Functional Safety and Development Process Capability for Intelligent Transportation Systems

Ansgar M. Meroth, Frank Tränkle, Bastian F. Richter, and Marko Wagner
Heilbronn University, Heilbronn, Germany
E-mails: ansgar.meroth@hs-heilbronn.de, frank.traenkle@hs-heilbronn.de, brichter@stud.hs-heilbronn.de, and marco.wagner@hs-heilbronn.de

Michael Neher and Jochen Lüling
Gigatronik Stuttgart GmbH, Stuttgart, Germany
E-mail: Jochen.Lueling@gigatronik.com

***Abstract*—**Intelligent transportation systems (ITS) require the extension of the systems engineering approach to a higher system level. Up to now, vehicles could be considered as closed systems while in future they have to be seen as components of an open system with ad hoc interfaces. Since ITS influence the series development and production of single vehicles, standards and procedures of process quality, systems engineering and functional safety apply for this higher level as well. The paper discusses the impact of those standards to the design of ITS and the role of model-based software and service oriented architectures (SOA) as means for process quality and functional safety assurance. It emphasizes on the mutual synergies between process capability standards, functional safety, model-based development and design for variability. Finally, it shows an approach for the design of ad hoc and distributed systems proposed by the authors.

© ISTOCKPHOTO.COM/RYCCIO

I. Introduction

Vehicles are the most complex systems that are available for a mass market especially since a variety of advanced driver assistance systems (ADAS) took place, that support or substitute the tasks that were traditionally in the responsibility of the driver. Commonly, the tasks are classified as *primary*, concerning the stabilization of the vehicle, navigation, and route keeping, *secondary*, concerning the operation of the vehicle devices (e.g. headlights, wipers etc.), and *tertiary* concerning the operation of information, entertainment

Digital Object Identifier 10.1109/MITS.2015.2474955
Date of publication: 23 October 2015

This paper has partially been presented during the ITSC 2014 [2].

1939-1390/15©2015IEEE

and comfort systems [1]. Up to now, the vehicle could be seen as a self-sufficient closed system with only little information and communication interfaces to the external world, at least for the primary and secondary driving task. In future, ADAS will more and more rely on communication between cars and between the car and the infrastructure (C2X). Hence, the car itself will be a component in a higher system level, requiring higher standards of functional safety and integrity on a new quality of system (see Figure 1) [2]. Furthermore, cars, retrofit equipment such as smart handheld and other commercial-off-the-shelf (COTS) devices together with infrastructure based components and services will establish ad-hoc and self-adaptive communication relations and thus create new services depending on location and situation requiring increased flexibility and fault tolerance of the system design. This is valid especially when the primary driving tasks are substituted by heavily automated vehicles. Those systems set up the requirement to allow changes of the software and system architecture at runtime. We call these highly distributed, dynamically changing systems Distributed Driver Assistance Systems (DDAS) [3].

Coping with the demands of reliability, integrity and safety of those complex systems requires not only a new understanding of the technical and legal issues of connected, automated cars, but also impacts the development processes on lower system levels. A thorough comprehension of the systems engineering approach is a prerequisite for the design of those processes.

An exemplary application showing the characteristics of DDAS are truck and trailer assistance systems [3]. Within this class of systems, components and software modules are distributed over both the towing vehicle and the trailer. This configuration adds the need for self-configuration features to handle the changes of the system at runtime, triggered for example by a hook-up or uncouple event. It also requires a high degree of self-description of the entities to ensure the setup of a properly functioning and efficient driver assistance system. All these properties have to be added to the system's components at development time and thereby require new approaches in automotive development procedures. Consequently, the purpose of this paper is to show how the original approaches of the authors, together with state of the art design methods and standardized development processes merge together. Therefor the paper first discusses major challenges of the systems engineering process and state of the art systems engineering principles. It then discusses how process capability models address those principles and later on, how requirements of functional safety are taken into account. In chapter V we discuss our technical interpretation of these standards and in chapter VI the original approach by some of us for addressing the ad hoc and distributed nature of ITS. Finally we exemplarily show how the midsize enterprise GIGATRONIK, where one of the authors is in charge of engineering process design, copes with the multiple demands of the related methods and processes.

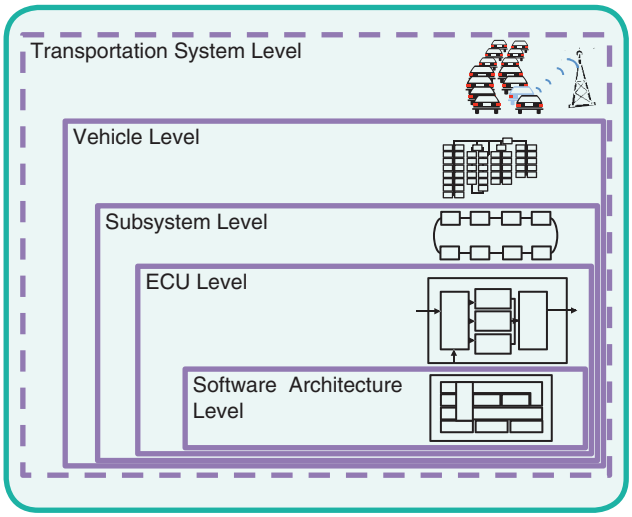


FIG 1 System levels [2].

II. Related Work

Studies on the synergy of the development procedures and standards as described in the sequel have already been published e.g. in [4]. The authors describe thoroughly the impact of the process design under the principles of ASPICE (see chapter IV of this paper) and ISO 26262 (see chapter V of this paper). More recently, Gallina et al. describe in [5] a process modelling scheme and tool chain that helps to adopt individual projects to the needs of the above mentioned standards. There have also been several studies from the technical point of view on enhancing fault tolerance, safety and reliability on ad-hoc systems taking into account the involvement of COTS components, among the most recent of them is [6]. Finally, Mladenovic and Abbas demonstrate a framework for self-organizing of participants in ITS scenarios using "the potentials of distributed in-vehicle computing power connected via wireless communications" [7].

In our paper, several aspects of the related work are discussed and are originally set in the context of well-established development schemes (such as model-based-design). Moreover, we propose the application of a new approach of service oriented software architecture and development schemes to the field of distributed vehicular systems.

III. Systems Engineering Processes

A. Challenges of Systems Engineering Processes

The main challenge of the systems engineering process is the complexity of requirements, interfaces and system states. Components have to be specified by means of premature system specifications since there is a time limit for the overall development phase. On the other hand, integration of the components has often to be done on the base of rather incomplete documentation and component test.

