, Effects and Criticality Analysis, November 1980.

[2] Military Standard Reliabilty Program for Systems and Equipment Development and Production, September 1980.

[3] Military Handbook Reliability Prediction of Electronic Equipment, December 1991.

[4] Department of Defense Handbook for Reliability Test Methods, Plans, and Environments for Engineering, Development Qualification, and Production, April 1996.

[5] Military Handbook Electronic Reliability Design Handbook, October 1998.

[6] Railway applications - The specification and demonstration of reliability, availability, maintainability and safety (RAMS), March 2000.

[7] A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and its Threats - A Taxonomy. In R. Jacquart, editor, *IFIP Congress Topical Sessions*, pages 91–120. Kluwer, 2004.

[8] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[9] International Electrotechnical Commission. Reliability data handbook - Universal model for reliability prediction of electronics components, PCBs and equipment. Technical Report IEC TR 62380, International Electrotechnical Commission, 2004.

[10] International Organization for Standardization. ISO 13053 Quantitative methods in process improvment – Six Sigma, 2011.

[11] ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.

[12] ISO - International Organization for Standardization. IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA) , 2006.

[13] ISO - International Organization for Standardization. IEC 61025 Fault tree analysis (FTA) , December 2006.

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

[15] ISO - International Organization for Standardization. ISO/IEC 15408. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security (2nd Ed.)*. Springer, 2011.

[16] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner. SAHARA: A Security-Aware Hazard and Risk Analysis Method. In *DATE'15: Proceedings of the Conference on Design, Automation & Test in Europe*, 2015.

[17] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114 – 117, April 1965.

[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] SAE International. Guidelines and Mehtods for Conductiong the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

[20] SAE International. Guidelines for Development of Civil Aircraft and Systems, 2010.

[21] K. Schmidt, P. Troeger, H. Kroll, and T. Buenger. Adapted Development Process for Security in Networked Automotive Systems. *SAE 2014 World Congress & Exhibition Proceedings*, (SAE 2014-01-0334):516 – 526, 2014.

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

[23] C. P. Simion, O. M. C. Bucovtchi, and C. A. Popescu. Critical Infrastructures Protection Through Threat Analysis Framework. *Annals of the ORADEA UNIVERSITY*, 1:351–354, May 2013.

[24] Special Committee 205 of RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.

[25] G. Tennant. *Six sigma: SPC and TQM in manufacturing and services*. Gower Publishing Ltd, 2001.

[26] D. Ward, I. Ibara, and A. Ruddle. Threat Analysis and Risk Assessment in Automotive Cyber Security. *SAE 2013 World Congress & Exhibition Proceedings*, pages 507–513, 2013.

SAHARA: A Security-Aware Hazard and Risk Analysis Method

Georg Macher*‖, Harald Sporer*, Reinhard Berlach*, Eric Armengaud‖ and Christian Kreiner*

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

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

***Abstract*—Safety and Security are two seemingly contradictory system features, which have challenged researchers for decades. Traditionally, these two features have been treated separately, but due to the increasing knowledge about their mutual impacts, similarities, and interdisciplinary values,they have become more important. Because systems (such as Car2x in the automotive industry) are increasingly interlaced, it is no longer acceptable to assume that safety systems are immune to security risks. Future automotive systems will require appropriate systematic approaches that will support security-aware safety development. Therefore, this paper presents a combined approach of the automotive HARA (hazard analysis and risk assessment) approach with the security domain STRIDE approach, and outlines the impacts of security issues on safety concepts at system level. We present an approach to classify the probability of security threats, which can be used to determine the appropriate number of countermeasures that need to be considered. Furthermore, we analyze the impact of these security threats on the safety analysis of automotive systems. This paper additionally describes how such a method has been developed based on the HARA approach, and how the safety-critical contributions of successful security attacks can be quantified and processed.**

***Keywords*—ISO 26262, HARA, STRIDE, automotive, safety, security.**

I. Introduction

The complexity of embedded systems in the automotive industry has grown significantly in recent years. Current luxury cars implement more than 90 electronic control units (ECU), which utilize nearly 1 Gigabyte of embedded software code [1]. By 2018, expert predict that 30% of the overall vehicle costs will be due to vehicle electronics [2]. This prediction is also strongly supported by the ongoing trend of replacing traditional mechanical systems with modern embedded systems. This enables the deployment of more advanced control strategies, thus providing additional benefits for both the customer and environment, such as reduced fuel consumption and better drivability. At the same time, the higher degree of integration and safety- and security-critical nature of the control application presents new challenges. Traditionally, safety and security features have been managed separately, because safety-critical systems have been assumed to be immune from security risks. Nevertheless, because automotive systems (such as Car2x) are increasingly interlaced this assumption is no longer valid. Future automotive systems will require appropriate systematic approaches to support security-aware safety development. Safety standards such as ISO 26262 [3] for road vehicles have been established to provide guidance during the development of safety-critical systems. Although safety might be interpreted as contradictory to security, a considerable overlap among safety and security methods exists. The contribution of this paper is to present a framework for the security-aware identification of safety hazards. SAHARA (Security-Aware Hazard Analysis and Risk Assessment) is an expansion of the inductive analysis method hazard analysis and risk assessment (HARA), and encompasses threats of the STRIDE Threat Model [5]. This approach enables the quantification of the probability of the occurrence and impacts of security issues on safety concepts (safety goals).

The document is organized as follows: In Section II, a description of the proposed SAHARA approach is provided. Section III includes an assessment of the contribution of our approach in relation to those of other related works dealing with (automotive) safety and security related topics. An application for the approach in an automotive battery management system (BMS) use-case scenario is presented in Section IV. Concluding remarks are found in Section V.

II. Proposed Approach

Safety and security were previously managed independently from one another due to the different application areas and technical solutions. However, due to increasing impact of the Internet of Things (IoT) on aspects of the automotive domain, it is no longer acceptable to assume that safety systems are immune to security risks. In the foreseeable future, automotive engineers will require appropriate, systematic approaches and interdisciplinary knowledge of both safety and security to support the development of security-aware safety methods. Currently, the automotive domain mainly focuses on the safety-critical nature of automotive embedded systems, and therefore, has several advanced methods and processes in place (e.g., hazard analysis and risk assessment (HARA), fault tree analysis (FTA), and failure mode and effects analysis (FMEA)). In addition, an industry-wide standard for assessing the functional safety of road vehicles, covering the whole product lifecycle (ISO26262 [3]) has been established. To the contrary, several approaches to exposing security design flaws exist in other industrial sectors, but have not yet been applied within the automotive industry. STRIDE, an acronym

for six security threat categories, is a threat modeling approach [5] that uses a technique called threat modeling to review system designs in a methodical way. This allows the identification of the type of threat to which the system is vulnerable. The first category includes *spoofing threats* attempt to successfully masquerade as another person or program in order to gain illegitimate advantages. The second includes *tampering attacks* that maliciously modify data or data orders. *Repudiation threats*, which fall into the third category, target systems that are unable to trace prohibited operations and counteract illegal operations. the fourth category comprises *information disclosure threats* that involve the exposure of sensitive information. *Denial of service attacks* (D.o.S), which simply deny valid services, and threats such as babbling idiot faults belong in the fifth category. Finally, *elevation of privilege threats* aim to access, compromise or destroy data that should be available only to privileged user or programs.

Each of these threat classes potentially has a safety impact (leads to new hazards) when applied to safety-critical applications. The threat model does not make a given design 100% secure, but enables the system designer to learn from mistakes and avoid repeating them. Therefore, this approach can be seen as a method by which the security of a system can be assessed equivalent to HARA attempt for safety.

		Threat Level 'T'			
Required Resources 'R'	**Required Know-How 'K'**	0	1	2	3
0	0	0	3	4	4
	1	0	2	3	4
	2	0	1	2	3
1	0	0	2	3	4
	1	0	1	2	3
	2	0	0	1	2
2	0	0	1	2	3
	1	0	0	1	2
	2	0	0	0	1
3	0	0	0	1	2
	1	0	0	0	1
	2	0	0	0	1

Fig. 1. SecL Determination Matrix - ascertains the security level from R, K, and T values

A. Problem Statement

The focus of this approach is placed on the early development phase - the so-called concept phase - of safety-critical embedded automotive systems, which is also addressed by ISO 26262 part 3 [3]. Safety-related or safety-critical automotive systems can potentially contribute to, or cause hazards for humans or the environment. During the concept phase, it is of utmost importance to identify ways in which the system might dangerously fail and to begin formulating safety constraints. For this reason, the automotive safety standard states that the system under development (SuD) must be analyzed using the HARA approach. The intention of such a HARA is to identify and categorize hazards and formulate high level safety requirements (safety goals) in order to avoid unreasonable risks. This approach focuses on the sources of problems that could occure due to malfunction or foreseeable user misuse. However, problems caused by malicious attacks (security issues) are not addressed by HARA within the ISO 26262 standard, although such attacks may also preempt safety strategies. In parallel, the STRIDE threat model provides a way to methodically review system designs and highlight security design flaws, but does not support the categorization of security hazards or methodically formulate security requirements in such a way as to avoid identified risks.

TABLE I. REQUIRED RESOURCE 'R' CLASSIFICATION - DETERMINATION OF THE 'R' VALUE FOR REQUIRED RESOURCES TO EXERT A THREAT

Level	**Required Resource**	**Example**
0	no additional tool or everyday commodity	randomly using the user interface, strip fuse, key, coin,
1	standard tool	screwdriver, multi-meter, multi-tool
2	simple tool	corrugated-head screwdriver, CAN sniffer, oscilloscope
3	advanced tools	debugger, flashing tools, bus communication simulators

B. Approach

A key outcome of the HARA approach is the definition of the automotive safety integrity levels (ASILs). The assigned ASIL determines the criticality level of the SuD and defines requirements and measures that must be applied for the remainder of the development lifecycle. For the purpose of determining the SuDs ASIL, potential hazards must be identified that endanger the system. Afterwards, these hazards are quantified according the severity of potential harm (S), probability of exposure (E), and controllability of the resulting hazardous event (C). The final step involves a formulation of high level safety requirements known as safety goals (for more detailed information see [3] part 3 Annex B).

Threat modeling using STRIDE [5] can be seen as a security pendant to HARA. The key goal of this threat modeling approach is to analyze each system component for its susceptibility to threats and, subsequently, mitigate all threats to these components in order to increase system security.

The first step of the SAHARA approach, combining security and safety analyses, is to quantify the STRIDE security threads of the SuD in an analog manner as is performed for safety hazards as part of the HARA approach. Threats are quantified with reference to the ASIL analysis, according to the resources (R) and know-how (K) that are required to pose the threat and the threats criticality (T).

Table I classifies the **required resources - 'R'** to threaten the SuDs security and gives some examples of tools that are required to successfully pose the security threat. Level 0 covers threats that do not require any tools or everyday commodities, and wich are available even in situations of unpreparedness.

TABLE II. REQUIRED KNOW-HOW 'K' CLASSIFICATION - DETERMINATION OF THE 'K' VALUE FOR REQUIRED KNOW-HOW TO POSE A THREAT

Level	Required Know-How	Example
0	no prior knowledge (black-box approach)	average driver, unknown internals
1	technical knowledge (gray-box approach)	electrician, mechanic, basic understanding of internals
2	domain knowledge (white-box approach)	person with technical training and focused interests, internals disclosed

TABLE III. THREAT CRITICALITY 'T' CLASSIFICATION - DETERMINATION OF THE 'T' VALUE OF THREAT CRITICALITY

Level	Threat Criticality	Example
0	no security impact	no security relevant impact
1	moderate security relevance	annoying manipulation, partial reduced availability of service
2	high security relevance	damage of goods, invoice manipulation, non-availability of service, privacy intrusion
3	high security and possible safety relevance	maximum security impact and life-threatening abuse possible

Level 1 tools can be found in any average household, while the availability of level 2 tools is more limited (such as special workshops). Tools assigned to level 3 are advanced tools to which accessibility is highly limited and not wide-spread.

Table II classifies the **required know-how - 'K'**. Here, level 0 requires no prior knowledge (the equivalent of the black-box approach). Level 1 addresses people with technical skills and a basic understanding of internals, and represents gray-box approaches. Finally, level 2 represents white-box approaches; it addresses people with focused interests and comprehensive domain knowledge.

An overview of the **criticality of a security threat - 'T'** is given in Table III. Level 0, in this case, indicates security impact that is irrelevant, such as when raw data can be visualized, but its meaning not determined. The threat impact of level 1 threats is limited to annoyances, such as reduction in availability of services, but does not imply any damage to products or manipulation of data or services; such threats and financial threats are found in level 2. Level 3 threats imply the invasion of privacy or result in impacts on human life (quality of life), as well as cover possible impacts on safety features.

These three factors determine the resulting security level (SecL). The SecL determination matrix is based on the ASIL determination approach and is illustrated in Figure 1. The quantification of required know-how and tools, instead of any estimation of likelihood (e.g., of success or failure rate of attacks) was chosen due to the fact that such a classification of these factors is more commonly performed in the automotive industry and because it will remain the same over the whole life-time of the SuD. In addition, the quantification of these two factors is related to the estimation of the likelihood that an attack will be carried out. The quantification of the impact of the threats, on the one hand, determines whether the threat is safety-related (threat level 3) or not (all other levels). This information is the trigger to hand over the threat for further analysis from the safety point of view. On the other hand, this quantification allows the determination of the limits of resources spent to protect the SuD from a specific threat (risk management in the case of security threats). Following this quantification, these threats may be then appropriately mitigated or prevented by appropriate design changes and the implementation of countermeasures.

In the case of safety-related security threats, such threats will be analyzed and the resulting hazards evaluated according to their criticality, exposure, and severity. This security analysis helps to improve the completeness of the HARA situation analysis by implying factors of reasonably foreseeable misuse (security treats) in a more structured way.

III. RELATED WORK

Safety and security of control systems are challenging research areas that are characterized by continuous development and steadily increasing importance. For this reason, many researchers and industrial experts have recently made efforts to combine security and safety.

Although only safety standards, such as the road vehicles functional safety norm ISO 26262 [3] and its basic norm IEC 61508, exist in the automotive industry, several safety and security norms and guidelines have been established in aeronautics industry. In addition to DO-178C [10], which addresses aeronautics software safety, ARP4754 [7] provides guidance for system level development and defines steps for the adequate refinement and implementation of requirements. Safety assessment techniques, such as failure mode and effects analysis (FMEA) and functional hazard assessment (FHA) anmong others, are specified by ARP4761 [6]. Security concerns in aeronautics industry are tackled e.g., by the Common Criteria [12] [4] approach and ED202 specification.

Some recent publications in the automotive domain also focus on security in automotive systems. The work of Schmidt et. al [8] presents a security analysis approach to identify and prioritize security issues, but only provides an analytical approach for networked connectivity. Alternatively, the work of Ward et. al [13] additionally mentions a method to assess security risks in the automotive domain, which is called threat analysis and risk assessment, and is based on the HARA. In contrast to these approaches, our approach combines hazard and threat analysis in order to identify threats that can contribute to the safety-concept or lead to violations of safety goals. The works of Schmittner et. al [9] and Steiner et. al [11] also deal with safety and security analysis, but focus on state/event fault trees or a failure mode and failure effect model to perform safety and security cause-effect analysis.

IV. APPLICATION OF THE APPROACH

This section describes the application of the SAHARA approach to an automotive battery management system (BMS).

Battery management systems are control systems inside of high-voltage battery systems used to power electric or hybrid vehicles.

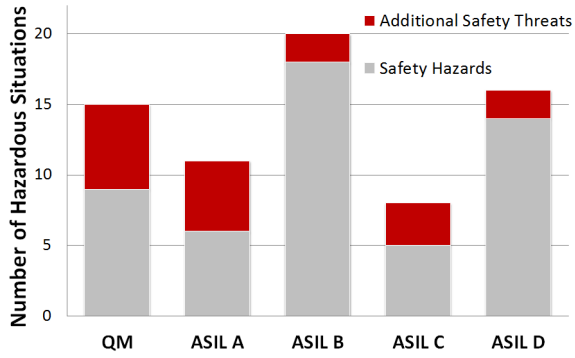


Fig. 2. Analysis of the SAHARA Approach for the BMS Use-Case - representation of safety hazards (identified using the common HARA approach) and additional hazards resulting from security threats (newly identified using the SAHARA approach)

The BMS is a safety-related system that is intended for installation in series production passenger cars and therefore within the scope of ISO 26262. Within the scope of this work, the focus was set on hazard analysis and risk assessment (HARA) in order to elaborate on a functional safety concept that had been developed using the SAHARA approach.

The SAHARA of the BMS use-case covered 52 hazardous situations, quantified the respective ASIL and assigned safety goals that were fully in line with the ISO 26262 standard. Additionally, 37 security threats were identified using the STRIDE approach and were quantified with their respective SecL. 18 of those security threats were classified as having possible impacts on safety concepts and were, therefore, further analyzed for their impacts on the safety of the SuD. Figure 2 presents the number of hazardous situations which were analyzed and quantified with ASILs, and highlights the additional safety hazards that were derived from the security threats. For this specific example, the SAHARA approach identified 34% more hazardous situations than the traditional HARA approach. SAHARA thus represents a systematic approach that can be taken to combine safety and security development, and takes into consideration the harmonization of safety and security methods.

V. Conclusion

Automotive embedded systems are safety-critical and, therefore, require appropriate risk identification and management throughout the development cycle. Moreover, these computing platforms are increasingly connected to their environment (e.g., on-board diagnosis, GPS, Car-2-Car or Car-2-infrastructure). In this context, safety-related functionalities rely on the trustworthiness of information gathered from the environment. Because security threats might have an impact on the safety of the system, automotive embedded systems cannot be considered immune to security attacks. Therefore, joint considerations of security and safety are necessary. This paper presents a combined approach, merging of the automotive HARA (hazard analysis and risk assessment) with the security domain STRIDE, proposing the security-aware hazard analysis and risk assessment (SAHARA) approach. The SAHARA approach is fully in line with the requirements of a HARA analysis according to the automotive safety standard, ISO 26262 [3], for road vehicles. SAHARA represents a systematic approach that addresses the need to synchronize the development of safety and security methods.

Acknowledgments

This work is partially supported by the INCOBAT and the MEMCONS projects.

The research leading up to these results was funded by the European Unions Seventh Framework Programme (FP7/2007-2013) under the grant agreement n 608988 and through financial support from 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) through the "COMET K2 - Competence Centers for Excellent Technologies Programme".

The authors thank Sara Crockett for assistance with language editing and proofreading.

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

[2] R. Hilbrich, J. Reinier van Kampenhout, and H.-J. Goltz. Modellbasierte Generierung statischer Schedules fuer sicherheitskritische, eingebettete Systeme mit Multicore-Prozessoren und harten Echtzeitanforderungen. *Informatik aktuell*, pages 29 – 38, 2012.

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

[4] ISO - International Organization for Standardization. ISO/IEC 15408. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security (2nd Ed.)*. Springer, 2011.

[5] Microsoft Corporation. The stride threat model, 2005.

[6] SAE International. Guidelines and Mehtods for Conductiong the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

[7] SAE International. Guidelines for Development of Civil Aircraft and Systems, 2010.

[8] K. Schmidt, P. Troeger, H. Kroll, and T. Buenger. Adapted Development Process for Security in Networked Automotive Systems. *SAE 2014 World Congress & Exhibition Proceedings*, (SAE 2014-01-0334):516 – 526, 2014.

[9] C. Schmittner, T. Gruber, P. Puschner, and E. Schoitsch. Security Application of Failure Mode and Effect Analysis (FMEA). In A. Bondavalli and F. Di Giandomenico, editors, *Computer Safety, Reliability, and Security*, volume 8666 of *Lecture Notes in Computer Science*, pages 310–325. Springer International Publishing, 2014.

[10] Special Committee 205 of RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.

[11] M. Steiner and P. Liggesmeyer. Combination of Safety and Security Analysis - Finding Security Problems That Threaten The Safety of a System. In *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[12] The Common Criteria Recognition Agreement Members. Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/, 2014.

[13] D. Ward, I. Ibara, and A. Ruddle. Threat Analysis and Risk Assessment in Automotive Cyber Security. *SAE 2013 World Congress & Exhibition Proceedings*, pages 507–513, 2013.

