The Next Generation of BMW's Electrified Powertrains: Providing Software Features Quickly by Model-Based System Design

Dr. Stefan **Kriebel**, Vincent **Moyses**, Dr. Georg **Strobl**
BMW Group, Munich, Germany

Dr. Johannes **Richenhagen**, Dr. Philipp **Orth**, Prof. Dr. Stefan **Pischinger**
FEV Europe GmbH, Aachen, Germany

Christoph **Schulze**, Timo **Greifenberg**, Prof. Dr. Bernhard **Rumpe**
RWTH Aachen University, Software Engineering, Aachen, Germany

Summary

Modern vehicles are complex systems dominated by continuously evolving software functions and highly cross-linked architectures. With shorter development cycles and increasing cost pressure, new approaches are required to provide safe and affordable mobility solutions. This paper tackles this by merging known computer science techniques with state of the art methods of mechanic and electric engineering: systems engineering is consistently done in four layers while retaining agile efficiency at the same time. Model-based requirements replace written text. Besides cost reduction through maintainable and reusable documentation, this approach enables semi-automated test case derivation cutting down testing effort. For the development of BMW's next generation electrified drivetrain efficiency and quality objectives can be achieved. Thus, cost, quality and adaptability targets can be met at the same time.

1 System and Software Business Model Evolution

Automotive system development is confronted with conflicting requirements on the one side and market conditions on the other. Customers require a quicker reaction of product development to new technical trends. This results in a shorter time-to-market dropping from average values of five years in the 1980s to three years in the 21st century [1]. Including upgrades and facelifts as well, this even drops to a level of one to two years. With each system upgrade, customers demand more functionality at a similar price level. If inflation is taken into account, this requires a required cost reduction of 4% every year pushing available development budgets to steadily lower levels [2].

However, concentration on cost efficiency does not resolve the trade-off to be made: Individual, mechanically driven vehicles are being replaced by integrated mobility systems including the electrified drivetrain, vehicle-wide functionalities, user experience and diverse traffic scenarios. The integration is realized by complex software systems. If these are developed with the same methods and processes as

[KMS+17] S. Kriebel, V. Moyses, G. Strobl, J. Richenhagen, P. Orth, S. Pischinger, C. Schulze, T. Greifenberg, B. Rumpe: The Next Generation of BMW's Electrified Powertrains: Providing Software Features Quickly by Model-Based System Design. In: 26th Aachen Colloquium Automobile and Engine Technology, 2017.
www.se-rwth.de/publications/

during the last decades, verification and validation effort will explode. E.g., the migration from a state-of-the-art adaptive cruise control function to an auto pilot results in a validation effort increase by a factor in the magnitude of 100000 [3]. Test effort, prototype vehicles and hence validation costs explode at abovementioned budget restrictions. As a result, quality assurance is performed risk-based where taken risks result in an explosion of software recall campaigns in the past decade [4].

For deriving right countermeasures, the weaknesses in current system engineering need to be analyzed (Fig. 1).

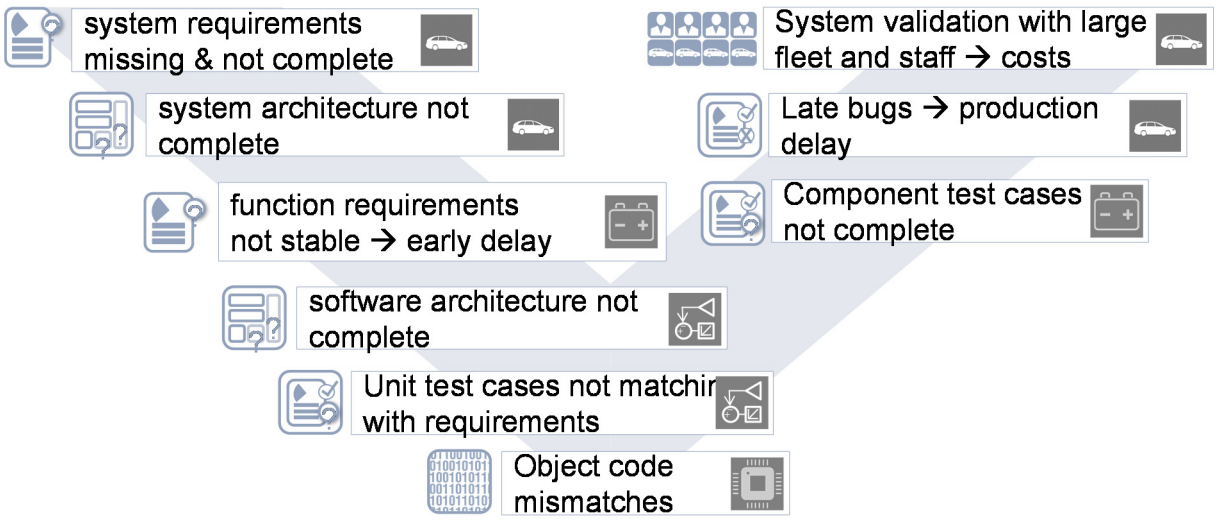

Fig. 1: Quality dilemma

At the begin of the development of new functionalities, system requirements and system architecture are difficult to be completely defined top down - they are usually being elaborated bottom up during prototype sessions to identify a desired behavior. However, description methods are missing to document doubtlessly for all later stages what shall be contained in the system. As a consequence, the entire system architecture is usually described incompletely and is therefore not maintainable. An incomplete or even missing system architecture leads, of course, to weak requirements on system level. Bottom up function level requirements have to be defined without taking consistent system interfaces into account. This causes multiple requirement alignment loops and hence early delay of project milestones. Hence, on software level unit tests are defined without consistent system and functional requirements which leads to additional integration loops caused by object code mismatches.