E.g., in [4] the authors find as the main reason of integration problems "Poorly communicated module objectives

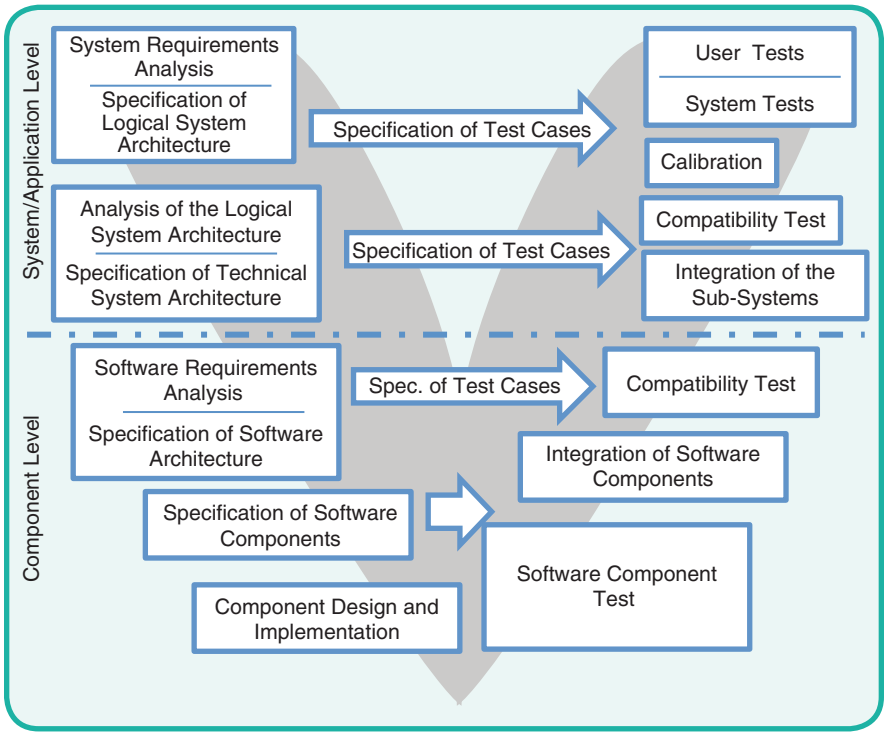


FIG 2 The CPSSD variant of the V-Model [2].

or requirements". This study, however, is mainly focused on aspects of human behavior in integration, and this is one reason why the car industry answers to the increasing safety, reliability and traceability requirements with a sophisticated development process organization.

On the other hand, ITS with ad-hoc communication impose a new quality of system integration since it is unlikely that a comprehensive specification of all possible communication and interaction processes between the Ego-vehicle and the infrastructure and/or other vehicles is available at integration time.

B. Systems Engineering Principles in ITS

Systems Engineering (SE) Principles are valid for ITS as well as for single vehicles, however, their importance increases due to the above mentioned challenges [9]. We consider the following SE principles as crucial for the development of ITS:

- Top-down approach for specification and bottom-up approach for test and integration (V-Model)
- Orientation in lifecycle phases and processes
- Optimization of interfaces and interface discipline
- Modularity and re-use
- Design for variability
- Model-based development and virtual prototyping including early simulation of the ITS in different configurations

C. V-Model

The V-Model, a phase-oriented process model, is the standard approach to solve the specification and integration problem of complex systems. It breaks down the system requirements to subsystem and component levels, of which the functionality and properties are directly derived from those of the preceding system level. A particular variant of the V-process is the so called "Core process for system and software development" (CPSSD) published by Schäuffele and Zurawka in [10]. On each of them, logical, behavioral and technical models are created which correspond to a set of test cases for the future integration task on the particular level (see Figure 2).

The V-Model allows the consideration of as many aspects of the system as possible in the specification phase and aims at comprehensive testing and integration.

As a disadvantage, all requirements and possible systems configurations in one particular systems level should preferably be known at the beginning of the development phase of this level, and requirements changes often lead to high additional work and development cost. Note, that the functional requirements of a vehicle are by far exceeded by non-functional including quality requirements, e.g. regarding efficiency, maintainability, portability, usability and safety.

In order to assure, for instance, functional safety, definitions of the safety lifecycle, a hazard analysis and risk assessment, and a functional safety concept in the beginning of the project are mandatory, as well as the subsequent specification of the technical safety requirements on each system level [11] each strongly depending on a full understanding of the technical boundary conditions and behavior of the system. In order to overcome those disadvantages, it is preferable to integrate an iterative approach in each process step which is accompanied by model-based prototyping and testing as shown in Figure 3. We will refer to that in chapter V. This can be achieved even on incomplete or premature system knowledge since the iterations can be repeated along the development lifecycle.

In comparison to other phase models the V-model and especially the CPSSD has no strict temporal sequences, but only activities and results defined.

D. Modularity and Interface Discipline

Another means to overcome the disadvantages of the V-Model is the introduction of modular architectures with standardized interfaces. They support the re-use of components or the usage of proprietary off-the-shelf or commercial-off-the

shelf (COTS) components. Modular architectures are also necessary for distributed development, e.g. between OEM, 1st Tier supplier and software and engineering service suppliers. Interface discipline, i.e. the exact definition and implementation of the interfaces and the behavior on interface level, supported by standardized compliance tests, is an inevitable prerequisite for modularity. For Off-the-shelf components and system parts the development process can then be significantly accelerated (Figure 4) when interface discipline is maintained.

In development of automotive electronic control units (ECU), the strive for standardized architecture, modular design and standardized interfaces has led to the wide spread AUTomotive Open System ARchitecture (AUTOSAR) approach [12]. AUTOSAR not only defines a common architecture model for a modular and re-usable runtime system but also describes the process of defining the interfaces between components on application level. Thus, AUTOSAR influences vehicle level, subsystem level, ECU level and software architecture level as denoted in Figure 1. Various tools exist that support the development process following the AUTOSAR approach.

As of now, there is yet no common approach for the design of Car-to-Car or Car-to-infrastructure communications. As there exist standards for the communication interfaces on physical and data link protocol layer, e.g. IEEE 802.11p [13], there is still an ongoing research of standards on the application/system level and on the design approach. The authors claim that the introduction of a common and open standard for the transmission of vehicle relevant data would accelerate the development of C2x services.

IV. Process Capability Models

The CPSSD is accompanied by a set of supporting and management processes that enable documentation, traceability and quality assurance of the processes and process outcomes (work products). Furthermore, specification and design changes re-use and variants have to be maintained and introduced into the design process.

In order to maintain the quality of those processes and their work products, requirements for process performance and output are documented and combined to process reference models (PRM). In order to measure the capability of an organization to fulfill a PRM, process assessment models (PAM) have been developed. In the automotive industry, CMMI ® [14] and Automotive SPICE (Software Process Improvement and Capability Determination) [15] according to ISO/IEC 15504 has received common acceptance. ASPICE distinguishes three process categories:

- Primary life cycle processes
- Supporting life cycle processes
- Organizational life cycle processes

In the first category, the Acquisition Process Group (ACQ) stands for the acquisition of subsystems and components, the Supply Process Group (SPL) for processes of the supply chain from the suppliers' point of view, the Engineering Process Group (ENG) stands for the main engineering process

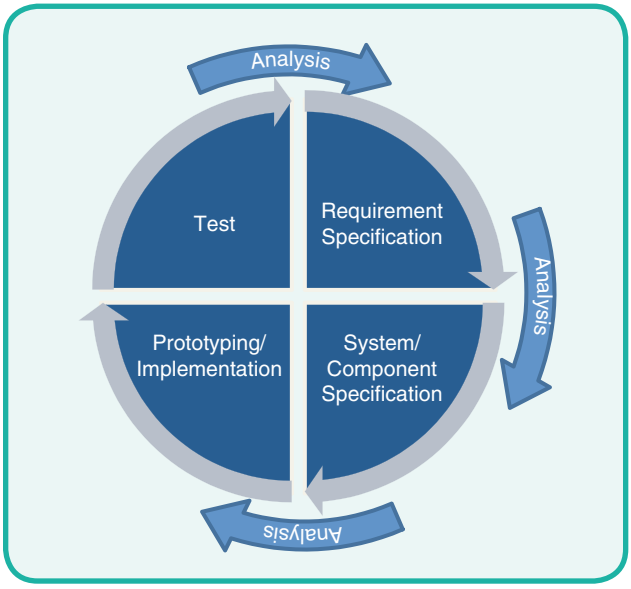


FIG 3 Iterative loop [2].