A Comprehensive Safety, Security, and Serviceability Assessment Method

Georg Macher[1,2], Andrea Höller[1], Harald Sporer[1], Eric Armengaud[2], and Christian Kreiner[1]

[1] Institute for Technical Informatics
Graz University of Technology
{georg.macher, andrea.hoeller, sporer, christian.kreiner}@tugraz.at
[2] AVL List GmbH
{georg.macher, eric.armengaud}@avl.com

Abstract. Dependability is a superordinate concept regrouping different system attributes such as reliability, safety, security, or availability and non-functional requirements for modern embedded systems. These different attributes, however, might lead to different targets. Furthermore, the non-unified methods to manage these different attributes might lead to inconsistencies, which are identified in late development phases. The aim of the paper is to present a combined approach for system dependability analysis to be applied in early development phases. This approach regroups state-of-the-art methods for safety, security, and reliability analysis, thus enabling consistent dependability targets identification across the three attributes. This, in turn, is a pre-requisite for consistent dependability engineering along the development lifecycle. In the second part of the document the experiences of this combined dependability system analysis method are discussed based on an automotive application.

1 Introduction

Embedded systems are already integrated into our everyday lives and play a central role in all domains including automotive, aerospace, healthcare, industry, energy, or consumer electronics. In 2010, the embedded systems market accounted for a turnover of almost 852 billion dollars and is expected to reach 1.5 trillion by 2015 (assuming an annual growth rate of 12%) [21]. For the automotive industry embedded systems components are responsible for 25% of vehicle costs, while the added value from electronics components range between 40% for traditional vehicle up to 75% for electric and hybrid vehicles [27].

The trend of replacing traditional mechanical systems with modern embedded systems enables the deployment of more advanced control strategies, such as reduced fuel consumption and better driveability, but at the same time the higher degree of integration and criticality of the control application bring new challenges. Future automotive systems will require appropriate systematic approaches to support dependable system engineering, rather than traditional engineering approaches managing safety, security, or reliability features separated.

System dependability features have mutual impacts, similarities, and interdisciplinary values in common and a considerable overlap among existing methods. Further to this, standards, such as ISO 26262 [16] in safety and Common Criteria [17] in the security domain, have been established to provide guidance during the development of dependable systems and are currently being reviewed for similarities and alignment. System dependability attributes have a major impact on product development and product release as well as for company brand reputation. Hence, incomplete dependability argumentation might be a show-stopper for the development of the entire system. On the other hand, smart countermeasures might represent unique selling points by reducing development costs and improving the quality of the system developed. In this document we define dependability according to [7] as an integrating concept that encompasses the following attributes:

a safety: absence of catastrophic consequences on the users and environment.
b security: the concurrent existence of availability for authorized users only, confidentiality, and integrity with improper meaning of unauthorized.
c serviceability: the combination of reliability (continuity of correct service) and maintainability (ability to undergo modifications and repairs).

This paper presents a combined approach for analysis of dependability features in early design phases of an embedded automotive system. We employed an approach which classifies the probability and impact of security threats using the STRIDE approach [20] and safety hazards using hazard analysis and risk assessment (HARA). This approach, described as the SAHARA approach in [19], quantifies the security impact on dependable safety-related system development at system level. We further extended the inductive analysis methods HARA and SAHARA to also enable the quantification of additional dependability features (such as reliability and availability). This service deterioration analysis (SDA), described in [18] gives further information about the deterioration resistance level (DRL) required for a specific system reliability/availability.

In the course of this document, a description of the state of the art analysis techniques and related works is given in Section 2. In Section 3 a description of the combination of methods and information handover for system dependability feature analysis is provided. Section 4 assesses an experience report of a system dependability analysis method for safety, security, and reliability attributes during the early development phases of an automotive battery management system (BMS) use-case. Section 5 concludes this work.

2 Related Work and Background

Dependability and security are superordinate concepts regrouping different system attributes such as reliability, safety, confidentiality, integrity, or availability. Systems dependability and security are thus challenging research domains inheriting a continuous development process and currently of growing importance. Dependable systems rely on mature quality management and development methods such as requirements / systems engineering, system analyses (e.g., FMEA), design and validation plans. For the automotive domain legacy (e.g., emission),

liability (e.g., safety and security), and reputation (e.g., reliability and availability) aspects have been identified as key aspects for sustainability in the market.

2.1 State-of-the-Art

Although safety standards only exist in the automotive domain (road vehicles functional safety norm ISO 26262 [16] and its basic norm IEC 61508 [13]), several safety and security norms and guidelines have been established in the avionic domain. In addition to DO-178C [28] for aerospace software safety, ARP4754 [24] gives guidance for system level development and defines steps for adequate refinement and implementation of requirements. Safety assessment techniques, such as failure mode and effects analysis (FMEA), and functional hazard assessment (FHA) among others, are specified by ARP4761 [23]. Security concerns in avionic domain are tackled e.g. by common criteria [31] approach and ED202 [9] specification.

Reliability and availability standards mainly originate from railways and the armaments industry. DIN EN 50126 [6] focuses on specification and demonstration of reliability, availability, maintainability, and safety (RAMS) of the railway system. In 1980 the US Department of Defense defined a standard reliability program for systems and equipment development and production (MIL-STD-785B [2]). Additionally, the military handbooks 338B [5] and 781A [4] assist with guidelines for electronic reliability design and reliability test methods, plans and environment for engineering. Nevertheless, most standards and guidelines, like the military handbook 217F [3] and the technical report TR 62380 [11] rely on reliability prediction of electronic equipment based on mathematical reliability models of the system components. Only a few works focus on quantification of dependability features (other than safety or security) in early stages of the development process.

Most reliability measures and work focus on estimation of probabilities and stochastic processes. All of this work requires detailed design information of the SuD and is therefore not applicable for an early design phase evaluation. Nevertheless, the process improvement techniques of Six Sigma [12], [30] aims at improving the quality of process outputs by identifying and removing the causes of defects (errors). The Six Sigma approach uses a set of quality management methods, including statistical methods. One of the Six Sigma methods CTQ trees (critical-to-quality trees) are the key measurable characteristics of a product or process and are used to decompose broad customer requirements into more easily quantified elements. These elements are then converted to measurable terms, this approach is also the basis for Service Deterioration Analysis [18] described later in this section.

The work of Gashi et al. [10] focuses on redundancy, diversity, and their effects on safety and security of embedded systems. This work is part of SeSaMo (Security and Safety Modeling for Embedded Systems) project, which focuses on synergies and trade-offs between security and safety through concrete use-cases.

In contrast to this, the work of Macher et. al. [19] proposes an approach involving a security-informed hazard analysis and describes an assessment process determining the impact of security attacks on safety features.

Some recent publications in the automotive domain also focus on security in automotive systems. On the one hand, the work of Schmidt et. al [25] presents a security analysis approach to identify and prioritize security issues, but provides only an analysis approach for networked connectivity. The work of Ward et. al [32], on the other hand, also mentions a risk assessment method for security risk in the automotive domain termed threat analysis and risk assessment, based on the HARA.

The works of Roth et. al [22] and Steiner et. al [29] also deal with safety and security analysis, but focus on state/event fault trees for modeling of the system under development. Schmittner et. al [26] presents a failure mode and failure effect model for safety and security cause-effect analysis. This work categorizes threats by using the STRIDE threat model, but focusing on IEC60812 conform FMEA.

Finally, the STRIDE threat model approach [20] developed by the Microsoft Corporation can be used to expose security design flaws. This approach uses a technique termed threat modeling. With this approach the system design is reviewed in a methodical way, which makes it applicable for integration into the HARA approach. Threat models, like STRIDE approach, may often not prove that a given design is secure, but they help to learn from mistakes and avoid repeating them, which is another commonality with HARA in safety domain.

To recap, there are approaches addressing safety, security, and reliability in early development stages. However, these methods are often performed independently and cross-dependencies and mutual impacts are often not considered. Our contribution here is a step towards filling this gap by presenting an approach for addressing safety, security, and reliability issues in early phases and on an interdisciplinary basis.

2.2 Dependability Analysis Methods

Each phase of the development process has a number of analysis methods for the different system features (e.g. safety, security or reliability) in place. In this work we focus solely on the system level rather than software or hardware development stages. The most commonly used analysis methods are hazard analysis and risk assessment (HARA), failure mode and effects analysis (FMEA), and fault tree analysis (FTA) in different variations for safety, security, or reliability/availability. FMEA [14] [1] and FTA [15] variation are common for all system features. Nevertheless a method is standardized for initial design assessment exclusively for safety analysis (HARA [16]). The proposed contribution of this work is to combine a security-aware hazard analysis and risk assessment (SAHARA) [19] and service deterioration analysis (SDA) [18] in a common approach. In the rest of this section both approaches are summarized (for standard hazard analysis and risk assessment as in use in the automotive domain please refer to [16]).

Safety-Aware Hazard Analysis and Risk Assessment (SAHARA)
The objective of SAHARA [19] is, analog to HARA, the systematic identification of possible attacks having a possible impact on the correctness of the system

Table 1: Classification Examples of Knowledge 'K', Resources 'R', and Threat 'T' Value of Security Threats

Level	Knowledge Example	Resources Example	Threat Criticality Example
0	average driver, unknown internals	no tools required	no impact
1	basic understanding of internals	standard tools, screwdriver	annoying, partial reduced service
2	internals disclose, focused interests	non-standard tools, sniffer, oscilloscope	damage of goods, invoice manipulation, privacy
3		advanced tools, simulator, flasher	life-threatening possible

(protecting the system against the human). Security attacks are identified and categorized according to their threat criticality (T), know-how (K), and resources (R) required to violate security barriers. The threat criticality (T) relates to the possible effects on the system, the know-how (K) relates to the required know-how to perform the attack, and the resources (R) relates to the required resources to perform the attack. Table 1 contains examples of resources, know-how, and threat levels for each quantification level of K, R, and T values. The three factors define a security level (SecL), as shown in Table 2 which is used to determine the appropriate number of countermeasures needed to be considered. Furthermore, a threat criticality of the highest level implies an impact on safety, has a SecL of minimum 1, and is added to the safety analysis.

Table 2: SecL Determination Matrix - Determination of the security level from R, K, and T values [19]

Required Resources 'R'	Required Know-How 'K'	Threat Level 'T'			
		0	**1**	**2**	**3**
0	0	0	3	4	4
	1	0	2	3	4
	2	0	1	2	3
1	0	0	2	3	4
	1	0	1	2	3
	2	0	0	1	2
2	0	0	1	2	3
	1	0	0	1	2
	2	0	0	0	1
3	0	0	0	1	2
	1	0	0	0	1
	2	0	0	0	1

Service Deterioration Analysis (SDA)
In analogy to the two previously mentioned analysis methods, the service deterioration analysis (SDA) [18] systematically analyzes the impact of component malfunction on the system availability. The deterioration resistance level (DRL) is categorized by the deterioration impact (I) on the system's dependability, the component's repair aggravation (A), and the operation profile (O). The deterioration impact (I) relates to the availability impact on the system, the component's repair aggravation (A) to the capability and easiness to undergo a repair, and the operation profile (O) to the intended harshness of the environment (and therefore the probability for break-down). The repair aggravation factor is determined aligned with the definition of inspection frequency and reliability life cycle degradation control of military handbook 338B [5]. The A factor is therefore the sum of the system's complexity ($high \rightarrow 1, low \rightarrow 0$), accessibility for maintenance ($hard \rightarrow 1, easy \rightarrow 0$), and diagnostic capability ($complex \rightarrow 1, manageable \rightarrow 0$). Table 3 shows the DRL determination matrix used to establish a quantitative indicator determining the impact on system's dependability. A description of the classification of deterioration impact (I), repair aggravation (A), and operation profile (O) values can be seen in Table 4.

Table 3: DRL Determination Matrix - Determination of DRL via I, O, and A values [18]

Operation Profile 'O'	**Repair Aggravation 'A'**	**Deterioration Impact 'I'**		
		0	**1**	**2**
0	0	0	0	1
	1	0	1	2
	2	0	2	3
	3	0	3	4
1	0	0	0	0
	1	0	0	1
	2	0	1	2
	3	0	2	3
2	0	0	0	0
	1	0	0	0
	2	0	0	1
	3	0	1	2

3 Combined Approach for Dependable System Development

In the automotive industry, dependability engineering is currently moving its center of gravity from mainly mechanical reliability towards functional safety

Table 4: Classification Examples of Deterioration Impact ('I'), Repair Aggravation ('A'), and Operation Profile ('O') Value

Level	**'I' Example**	**'A' Example**	**'O' Example**
0	no impact	can be performed by end-user	normal / intended operation environment
1	minor impacts	can be performed at any workshop (trained skills required)	unplanned / harsh environment
2	major impacts / reputation compromised	need to be performed at the production center (specialized skills required)	misuse
3		no repair possible (repair action not foreseen, product not useable anymore)	

and security of the control system. While the target is still to provide a convincing argumentation that the system can justifiably be trusted [7], the hazards to consider as well as the development methods must be adapted accordingly. Hence, dependability is seen according to [8] and [7], as an integrating concept that encompasses more different attributes. Fig. 1 provides an overview of the attributes (aspects) of dependability, the analysis methods available for each attribute, and a common dependable development block indicating the fact that each aspect needs to be addressed within a consistent engineering framework. Indeed, a common analysis method delivering consistent dependability targets over the different attributes is the basis to perform consistent dependability engineering during the entire product development. It can also be seen that security is a composition of the attributes of confidentiality, integrity, and availability. Security is the combination of confidentiality, the prevention of unauthorized information disclosure and amendment or deletion of information, plus availability and also the prevention of unauthorized withholding of information [7].

The common engineering basis for all dependability aspects raises the requirement for a combination of the different analysis methods and targets, thus a mutual understanding of focuses and language concepts. Table 5 shows a mapping of safety, security, and service oriented engineering terms.

Combining the different dependability feature analysis methods and dependability targets is of high importance. The SAHARA approach [19] already implies an identification of security threats having a possible impact on safety and an information exchange between the security and safety domain. Nevertheless, the approach described in this paper relies on the combination of the outcomes (targets and classifications) of HARA, SAHARA, and SDA to raise the level of completeness of the analysis and consistency between mutual dependencies. Fig. 2 shows an overview of the described approach and highlights the distinctive features of the presented approach (broad red arrows).

The mutual impact of serviceability analysis considerations and safety considerations (see Fig. 2 - arrow I) exists between safe states and reliability targets of the system. A tradeoff between higher availability and higher safety of the

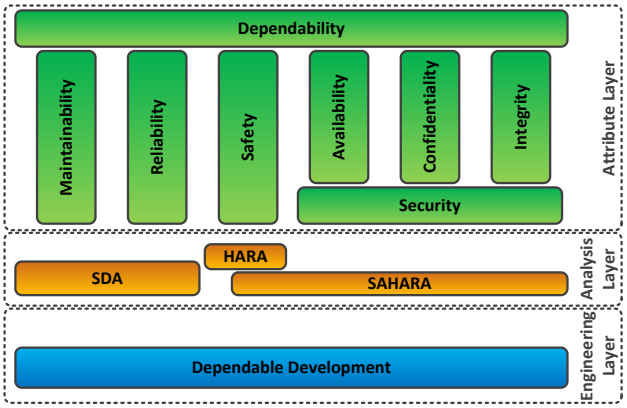


Fig. 1: General Overview of Dependability Attributes and Analysis Methods

Table 5: Mapping of Safety, Security, and Service Oriented Engineering Terms

		Safety Engineering	**Security Engineering**	**Service-Oriented Engineering**
Analysis Subjects	risk	hazard	threat	warranty claim, unplanned maintenance
	system inherent deficiency	malfunction	vulnerability	service loss or degradation
	external enabling condition	hazardous situation	attack	(mis)usage profile
Analysis Categories	impact analysis	severity	threat criticality	reputation loss, deterioration impact
	external risk control analysis	controllability	attacker skills, know-how	repair efforts, repair aggravation
	occurrence analysis	exposure	point of attack, attack resources	operation condition spectrum
Analysis Results	design goal	safety goal	security target	dependability target
	design goal criticality	ASIL	SecL	DRL

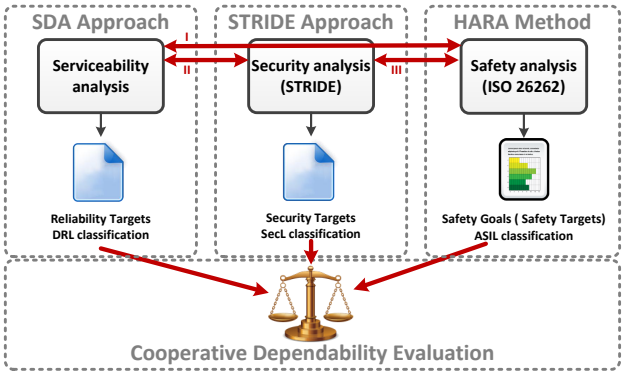


Fig. 2: Overview of the Described Approach with Highlighting of Distinctive Features

system impacts the design of safe states (e.g. a system shutdown in case of uncertainty of the actual system state can be a good option from a safety or cost point of view, but has negative effects on system availability). The targets and target classifications found in SDA, therefore have an impact on the design of safe states (e.g. fail-silent vs. fail operational). On the other hand, higher safety levels (ASIL) require higher reliability of the related components. This affects the design and quality requirements of components which might have not been in focus of SDA. Formulating these mutual dependencies between serviceability and safety leads to two regulations:

$DRL \geq 3 \rightarrow$ component related safe states need to be reviewed, eventual adaptation of degradation concept required
$ASIL > QM \rightarrow$ specific component reliability required, eventual adaptation of DRL of neglected component required

Preventing unauthorized access to control interfaces by security password affects the system security positively, beside this, it reduces the availability and controllability of the whole system (see Fig. 2 - arrow II). Requiring authentication of each system component positively affects the overall security, but simultaneously increases the maintenance effort and time requirement, which further impacts system availability. On the other hand easing the maintenance burden by reducing security authentication discloses attack vectors probably not considered during security analysis (e.g. the simple replacement of encryption chip). The regulation of the mutual dependencies between serviceability and security leads to the following:

$DRL \geq 3 \rightarrow$ component related security targets need to be reviewed
$SecL \geq 3 \rightarrow$ serviceability might be affected, review required to determine impact on operational readiness

The third mutual impact (see Fig. 2 - arrow III) exists between safety and security. Safety and security features frequently appear to be in total contradiction to the overall system features. An example of this contradiction can be

Security Risk description			Security Risk classification				Security Risk related Safety goal description					
Security Hazard ID	STRIDE Function	attacker generated malfunction	Required Resources 'R'	Required Know-How 'K'	Threat Level 'T'	Resulting SecL	Severity 'S'	Exposure 'E'	Controllability 'C'	Resulting ASIL	Safety Goal	Safe State
SH_4	Spoofing	extending safe operation areas of HV battery (temp, cell current, cell voltages)	1	1	3	3	3	4	2	ASIL C	Estimate correct cell status information	assume worst case, No torque provided, driver warning
SH_5	Denial of service	communication with charger jammed	2	2	3	1	3	2	3	ASIL B	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_6	Denial of service	immobilizing of driving functionality	1	0	2	3	0	0	0	QM	--- no safety impact --	--- no safety impact --
SH_7	Denial of service	emergency kill switch without function	1	2	3	2	3	2	3	ASIL B	Manual main switch off must be possible	Disconnect HV battery, driver warning
SH_8	Denial of service	replace fuse with non current limiting element	1	1	3	3	3	4	3	ASIL D	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_9	Spoofing	HV system ready without ensured overall system safety	2	2	3	1	3	2	3	ASIL B	Detect short circuit and isolation faults, Prevent from electric shock	Disconnect HV battery, driver warning
SH_10	Tampering	extending safe operation areas of HV battery (temp, cell current, cell voltages)	3	3	3	0	3	4	3	ASIL D	Battery outgasing and fire shall be prevented	Disconnect HV battery, driver warning
SH_11	Spoofing	extending safe operation areas of HV battery (temp, cell current, cell voltages)	1	2	3	2	3	4	2	ASIL C	Correct amount of power shall be maintained	No torque provided, driver warning
SH_12	Denial of service	bypass HVIL	1	2	3	2				ASIL A	Detect short circuit and isolation faults, Prevent from electric shock	Disconnect HV battery, driver warning

Fig. 3: Excerpt of the Application of the SAHARA Analysis of the BMS Use-Case

shown by the electrical steering column lock system. In the security context the system locks the steering column when in doubt, because this doubt area might result from an attack. From the safety perspective, however, it might not be the best approach to lock the steering column as a fallback, since the issue involved might well be an occurrence directly before a high speed corner turn. Mutual dependencies between safety and security can be prescribed as:

$ASIL > QM \rightarrow$ safe state need to be reviewed for possible deactivation of security
$T = 3 \rightarrow$ safety might be affected, a review is required to determine impact on operational readiness

These regulations of the mutual dependencies, the required information handover, and the mapping of the different engineering domain terms allow a cooperative dependability evaluation by cross-domain expert teams and provide traceable measures for early design decisions.

4 Application of the Approach

This section describes the application of the approach as described above for an automotive battery management system (BMS). The BMS use-case is an illustrative material, reduced for training purpose of both students and engineers. The technology-specific details have thus been abstracted for commercial sensitivity and the analysis results presented are not intended to be exhaustive.