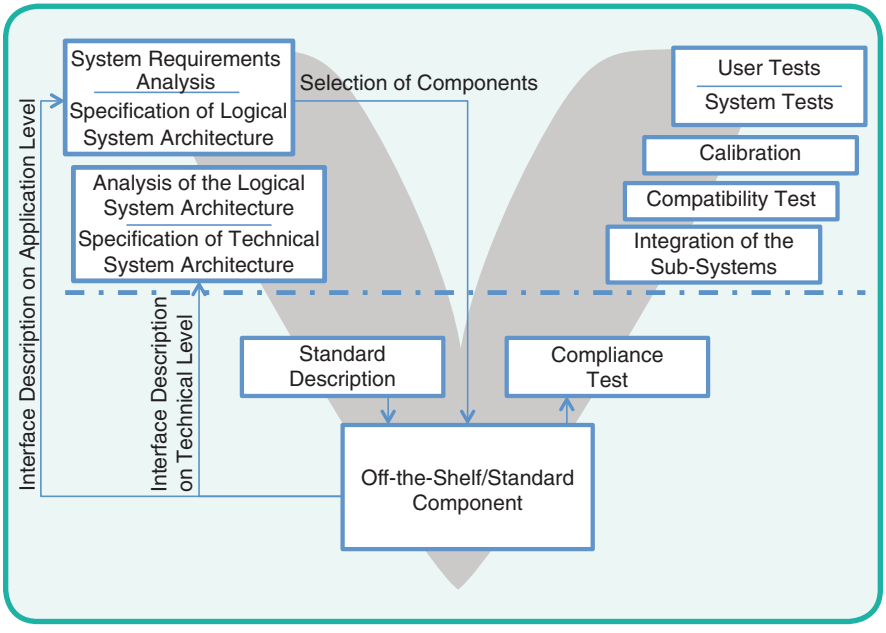


FIG 4 Use of standard components in the CPSSD [2].

as denoted above. The supporting life cycle processes are combined in the Supporting Process Group (SUP), and the organizational processes are divided into Management (MAN), Re-use (REU) and Process Improvement (PIM) processes. As an example, we list important processes of the three categories in Fig. 5, according to the so called HIS-scope (OEM's initiative for software development).

As a measure of the organizational capability to perform the processes in the PRM, a staged capability model together with a PAM is introduced by ASPICE. It comprises six levels, where Level 0 means that no or not all processes are carried out. Hence, certain important work products (plans, specifications, design documents, test descriptions etc.) are incomplete or do not exist. Level 1 denotes that all important documents are available, in Level 2 all processes and outcomes for single projects are systematically planned and tracked, Level 3 claims that there are uniform guidelines throughout the complete organization, and in the upper Levels 4 and 5 the processes are statistically measured and optimized [15].

V. Processes and Process Models for Functional Safety

While the above mentioned process standards address the reproducibility, traceability and quality of the development processes and claim to be a prerequisite of work product quality, the functional safety of the work product, i.e. the vehicle system, subsystems and components has to be considered additionally. Therefore, ISO 26262 as a modification of the IEC 61508 has been introduced as an adapted standard to the automotive industry. It focusses on the functional safety of safety-related electrical and/or electronic (E/E) systems within road vehicles. Furthermore, the standard contributes to the continuous "State of Science and Technology" since its publication. On the one hand obeying the standard serves as a proof, on the other hand it is an appraisal to ensure compliance of security policies in development processes as well as the safety of the product to be developed. ISO 26262 not only considers the development of the product itself (system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied) or individual elements (system or part of a system including components, hardware, software, hardware parts, and software units). Moreover the entire lifecycle of the product is taken into account. This includes domains such as management, development, production, operation, service, decommissioning and supports observing and verifying necessary steps [8].

The aim is to develop an overall functionally safe system according the current State of Science and Technology based on an ISO 26262 standard-compliant hazard and risk analysis (HARA). The HARA classifies potential hazards and risks, taking into account possible effects in case of an undesired system behavior under all foreseeable conditions of use. For this purpose, a specific automobile mandatory rating system is used, called Automotive Safety Integrity Level (ASIL).

The ASIL measures the safety relevance of a malfunction of an E/E-System in the automotive sector. Non-relevant malfunctions are classified by regular quality management. In this case, a standard qualification management process is sufficient. If there are significant errors the rating goes from A "low" to D "very high". The assessment is made by the factors severity, for the severity of an accident, exposure, for the probability of occurrence and controllability, for the controllability in case of an error [17].

The advantage of this classification is to identify and efficiently avoid risks by applying certain requirements of ISO 26262 *ab ovo*. Here, the standard not only refers to the product lifecycle but to all interfaces in the product development process (PDP). Among them, topics such as requirements specification, design, implementation, integration, verification, validation and configuration are included. Therefore, the ISO falls back on the procedure of the V-Model continuously and links the aforementioned domains profoundly among each other. In particular it goes into details in levels of development of systems, hardware and software – see Fig. 1 in [11].

As mentioned in Chapter II part 8 "Supporting Processes" of the ISO 26262 deals with the subject matter software tools next to threads as tests and reviews. With increase of software-based functions in vehicles also the probability of occurrence of software errors leading to failures will increase. Approximately 50% of all vehicle breakdowns can be traced back to electronics and software failures – with upward tendency [18]. Therefore, all tools involved in the development project have to be listed, evaluated and tool analysis must be performed as evidence in the documentation process, called the safety case [19].

ENG.2	System Requirements Analysis
ENG.3	System Architectural Design
ENG.4	Software Requirements Analysis
ENG.5	Software Design
ENG.6	Software Construction
ENG.7	Software Integration Test
ENG.8	Software Testing
ENG.9	System Integration Test
ENG.10	System Testing

SUP.1	Quality Assurance
SUP.8	Configuration Management
SUP.9	Problem Resolution Management
SUP.10	Change Request Management
ACQ. 4	Supplier Monitoring

MAN.3	Project Management

FIG 5 Important processes of ASPICE according to [16].

The ISO 26262 provides the so-called Tool Confidence Level (TCL) for a uniform evaluation process of tools. It is crucial for the respective safety-critical function to accomplish the designated ASIL.

The TCL is calculated from the factors Tool Impact (TI) and Tool Error detection (TD). TI describes the possibility whether malfunctions or the outputs of the corresponding tools have any impact on the security requirements of a function. TD is the probability of detection of a malfunction or faulty output of the corresponding tools. Depending on the results, a tool is evaluated with TCL1 if "no tool qualifications is necessary" and up to TCL3 for "a comprehensive qualification is necessary". Elaborate tool validation steps increase with higher TCL. According to the ISO 26262 no distinction are made between application software and created program code itself. Proof must be furnished that the software complies with standards, while developing one's own software tools. It has to be:

- specified and identified
- meets the needs and requirements
- suitable for the intended application
- developed on a standard, for example ASPICE [20].

The use of Open Source Software (OSS) in Embedded Systems (ES) has been established over the years. Due to the requirements of ISO 26262, this tendency is now facing vast challenges. Basically, the standard prohibits the use of OSS as it imposes that the use of different software components of different ASIL the ES must meet the highest ASIL. [21] These security requirements neither can be fulfilled nor proofed accurately for OSS. However, also for this purpose the ISO standard provides a few ways to assess ES.

If components of a system can be reused or were used successfully without errors before, the ISO 26262 becomes operative. Hence, certain measures can be omitted and the development process is effectively accelerated – the so-called "Proven in use" argument. A user-originated product monitoring and failure analysis is prerequisite. This approach is also part of the Supporting Processes of the ISO 26262 standard.

Starting from that basis an overall effective process management is required in order to establish an ISO 26262 compliant development process for functional safety successively, thus resisting the topic producer liability.

VI. Model-Based Design as a Pacemaker for Quality and Functional Safety

Without model-based methods in the development of automotive systems the innovations of recent years would have not been possible. Recent surveys show that 73% of the in-car application software has been developed with model-based methods [22]. Major benefits of model-based development are seen in the early detection of errors and the facilitation of model exchange and re-use.

During model-based development, the embedded software as well as system components is defined by executable computer models. For the modeling of software, signal flow charts and finite state machines are applied. The plant models of system components are built up from differential-algebraic equation systems, discrete event models and signal flow charts. In simulation runs, the software models are tested in open-loop or in closed-loop with the system components. Once the software models fulfill the quality requirements, embedded C software components are generated by auto code generators.

ISO26262 requires the following phases of verification and validation when applying model-based methods in the development of functional safe systems [23][24]. First, the tool chain for model-based software development including graphical editors, model libraries, simulation tools, auto-code generators and test environments must be certified. Second, the software models must be verified and validated. Third, the embedded C software components that are generated from the software models must be verified in back-to-back tests. In addition, the complete software that is integrated from software components has to be validated in integration tests.

In particular, the development of functionally safe ADAS requires to further advance the model-based methods into the area of requirements specification, verification and validation. Formal specification methods of system and function requirements enable computer-based automated requirements verification and validation.

Decision trees and finite state machines that are key components in ADAS software can be specified by means of first-order predicate logic in connection with mathematical expressions [25]. By simulation or static analysis these requirement specifications are verified. I.e., contradictions or incompleteness of requirements are detected. Further, test vectors for the embedded software are derived for software model and component verification. With these formal model-based methods, the major requirement of ISO26262 to ADAS can be fulfilled: Only wanted and no unwanted software is executed in the system.

In the model-based development process, the software models evolve to embedded code. MiL development environments as depicted in Figure 6 enable the validation and verification of ITS on development PCs. For this, software models of the ITS (M) and vehicle, driver, sensor, actuator and environment models (M) are simulated in closed loops.

The ITS software models contain all information for an automated code generation of the application layer of the ECU software. The generated embedded C code is compiled to runnables on the development PC and tested in SiL development environments (see Figure 7) which are based on the MiL development environments.

In HiL development environments (see Figure 8) the simulated ECUs are replaced by real components that run the embedded code (C) which has been auto-generated in the

previous steps. For this, the vehicle, driver, sensor, actuator and environments are simulated on real-time hardware-in-the-loop tests systems (RT) that stimulate the ECU input signal lines and measure the ECU output signal lines.

The MiL, SiL and HiL development environments all allow the interconnection of distributed ITS where the overall system functionality is divided into software components running on distributed, interconnected ECUs.

In order to test collaborative algorithms in ITS, the authors propose a collaboration environment, in which multiple MiL/SiL/HiL entities (called Ego Car *n*) – one for each participating vehicle – are running independently, delivering communication and localization data to a dispatcher structure that executes further coordination activities. Hence, each Ego Car is visible as a traffic object in the environment model of the other Ego Cars as depicted in Figure 9.

This model allows repeated tests in highly reproducible scenarios, and is especially suitable for regression tests.

VII. Systems Engineering for Distributed and Ad-Hoc-Systems

The complexity of distributed and Ad-hoc-Systems is founded by the lack of comprehensive knowledge of all system states and requirements during the development phase. Hence, the classical V-Model approach as described above seems to be not accomplishable for systems with incomplete specification in the beginning of the design phase. Furthermore, AUTOSAR does not support ad hoc communication with components that were not specified during the system design phase, and ISO 26262 demands for a thorough HARA and systems specification prior to the systems design.

A solution to that problem for ITS has recently been proposed by two of the authors in [3]. Instead of defining all possible system states, the so called SOMA4DDAS approach (Service-oriented Modeling and Architecture [26] for DDAS) defines the interfaces and behavior on interface level for all contributing sub-systems and leaves the system configuration to the runtime by means of service oriented

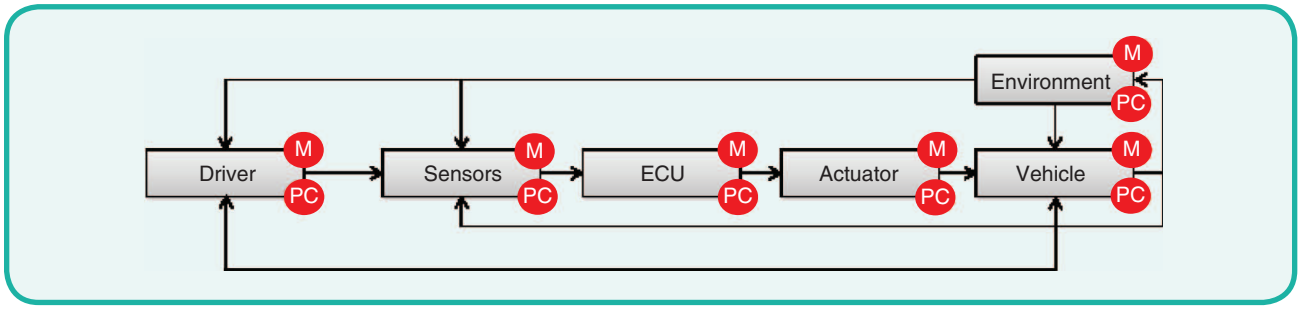


FIG 6 Model-in-the-Loop (MiL) development environment.

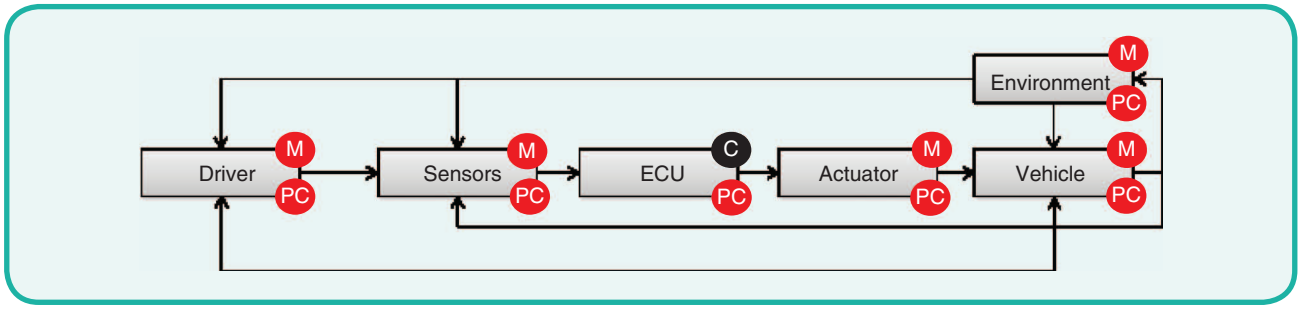


FIG 7 Software-in-the-Loop (SiL) development environment.