Battery management systems are control systems inside of high-voltage battery systems used to power electric or hybrid vehicles. The BMS consists of several input sensors, sensing e.g., cell voltages, cell temperatures, output current, output voltage, and actuators (the battery main contactors).

For the scope of this work the focus is set on early design decision evaluation. This evaluation includes an ISO 26262 [16] aligned HARA safety analysis, a security analysis based on the SAHARA approach, and a serviceability analysis based on the SDA approach. An excerpt of the SAHARA analysis of the BMS

use-case is shown in Fig. 3. The excerpt highlights (a) the threat level classification 'T' triggering further analysis of the threat for safety impact and (b) a security hazard aiming on denial of service of the HV fuse. The HARA of the BMS use-case covers 52 hazardous situations, quantifies the respective ASIL and assigns safety goals fully in line with the ISO 26262 standard. Additionally, 37 security threats have been identified using the SAHARA approach, 18 of these security threats have been classified as having possible impacts on safety concepts. Furthermore, 63 service deterioration scenarios have been analyzed using the SDA approach.

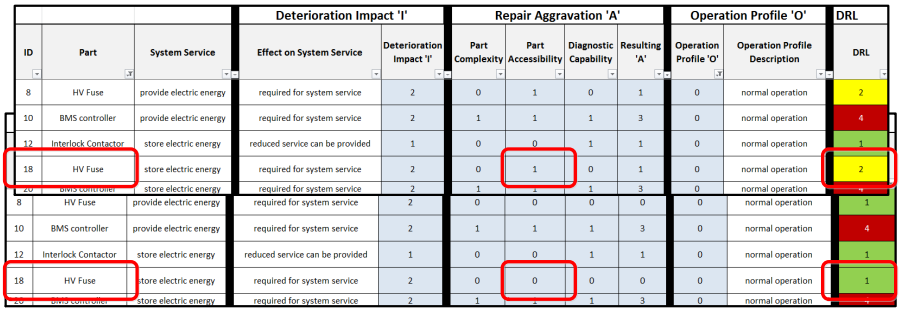
Fig. 4 shows an excerpt of the SDA for normal operation modes of the BMS use- case also highlighting the HV fuse data. The overlaid excerpt show the impact of the security countermeasure against the threat 'replace fuse with non current limiting element'. As can be seen in the overlay, using corrugated-head screws for the fuse cover decreases the security risk of 'replace fuse with non current limiting element', but increases the repair aggravation value of the HV fuse, which also results in a higher DRL.

Table 6 shows the analysis results related to the three focused elements for this application example (HV fuse, BMS controller, and interlock circuit). The HV fuse related part of Table 6 highlights how a security threat countermeasure which has not affected the ASIL nevertheless affects the DRL of the system, as already shown in Fig. 4. Part two of the table, related to the interlock circuit, indicates that a component of less importance from the safety or service perspective can have huge impact on the system's safety. The BMS controller related part of the table, highlights on one hand the opposite. The data encryption chip does not affect the system's safety, but is related to system's security and also the reputation of the system. On the other hand the table also indicates, that components (such as the BMS controller) which have major impact on the system's safety commonly also demand higher security and serviceability requirements.

Fig. 5 additionally presents the number of hazardous situations which have been analyzed and quantified with ASILs and highlights the additional portion of safety hazards derived from security threats.

5 Conclusion

Dependability engineering aims at providing a convincing and explicit argumentation that the system under development has achieved an appropriate level of maturity. Dependability engineering in the automotive domain, while relying on a strong and long-term body of experience, is still an emerging trend. Although, several approaches exist for addressing safety, security and reliability in early development stages, many of these methods are often performed independently and cross-dependencies and mutual impacts are often not considered. This paper presents a concept for a cooperative dependability analysis in early development phases and an application of this analysis for an automotive battery management use-case. The approach conjointly combines security, safety, and serviceability analysis concepts (SAHARA, HARA, and SDA) to enable a common analysis and language base enabling dependable system development. In the course of this study document, (a) a mapping of safety, security, and



			Deterioration Impact 'I'		Repair Aggravation 'A'				Operation Profile 'O'		DRL
ID	Part	System Service	Effect on System Service	Deterioration Impact 'I'	Part Complexity	Part Accessibility	Diagnostic Capability	Resulting 'A'	Operation Profile 'O'	Operation Profile Description	DRL
8	HV Fuse	provide electric energy	required for system service	2	0	1	0	1	0	normal operation	2
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	1	0	1	0	normal operation	2
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4
8	HV Fuse	provide electric energy	required for system service	2	0	0	0	0	0	normal operation	1
10	BMS controller	provide electric energy	required for system service	2	1	1	1	3	0	normal operation	4
12	Interlock Contactor	store electric energy	reduced service can be provided	1	0	0	1	1	0	normal operation	1
18	HV Fuse	store electric energy	required for system service	2	0	0	0	0	0	normal operation	1
20	BMS controller	store electric energy	required for system service	2	1	1	1	3	0	normal operation	4

Fig. 4: Excerpt of the SDA Application of the BMS Use-Case

service oriented engineering terms, (b) a combination of the different analysis methods and targets, and (c) regulations of the mutual dependencies and the required information handover between the different dependability aspect analysis have been presented.

The application of the approach presented has been demonstrated utilizing a simplified automotive BMS use-case. While the authors does not claim completeness of the analysis (due to confidentiality issues), the benefits of the approach are already evident. First, the dependencies between safety / security and reliability analysis are made explicit and can be handed over from one domain to the other according to identified rules. Second, and this is possibly the most important issue, the proposed cooperative dependability evaluation enables consolidation of the different dependability targets in a consistent and at an early design phase, thus providing a single source of dependability targets for the rest of the development.

Acknowledgments

This work is partially supported by the INCOBAT and the MEMCONS projects.

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement n 608988 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).

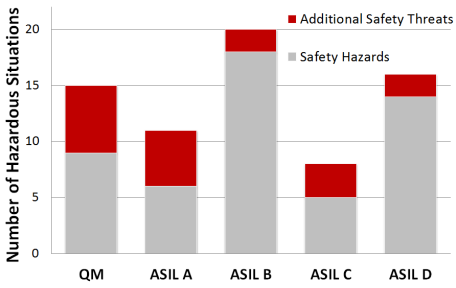
We are grateful for the contribution of the SOQRATES Safety AK experts and the expertise gained in SafEUr professional trainings.

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.

Table 6: Component Related Analysis Results of BMS Examples

Component	ASIL	SecL	DRL
HV Fuse	B(D)	3 2	1 2
Interlock Circuit	D	2	2
BMS Controller	D	3	4
Data Encryption Chip	QM	4	4

Fig. 5: Safety Analysis of the BMS Use-Case (using SAHARA approach) [19]



References

1. Military Standard Procedures for Performing a Failure Mode, Effects and Criticality Analysis, November 1980.
2. Military Standard Reliabilty Program for Systems and Equipment Development and Production, September 1980.
3. Military Handbook Reliability Prediction of Electronic Equipment, December 1991.
4. Department of Defense Handbook for Reliability Test Methods, Plans, and Environments for Engineering, Development Qualification, and Production, April 1996.
5. Military Handbook Electronic Reliability Design Handbook, October 1998.
6. Railway applications - The specification and demonstration of reliability, availability, maintainability and safety (RAMS), March 2000.
7. A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and its Threats - A Taxonomy. In R. Jacquart, editor, *IFIP Congress Topical Sessions*, pages 91–120. Kluwer, 2004.
8. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.
9. European Organization for Civil Aviation Equipment (EUROCAE WG-72) and Radio Technical Commission for Aeronautics (RTCA SC-216). Airworthiness security process specification, ED-202, 2010.
10. I. Gashi, A. Povyakalo, L. Strigini, M. Matschnig, T. Hinterstoisser, and B. Fischer. Diversity for Safety and Security in Embedded Systems. In *International Conference on Dependable Systems and Networks*, volume 26, 06 2014.
11. International Electrotechnical Commission. Reliability data handbook - Universal model for reliability prediction of electronics components, PCBs and equipment. Technical Report IEC TR 62380, International Electrotechnical Commission, 2004.
12. International Organization for Standardization. ISO 13053 Quantitative methods in process improvment – Six Sigma, 2011.
13. ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.
14. ISO - International Organization for Standardization. IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA) , 2006.

15. ISO - International Organization for Standardization. IEC 61025 Fault tree analysis (FTA) , December 2006.
16. ISO - International Organization for Standardization. ISO 26262 Road vehicles Functional Safety Part 1-10, 2011.
17. ISO - International Organization for Standardization. ISO/IEC 15408. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security (2nd Ed.)*. Springer, 2011.
18. G. Macher, A. Hoeller, H. Sporer, E. Armengaud, and C. Kreiner. Service Deterioration Analysis (SDA): An Early Development Phase Reliability Analysis Method. In *In Review at 45th Annual International Conference on Dependable Systems and Networks (DSN) - RADIANCE Workshop*, 2015.
19. G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner. SAHARA: A security-aware hazard and risk analysis method. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, pages 621–624, March 2015.
20. Microsoft Corporation. The stride threat model, 2005.
21. 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.
22. M. Roth and P. Liggesmeyer. Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees. In *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.
23. SAE International. Guidelines and Mehtods for Conductiong the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.
24. SAE International. Guidelines for Development of Civil Aircraft and Systems, 2010.
25. K. Schmidt, P. Troeger, H. Kroll, and T. Buenger. Adapted Development Process for Security in Networked Automotive Systems. *SAE 2014 World Congress & Exhibition Proceedings*, (SAE 2014-01-0334):516 – 526, 2014.
26. C. Schmittner, T. Gruber, P. Puschner, and E. Schoitsch. Security Application of Failure Mode and Effect Analysis (FMEA). In A. Bondavalli and F. Di Giandomenico, editors, *Computer Safety, Reliability, and Security*, volume 8666 of *Lecture Notes in Computer Science*, pages 310–325. Springer International Publishing, 2014.
27. G. Scuro. Automotive industry: Innovation driven by electronics. http://embedded-computing.com/articles/automotive-industry-innovation-driven-electronics/, 2012.
28. Special Committee 205 of RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.
29. M. Steiner and P. Liggesmeyer. Combination of Safety and Security Analysis - Finding Security Problems That Threaten The Safety of a System. In *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.
30. G. Tennant. *Six sigma: SPC and TQM in manufacturing and services*. Gower Publishing Ltd, 2001.
31. The Common Criteria Recognition Agreement Members. Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/, 2014.
32. D. Ward, I. Ibara, and A. Ruddle. Threat Analysis and Risk Assessment in Automotive Cyber Security. *SAE 2013 World Congress & Exhibition Proceedings*, pages 507–513, 2013.

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.

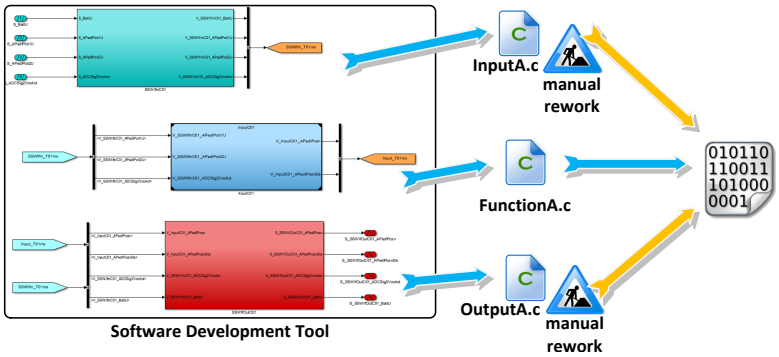


Figure 1: ICC1 AUTOSAR Approach Methodology with Required Manual Intervention

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.

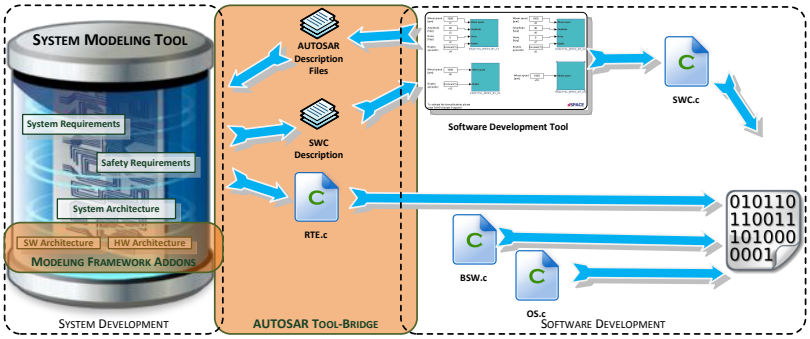


Figure 2: Portrayal of the Bridging Approach Transferring System Development Artifacts to SW Development Phase

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-

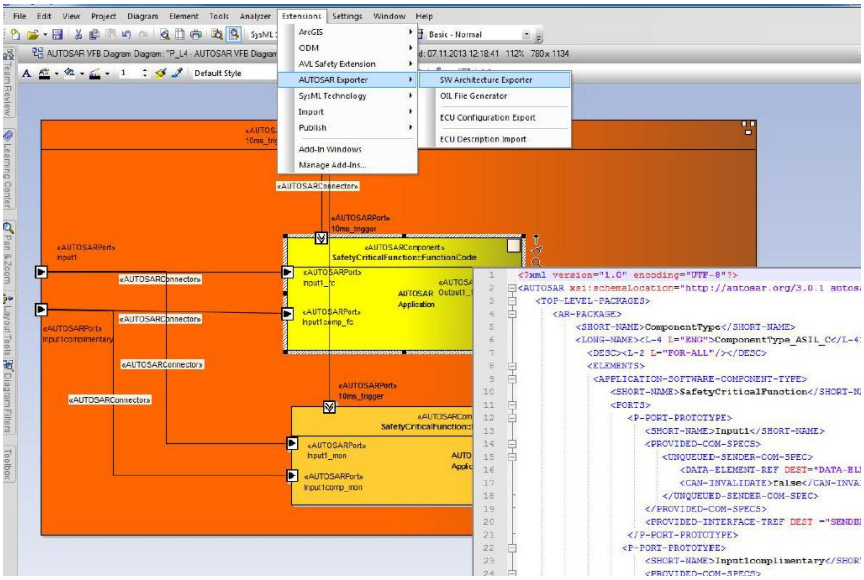


Figure 3: Screenshot of the SW Architecture Representation within the System Development Tool and Extension of Bridging Approach

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.

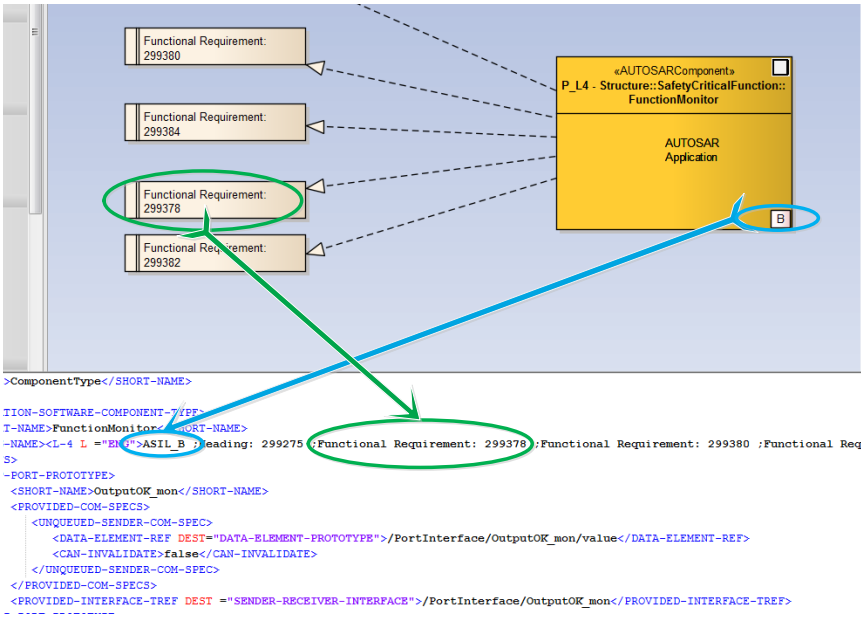


Figure 4: Traceability of Artifacts between System Engineering and SW Development

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.

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

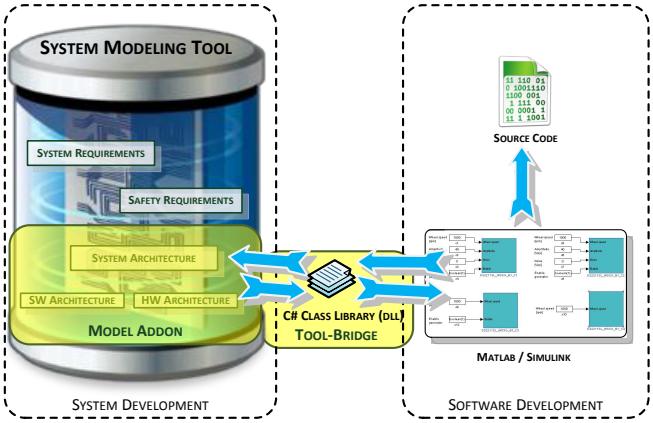


Fig. 1. Portrayal of the Bridging Approach Transferring System Development Artifacts to SW Development Phase

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

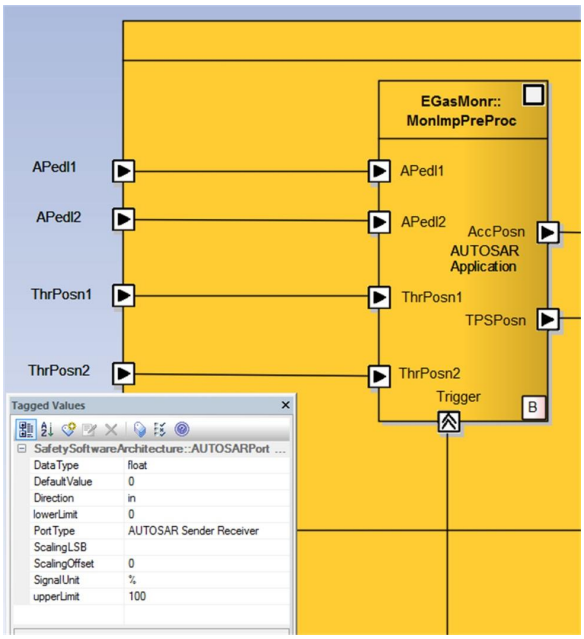


Fig. 2. Screenshot of the SW Architecture Representation within the System Development Tool and Representation of the Interface Information

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.

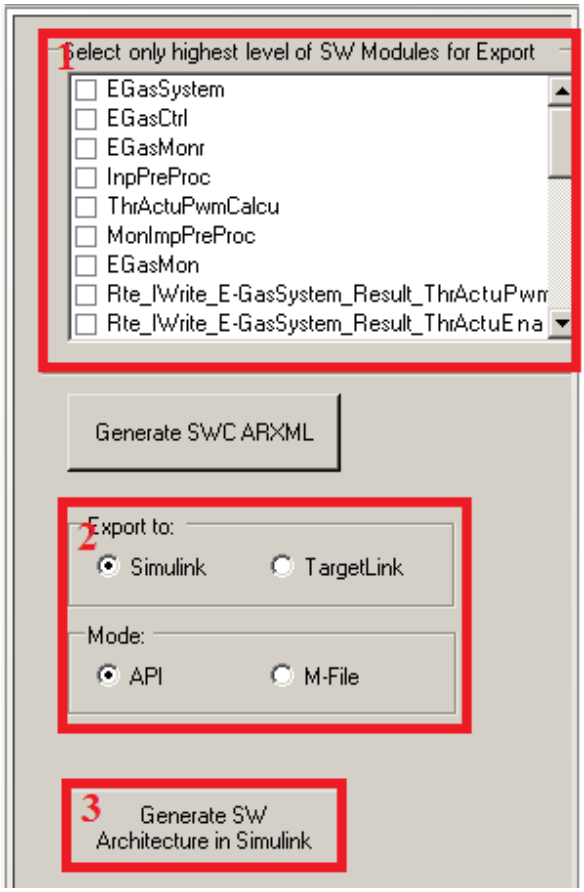


Fig. 3. Screenshot of the SW Architecture Exporter GUI

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

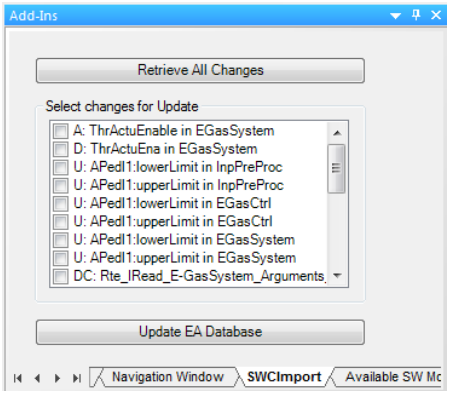


Fig. 4. SW Architecture Importer User Interface

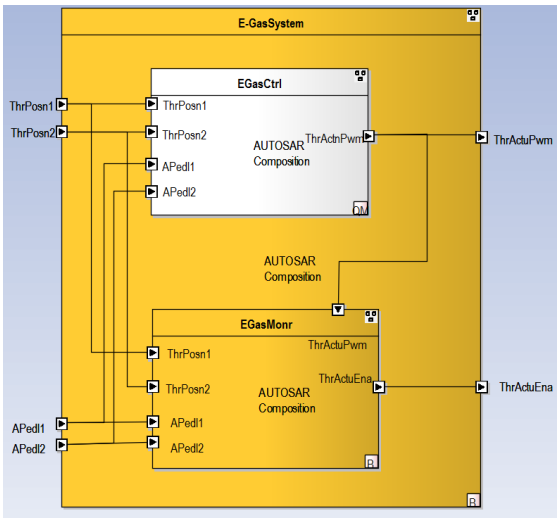


Fig. 5. Top-Level Representation of Demonstration Use-Case in Enterprise Architect

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.