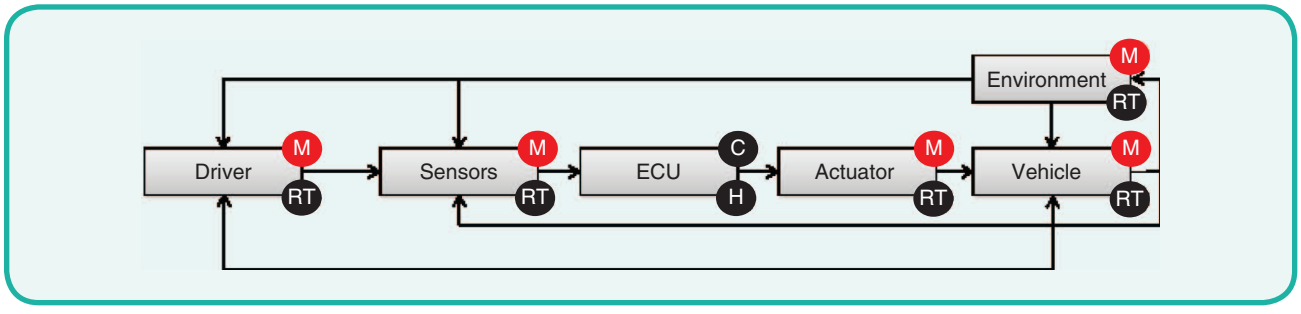


FIG 8 Hardware-in-the-Loop (HiL) development environment.

architecture (SOA) [27]. Therefore, after the interface specification, a service contract has to be defined that describes the roles of the service partners and their communication protocol. The service Architecture illustrates the relationships of the participants involved using their interfaces and contracts. Finally the participants are implemented and integrated.

While the cooperative applications created hereby are underlying ongoing changes, the behavior of the different functional entities is stable.

This approach fits easily into the CPSSD as described above (see Figure 10) but introduces flexibility and changeability of the system configuration during runtime as imposed by C2X communication. However, it requires a strong model-based approach and tool support.

In case of collaborative cars or coordinated traffic, each vehicle is seen as an entity running participants to offer services or request those of other cars in reach. Each of these services is providing specific functionality which is modeled using a capability. Furthermore, it is reacting according to the behavioral model specified within a contract. At runtime, the different participants are detecting other ones in reach using a so called service discovery by sending out discovery requests to other participants in reach. The discovery mechanism creates a temporal view of what functionality is available at the very moment locally at each participant [28]. This enables the participants to select services to be used, connect to them and thereby include them into their application execution graph. While the cooperative applications created hereby are underlying ongoing changes, the behavior of the different functional entities is stable.

As there might be more than one service available in the current scenario with the same functionality, an optimization problem for the best service quality has to be solved by an appropriate composition algorithm that decides for each requested Interface the optimal selection of available services.

Having completed the Service Discovery Process, the distributed application takes place until a participant vanishes from the scene. As an example, various vehicles appear at an intersection, requiring the location and expected trajectory from each other and the indication of the presence and trajectory of other traffic objects such as pedestrians and bikes from the infrastructure as in Figure 11.

In the above example as well as in the example mentioned in the introduction of this paper, an activity diagram of the desired service identifies the need for services calculating the trajectory of several participants. Those trajectories are not independent of each other being mechanically coupled in case of the truck-and-trailer assistance systems or being coupled by decisions being made by the drivers or a control unit in order to avoid collisions in Figure 11. Thus, the interfaces of those services have to take into account mutual dependencies.

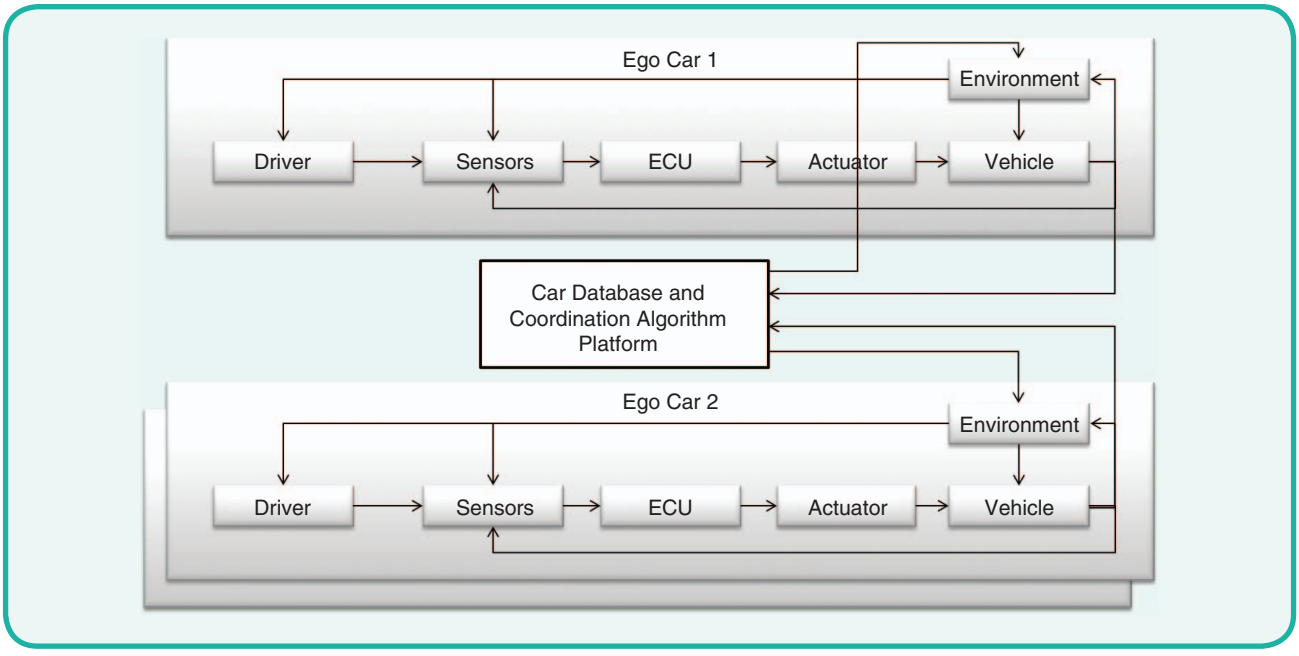


FIG 9 Multiple car simulation environment.