RTE Generation and BSW Configuration Tool-Extension for Embedded Automotive Systems

Georg Macher*‖,Rene Obendrauf‖, Eric Armengaud‖, Eugen Brenner* and Christian Kreiner*

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

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

***Abstract*—Automotive embedded systems have become very complex, are strongly integrated and the safety-criticality and real-time constraints of these systems are raising new challenges. Distributed system development, short time-to-market intervals, and automotive safety standards (such as ISO 26262 [8]) require efficient and consistent product development along the entire development lifecycle. The challenge, however, is to ensure consistency of the concept constraints and configurations along the entire product life cycle. So far, existing solutions are still frequently insufficient when transforming system models with a higher level of abstraction to more concrete engineering models (such as software engineering models).**

The aim of this work is to present a model-driven system-engineering framework addon, which enables the configurations of basic software components and the generation of a runtime environment layer (RTE; interface between application and basic software) for embedded automotive system, consistent with preexisting constraints and system descriptions. With this aim in mind a tool bridge to seamlessly transfer artifacts from system development level to software development level is described. This enables the seamless description of automotive software and software module configurations, from system level requirements to software implementation and therefore ensures both consistency and correctness for the configuration.

Keywords*—automotive, embedded systems, Model-based development, basic software configuration, traceability, model-based software engineering.

I. Introduction

Embedded systems are already integrated into our everyday lives and play a central role in all domains including automotive, aerospace, healthcare, manufacturing industry, energy, or consumer electronics. Current premium cars implement more than 90 electronic control units (ECU) per car with close to 1 Gigabyte software code [4], these are responsible for 25% of vehicle costs and bring an added value between 40% to 75% [18]. This trend of making use of modern embedded systems, which implement increasingly complex software functions instead of traditional mechanical systems is unbroken in the automotive domain. Similarly, the need is growing for more sophisticated software tools, which support these system and software development processes in a holistic manner. As a consequence, the handling of upcoming issues with modern real-time systems, also in relation to ISO 26262 [8], model-based development (MBD) would appear to be the best approach for supporting the description of the system under development in a more structured manner. Model-based development approaches enable different views for different stakeholders, different levels of abstraction, and provide a central storage of information. This improves the consistency, correctness, and completeness of the system specification. Nevertheless, such seamless integrations of model-based development are still the exception rather than the rule and frequently MBD approaches fall short due to the lack of integration of conceptual and tooling levels [3].

The aim of this paper is to present a tool approach which enables a seamless description of safety-critical software, from requirements at the system level down to software component implementation in a bidirectional way. With the presented tool available hardware- software interfacing (HSI) information can be used to generate basic software (BSW) component configurations, as well as, automatic generation of the run-time environment layer (RTE; interface between application software (ASW) and basic software).

The tool consists of a basic software configuration generator and a software interface generator producing $.c$ and $.h$ files for linking ASW and BSW. To ensure more versatility of the tool the required HSI information can either be imported from a HSI spreadsheet template or the system model representation. The goal is, on one hand, to support a consistent and traceable refinement from the early concept phase to software implementation, and on the other hand, to combine 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 to support semi-automatic generation of BSW configuration and the SW-SW interface layer (in AUTOSAR notation known as runtime environment - RTE).

The document is organized as follows:
Section II presents an overview of related works as well as the fundamental model-based development tool chain on which the approach is based. In Section III a description of the proposed tool and a detailed depiction of the contribution parts 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 presented approach.

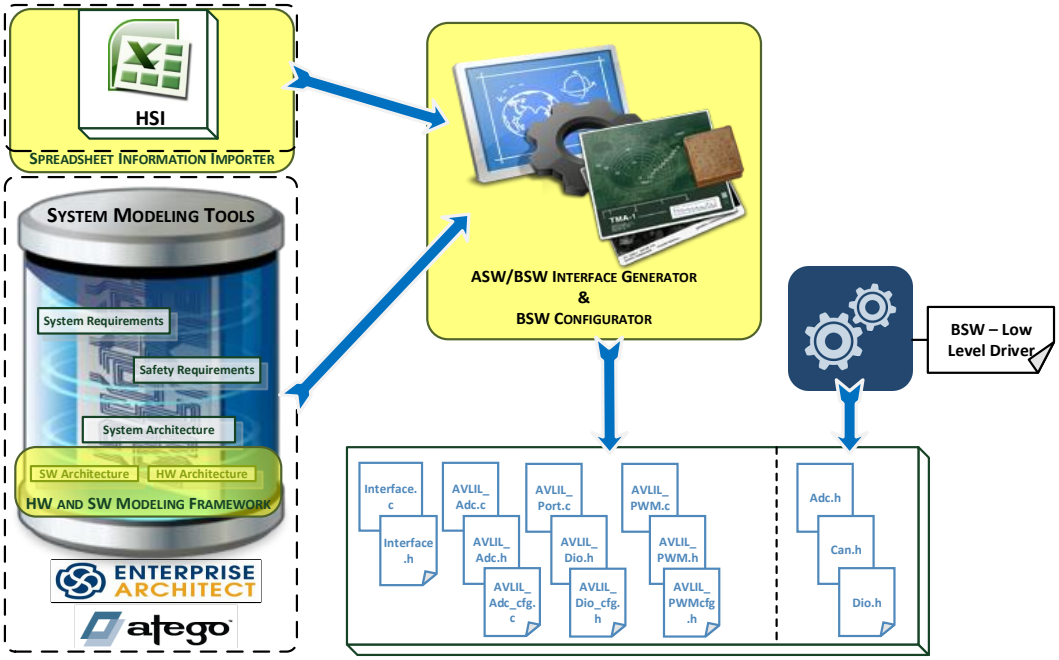


Fig. 2. Portrayal of the Approach for Generation of BSW Configuration and SW Interfaceing Files for the SW Development Phase

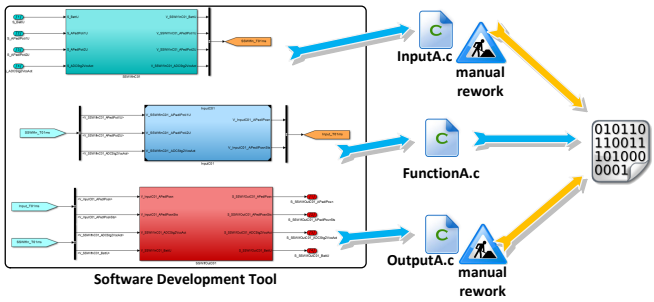


Fig. 1. ICC1 AUTOSAR Approach Methodology with Required Manual Intervention

II. Related Work

Development of automotive embedded software as well as the configuration of the underlying basic software and embedded systems are engineering domains and research topics aimed at moving the development process to an automated work-flow for improving the consistency and tackling the complexity of the software development process across expertise and domain boundaries. Recent publications are mainly based on AUTOSAR [1] methodology.

Due to the ever increasing software complexity of the last few years more and more efforts are becoming necessary to manage the development process of automotive embedded software. To handle this complexity the AUTOSAR consortium was founded and the AUTOSAR methodology provides standardized and clearly defined interfaces between different software components. The AUTOSAR approach features three different classes of implementation (ICC - implementation conformance class). The main benefit of the AUTOSAR ICC1 approach clearly relies on the time-saving in terms of no additional familiarization with usually very complex and time-consuming AUTOSAR tools compared to the full AUTOSAR approach (ICC3). The ICC1 approach does not take advantage of the AUTOSAR benefits from the full AUTOSAR tool-chain supporting tools for RTE configuration and communication interfaces, but standardized component interfaces for exchange of data between the ASW and BSW and therefore features the separation of application specific and hardware specific software parts (like native non-AUTOSAR development). 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, as indicated in Figure 1. The tool approach introduced in this work enhances this aspect by providing a framework for the visualization of both ASW and BSW interface configuration and automated generation of these interfacing *.c* and *.h* files (see Figure 2). Furthermore, the available hardware- software interfacing (HSI) information can be used to generate basic software (BSW) components configurations and the HSI information import functionality can also handle HSI spreadsheet templates to ensure more versatility of the tool.

An approach for an AUTOSAR migration of existing automotive software is described in the work of Kum et. al [10]. The authors highlight the benefits of separating the application software and the basic software and present an approach to configuration of basic software modules instead of time consuming and error-prone manual coding of embedded software. The automatic generation of automotive embedded software and the resultant configuration of the embedded systems thus improves quality as well as re-usability.

In [11], the authors describe a framework for a seamless configuration process for the development of automotive em-

bedded software. The framework is also based on AUTOSAR which defines the architecture, methodology, and application interfaces. The configuration process is established via system configuration and ECU configuration. All the configurations and descriptions used are stored in separate XML (Extensible Markup Language) files, containing described and classified parameters and the associated information. The authors additionally specify a meta-model for the AUTOSAR exchange formats that describe the ECU configuration parameter definition and the ECU configuration description.

Jo et al. [9] describe an approach for the design of a vehicular code generator for distributed automotive systems. The increasing complexity during development of an automotive embedded software and systems and the manual generation of software have the effect of leading to more and more software defects and problems. The authors thus integrated a RTE module into their earlier development phase tool to design and evolve an automated embedded code generator with a predefined generation process. The presented approach saves time through automated generation of software code, compared to manual code generation, it reduces error-prone and time-consuming tasks and is also based on an AUTOSAR aligned approach. The output of the code generator tool is limited to the RTE source code and the application programming interface (API) of the input information. As in our approach, the configuration of software modules, is not focused.

Piao et al. [15] illustrate a design and implementation approach of a RTE generator for automotive embedded software. The RTE layer is located in the middle-ware layer of the AUTOSAR software architecture and combines the top layer mentioned as application software with the underlying hardware and basic software. Automated code generation aims at moving the development steps closer together and thus improving the consistency of the software development process. The output of the automated RTE generator are communication API functions for AUTOSAR SW components of the ASW.

Focusing on software complexity, Jo et al. [7] presents a design for a RTE template structure to manage and develop software modules in automotive industry. The authors focus on the design of a RTE structure also based on the AUTOSAR methodology. Within this design they describe the Virtual Functional BUS (VFB) which establishes independence between the Application Software (ASW) and the underlying basic software (BSW) and hardware.

In [14], an approach for realizing location-transparent interaction between software components is shown. The proposed work illustrates the relationship between the RTE and the VFB and shows which artifacts of the VFB are necessary for the generation of the RTE.

A work depicting the influence of the AUTOSAR methodology on software development tool-chains is presented by Voget [19]. The tool framework presented, named ARTOP (AUTOSAR Tool Platform), is an infrastructure platform that provides features for the development of tools used for the configuration of AUTOSAR systems. The implemented features are base functionalities required for different AUTOSAR tool implementations. The work does not, however, focus on a specific tool integration.

To summarize, none of the approaches described above supports (1) the generation of source code and (2) configuration of the basic software from information available at system level and from system models. The approach we present by contrast, supports not only the automatic generation of the RTE source code, but also the automated generation of basic software configuration of embedded systems from system models.

III. Basic Software Interface and Configuration Generation Approach

The underlying concept of the approach is to have a consistent information repository as a central source of information, to store all information of all engineering disciplines involved in embedded automotive system development in a structured manner [13]. The concept focuses on allowing different engineers to do their job in their own specific way, but providing traces and dependency analysis of features concerning the overall system, e.g. safety, security, or dependability. The approach stirs out of common AUTOSAR based approaches and additionally supports a non-AUTOSAR or AUTOSAR ICC1 approach, which are frequently hampered due to a lack of supporting tools. The decision of not fostering a full AUTOSAR approach is based on the one hand on focusing not only AUTOSAR based automotive software development and on the other hand, experiences we have made with our previous approach [12] confirm the problem mentioned by Rodriguez et al. [16]. Not all development tools fully support the entire AUTOSAR standard, because of its complexity, which leads to several mutual incompatibilities and interoperability problems.

Figure 2 shows an overview of the approach and highlights the main contributions. For a more detailed overview of the orchestration for the overall development tool-chain see [13].

The tool approach introduced in this work provides a framework for the visualization of ASW and BSW interface configuration and automated generation of these interfacing $.c$ and $.h$ files (see Figure 2). Furthermore, the available hardware- software interfacing (HSI) information can be used to generate basic software (BSW) component configurations and the HSI information import functionality can also handle HSI spreadsheet templates to ensure more versatility of the tool. More specifically, the contribution proposed in this work consists of the following parts:

- *AUTOSAR aligned 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 abstraction layer), see Figure 2 – HW and SW Modeling Framework.
- *BSW and HW module modeling framework*:
  Enhancement of an UML profile to describe BSW components and HW components. To ensure consistency of the specification and implementation for the entire control system, see Figure 2 – HW and SW Modeling Framework.
- *RTE generator*:
  Enables the generation of interface files ($.c$ and $.h$) between application-specific and hardware-specific software functions, see Figure 2 – ASW/BSW Interface Generator .

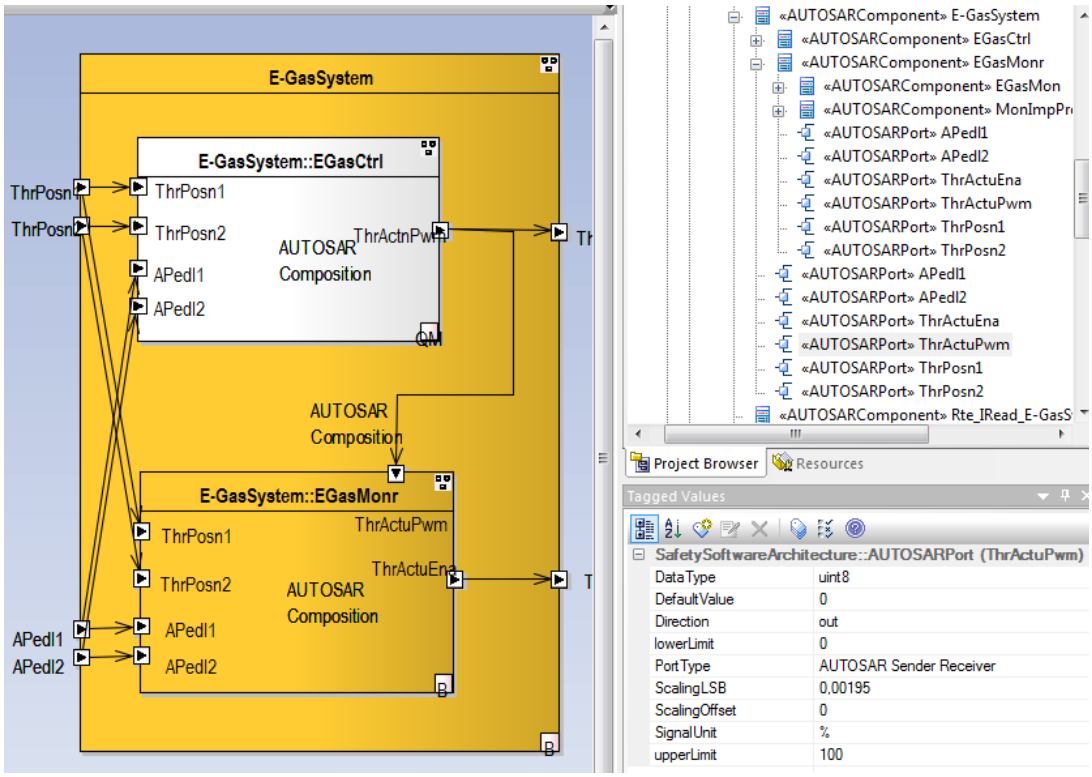


Fig. 3. Screenshot of the SW Architecture Representation within the System Development Tool and Representation of the Interface Information

- *Basic software configuration generator*:
  Generates BSW configurations according to the specifications within the HSI definition, see Figure 2 – BSW Configurator.
- *Spreadsheet information importer*:
  Enables the import of HSI definition information done in spreadsheet format, see Figure 2 – Spreadsheet Information Importer.

This proposed approach closes the gap between system-level development of abstract UML-like representations and software-level development, also mentioned by Giese et al. [5], Holtmann et al. [6], and Sandmann and Seibt [17] by supporting consistent information transfer between system engineering tools and software engineering tools. Furthermore the approach minimizes redundant manual information exchange between tools and contributes to simplifying seamless safety argumentation according to ISO 26262 for the developed system. 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 mentioned by Broy et al. [3].

The contribution proposed in this work is part of the framework presented in [13] aiming towards software development in the automotive context. The implementation of the approach is based on versatile C# class libraries (dll) and API command implementations to ensure tool and tool version in-dependence of the general-purpose UML modeling tool (such as Enterprise Architect or Artisan Studio) and other involved tools (such as spreadsheet tool and software development framework). The following sections describe those parts of the approach that make key contributions in more details.

A. AUTOSAR aligned UML modeling framework

The first part of the approach is a specific UML modeling framework enabling software architecture design in AUTOSAR like representation within a state-of-the-art system development tool (in this case Enterprise Architect). A specific UML profile to limit the UML possibilities to the needs of software architecture development of safety-critical systems and enable software architecture design in AUTOSAR like representation within the system development tool (Enterprise Architect). In addition to the AUTOSAR VFB abstraction layer [2], the profile enables an explicit definition of 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). Hence the SW architecture representation within EA can be linked to system development artifacts and traces to requirements can be easily established. This brings further benefits in terms of constraints checking, traceability of development decisions (e.g. for safety case generation), reuse and ensures the versatility to also enable AUTOSAR aligned development as proposed in [12]. Figure 3 shows an example of software architecture artifacts and interface information represented in Enterprise Architect. As can be seen in the depiction, all artifacts required to model the SW architecture are represented and inherit the required information as tagged values.

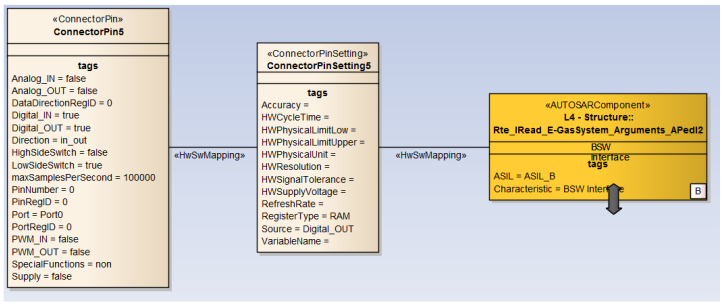


Fig. 4. Screenshot of the BSW and HW Pin Representation within the System Development Tool

B. BSW and HW Module Modeling Framework

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. The hardware profile of the approach allows a graphical representation of hardware resources (such as ADC, CAN), calculation engines (core), and connected peripherals which interact with the software. Special basic software (BSW) and hardware module representations are assigned to establish links to the underlying basic software and hardware layers. This enables an intuitive graphical means of establishing software and hardware dependencies and a hardware-software interface (HSI), as required by ISO 26262. Software signals of BSW modules can be linked to HW port pins via dedicated mappings. On the one hand this enables the modeling and mapping of HW specifics and SW signals, see Figure 4 and on the other hand this mapping establishes traceable links to port pin configurations. A third point is that this HW dependencies can be used to interlink scheduling and task allocation analysis tools for analysis and optimization of resource utilization.

C. Runtime Environment Generator

The third part of presented approach is the SW/SW interface generator. This dll- based tool generates $.c$ and $.h$ files defining SW/SW interfaces between application software signals and basic software signals based on modeled HSI artifacts. In addition, this generation eliminates the need for manual SW/SW interface generation without adequate syntax and semantic support and ensures the reproducibility and traceability of these configurations.

Figure 5 shows the conceptual overview of generated files. The $.c$ and $.h$ files on application software level are generated via a model-based software engineering tool, such as Matlab/Simulink. The files on the basic software level are usually provided by the hardware vendor. While the files referred to in the SW/SW interface layer are generated by our approach.

The generated files are designed in a two-step approach. The first step of the interfacing approach ($interface.c$ and $interface.h$) establishes the interface between ASW and BSW based on AUTOSAR RTE calls. The second step ($AVLIL\_BSWa.c$ and $AVLIL\_BSWa.h$) maps these AUTOSAR RTE based calls to the HW specific implementation

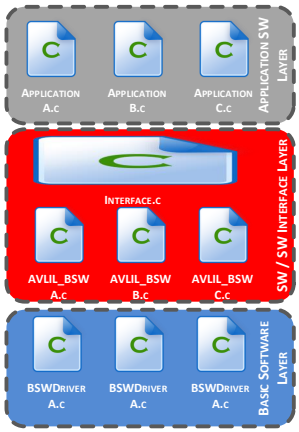


Fig. 5. Overview of Architecture Level Files Generated by the Interface Generator

of basic SW drivers. This ensures independence from implementation of the BSW drivers and also features an AUTOSAR ICC1 approach if needed.

D. Basic Software Configuration Generator

The basic software configuration generator is also part of the dll- based tool, which generates BSW driver specific $*\_cfg.c$ files. These files configure the vendor specific low-level driver (basic software driver) of the HW device according to the HSI specifications. The mapping of HSI specifications to low-level driver configuration is hardware and low-level driver implementation specific and needs to be done once per HW device and supported low-level driver package.

E. HSI Spreadsheet Information Importer

The HSI definition requires mutual domain knowledge of hardware and software and is to be a work product of a collective workshop of hardware, software, and system experts and will act as the linkage between different levels of development. Consistency and evidence of correct implementation of HSI can be challenging in case of concurrent HW and SW development and cross-dependencies of asynchronous update intervals. Therefore, this approach enables a practicable and intuitive way of engineering HSI definitions in a spreadsheet tool (Excel) and transforms them to a reusable and versionable representation in the MDB tool (Enterprise Architect). The spreadsheet template defines the structure of the data representation in a project-specific customizable way. On the one hand this enables a practicable and intuitive means of engineering HSI definitions with spreadsheet tools, while their machine- and human-readable notation ensures a cost- and time-saving alternative to the usually complex special-purpose tools, while on the other hand it enables the generation of SW/SW interface files and BSW configurations without the need for a model-based development toolchain in place. Figure 6 depicts the project-independent template structure for HSI definition data preparation.

IV. Application of the Proposed Approach

This section demonstrates the benefits of the introduced approach for development of automotive embedded systems.

HSI definition		<project/customer logo>		<project name>
Sensor/Actuator		throttle sensor 1	throttle sensor 2	throttle actuator 1
Direction	common	in	in	out
Source(CAN/ANA/DIG)		ANA	CAN	DIG
Signaltype (V / % / deg)		°	%	PWM
Signalname		ThrPos1	ThrPos2	ThrActuPWM
signal range lower limit		0	0	10
signal range upper limit		60	100	90
Scaling LSB				
Scaling Offset				
accuracy	SW	0,05	0,5	-
ASIL		ASIL B	ASIL B	ASIL QM
default value		0	0	0
type		float	uint8	uint8
refresh rate	ms	10	10	100
register-type		RAM	RAM	RAM
variable name		v_APS1	s_APS2	ThrActuPWMout
unit		V	V	V
physical range lower limit		0,5	1	0
physical range upper limit		4,5	4	5
cycle time	ms	1	1	100
supply voltage	HW	5	5	-
signal tolerance		0,25	0,5	-
resolution	bit	16	8	8
port		PortA	PortB	PortC
pin		12	1	4

Fig. 6. Example of a project-independent spreadsheet template structure for HSI definition

TABLE I. OVERVIEW OF THE EVALUATION USE-CASE SW ARCHITECTURE

Object type	**Element-count**	**Configurable Attributes per Element**
ASW Modules	10	3
BSW Modules	7	3
ASW Module Inputs	54	10
ASW Module Outputs	32	10
ASW/ASW Interfaces	48	-
ASW/BSW Interfaces	19	-
HW/SW Interfaces	19	13

We used an automotive battery management system (BMS) as the use-case for the evaluation of the approach. This use-case is an illustrative material, reduced for internal training purposes and is not intended to be either exhaustive in scope or to represent leading-edge technology.

The definition of the software architecture is usually done 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 (as shown in Figure 3). This does not hamper the work of the software architect but enables the possibility to also link existing HSI mapping information to the SW architecture (as shown in Figure 4).

The use-case consists of 10 ASW modules and 7 BSW modules with 19 interface definitions between ASW and BSW and makes use of the 3 fundamental low-level HW functions (digital input/output, analog input/outputs, and PWM outputs). A more complete overview of use-case is given in Table I.

The definition of the 19 HW/SW interfaces with 10 parameters for each SW signal and 13 parameters for each HW pin sums up to 437 parameter configurations within the HSI spreadsheet template or in the MDB tool, which can be used to generate ASW/BSW interfaces and BSW configurations.

This results in the file architecture depicted in Figure 7. With the use of the approach 8 additional interfacing files with 481 lines of code (LoC) source and 288 LoC configuration have been generated.

In terms of getting started with AUTOSAR aligned development or supporting non-AUTOSAR SW development our approach features a smooth first step approach of the ICC1 AUTOSAR and generates an interface layer (similar to AUTOSAR RTE) without relying on full AUTOSAR tooling support. In terms of safety-critical development the approach presented supports traceability links between BSW configurations to HSI information and eliminates the need of manual interface source code rework, which further surmounts the main drawbacks of the ICC1 AUTOSAR approach.

V. CONCLUSION

An important challenge for the development of embedded automotive systems is to ensure consistency of the design decisions, SW implementations, and driver configurations, especially in the context of safety-related development. This work presents an approach which seamlessly describes safety-critical software, from requirements at the system level down to software component implementation in a traceable manner. The available hardware- software interfacing (HSI) information can thus be used to generate basic software (BSW) component configurations, as well as automatic software interface layer generation (interface between application software and basic software). With this aim in mind a framework consisting of a basic software configuration generator and a software interface generator producing $.c$ and $.h$ files for linking ASW and BSW has been presented, which can also be used in combination with a spreadsheet based HSI definition. The main benefits of this enhancement are: improved consistency and traceability from the initial design at the system level down to the single CPU driver configuration, together with a reduction of cumbersome and error-prone manual work along the system development path. Further improvements of the approach include the progress in terms of reproducibility and traceability of configurations for software development (such as driver configurations and SW-SW interfaces).

The application of the presented approach has been demonstrated utilizing an automotive BMS use-case, which is intended to be used for training purposes for students and engineers and does not represent either an exhaustive or a commercial sensitive project. While the authors do not claim completeness of the analysis (due to confidentiality issues), the benefits of the approach are already evident.

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),

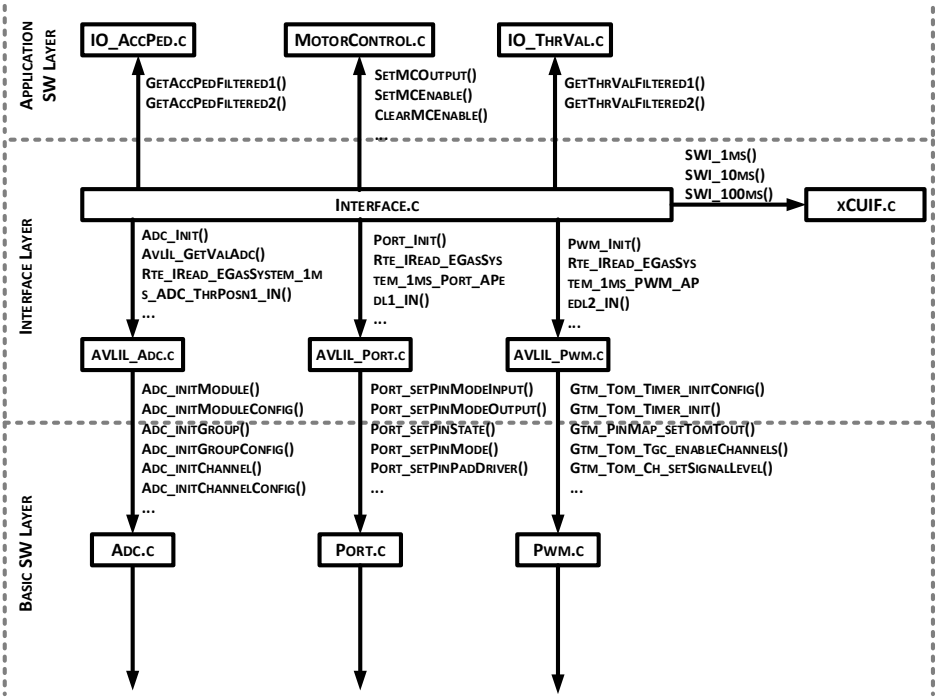


Fig. 7. Excerpt of Generated Files for the BMS Use-Case

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] AUTOSAR Development Cooperation. Virtual Functional Bus. online, 2013.

[3] 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.

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

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

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

[7] J. Hyun Chul, P. Shiquan, C. Sung Rae, and J. Woo Young. RTE Template Structure for AUTOSAR based Embedded Software Platform. In *Basic Research Program of the Ministry of Education, Science and Technology*, pages 233–237, 2008.

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

[9] H. C. Jo, S. Piao, and W. Y. Jung. Design of a Vehicular code generator for Distributed Automotive Systems. In *Seventh International Conference on Information Technology*. DGIST, 2010.

[10] D. Kum, G.-M. Park, S. Lee, and W. Jung. AUTOSAR Migration from Existing Automotive Software. In *International Conference on Control, Automation and Systems*, COEX, Seoul, Korea, 2008. DGIST.

[11] J.-C. Lee and T.-M. Han. ECU Configuration Framework based on AUTOSAR ECU Configuration Metamodel. 2009.

[12] 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.

[13] 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.

[14] N. Naumann. Autosar runtime environment and virtual function bus. Department for System Analysis and Modeling.

[15] S. Piao, H. Jo, S. Jin, and W. Jung. Design and Implementation of RTE Generator for Automotive Embedded Software. In *Seventh ACIS International Conference on Software Engineering Research, Management and Applications*. DGIST, 2009.

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

[17] 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.

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

[19] S. Voget. AUTOSAR and the Automotive Tool Chain. In *DATE10*, 2010.

2015-01-0183
Published 04/14/2015

doi:10.4271/2015-01-0183
saepcelec.saejournals.org

A Model-Based Configuration Approach for Automotive Real-Time Operating Systems

Georg Macher
Graz University of Technology

Muesluem Atas and Eric Armengaud
AVL List GmbH

Christian Kreiner
Graz University of Technology

ABSTRACT

Automotive embedded systems have become very complex, are strongly integrated, and the safety-criticality and real-time constraints of these systems raise new challenges. The OSEK/VDX standard provides an open-ended architecture for distributed real-time capable units in vehicles. This is supported by the OSEK Implementation Language (OIL), a language aiming at specifying the configuration of these real-time operating systems. The challenge, however, is to ensure consistency of the concept constraints and configurations along the entire product development.

The contribution of this paper is to bridge the existing gap between model-driven systems engineering and software engineering for automotive real-time operating systems (RTOS). For this purpose a bidirectional tool bridge has been established based on OSEK OIL exchange format files. The approach makes use of existing high-level control system information in SysML format to generate the configuration of automotive real-time operating systems. The implemented tool extension transfers artifacts from system engineering domain to software development frameworks for RTOS configuration, thereby creating traceable links across domain and tool boundaries, and relying on standardized OSEK OIL exchange files.

CITATION: Macher, G., Atas, M., Armengaud, E., and Kreiner, C., "A Model-Based Configuration Approach for Automotive Real-Time Operating Systems," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(2):2015, doi:10.4271/2015-01-0183.

INTRODUCTION

The number of embedded systems in the automotive domain has grown significantly in recent years. Current premium cars implement more than 90 electronic control units (ECU) with close to 1 Gigabyte software code [1]. In 2018 30% of the overall vehicle costs are predicted to stem from vehicle electronics [2]. This trend is also strongly supported by the ongoing replacement of traditional mechanical systems with modern embedded systems. This enables the deployment of more advanced control strategies, thus providing new benefits for the customer and environment, such as reduced fuel consumption and better driveability. At the same time, the higher degree of integration and the safety-criticality of the control application raise new challenges. Hence, the correctness of the different applications, both in the time domain and value domain, possibly running on the same computing platform, has to be guaranteed. 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 [3] for road vehicles have been established to provide guidance during the development of safety-critical systems. These standards rely on risk identification and mitigation strategies, further supported by appropriate quality management. They target early hazard identification as well as solid counter measure specification, implementation and validation along the entire product life cycle.

The challenge in this context is to provide evidence of consistency during product development among the different work-products. To handle upcoming issues with modern real-time systems in relation to ISO 26262, model-based development supports the description of the system under development in a more structured way. Model-based development approaches enable different views for different stakeholders, different levels of abstraction, and central storage for information. This improves the consistency, correctness, and completeness of the system specification and thus supports the demands of time-to-market (first time right).

The contribution of this paper is to bridge the existing gap between model-driven system engineering tools and software engineering tools for automotive real-time operating systems (RTOS). More especially, the approach relies on the generation of standardized OSEK implementation language files (OIL files) [4] for automotive RTOS based on existing high-level control system information in SysML format. The goal is to support a consistent and traceable refinement from the early concept phase to individual configurations of the RTOS.

The document is organized as follows:

The following section presents an introduction to OSEK/VDX and OSEK OIL as well as model-based development and integrated tool chains. In the third section, a description of the proposed approach for the generation of RTOS configuration files according to the OIL standard is provided. An application and evaluation of the approach is presented in the fourth section. Finally, this work is concluded with an overview of the presented approach.

OSEK/VDX AND MODEL-BASED DEVELOPMENT OVERVIEW

OSEK is the German abbreviation for open systems and their interfaces for electronics in motor vehicles, a consortium founded in 1993 by several German automotive companies. VDX (Vehicle Distributed eXecutive) was the French pendant from the French car manufacturers' side, which, in 1994, regrouped the OSEK/VDX consortium.

OSEK/VDX is an open standard for specifications for embedded real-time operating systems (RTOS) designed to provide standard software architecture for the various electronic control units (ECUs) and partially standardized in ISO 17356. The work of the OSEK/VDX consortium is now continued by the AUTOSAR consortium [5], which is based on OSEK/VDX specifications. Therefore, several publications deal with OSEK/VDX approaches and different tool vendors provide OSEK/VDX tools, which are supporting at different development stages.

The OSEK implementation language files (OIL) are intended to be used to describe OSEK RTOS. These files may be generated manually or via configuration tools. In both cases, the OIL files include all object containers and information required to configure the RTOS of one specific ECU. This is possible due to the fact that automotive real-time systems are typically statically configured. In accordance with this, features of an OSEK implementation are configured at compile-time and changes at run-time are prohibited. Only static priorities are allowed for tasks and FIFO scheduling is used for tasks with equal priority. Deadlocks and priority inversion are prevented by a priority ceiling approach.

The following paragraph provides a brief overview of OIL files and included objects to configure automotive RTOS. The next paragraph briefly describes both OSEK/VDX supporting tools and publications related to OSEK OIL. Nevertheless, this is not intended to be a full description or complete overview; the main focus is set on RTOS configuration with OIL files. The third paragraph of this section provides an overview of publications dealing with model-based development approaches and integrated tool chains for real-time systems. The final paragraph highlights the basic framework for the proposed approach.

OSEK Implementation Language Files

As mentioned previously, the OIL files inherit a normalized description language for OS configuration and related objects. OIL files are commonly used in the automotive domain to configure the real-time operating systems of individual ECUs. This is frequently done manually, due to the simple human readable structure of OIL files and the lack of tools supporting an automated information exchange.

OIL files typically consist of implementation specific definitions, which are closely related to the hardware (ECU) in use and specify the OIL object and all possible attribute properties. Secondly, OIL files consist of application specific definitions that are, as the name implies, specific to one application and specify the attribute of the OIL object.

OIL objects and their descriptions:

- ***CPU*** - specification of the core which runs the application
- ***OS*** - specification of the RTOS of the specific core
- ***APPMODE*** - definition of different application modes to control task features or support reduced functionalities
- ***ISR*** - specification of interrupt service routines
- ***RESOURCE*** - specification of different resources of the ECU
- ***TASK*** - software functionality handled by the RTOS
- ***COUNTER*** - a SW/HW resource for alarms
- ***EVENT*** - synchronization mechanism for tasks
- ***ALARM*** - notification mechanism
- ***COM*** - definition of the OSEK communication subsystem
- ***MESSAGE*** - definition of data exchange between tasks
- ***NETWORK MESSAGE*** - definition of data exchange between different CPUs
- ***NM*** - network management subsystem configurations

As mentioned previously, the creation of OIL files is done for each CPU individually and very often manually, due to the project-specific nature of OIL files. Nevertheless, due to the introduction of multi-core real-time systems and current awareness of safety-criticality of such system configurations, tool support and automation of OIL generation becomes increasingly relevant.

OSEK/VDX Related Publications and Tools

To our knowledge, most development frameworks do not include a tool for automatic OIL file generation of previous development stages at higher abstraction level, e.g. systems engineering. Most frameworks either require manual generation of OIL files by the developer, or they provide a dedicated graphical user interface for support and guidance while generating the OIL file. Such representations do not support the automatic generation of OIL files from available information of previous development steps, nor do

they significantly speedup the configuration process of RTOS. Many available OIL file configurators provide such a representation of the OIL information. They thereby provide guidance to minimize configuration failures, but do not significantly reduce workload or speed up the generation of OIL files. Also the import of prior available information is very limited. Vector's OIL Configurator and GOB - GUI based OIL Builder are mentioned as representatives of such commercial and non-commercial tool implementations. This group of tools will be referenced as SW Development Level Configurators in the course of this document.

A DSL (domain specific language) approach was applied with SmartOSEK's visual designer [6], a part of the SmartOSEK platform. The visual design tool enables modeling of applications in a graphical way and automatically generates the designed OIL file. The graphic design helps developers devote their minds to the modeling of the application rather than applying correct OIL syntax and semantics. Drawbacks of this approach are the limited usability, solely for SmartOSEK platform, and the missing availability of feedback information into the model.

Most OIL configurators, like the previously mentioned Vector OIL Configurator, GOB, and Freescale's OSEK Builder Tool focus on generation of OIL files at a software development level. The work of Koester et al. [7] is another example of achieving OSEK-compliant code after software development.

This implies the drawback that information of prior development phases cannot be used for timing analysis or has to be transferred manually to the corresponding OIL files.

The publication of Kim et al. [8] suggests a lightweight AUTOSAR software platform and additional extensions of OSEK OIL files. This approach offers the possibility of adding system protection functions, real-time scheduling functionalities, and additional timing attributes to the standard OIL files. The platform is improved for development of time-critical application software. Nevertheless, the presented approach focuses on adding extensions to OIL files, rather than supporting automated generation of OIL files or the origin of such information.

Yang et al. [9] base their work on the SmartOSEK platform mentioned earlier in this section and present the conversion of UML models into OSEK/VDX models for simulation and optimization of the system design. The authors claim that by converting UML into OSEK/VDX models productivity can be improved, correctness of development artifacts can be, more easily ensured and documentation can be provided with less effort and better quality.

Gu et al. [10] focus on the mapping of RT-UML (UML real-time profile) to OSEK APIs. The authors focus on the description of automated mechanisms for generation of application code and seamless integration of models for software development, but do not provide methods of the transformation of UML and OSEK artifacts.

In contrast to the last two publications, our approach focuses on the safety-critical development of multi-core systems. The two approaches have solely the ability to provide a graphical representation of an OSEK OIL file and thereby improve productivity. Our approach, in contrast, enables an automated information exchange and generation also for safety and multi-core constraints (non-functional requirements which are hard to trace and verify). This supports several factors required for safety-critical system development (e.g. tracing of automotive safety integrity levels (ASIL) and ASIL decompositions, proof of freedom from interference, proof of separation, support of safety-case generation).

Model-Based Development and Integrated Tool Chains

Broy et al. [11] mention concepts and theories for the model-based development of embedded software systems. The authors also claim model-based development to be 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 problems that arise through the use of an inadequate tool chain (e.g. redundancy, inconsistency and lack of automation). The focus of their work is to present basic ideas and concepts, although no detailed solutions for the automotive domain are presented.

The work of Quadri and Sadovykh [12] presents a real-time embedded system for avionics and a model-driven engineering approach aiming to develop novel model-driven techniques and new tools supporting design, validation, and simulation. They defined profiles using a subset of UML and SysML for their approach and mentioned the usage of effective design tools and methodologies as crucial to be capable of managing complex real-time embedded systems. Furthermore, they highlighted the possibility of high-level model analysis for schedulability.