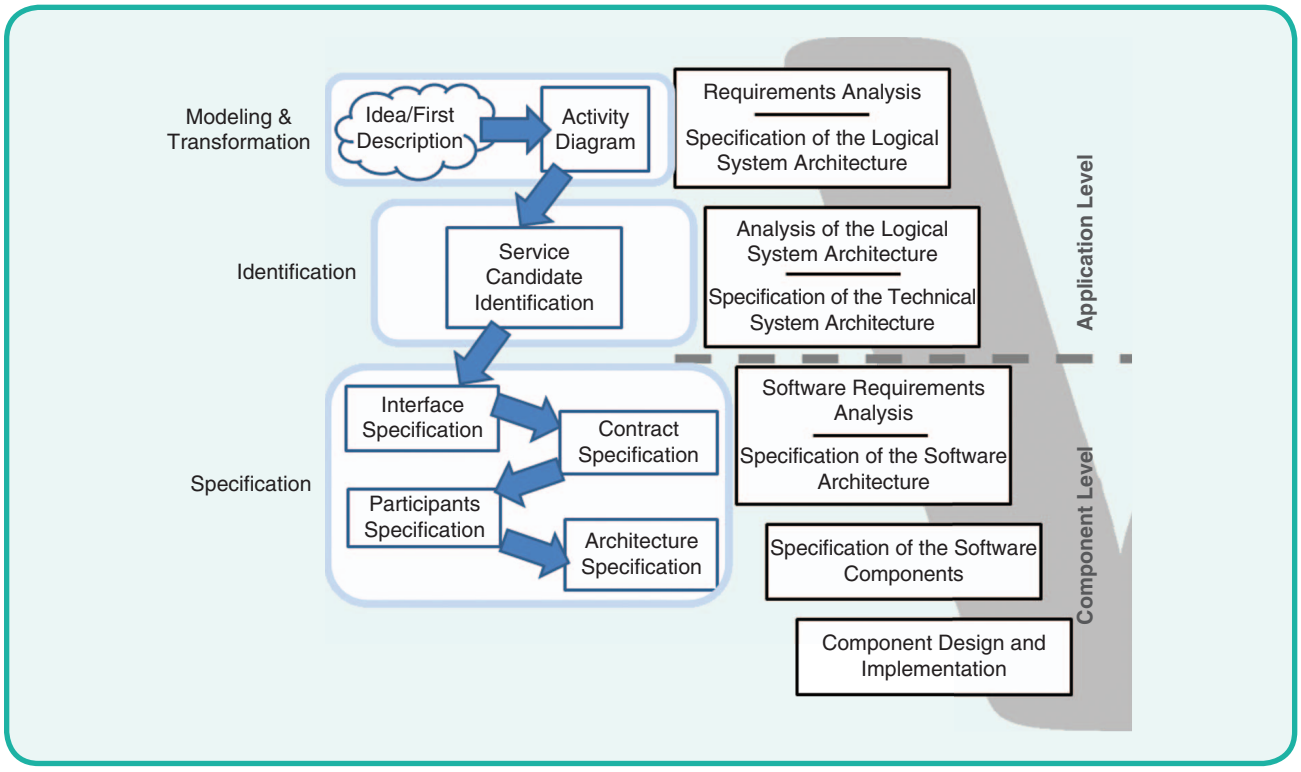


FIG 10 V-Model and Service-oriented design [3].

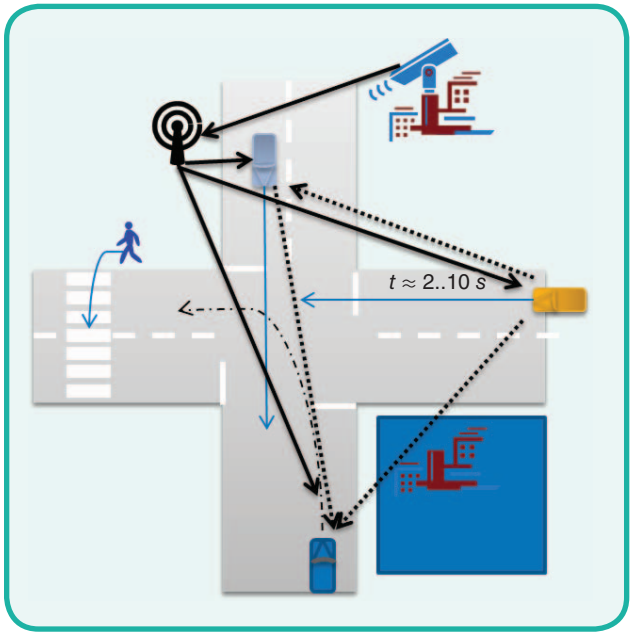


FIG 11 Distributed services scenario.

The identification and specification of services candidates follow a formal process supported by according modeling schemes as exemplarily shown in Figure 12.

Of course, the compliance of ad-hoc and distributed systems to ISO 26262 raises several questions. The main of them is how to validate the safety of a system that is not complete during development time, since the final system functionality is established at runtime (Safety validation 4–9 in ISO 26262 [11]). From the point of view of a service oriented approach, we can reason that:

- On component level, all contributing components can be validated as in a statically defined system on their according ASIL. Moreover, services can communicate via their interface, to which ASIL they are certified. From the point of view of a vehicle as a component, it has to maintain functional safety even though there is no or erroneous information from the ITS available.
- On system level, the overall system architecture, i.e. the architecture that includes all possible services, can be validated on functional level when methods of formal specification and model-based design are applied. There exist methods for formal verification to which one of the authors contributed e.g. in [29]. A new service broad into a scenario would lead to a system architecture extension as described in chapter VI, and therefore to a new formal verification.
- Implementation verification on system level can only be verified by example implementations, e.g. concrete traffic situations. The participants must make sure that their functionality in new system constellations would not lead to unwanted behavior. At this point, we see potential for further research and standardization.

VIII. Synergies of a Common Process Strategy

The development of complex systems such as ADAS often takes several years. The various development stages require an equally strong focus on a steered and optimized development process. At the same time, the risks of such systems should not be ignored and must be continuously monitored.

These two goals are sought by the ASPICE processes and ISO 26262.

An outstanding advantage is achieved when both methods are used along the entire development process, as they develop in this case, most similarities and synergies. Even more pronounced are these synergies, when they are used within an entire enterprise.

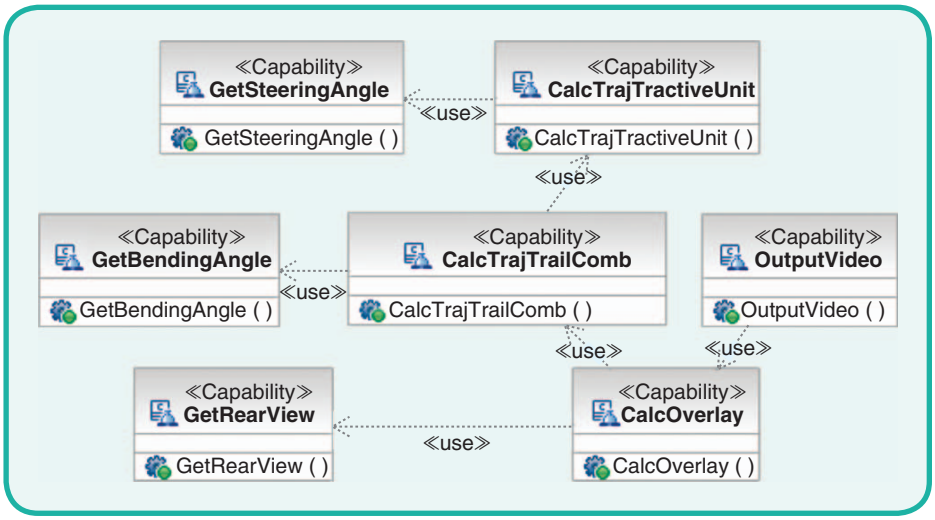


FIG 12 Services candidates for truck-and-trailer trajectory calculation [3].

The synergies of these two standards and their interfaces to the development processes of higher system levels are crucial for a common process strategy, hence to implement an effective and efficient development process on the lower system level. On the lower system level the engineering processes respectively the V-Model, the supporting processes and the management processes should be regarded separately.

The most synergies are especially for the software development part in the engineering processes, as shown in Table 1.

In this ASPICE process area the work products and outcomes of the ASPICE processes are the basis for the ISO 26262 part 6 software development.

ASPICE doesn't require specific methods since it can be regarded as a framework for the development process.

Hence the overhead to fulfill the ISO 26262 standard is to elect the sufficient methods for the software development and hardware development. These elected methods are requested by the ASPICE standard as well.

A crucial part for development projects is the requirement management, because the established requirement engineering is the basis for a high quality product and an effective and efficient development.

The safety requirements and the other functional requirements can be documented, traced and maintained within a common process and workflow as long as the safety requirements are classified with the relevant ASIL-level.

The most important part in the requirement management is the traceability along the whole requirement cascade, from high-level to low-level. This is requested by the ISO 26262 and ASPICE. There are many tools for requirement management which support the bidirectional traceability between the different requirement artefacts.

Therefore, another synergy can be achieved by choosing the adequate development tools, since the ISO 26262 sets requirements to Tool Confidence Levels as described in chapter V.

The used tools should be chosen at the beginning of the project and if they fulfill the ISO26262 standard they automatically fulfill the ASPICE standard.

Table 1. HIS scope of ASPICE and ISO26262 [30].

ASPICE (HIS)	ISO 26262
ENG.2	Specification of technical requirements
ENG.3	System design
ENG.4	Specification of software requirements
ENG.5	Software architectural design
ENG.6	Software unit design and implementation software unit testing
ENG.7	Software integration and testing
ENG.8	Verification of software safety requirements
ENG.9	Item integration and testing
ENG.10	Safety validation
SUP.1	(No specific requirements)
SUP.8	Configuration management
SUP.9	(No specific requirements)
SUP.10	Change management
MAN.3	(No specific requirements)
ACQ.4	(No specific requirements)

The supporting processes "configuration management" and "change request management" are required by both standards, so their implementation is mandatory for a development project on lower system levels.

IX. Conclusion

The development of ITS with C2X communication imposes new demands on systems and software engineering processes. Organizing tasks and activities logically, executing them in ordered sequence and determine their results is crucial for developing safety-relevant systems. In addition, a further important aspect is the clear definition and assignment of responsibilities to all contributors over the supply chain.

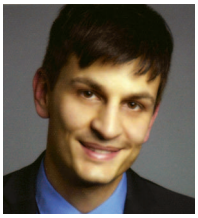
It is commonly understood that ISO 26262 is a mandatory standard in developing functionally safe systems and it is foreseeable that the processes and methods introduced by ISO 26262 extend to the ITS system level. On the other hand, service oriented architectures and development procedures will help to cope with the complexity of dynamically reconfigurable settings resulting from C2X services. Both fit to the commonly accepted V-Model and reflect the demands from PRMs such as ISO 15504 and Automotive Spice. Formal approaches for requirement specification, a modular architecture with standardized if not open interfaces, the usage of model-based design methods and tools and last but not least a strong process orientation support the achievement of the demands for process and product quality and functional safety. It has been demonstrated, that these means cannot be considered independently. In fact, it is advisable to take the advantage of synergies arising from a simultaneous consideration of the process standards in the framework of systems engineering. Moreover, a careful identification and selection and the proper use of the tool environment is essential for the future advances in ITS. As an example, the role of model-based design and service oriented architecture and development schemes has been demonstrated for distributed driver assistance systems.

Finally, the authors have demonstrated that the simultaneous introduction of ASPICE and ISO 26262 on company level can be advantageous in terms of process management efficiency and process quality.

About the Authors

Ansgar M. Meroth was born in Freiburg, Germany. He received the Dipl.-Ing. in Electrical Engineering and the Dr.-Ing. (PhD) in Electrical Engineering at the University of Karlsruhe (now Karlsruhe Institute of Technology KIT) in Karlsruhe, Germany. He worked as a system and software developer in Car Multimedia and Driver Information systems at Robert Bosch GmbH and in the corporate human resource development department of the same company – there on system development processes. Since 2003 he is Professor for Automotive Systems Engineering at Heilbronn University, Heilbronn, Germany and served there as well as vice dean and vice president. He is local representative of the German VDI (Germany Engineers Association)

Frank Tränkle received his Master of Science degree in chemical engineering at the University of Wisconsin in 1993 and his diploma in engineering cybernetics at the University of Stuttgart in 1994. In his PhD thesis at the University of Stuttgart he developed the software application PROMOT for the object-oriented modeling and dynamic simulation of chemical processes. After his dissertation in 1999 he started as a software engineer at ETAS in Stuttgart. In 2001 he assumed the leadership of the LABCAR® software and model development team. Since 2006 Frank Tränkle had been the head of the function development and simulation team and later department at GIGATRONIK in Stuttgart. In 2012 Frank Tränkle received his professorship for control engineering at the Heilbronn University for Applied Sciences. His research and teaching areas are model-based controller design, model-based software development, embedded microcontroller and Linux targets in Simulink®, advanced driver assistance systems, vehicle simulation, as well as embedded software validation and verification.



Bastian F. Richter was born in Wuerzburg, Germany. He received the B.Eng. degree in automotive systems engineering, in 2012 and the M.Eng. degree in electronic systems engineering from the University of Heilbronn, Germany, in 2014. During his studies he conducted an elaboration in the field of applied research, how to establish ISO 26262 compliant development processes in an existing process landscape and published it as co-author at ITSC 2014.

His main areas of research interests cover the methodological design and simulation of electronic systems as well as their control and their functional safety software development. For the last 5 years he has been working for the Robert Bosch GmbH in different development and research departments dealing with new and innovative braking systems. Since his graduation in 2014, he is working as a software developer.

Marco Wagner was born in Heilbronn, Germany in 1983. He received his diploma (Dipl.-Ing. (FH)) in Automotive Systems Engineering from Heilbronn University, Heilbronn, Germany in 2008.

In 2008 he joined Heilbronn University as a researcher. His research interests include adaptive, distributed, embedded automotive

systems. Using the principles of Service-oriented Computing (SOC) he designed a software framework to support adaptive driver assistance systems for truck and trailer combinations. In 2014 he left the University to continue his work in industry.

Michael Neher received his Diploma in Physics at the University of Heidelberg in 2008. From 2008 until 2011 he worked as a system analyst at CETEQ GmbH & Co KG and from 2011 to 2014 at GIGATRONIK Stuttgart GmbH in the field of project and process management. He is actually seeking his MBA degree at the ESB business school in Reutlingen. After his contribution to this paper he left GIGATRONIK for an affiliation in a consulting company as a process consultant.