The work of Holtmann et al. [13] 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 no guidance which model is to be used at which specific development stage. A model-based development process is presented that conforms to the process reference model of Automotive SPICE. 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 that require manual synchronization and which are therefore often inconsistent, rely on redundant information and, due to a lack of automation, require redundant manual work.

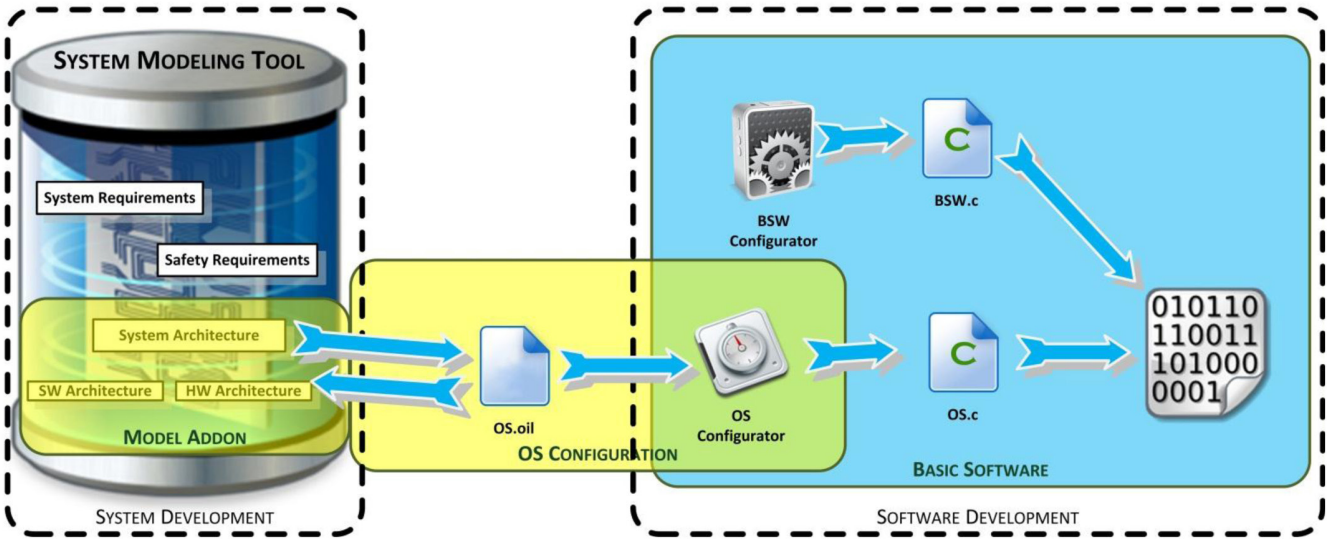


Figure 1. Overview of the bridging approach ensuring boundless information exchange.

This issue is also addressed by Giese et al. [14]. They address the problems of different models along the development process, with each designed for a specific issue. They highlight the step from system design to software design as critical. System design models have to be correctly transferred to the software engineering model, and later changes must be kept consistent.

Fabbrini et al. [15] provide an overview of software engineering in the European automotive industry and present tools, techniques and countermeasures to prevent faults. The authors also highlight the importance of tool integration and model-based development approaches.

Basic Framework

This paragraph gives a short overview of the model-based development tool chain in use and the related preliminary work that provides the base for the proposed approach. The prototype of our tool chain, proposed by Mader [16], is a specific implementation of a tool-independent and language-independent methodology to support continuous safety analyses of system architecture development according to ISO 26262. This approach stems from the CESAR Project [17] and was further improved by a feasibility study [16] to support development at concept phase and system development level. A UML profile, tailored to the needs of automotive safety engineering, is used to define the model of the system under development. This model can be refined from the initial development phase down to system development level according system development and safety engineering needs.

The basic concept behind this framework is to have a consistent information repository as a central source of information. This concept allows different engineers to do their job in their specific manner, provides traces between different artifact types, and ensures timeliness of data. Proprietary extensions for the modeling tool ensure seamless and consistent transition of information between the repository and various adequate special-purpose tools (such as software development tools). This enables reorganization from a document-centric development approach to a seamless model-based development approach. For a more detailed overview of the concept as a whole, see [18].

OVERVIEW OF THE OSEK OIL GENERATOR

The contribution proposed in this work is an extension of the previously mentioned framework towards RTOS configuration. More specifically, our contribution consists of the following parts:

- ***UML modeling framework extension***: Enhancement of the software UML profile with the capability of visualizing and processing OSEK OIL objects. The profile extends the AUTOSAR aligned model [19] we proposed for software development to enable configuration of the OSEK OS by prior available constraints. This is required for consistent SW system description, see figure 1 - model add-on.
- ***OIL configuration generation***: An extractor that automatically generates OIL files from existing information at a system development level. This is required to ensure consistency of the specification and implementation for the RTOS, see figure 1 - OS configuration.
- ***OIL configuration importer***: Importers support round-trip engineering through re-importation of information updates from the software development tool via the OIL file, see figure 1 - OS configuration.

This proposed extension is a constituent of the proposed tool chain in [18] to close the gap between system-level development with abstract UML-like representations and software-level development modeling tools (such as OS configuration tools). This bridging guarantees consistency of information, due to the single source of information principle and shares information more precisely and accurately. The approach minimizes redundant manual information exchange between tools and also takes ISO 26262 requirements (especially traceability) and ISO 26262 constraints into account.

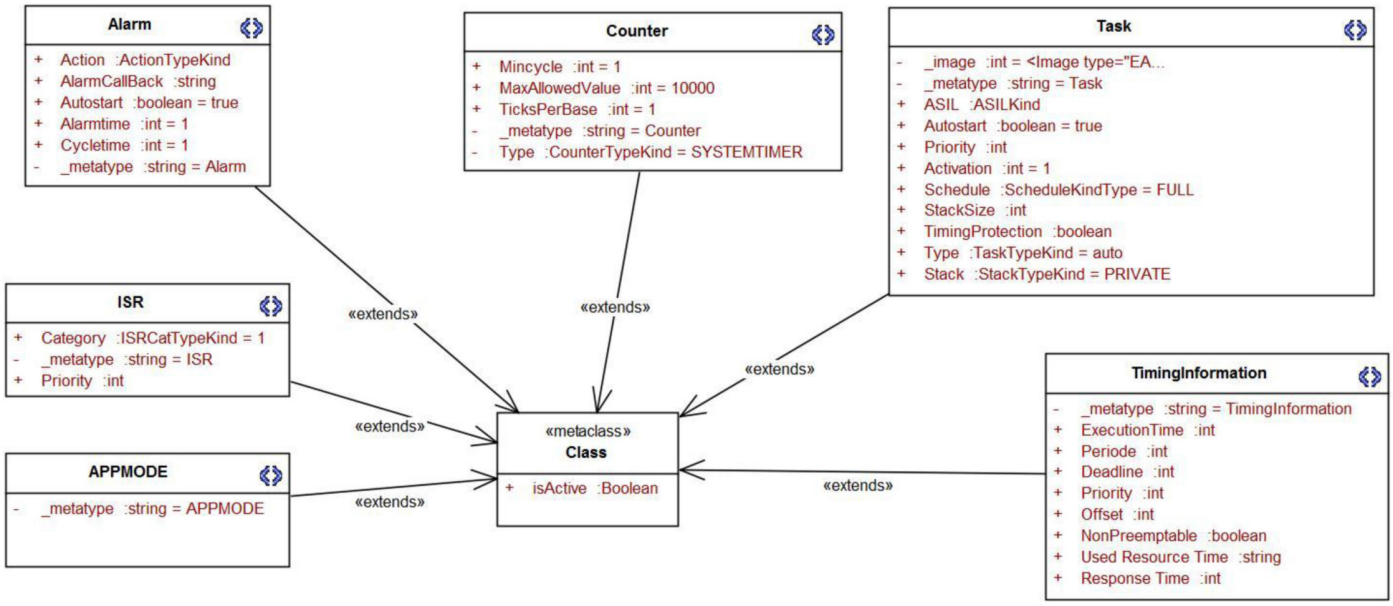


Figure 1 shows the conceptual overview of the tool-chain and highlights the OS configuration parts. As can be seen in the figure, a lack of tool support for information transfer between system development tools and basic software development tools exist, which has been bridged by the proposed approach. Therefore, the proposed approach simplifies the information handover and reduces competence conflict potentials between system development and hardware and software development phases of the automotive product development cycle. OIL files are generated for each CPU individually, but due to the introduction of multi-core real-time systems in the automotive domain and the recent awareness of safety-criticality of such systems, tool support and automation of OIL generation is increasingly relevant. The tight linking of the independent system development and OS configuration tools to a seamless model-based development tool chain interacting via OIL files further allows the inclusion of additional tools, such as scheduling analysis tools, seamlessly into the tool chain.

UML Modeling Framework Extension

A UML profile has been regenerated for this approach to allow a graphical visualization and processing of OSEK OIL objects. This profile extension ensures the accumulation of additional information which enable the mapping of tasks to a specific core and clear arrangement of dependencies and shared resources in terms of multi-core development. Figure 2 shows a small selection of the additional profile elements and their accumulated element information. The regenerated profile offers an intuitive, graphical way of generating OSEK OS configurations and a highlighting functionality of safety-related software tasks and resources. This also enables the possibility of a traceable automatic OIL file configuration generation instead of the typical manual definition, which inherits increasing significance in terms of safety-critical system development according ISO 26262. Note that this profile has been integrated in the existing framework described previously in this paper.

This prior profile allowed development of software architectures fully in line with the AUTOSAR approach [5] and is now enhanced with the ability to add additional information regarding safety considerations and basic OS configurations. Consequently, the system description can be refined down to the operating system, thus improving architecture consistency over the skills boundaries (systems engineering and software engineering). Figure 2. Selection of UML Elements of the UML Framework Extension.

OIL Configuration Generator

The second part of the approach is an exporter that is able to export the RTOS configuration previously available (from the SysML model) to an OIL file. The exporter generates OIL files enriched with the available system and safety development artifact traces (such as required ASIL of task implementation). Most of the state-of-the-art basic software development frameworks are capable of configuring the RTOS according to the specifications within such an OIL file. Consequently, the use of this exporter additionally improves communication of the (safety) context to the software experts into their native development tools, thus improving the consistency of the product development. Furthermore, the tool chain is capable of multi-core or multi-system development; therefore the generation of OIL file is selectable for individual CPUs.

OIL File Importer

The third 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 updates of the representation in the system development tool and the software development tool. This ensures consistency between system development artifacts and changes done in the software development tool that may occur due to the typical iterative automotive development cycles phases. The importer also implies an overview of changes between database and re-imported OIL files; this offers the possibility of selective database updates. Finally, the importer enables

the reuse of available RTOS configurations, guarantees consistency of information in combination with the export functionality, and shares information more precisely and less ambiguously.

Figure 3 shows a screenshot of the import, selective update, and difference highlighting functionality.

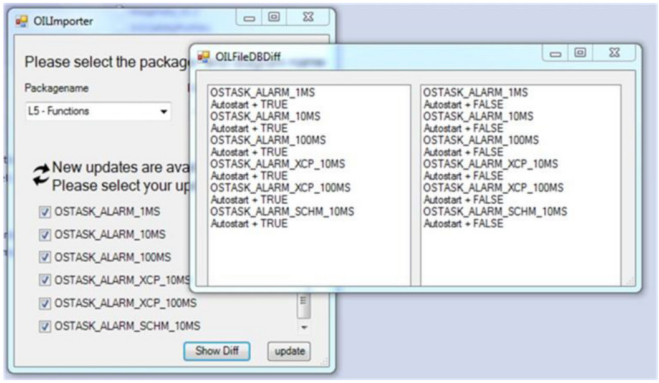


Figure 3. Screenshot of the importer functionalities and the selective artifact update feature.

APPLICATION OF THE PROPOSED APPROACH

This section demonstrates the benefits of the introduced approach. By using our approach for the definition of the software architecture, this work-package is included in the system development tool in an AUTOSAR-aligned representation. Usually this work would have been done by a software system architect within the software development tool (such as Matlab/Simulink). This tool change does not affect the work of the software system architect, due to the AUTOSAR-aligned representation [18], but offers a significant benefit for development of safety-critical software in terms of traceability and replicability of design decisions.

With the additional improvements presented in this paper, the extra facility of mapping SW tasks to dedicated ECU cores and their required resources enables the possibility to unambiguously visualize dependencies and analyze scheduling variants in early development phases. Furthermore, safety-related software artifacts can be explicitly highlighted and dependencies linked in a graphical way.

To provide a comparison of the improvements of our approach, we selected a simplified multi-core use-case solely consisting of tasks, alarms, counter, OS, CPU and application modes. Other OIL objects have been omitted for reasons of the variable multiplicity of these objects (such as resources of a task). The use-case consists of the OIL objects expressed in Table 1.

Table 1. Figures of modeling artifacts representing the evaluation use-case.

OIL Object	Number of elements	Configurable Attributes
CPU	4	2
OS	1	15
APPMODE	2	1
TASKS	6	9
COUNTER	1	5
ALARMS	6	6

This results in a total count of 20 OIL objects and 46 relations between the elements. This small example already indicates that the relations between the elements quickly sum up to a large amount difficult to manage. To overcome this issue, the model-based development approach offers the possibility to hide specific relations. It might be argued that this approach also does not reduce the workload or speed up the generation of OIL files significantly, due to high number of relations that need to be established. However the approach, similar to other examined tools, provides guidance to minimize configuration failures. Additionally, it supports round-trip engineering features, which split workloads among different development phases and thus simplifies reuse. Table 2 compares the proposed approach with other tools presented in the introduction section of this paper and discusses several improvement indicators.

The labels for categorizations are:

- \+ supported or positive effects
- \- not supported or negative effects
- **o** possible or no effects

To specify the advantages mentioned in table 2 in more detail, the following paragraphs take a closer look at the individual approach improvement indicators.

OIL syntax and semantic checks are not supported with the manual approach. The other approaches are capable of supporting these checks and thus improve correctness and completeness of the specific configuration.

Reuse is one key feature solely supported by our approach due to the fact that the OS configuration profile has been integrated in the existing framework and the SW architecture profile described in previous section of this document. This allows development of software architectures fully in line with the AUTOSAR approach and also inherits the ability of adding information regarding safety considerations and basic OS configurations, which simplifies proven-in-use argumentation.

As already mentioned, significant time savings could not be achieved by any of the investigated approaches. This is mainly based on the simple human readable structure of OIL files and the fact, that developers make use of text templates for the manual approach.

Table 2. Comparison of the approach with other common OIL file generation approaches based on feature support.

Approach Improvement Indicators	Proposed Approach	Visual DSL Approach	SW Development Level Configurators	Manual Approach
OIL syntax and semantic checks	+	+	+	-
Reuse	+	o	o	o
Speed-up	o	o	o	o
Distribution of configuration activities	+	o	o	-
Automated configuration from available information	+	+	-	-
Additional constraints	+	-	-	o
Consistency, correctness, and completeness checks	+	o	o	-
Round-trip engineering support and bi-directional update features	+	+	-	-
Traceability of decision making process	+	-	-	-
Multi-core systems	+	o	o	o

However, the automated configuration ability, which is able to generate OS configurations from previously available information and is supported by our approach and the visual DSL approach, also indicates a positive speed-up effect in the case of safety case evidence creation in terms of safety-critical system development.

Nevertheless, in the case of distributed configuration activities (e.g. rough task allocation due to safety constraints done by the system/ safety engineers and detailed OS configuration done by the software integrator), our approach enables easier cooperation and reduces cooperation ramp up times.

Another key feature of our approach, adding additional constraints to the configuration (e.g. ASIL, separated core allocation constraints for independence features, combination constraints for prevention of shared resources), is also possible with the manual approach, but requires in this case additional effort to ensure correct syntax and consistency of information. Nevertheless, these features, among the OS configurations, can be automatically kept consistent solely by our approach.

In terms of safety-critical development and reuse, the presented approach supports crucial additional features such as round-trip engineering by tool-supported information transfer between separated tools and links to support safety-relevant information. Furthermore, the approach eliminates the need for manual generation of OIL files without adequate syntax and semantic support and ensures reproducibility and traceability arguments.

In addition, our approach also enables automated support for multi-core systems and multiple OS instance configurations, which has not been supported by any of the other approaches so far. Visual DSL and SW development level configurator approaches may also support multi-core system and multiple OS instance configuration, but our approach and its ability to take additional constraints into account (mentioned earlier in this section) additionally improves safety-related multi-core development and supports the certification of such multi-core systems (such as traceability of task allocation and safety constraint consistency).

Nevertheless, future (safety-) critical multi-core systems will require the utilization of benefits of the various approaches and usage of the manual based approach will become more and more infeasible due to the number of constraints to comply with.

CONCLUSIONS

An important challenge for the development of safety-critical real-time automotive systems is to ensure the consistency of the safety relevant artifacts (e.g., safety concepts, requirements and configurations) 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 a safety engineering framework and software engineering tools, based on domain standard OSEK. The implemented tool extension transfers artifacts from system development tools to software development frameworks for RTOS configuration, thereby creating traceable links across tool boundaries, and relying on standardized OSEK OIL 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 CPU configuration, as well as a reduction of cumbersome and error-prone manual work along the system development path. Further improvements of the approach include progress in terms of reproducibility and traceability of safety-critical arguments, configurations for software development and support of multi-core system development.

REFERENCES

1. Ebert, C. and Jones, C., “Embedded Software: Facts, Figures, and Future”, Computer, vol.42, no. 4, pp. 42-52, April 2009, doi:10.1109/MC.2009.118.
2. Hilbrich, R., Reinier van Kampenhout, J. and Goltz, H., “Modellbasierte Generierung statischer Schedules fuersicherheitskritische, eingebettete Systeme mit Multicore-Prozessoren und harten Echtzeitanforderungen,” Herausforderungen durch Echtzeitbetrieb Informatik aktuell, 2012, doi:10.1007/978-3-642-24658-6_4.
3. The International Organization for Standardization (ISO), “Road Vehicles Functional Safety Part 1-10,” ISO 26262, 2011.

4. Zahir, A., "OIL-OSEK Implementation Language," OSEK/VDX Open Systems in Automotive Networks (Ref. No. 1998/523), IEE Seminar, 1998, doi:10.1049/ic:19981079.
5. AUTOSAR Development Cooperation. "AUTOSAR AUTomotive Open System Architecture," http://www.autosar.org, 2014.
6. Zhao, M., Wu, Z., Yang, G., Wang, L. and Chen, W., "SmartOSEK: a real-time operating system for automotive electronics," Proceedings of the First international conference on Embedded Software and Systems (ICESS'04), 2004, doi:10.1007/11535409_63.
7. Köster, L., Thomsen, T., and Stracke, R., "Connecting Simulink to OSEK: Automatic Code Generation for Real-Time Operating Systems with TargetLink," SAE Technical Paper 2001-01-0024, 2001, doi:10.4271/2001-01-0024.
8. Kim, J., Lee, J., Son, J., Kwon, K. and Kim, G., "Lightweight AUTOSAR Software Platform for Automotive," IEEE International Conference on Consumer Electronics, 2012, doi:10.1109/ICCE.2012.6161881.
9. Yang, G., Zhao, M., Wang, L. and Wu, Z., "Model-based Design and Verification of Automotive Electronics Compliant with OSEK/VDX," Proceedings of the Second International Conference on Embedded Software and Systems (ICESS '05),2005.
10. Gu, Z., Wang, S. and Shin, K., "Issues in Mapping from UML Real-Time Profile to OSEK," Proc SVERTS: Workshop on Specification and Validation of UML models for Real Time and Embedded Systems, Oct. 2003.
11. Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S. and Ratiu, D., "Seamless Model-based Development: From Isolated Tool to Integrated Model Engineering Environments," Proceedings of the IEEE Volume 98 Issue 4, 2010, doi:10.1109/JPROC.2009.2037771.
12. Quadri, I. and Sadovykh, A., "MADES: A SysML/MARTE high level methodology for real-time and embedded systems," Reconfigurable Communication-centric Systems-on Chip, 2012, doi:10.1109/ReCoSoC.2012.6322882.
13. Holtmann, J., Meyer, J. and Meyer, M., "A Seamless Model-Based Development Process for Automotive Systems," Lecture Notes in Informatics, 2011, ISBN 978-3-88579-278-9.
14. Giese, H., Hildebrandt, S. and Neumann, S., "Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent," Lecture Notes in Computer Science 5765, 2010, ISBN 978-3-642-17321-9, doi:10.1007/978-3-642-17322-6_24.
15. Fabbrini, F., Fusani, M., Lami, G. and Sivera, E., "Software Engineering in the European Automotive Industry: Achievements and Challenges," COMPSAC, 2008, 2013 IEEE37th Annual Computer Software and Applications Conference, doi:10.1109/COMPSAC.2008.140.
16. Mader, R.,, "Computer-Aided Model-Based Safety Engineering of Automotive Systems," PhD thesis, Graz University of Technology, 2012.
17. Rajan, A. and Wahl, T., "CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems," Springer Wien, Rev, ISBN 978-3-7091-1386-8, doi:10.1007/978-3-7091-1387-5.
18. Macher, G., Armengaud, E., and Kreiner, C., "Bridging Automotive Systems, Safety and Software Engineering by a Seamless Tool Chain," Proceedings European Congress Embedded Real Time Software and Systems, 2014.
19. Macher, G., Armengaud, E. and Kreiner, C., "Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures," Lecture Notes in Informatics, 2014.

CONTACT INFORMATION

Georg Macher
AVL List GmbH
Powertrain Engineering - Research & Development
Graz University of Technology
Institute for Technical Informatics
Tel.: +43 316 787 2974
georg.macher@avl.com
http://www.avl.com
http://iti.tugraz.at/

Muesluem Atas
AVL List GmbH
Powertrain Engineering - System Development Engineer
Tel.: +43 316 873 7434
muesluem.atas@avl.com
http://www.avl.com

Eric Armengaud
AVL List GmbH
Powertrain Engineering - Research & Development
Tel.: +43 316 787 6945
eric.armengaud@avl.com
http://www.avl.com

Christian Kreiner
Graz University of Technology
Institute for Technical Informatics
Tel.: +43 316 873 6408
christan.kreiner@tugraz.at
http://iti.tugraz.at/

ACKNOWLEDGMENTS

This work is partially supported by the INCOBAT and the MEMCONS projects.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 608988 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.

SAE INTERNATIONAL

A Versatile Approach for an ISO26262 Compliant Hardware-Software Interface Definition with Model-Based Development

2015-01-0148
Published 04/14/2015

Georg Macher and Harald Sporer
Graz University of Technology

Eric Armengaud
AVL LIST GmbH

Christian Kreiner
Graz University of Technology

CITATION: Macher, G., Sporer, H., Armengaud, E., and Kreiner, C., "A Versatile Approach for an ISO26262 Compliant Hardware-Software Interface Definition with Model-Based Development," SAE Technical Paper 2015-01-0148, 2015, doi:10.4271/2015-01-0148.

Abstract

Increasing demands for safety, security, and certifiability of embedded automotive systems require additional development effort to generate the required evidences that the developed system can be trusted for the application and environment it is intended for.

Safety standards such as ISO 26262 for road vehicles have been established to provide guidance during the development of safety-critical systems. The challenge in this context is to provide evidence of consistency, correctness, and completeness of system specifications over different work-products. One of these required work-products is the hardware-software interface (HSI) definition. This work-product is especially important since it defines the interfaces between different technologies. Model-based development (MBD) is a promising approach to support the description of the system under development in a more structured way, thus improving resulting consistency.

Therefore, this paper presents a tool approach for an ISO 26262 aligned hardware-software interface definition. More specifically, 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) bidirectionally. The approach is capable of defining an ISO 26262 compliant HSI definition and enables automatic derivation of basic software configurations according to the HSI definition. This simplifies concurrent development of software and hardware across domain and company borders.

Introduction

Automotive OEMs are investing large sums in the development of (hybrid) electrified vehicles and networked automotive systems (such as Car2x systems). Future aims concerning autonomous driving and the currently ongoing replacement of traditional mechanical systems with modern embedded systems lead to the significantly increasing complexity of the embedded control systems.

Premium cars in 2009 utilized more than 90 electronic control units (ECU) implementing close to 1 Gigabyte software code. For 2018, 30% of the overall vehicle costs are predicted to stem from vehicle electronics [1].

At the same time, the higher degree of integration and the safety-criticality of the control application raise new challenges. Evidence of correctness of the different applications, possibly running on the same computing platform, has to be guaranteed. In parallel, new computing architectures with services integrated in hardware require new software architectures and safety concepts. On one hand, the development of such systems has to face many cost challenges and is required to support the demands of time-to-market (first time right). On the other hand, increasing demands for safety, security, and certifiability require additional development efforts.

Safety standards such as ISO 26262 [2] for electrical and electronic 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 define a long list of work-products to be generated [3]. An important challenge in this context is to provide evidence of consistency throughout the entire product development cycle among the different work-products.

One of these required work-products is the hardware-software interface (HSI) definition. The HSI specifies the hardware and software interactions in consistency with the technical safety concept and must include the hardware components that are controlled by software and support the software execution. The HSI is specified

during the system design phase and further refined during hardware and software development phases [2]. This ISO 26262 statement highlights the importance and essentiality of this work product.

The HSI document is the last development artifact of the system development and the starting point for parallel development of hardware and software. HSI definition requires mutual domain knowledge of hardware and software and is to be a work product of a collective workshop of hardware, software, and system experts. Furthermore, HSI is used to agree on topics relevant to both, hardware and software development, and acts as the linkage between different levels of development.

Insufficient definition of the HSI can cause several additional iteration cycles and communication issues between development teams.

To handle these issues, model-based development supports the description of the system under development in a more structured way, enables different views for different stakeholders, different levels of abstraction, and a central source of information. Nevertheless, seamless model-based solutions have not been achieved so far, due to problems arising from inadequate tool-chains (e.g. redundancy, inconsistency and lack of automation). This hampers MBD approaches in tapping their full potential.

Therefore, this paper presents a tool approach for ISO 26262 aligned hardware-software interface definition. More specifically, 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) 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 existing model-based systems and safety tool chain on which this work is based is presented in the third section. The fourth section provides a description of the proposed enhancement for HSI definition. Section five evaluates the presented approach with an automotive use case. Finally, the last section concludes this work with an overview of what has been achieved.

State-of-the-Art HSI Definition

This section briefly describes the current state-of-the-art approaches for defining hardware-software interfaces. Although the topic is of high importance for the automotive domain, only few recent publications exist.

In contrast to this, Hardware-Software Interface co-design has a long history in development of System-on-Chip (SoC) systems. Already in 2005 Jerraya et. al [4] postulated that co-design of HW and SW interfaces will fundamentally improve the SoC design process and increase both hardware and software quality of SoC development.

Kecheng and Fei [5] realized a component-based approach to HW/SW interface design of embedded systems with a platform-specific bridge specification language (BSL). Nevertheless, their work addresses modeling system models as SystemC and introduces another abstraction layer for hardware abstraction.

King et. al [6] postulated the problem of defining HW/SW interfaces in early development steps. First, a detailed interface is difficult to specify without detailed knowledge of software and hardware. Second, this specification prevents later migration of interface functionalities and the addition of features.

The reason why HSI development was not the main focus in the automotive industry originates from the fact that automotive hardware and software development significantly differ in cycle times. Furthermore, automotive software is typically separated into several abstraction layers (such as application software, microcontroller abstraction layers, basic functionality drivers). This approach usually hides hardware details and establishes software development teams with specific software focus (e.g., basic software developer, application software engineers, and software integrators).

The AUTOSAR architectural approach [7] explicitly forces such an approach to support hardware independent development of application software modules till a very late development phase and therefore enables parallel working of application software developers, basic software developers, and hardware developers. The intention of the AUTOSAR specifications is to support the exchange and reuse of software, by defining software architectures, interfaces, and exchange formats.

An emerging domain-independent paradigm for interface definition is the contract-based design paradigm. Here the contracts specify the input assumptions of a component and provide a guaranteed output behavior [8]. Such an approach can be used for safety contracts of software components [9], as well as contract-based embedded system development [10].

Another emerging paradigm is system of systems (SoS) engineering [11]. SoS are integrations of heterogeneous systems delivering capabilities and services without exact knowledge of the internal workings of an involved subsystem. Basically all current SW architecture in the automotive domain can be seen as such SoS, due to their distributed development of SW layers. A promising method for definition of SoS architectures lies in interface specification and a quasi-contract-based development. Bryans et. al [11] establish a semi-formal notation to model SoS architectures with SysML. Such a notation of software architectures is also part of our approach, for more details see [12].

Although, these contract-based approaches foster model-based development and traceability of development decisions, they are not simple and easy enough to be used for HSI definition workshops. Because of this reason and the project specific nature of HSI definitions many hardware-software interface definitions are still done within spreadsheet tools or in textual form within a requirement management tool. Although Chen et. al [13] claim that social and text-based communication does not scale for handling future advanced embedded automotive systems.

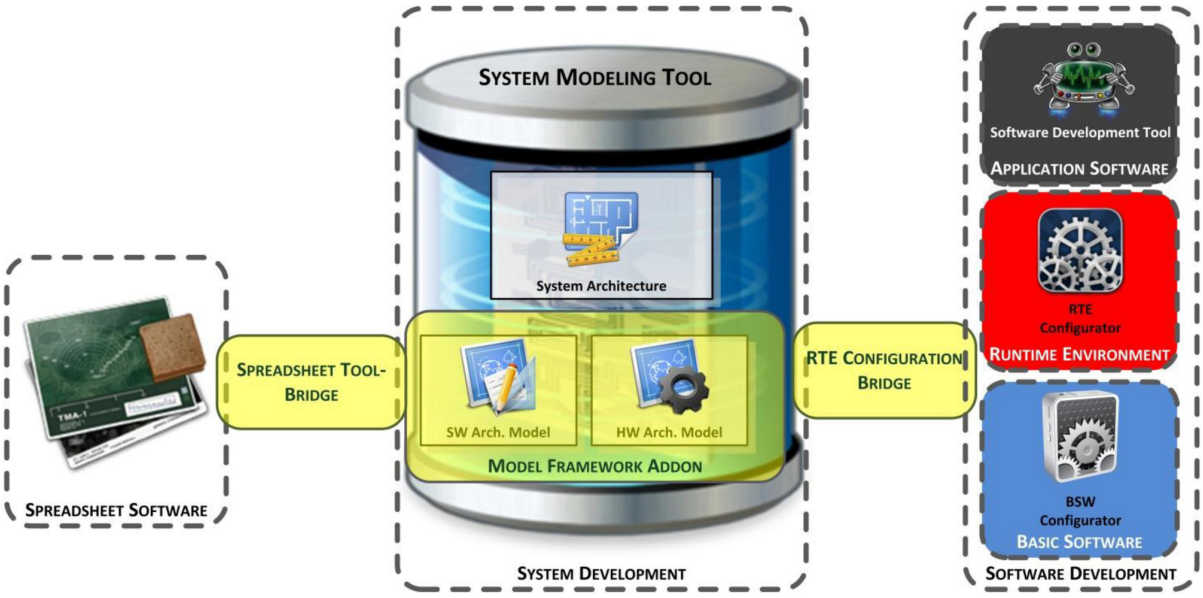


Figure 1. Conceptual overview of the HSI definition approach

Model-Based Development Fundament

This paragraph gives a short overview of the model-based development tool chain in use and the related preliminary work that provides the base for the proposed approach.

The basic concept behind the MBD framework is to have a consistent information repository as a central source of information [14]. This concept allows different engineers to do their job in their specific manner, provides traces between different artifact types, and ensures timeliness of data. Proprietary extensions for the modeling tool ensure seamless and consistent transition of information between the repository and various adequate special-purpose tools (such as software development tools). A UML profile, tailored to the needs of automotive safety engineering, is used to define the model of the system under development.

This enables reorganization from a document-centric development approach to a seamless model-based development approach. An approach, Broy et al. [15] claim to be best to manage large complexity of modern embedded automotive systems. Fabbrini et al. [16] also highlight the importance of model-based development approaches in their overview of software engineering in the European automotive industry.

For a more detailed overview of the model-based development concept in use, see [17].

Hardware-Software Interface Definition Approach

The contribution proposed in this work is a tool approach for ISO 26262 aligned hardware-software interface definition. The approach enables a practicable and intuitive way of engineering HSI definitions in a spreadsheet tool (such as Excel) and transforms them to a reusable and version able representation in the MDB tool (such as Enterprise Architect). With this approach, the spreadsheet document and system model can be bidirectionally aligned via program-specific APIs, which support tool-independence of the approach.

More specifically, our contribution consists of the following parts:

- ***Application SW modeling framework:*** Enhancement of a 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 and modeling of software signals, see figure 1 - modeling framework add-on.
- ***Basic SW and HW module modeling framework:*** Enhancement of a UML profile to describe basic software (BSW) components and HW component interfaces. This is required to ensure consistency of the specification and configuration of BSW components and modeling of the hardware component interfaces, see figure 1 - modeling framework add-on.
- ***HSI definition exporter:*** MBD-tool extension to export the resulting HW/SW interface definitions to spreadsheet tools for further reworking without the need of special tools, see figure 1 - spreadsheet tool bridge.
- ***Spreadsheet importer:*** Importer to integrate HW/SW interfaces defined with a spreadsheet template, see figure 1 - spreadsheet tool bridge.
- ***Spreadsheet template:*** The spreadsheet template defines the representation structure for the input data and is customizable for each project.
- ***SW/SW interface generator:*** Generator to automatically define SW/SW interfaces between application software signals and basic software signals, see figure 1 - RTE configuration bridge.

Application SW Modeling Framework

The first contribution is the development of a specific UML modeling framework enabling software architecture design in AUTOSAR such as 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 the software architecture and ensures proper definition of the communication between the architecture artifacts, including interface specifications (e.g. upper limits, initial values, formulas). 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 2 shows an example of a software module (AUTOSAR Composition) and its signal interface definitions. This integrated definition of system artifacts and software module in one tool furthermore supports the work of safety engineers by adding values, visual labels for safety-relevant software modules, and enables the automatic generation of interfaces to BSW modules. Furthermore the model representation enables constraint-checking features and supports signal traces to HSI definition and requirements.

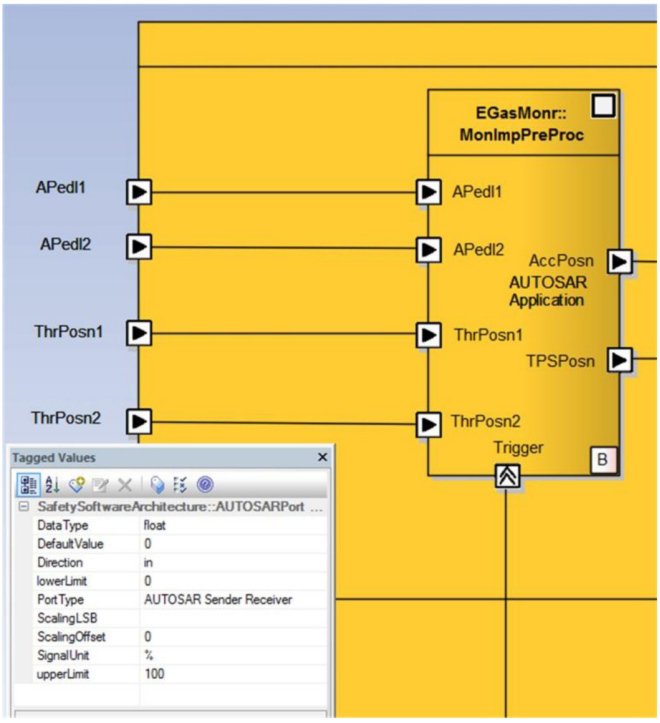


Figure 2. Example of an application software (ASW) component and configurations of a software signal interface.

Basic SW 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. The hardware profile allows a graphical representation of hardware resources (such as ADC, CAN), calculation engines (core), and connected peripherals that interact with the software. This 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. Software signals in BSW modules can be linked to HW port pins via dedicated mappings. This on one hand enables the modeling and mapping of HW specifics and SW signals, see figure 3. On the other hand, this mapping enables traceable links to port pin configurations.

HSI Definition Exporter

The HSI exporter (a MBD-tool extension) establishes an API link to spreadsheet tools and enables the export of modeled HW/SW interfaces to spreadsheet documents. This API link ensures freedom from version dependence of the specific spreadsheet tool. Furthermore, the MBD-tool extension is developed in the form of a dll class library, which provides means for reuse by multiple programs and ensures MDB-tool independence of the exporter.

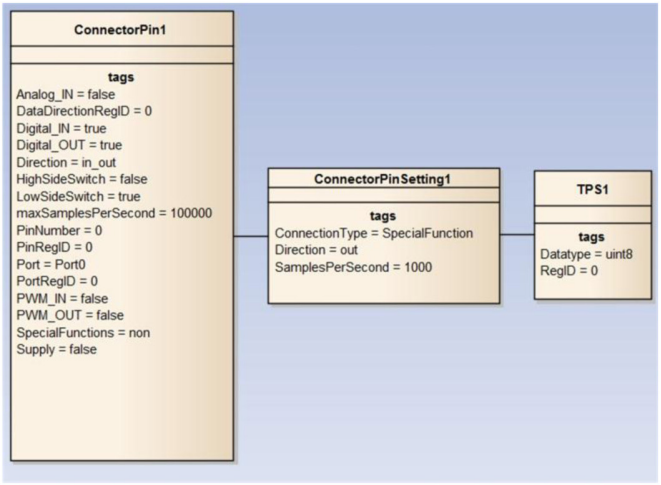


Figure 3. Example of a port pin connector artifact, basic software (BSW) signal, and the corresponding connector pin settings.

Spreadsheet Importer

The MBD-tool import-extension is the corresponding counterpart to the HSI exporter. It is also a dll class library using the spreadsheet tools API and enables the import of information and the selective update of HW/SW interface model artifacts. This enables round-trip engineering of HSI definition within the spreadsheet and MDB tool.

Spreadsheet Template

The spreadsheet template defines the structure of the data representation in a project specific customizable way. This, on one hand, enables a practicable and intuitive way of engineering HSI definitions with spreadsheet tools and transformation to a reusable and version able representation in the MDB tool. On the other hand, this approach unifies the project-dependent process for HSI definitions across the variety of different projects and contributing partners without requiring exactly the same development tools or processes in place. Thirdly, the machine- and human-readable notation of a spreadsheet ensures a cost- and time-saving alternative to usually complex special-purpose tools. Figure 4 depicts the project-independent template structure for HSI definition data preparation.

SW/SW Interface Generator

The last part of the presented approach is the SW/SW interface generator. This dll- based MDB-tool extension generates .c and .h files defining SW/SW interfaces between application software signals and basic software signals from the modeled HSI artifacts. In addition this generation eliminates the need for manual SW/SW interface generation without adequate syntax and semantic support and ensures reproducibility and traceability of these configurations.

HSI definition		<project/customer logo>		<project name>
Sensor/Actuator		throttle sensor 1	throttle sensor 2	throttle actuator 1
Direction	common	in	in	out
Source(CAN/ANA/DIG)		ANA	CAN	DIG
Signaltype (V / % / deg)		°	%	PWM
Signalname		ThrPos1	ThrPos2	ThrActuPWM
signal range lower limit		0	0	10
signal range upper limit		60	100	90
Scaling LSB				
Scaling Offset				
accuracy	SW	0,05	0,5	-
ASIL		ASIL B	ASIL B	ASIL QM
default value		0	0	0
type		float	uint8	uint8
refresh rate	ms	10	10	100
register-type		RAM	RAM	RAM
variable name		v_APS1	s_APS2	ThrActuPWMout
unit		V	V	V
physical range lower limit		0,5	1	0
physical range upper limit		4,5	4	5
cycle time	ms	1	1	100
supply voltage	HW	5	5	-
signal tolerance		0,25	0,5	-
resolution	bit	16	8	8
port		PortA	PortB	PortC
pin		12	1	4

Figure 4. Example of a project-independent spreadsheet template structure for HSI definition.

Application of the HSI Definition Approach

In order to evaluate the approach, an automotive use-case of a central control unit (CCU) for a battery management system (BMS) prototype for (hybrid) electric vehicle has been chosen. Project-specific details have been abstracted for reasons of commercial sensitivity.

Figure 5 shows a generalized representation of the CCU SW architecture. As can be seen from the picture, 10 SW modules on the ASW layer and 7 SW modules on the BSW layer have been defined. Table 1 summarizes the number of interfaces module and artifacts representing the CCU SW architecture.

Table 1. Summary of model artifacts representing the CCU SW architecture, required to model the BMS HSI definition.

Number of ASW module	10
Number of BSW modules	7
Number of ASW module inputs	54
Number of ASW module outputs	32
SW/SW interfaces on ASW layer	48
Amount of ASW/BSW interfaces	19
Amount of HW/SW interfaces	19

As can be seen in table 1, 19 ASW/BSW interfaces (7 input and 12 output interfaces) need to be defined. This definition adds up to more than 30 lines of code (LoC) that can be generated automatically with the presented approach into interface.c and interface.h files.