Jochen Lüling was born in Stuttgart, Germany. He obtained a degree in mechanical engineering from the University Stuttgart, Germany, in 1991. During his studies one main focus of research was programming of robots and production equipment. He started his career as a software developer for the Cap Gemini group. Later he was team leader for IVM Automotive Stuttgart GmbH, responsible for developing database applications for the automotive industry.

He is now working in the project management at the GIGATRONIK Stuttgart GmbH and therefor responsible for different projects mainly for the automotive industry. One of those projects e.g. is the serial production of an electronic device, which is integrated in a battery for an electric car. His main area of research interests covers the development of an efficient software development process in combination with the right tool for that issue.

He is an intacs™ certified ISO/IEC 15504 Provisional Assessor Automotive SPICE®. He is also member of the board of the competence center ITS Baden-Württemberg e.V. (Integrated Telematic Systems)

References

[1] H. Bubb, "Der Fahrprozess—Informationsverarbeitung durch den Fahrer," in *Tagungsband VDA Technischer Kongress*, Stuttgart, Germany, 2002.

[2] A. Meroth, F. Tränkle, and B. Richter, "Optimization of the development process of intelligent transportation systems using Automotive SPICE and ISO 26262," in *Proc. IEEE Intelligent Transportation Systems Conf.*, Qingdao, Shandong, Oct. 2014, pp. 1481–1486.

[3] M. Wagner, A. Meroth, and D. Zöbel, "Developing self-adaptive automotive systems," *Des. Automat. Embedded Syst.*, vol. 18, no. 3, pp. 199–221, 2013.

[4] E. Petry, "How to upgrade SPICE-compliant processes for functional safety," in *Proc. 10th Int. Conf. Software Process Improvement Capability Determination*, Pisa, Italy, May 18–20, 2010.

[5] B. Gallina, S. Kashiyarandi, H. Martin, and R. Bramberger, "Modeling a safety- and automotive-oriented process line to enable reuse and flexible process derivation," in *Proc. IEEE 38th Int. Computer Software Applications Conf. Workshops*, July 21–25, 2014, pp. 504–509.

[6] S. di Carlo, G. Gambardella, P. Prinetto, F. Reichenbach, T. Lokstad, and G. Rafiq, "On enhancing fault injection's capabilities and performances for safety critical systems," in *Proc. 17th Euromicro Conf. Digital System Design*, Aug. 27-29, 2014, pp. 583–590.

[7] M. N. Mladenovic and M. M. Abbas, "Self-organizing control framework for driverless vehicles," in *Proc. 16th Int. IEEE Conf. Intelligent Transportation Systems*, Oct. 6–9, 2013, pp. 2076–2081.

[8] A. Zafar, S. Ali, and R. K. Shahzad, "Shahzad Investigating integration challenges and solutions in global software development," in *Frontiers Information Technology*, Piscataway, NJ: IEEE Press, 2011, pp. 291–297.

[9] P. Gonzales, B. Christie, and J. White. (2011, Apr.). Applying systems engineering principles to the development of transportation communication standards. ITS joint program office, Final Rep. [Online]. Available: www.its.dot.gov/index.htm

[10] J. Schäuffele and T. Zurawka, "Automotive software engineering," 5th ed. Berlin, Germany: Springer-Verlag, 2013.

[11] *Road Vehicles—Functional Safety—Part 1: Vocabulary*, ISO 26262-1:2011.

[12] Autosar (2013). AUTOSAR—The worldwide automotive standard for E/E system. [Online]. Available: www.autosar.org

[13] A. Sikora and M. Schappacher, "A highly scalable IEEE802.11p communication and localization subsystem for autonomous urban driving," in *Proc. Int. Conf. Connected Vehicles Expo.*, Las Vegas, NV, 2013, pp. 775–780.

[14] CMMI Product Team "CMMI® for Development, Version 1.3," SEI, Carneghie Mellon Univ., Pittsburgh, PA, Tech. Rep., 2011.

[15] AUTOSIG. (2010). Automotive SPICE® Process Reference Model/Process Assessment Model. [Online]. Available: http://www.automotivespice.com

[16] (2014, June 20). HIS Process Scope Automotive Spice. [Online]. Available: http://portal.automotive-his.de

[17] *Road Vehicles—Functional Safety—Part 3: Concept Phase*, ISO 26262-3:2011.

[18] A. Dold and M. Trapp, "Herausforderungen und erfahrungen eines OEM bei der gestaltung sicherheitsgerechter prozesse," in *Gesellschaft für Informatik -GI-, Bonn: Informatik, Informatik trifft Logistik. Bd.2 : Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, R. Koschke, Ed. Sept. 2007, pp. 24–27.

[19] *Road Vehicles—Functional Safety—Part 2: Management of Functional safety*, ISO 26262-2, 2011.

[20] *Road Vehicles—Functional Safety—Part 8: Supporting Processes*, ISO 26262-8, 2011.

[21] *Road Vehicles—Functional Safety—Part 6: Product Development: Software Level*, ISO 26262-6, 2011.

[22] M. Broy, S. Kirstan, H. Krcmar, and B. Schätz, "What is the benefit of a model-based design of embedded software systems in the car industry?" C. Bunse and J. Rech, Eds. in *Emerging Technologies for the Evolution and Maintenance of Software Models*, Hershey, PA: IGI Global, pp. 343–369.

[23] M. Beine, "Modellbasierte entwicklung und automatische code-generierung für sicherheitskritische anwendungen," in *Proc. GI Jahrestagung*, 2009, pp. 2659–2667.

[24] C. Kurutas, "ISO 26262 konforme software-entwicklung mit modelbased design," in *Schritte in Die Künftige Mobilität*, Berlin, Germany: Springer-Verlag, 2013, pp. 7–17.

[25] F. Tränkle, R. Bechtold, and S. Harms, "Testbasierte entwicklung von steuergerätefunktionen," in *Proc. Tagungsband 4. Dagstuhl-Workshop MBEES*, 2008, pp. 95–101.

[26] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for developing service-oriented solutions," *IBM Syst. J.*, vol. 47, no. 3, pp. 377–396, 2008.

[27] O. Thomas, K. Leyking, and M. Scheid, "Serviceorientierte vorgehensmodelle: überblick, klassifikation und vergleich," *Informatik-Spektrum*, vol. 33, no. 4, pp. 363–379, 2010.

[28] M. Wagner, D. Zöbel, and A. Meroth, "Re-configuration in SOA-based adaptive Driver Assistance Systems," in *Proc. 6th Workshop Adaptive Reconfigurable Embedded Systems conjunction IEEE/ACM CPSWeek*, Berlin, Germany, 2014.

[29] C. Schwarz, M. Wagner, and D. Zöbel, "Formal verification of service-oriented adaptive driver assistance systems," in *Proc. 5th Workshop Adaptive Reconfigurable Embedded Systems conjunction IEEE/ACM CPSWeek*, Apr. 2013.

[30] J. Wolf, B. Spanfelner, and D. Wild. (2014). TüV Süd Foliensatz, ASPICE and TüV Süd 'Fit-for-purpose certificate' Tool Qualification Symposium. [Online]. Available: http://www.validas.de/TQS/2014/program.html#talks

ITS