Key aspects for ISO 26262 aligned HSI definition are correct, consistent, and complete interface descriptions. Furthermore, evidences of correct implementation of these interfaces are required. This can be challenging in case of concurrent HW and SW development and cross-dependencies of asynchronous update intervals. Therefore, the presented approach provides a single source of HSI definitions, available as graphical HSI model and spreadsheet and automatically generates code configurations. Regarding the SW/SW Interfaces on the ASW layer, consistency checks for the 32 output interfaces and 54 input interfaces ensure point-to-point consistency of the signal routing. For 9 definable features per signal, this adds up to 774 definitions, which are automatically checked for consistency with this approach.

The actual HW/SW interface, mapping of BSW signals to HW pins, also consists of 19 interfaces for this specific SW architecture. This mapping includes 23 settings per mapping and can be automatically exported to the spreadsheet tool.

In case of changes of the HSI mapping within the spreadsheet, e.g. due to necessary pin reconfiguration, this information can be kept consistent with the model-representation via the importer functionality. This ensures actuality of development artifacts and simplifies tracing of development decisions. Table 2 sums up the additional features for the BMS use-case supported by the presented approach and provides an overview of the amount of data affected by each iteration step.

Table 2. Additional features provided by the approach and number of presence for the BMS use-case

Automatically extracted settings to Excel	437
Consistency checks of ASW 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 ASW/BSW mappings	33 LoC

Summary

This paper presented a tool approach for ISO 26262 aligned hardware-software interface definition and a brief evaluation of the approach for a BMS use-case.

The approach combines the versatility and intuitiveness of spreadsheet tools and the properties of MDB tools (e.g. different views, levels of abstraction, central source of information, and information reuse) in a bidirectional way. This, on one hand, enables a practicable, tool-independent, and intuitive way of engineering HSI definitions with spreadsheet tools and transforms the generated information to a reusable and version-able representation in the MDB tool.

On the other hand, this approach unifies the project-dependent process for HSI definitions across the variety of different projects and contributing partners without requiring exactly the same development tools or processes in place.

Thirdly, the machine- and human-readable notation of spreadsheets ensures a cost- and time-saving alternative to usually complex special-purpose tools. This facilitates collaboration especially for small budget projects, projects with small and medium-sized companies or research projects, and collaboration with academic partners.

Finally, the approach is capable of defining an ISO 26262 compliant HSI definition and enables automatic derivation of basic software configurations according to the HSI definition.

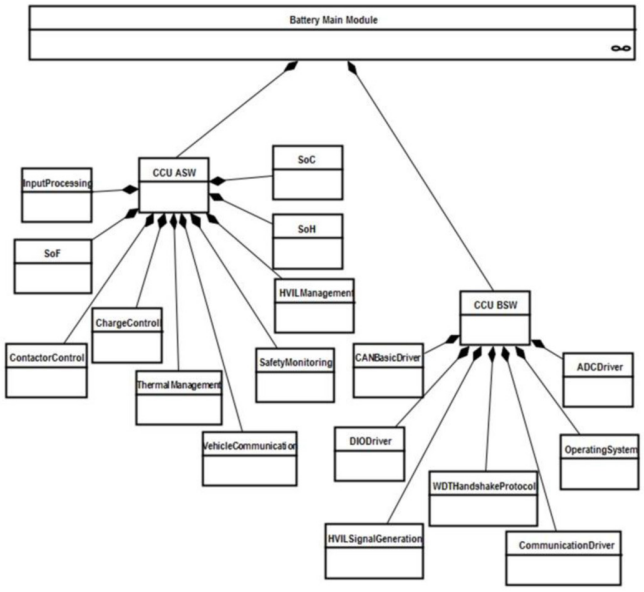


Figure 5. Generalized representation of the BMS software architecture; subdivided into basic software modules and application software modules.

Key aspects for ISO 26262 aligned HSI definition are correct, consistent, and complete interface descriptions. This is in particular crucial in case of concurrent HW and SW development and synchronization of HSI changes and updates. By automatic bidirectional updating of the information resources (model and spreadsheet) and automatic generation of code, the presented approach minimizes manual work efforts and improves traceability of HSI specification to code implementation. This helps to simplify concurrent development of software and hardware across domain, company, and tool borders.

References

1. Riel, A., Bachmann, O., Dussa-Zieger, K., Kreiner, C., Messnarz, R., Nevalainen, R., Sechser, B., smf Tichkiewitch, S. EU Project SafEUr - Competence Requirements for Functional Safety Managers. In EuroSPI Proceedings, volume 301 of Communications in Computer and Information Science, pages 253-265. Springer, 2012.
2. The International Organization for Standardization (ISO), "Road Vehicles Functional Safety Part 1-10," ISO 26262, 2011.
3. Messnarz, R., Kreiner, C., Bachmann, O., Riel, A., Dussa-Zieger, K., Nevalainen, R., and Tichkiewitch, S. Implementing Functional Safety Standards Experiences from the Trials about Required Knowledge and Competencies (SafEUr). In Systems, Software and Services Process Improvement, volume 364 of Communications in Computer and Information Science, pages 323-332. Springer Berlin Heidelberg, 2013.
4. Jerraya, A. and Wolf, W., "Hardware/Software Interface Codesign for Embedded Systems," Computer, vol.38, no. 2, pp. 63-69, February 2005, doi:10.1109/MC.2005.61.
5. Kecheng, H. and Xie, F., "Componentizing Hardware/Software Interface Design," DATE 2009, ISBN 978-3-9810801-5-5
6. King, M., Nirav, D. and Arvind, "Automatic Generation of Hardware/Software Interfaces," ASPLOS 2012: Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, doi:10.1145/2150976.2151011.
7. AUTOSAR Development Cooperation. "AUTOSAR AUTomotive Open System Architecture," http://www.autosar.org, 2014.
8. Cimatti, A. and Tonetta, S., "A Property-Based Proof System for Contract-Based Design," Proc. 36th EUROMICRO Conference on Software Engineering and Advanced Applications, 2012.
9. Johansson, R., "Safety Contract Based Design of Software Components", ISSREW, 2013, 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2013, doi:10.1109/ISSREW.2013.6688922.
10. Damm, W., Hungar, H., Josko, B., Peikenkamp, T. and Stierand, I., "Using Contract-Based Component Specifications for Virtual Integration Testing and Architecture Design," DATE, 2011, Design, Automation & Test in Europe Conference & Exhibition, doi:10.1109/DATE.2011.5763167.
11. Bryans, J., Payne, R., Holt, J. and Perry, S., "Semi-Formal and Formal Interface Specification for System of Systems Architecture," Systems Conferecne (SysCon), 2013, ISBN 978-1-4673-3107-4, doi:10.1109/SysCon.2013.6549946.
12. Macher, G., Armengaud, E. and Kreiner, C., "Automated Generation of AUTOSAR Description File for Safety-Critical Software Architectures," Lecture Notes in Informatics, 2014.
13. Chen, D., Johansson, R., Loenn, H., Papadopoulos, Y., Sandberg, A., Toerner, F. and Toerngren, M., "Modelling Support for Design of Safety-Critical Automotive Embedded Systems," SAFECOMP 2008, 2008.
14. Rajan, A. and Wahl, T., "CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems," Springer Wien, Rev, ISBN 978-3-7091-1386-8, doi:10.1007/978-3-7091-1387-5.
15. Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S. and Ratiu, D., "Seamless Model-based Development: From Isolated Tool to Integrated Model Engineering Environments," Proceedings of the IEEE Volume 98 Issue 4, 2010, doi:10.1109/JPROC.2009.2037771.
16. Fabbrini, F., Fusani, M., Lami, G. and Sivera, E., "Software Engineering in the European Automotive Industry: Achievements and Challenges," COMPSAC, 2008, 2013 IEEE 37th Annual Computer Software and Applications Conference, doi:10.1109/COMPSAC.2008.140.
17. Macher, G., Armengaud, E., and Kreiner, C., "Bridging Automotive Systems, Safety and Software Engineering by a Seamless Tool Chain," Proceedings European Congress Embedded Real Time Software and Systems, 2014.

Contact Information

Georg Macher
AVL List GmbH
Powertrain Engineering - Research & Development
Graz University of Technology
Institute for Technical Informatics
Tel.: +43 316 787 2974
georg.macher@avl.com
http://www.avl.com
http://iti.tugraz.at/

Harald Sporer
Graz University of Technology
Institute for Technical Informatics
Tel.: +43 316 873 6409
sporer@tugraz.at
http://iti.tugraz.at/

Eric Armengaud
AVL List GmbH
Powertrain Engineering - Research & Development
Tel.: +43 316 787 6945
eric.armengaud@avl.com
http://www.avl.com

Christian Kreiner
Graz University of Technology
Institute for Technical Informatics
Tel.: +43 316 873 6408
christan.kreiner@tugraz.at
http://iti.tugraz.at/

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).

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.

Definitions/Abbreviations

ECU - Engine control unit

HSI - Hardware-software interface

MDB - Model-based development

AUTOSAR - Automotive open system architecture

SoC - System on chip

SW - Software

ASW - Application software

BSW - Basic software

HW - Hardware

VFB - Virtual function bus

BMS - Battery management system

CCU - Central Control Unit

LoC - Lines of Code

The Engineering Meetings Board has approved this paper for publication. It has successfully completed SAE's peer review process under the supervision of the session organizer. The process requires a minimum of three (3) reviews by industry experts.

ISSN 0148-7191

http://papers.sae.org/2015-01-0148

Bibliography

[1] Military Standard Procedures for Performing a Failure Mode, Effects and Criticality Analysis, November 1980.

[2] Military Standard Reliabilty Program for Systems and Equipment Development and Production, September 1980.

[3] Military Handbook Reliability Prediction of Electronic Equipment, December 1991.

[4] Department of Defense Handbook for Reliability Test Methods, Plans, and Environments for Engineering, Development Qualification, and Production, April 1996.

[5] Military Handbook Electronic Reliability Design Handbook, October 1998.

[6] Railway applications - The specification and demonstration of reliability, availability, maintainability and safety (RAMS), March 2000.

[7] Albert Albers and Christian Zingel. Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML. In Michael Abramovici and Rainer Stark, editors, *Smart Product Engineering*, Lecture Notes in Production Engineering, pages 83–92. Springer Berlin Heidelberg, 2013.

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

[9] AUTOSAR Development Cooperation. Technical Safety Concept Status Report. Technical Report Document Version: 1.1.0, Revision 2, AUTOSAR development cooperation, October 2010.

[10] AUTOSAR Development Cooperation. Guide to Multi-Core Systems. online, 2013.

[11] AUTOSAR Development Cooperation. Virtual Functional Bus. online, 2013.

[12] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Dependability and its Threats - A Taxonomy. In Rene Jacquart, editor, *IFIP Congress Topical Sessions*, pages 91–120. Kluwer, 2004.

[13] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[14] Volodymyr Bendiuga. Multi-Core Pattern. Master's thesis, School of Innovation, Design and Engineering Malardalen University Sweden, December 2012.

[15] 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.

[16] Kurt Boehringer and Markus Kroh. Funktionale Sicherheit in der Praxis, July 2013.

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

[18] Marc Born, John Favaro, and Olaf Kath. Application of ISO DIS 26262 in Practice. *CARS 2010*, 2010.

[19] Baert Brecht and Steven Luys. Application of multi-core CPUs in a safety-critical context. Technical report, Barco Defense and Aerospace, March 2011.

[20] 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.

[21] D. Chen, R. Johansson, H. Loenn, H. Blom, M. Walker, Y. Papadopoulos, S. Torchiaro, F. Tagliabo, and A. Sandberg. Integrated safety and architecture modeling for automotive embedded systems. In *Elektrotechnik & Informationstechnik*, volume 128/6, pages 196 – 202, February 2011.

[22] Philippe Cuenot, Patrick Frey, Rolf Johansson, Henrik Lönn, Mark-Oliver Reiser, David Servat, Ramin Tavakoli Kolagari, and DeJiu Chen. Developing Automotive Products Using the EAST-ADL2 : an AUTOSAR Compliant Architecture Description Language. *Ingenieurs de l'Automobile (Automobile Engineers)*, (793), 2008. QC 20120510.

[23] Christof Ebert. Functional Safety Industry Best Practices for Introducing and Using ISO 26262. In *SAE Technical Paper*. SAE International, 04 2013.

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

[25] European Automobile Manufacturers Association. The Automobile Industry Pocket Guide. Technical report, European Automobile Manufacturers Association, 2015.

[26] European Organization for Civil Aviation Equipment (EUROCAE WG-72) and Radio Technical Commission for Aeronautics (RTCA SC-216). Airworthiness security process specification, ED-202, 2010.

[27] Michael J. Flynn. Some Computer Organizations and Their Effectiveness. *IEEE Trans. Comput.*, 21(9):948–960, September 1972.

[28] Ilir Gashi, Andrey Povyakalo, Lorenzo Strigini, Martin Matschnig, Thomas Hinterstoisser, and Bernhard Fischer. Diversity for Safety and Security in Embedded Systems. In *International Conference on Dependable Systems and Networks*, volume 26, 06 2014.

[29] Vera Gebhardt, Gerhard Rieger, Juergen Mottok, and Christian Giesselbach. *Funktionale Sicherheit nach ISO 262626 - Ein Praxisleitfaden zur Umsetzung*, volume 1. Auflage. dpunkt.verlag, 2013.

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

[31] Jongtaek Han, Jin Seo Park, Michael Deubzer, Jens Harnisch, and Patrick Leteinturier. Efficient Multi-Core Software Design Space Exploration for Hybrid Control Unit Integration. In *SAE Technical Paper*. SAE International, 04 2014.

[32] Hansen, Finn Overgaard and i Århus, Ingeniørhøjskolen. SysML–a modeling language for systems engineering. 2010.

[33] Robert Hilbrich, J. Reinier van Kampenhout, and Hans-Joachim Goltz. Modellbasierte Generierung statischer Schedules fuer sicherheitskritische, eingebettete Systeme mit Multicore-Prozessoren und harten Echtzeitanforderungen. *Informatik aktuell*, pages 29 – 38, 2012.

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

[35] International Electrotechnical Commission. Reliability data handbook - Universal model for reliability prediction of electronics components, PCBs and equipment. Technical Report IEC TR 62380, International Electrotechnical Commission, 2004.

[36] International Organization for Standardization. ISO 13053 Quantitative methods in process improvment – Six Sigma, 2011.

[37] ISO - International Organization for Standardization. IEC 61508 Functional safety of electrical/ electronic / programmable electronic safety-related systems.

[38] ISO - International Organization for Standardization. IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA) , 2006.

[39] ISO - International Organization for Standardization. IEC 61025 Fault tree analysis (FTA) , December 2006.

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

[41] Daniel Kaestner, Marc Schlickling, Markus Pister, Christoph Cullmann, Gernot Gebhard, Reinhold Heckmann, and Christian Ferdinand. Meeting Real-Time Requirements with Multi-Core Processors. In *SAFECOMP 2012 Workshops, LNCS 7613*, pages 117 – 131, 2012.

[42] Markus Krallinger. CDL - An Extensible Framework to Create Constraints in Model-Based Development. Master's thesis, Graz University of Technology - Institute for Technical Informatics, 2012.

[43] Daehyun Kum, Gwang-Min Park, Seonghun Lee, and Wooyoung Jung. AUTOSAR Migration from Existing Automotive Software. In *International Conference on Control, Automation and Systems*, COEX, Seoul, Korea, 2008. DGIST.

[44] Joo-Chul Lee and Tae-Man Han. ECU Configuration Framework based on AUTOSAR ECU Configuration Metamodel. 2009.

[45] Peter Liggesmeyer and Mario Trapp. Trends in embedded software engineering. *IEEE Software*, 26(3):19–25, May 2009.

[46] Tomislav Lovric, Manuel Schneider-Scheyer, and Samir Sarkic. SysML as Backbone for Engineering and Safety - Practical Experience with TRW Braking ECU. In *SAE Technical Paper*. SAE International, 04 2014.

[47] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner. SAHARA: A security-aware hazard and risk analysis method. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, pages 621–624, March 2015.

[48] Georg Macher, Eric Armengaud, Eugen Brenner, and Christian Kreiner. An Automotive Engineering Model to Improve the Architectural Design Interchange Continuity. In *under review at SAE Technical Paper*, pages 001 –007, 2016.

[49] Georg Macher, Andrea Hoeller, Harald Sporer, Eric Armengaud, and Christian Kreiner. Service Deterioration Analysis (SDA): An Early Development Phase Reliability Analysis Method. In *In Review at 45th Annual International Conference on Dependable Systems and Networks (DSN) - RADIANCE Workshop*, 2015.

[50] Microsoft Corporation. The stride threat model, 2005.

[51] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114 – 117, April 1965.

[52] Bryon Moyer. *Real World Multicore Embedded Systems*. Expert guide. Newnes, Newton, MA, USA, 1st edition, 2013.

[53] OSEK/VDX Steering Committee. OSEK/VDX System Generation OIL: OSEK Implementation Language. http://portal.osek-vdx.org/files/pdf/specs/oil25.pdf, 2004.

[54] Roberto Passerone, Imene Ben Hafaiedh, Susanne Graf, Albert Benveniste, Daniela Cancila, Arnaud Cuccuru, Sebastien Gerard, Francois Terrier, Werner Damm, Alberto Ferrari, Leonardo Mangeruca, Bernhard Josko, Thomas Peikenkamp, and Alberto Sangiovanni-Vincentelli. Metamodels in Europe: Languages, Tools, and Applications. *IEEE Design and Test of Computers*, 26(3):38–53, 2009.

[55] Alexander Pretschner, Manfred Broy, Ingolf H. Kruger, and Thomas 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.

[56] Imran Rafiq Quadri and Andrey Sadovykh. MADES: A SysML/MARTE high level methodology for real-time and embedded systems, 2011.

[57] F. Reichenbach and A. Wold. Multi-core Technology – Next Evolution Step in Safety Critical Systems for Industrial Applications? In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 339–346, Sept 2010.

[58] Kai Richter and Simon Schliecker. Sicher auf Multi-Core umsteigen. *HANSER automotive*, 10:38–42, 10 2013.

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

[60] Michael Roth and Peter Liggesmeyer. Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees. In *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[61] SAE International. Guidelines and Mehtods for Conductiong the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

[62] SAE International. Guidelines for Development of Civil Aircraft and Systems, 2010.

[63] SafEUr Training Material Committee. ECQA Certified Functional Safety Manager Training Material. training dossier, April 2013.

[64] Alberto Sangiovanni-Vincentelli and Marco Di Natale. Embedded System Design for Automotive Applications. *Computer*, 40(10):42–51, 2007.

[65] Oliver Scheickl, Christoph Ainhauser, and Peter Gliwa. Tool Support for Seamless System Development based on AUTOSAR Timing Extensions, 2011.

[66] K. Schmidt, P. Troeger, H. Kroll, and T. Buenger. Adapted Development Process for Security in Networked Automotive Systems. *SAE 2014 World Congress & Exhibition Proceedings*, (SAE 2014-01-0334):516 – 526, 2014.

[67] Christoph Schmittner, Thomas Gruber, Peter Puschner, and Erwin Schoitsch. Security Application of Failure Mode and Effect Analysis (FMEA). In Andrea Bondavalli and Felicita Di Giandomenico, editors, *Computer Safety, Reliability, and Security*, volume 8666 of *Lecture Notes in Computer Science*, pages 310–325. Springer International Publishing, 2014.

[68] Rolf Schneider, Wolfgang Brandstaetter, Marc Born, Olaf Kath, Tobias Wenzel, Rafael Zalman, and Mayer Johann. Safety Element out of Context - A Practical Approach. In *SAE International Technical Papers*, number 2012-01-0033, April 2012.

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

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

[71] Special Committee 205 of RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011.

[72] Max Steiner and Peter Liggesmeyer. Combination of Safety and Security Analysis - Finding Security Problems That Threaten The Safety of a System. In *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.

[73] Geoffrey Tennant. *Six sigma: SPC and TQM in manufacturing and services*. Gower Publishing Ltd, 2001.

[74] The Common Criteria Recognition Agreement Members. Common Criteria for Information Technology Security Evaluation. http://www.commoncriteriaportal.org/, 2014.

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

[76] Judith Thyssen, Daniel Ratiu, Wolfgang Schwitzer, Er Harhurin, Martin Feilkas, Technische Universitaet Muenchen, and Eike Thaden. A system for seamless abstraction layers for model-based development of embedded software. In *Software Engineering Workshops*, pages 137–148, 2010.

[77] Stefan Voget. AUTOSAR and the Automotive Tool Chain. In *DATE10*, 2010.

[78] D. Ward, I. Ibara, and A. Ruddle. Threat Analysis and Risk Assessment in Automotive Cyber Security. *SAE 2013 World Congress & Exhibition Proceedings*, pages 507–513, 2013.

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