On component level test cases are not complete, causing similar integration issues on the vehicle level. These are mitigated by applying large vehicle fleets with intense staff involvement: error identification time is increased through failure detection on system level only, cost limits require a prioritization of test maneuvers.

2 Model-Based System Design: System Design based on Software Development Approaches

One main strategy to handle the mentioned conflicts between time-to-market, cost reductions, arising quality issues and the increased validation time on vehicle level is frontloading. This means it is necessary to increase the efforts in early development steps like requirements development, functional design and architectural design [5]. In fact, frontloading is not a new idea as it shall help to reduce efforts on the more cost-intensive verification and validation tasks on Hardware-in-the-Loop (HiL) level or vehicle level. Furthermore, frontloading is recommended by many standards like ISO/IATF 16949 and ISO 26262. However, it cannot be easily applied as it needs an interdisciplinary approach combining methods from known mechanic and electric engineering techniques as well as computers science approaches.

Model-based systems engineering [6, 7] focuses on a continuously evolving model-based development of architectures and requirements [8, 9]. While a systematic and semi-formal representation of requirements is more intensive in a first step, experience from large scale software projects shows an increased product quality. The higher quality of documents resulting from frontloading activities reduces following costs for validation significantly. High quality models will reduce communication efforts, inconsistencies between requirements and reduce the amount of defects and failures which would otherwise only be detected on later verification levels. An easier communication is especially important in the context of large teams, interdisciplinary exchange and collaborations with external partners and internal departments (aspects, which are all common in the automotive domain).

The System Modeling Language (SysML) [10] is derived from the Unified Modelling Language (UML) [10, 11] which was introduced in the 1990ies. It provides a larger set of structural and behavioral diagram languages to specify systems from different viewpoints on different abstraction levels, but does not provide a concrete process or detailed guidelines which diagram types are to be used in which order or for which level of abstractions. Similar to EAST-ADL [12], BMW has proposed a model-based system engineering approach called SMArDT ("Specification Method for Architecture, Design and Test") which comprises four levels: Requirements level, function design level, architectural level, hardware resp. software design level (EAST-ADL: vehicle, analysis, design and implementation level), as shown in Fig. 2.

SMArDT combines a systematic vertical refinement approach from layer to layer with a horizontal hierarchical composition from the context of the overall engine to specific modules. From layer to layer additional requirements are identified and derived from higher level requirements, where suitable, to establish a complete tracing (as demanded by standards like CMMI [13] or ISO26262 [14]). On the technical level a first separation between hardware and software aspects is performed, which are then implemented on the fourth level.

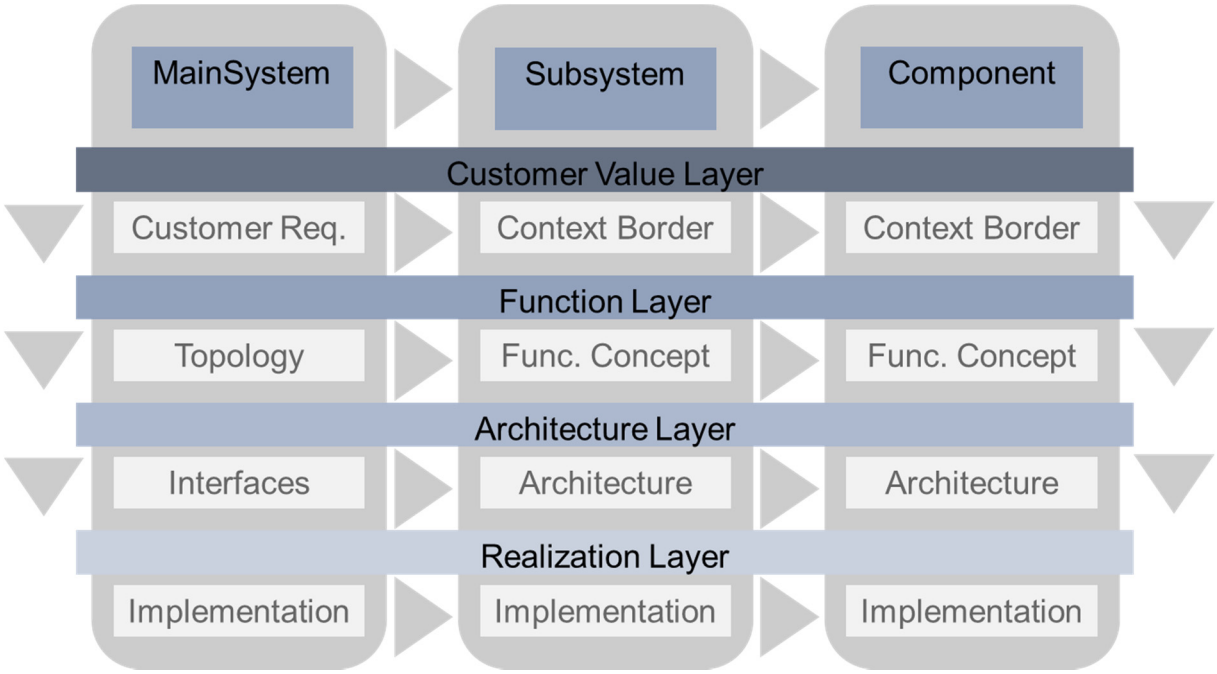

Fig. 2: SMArDT methodology overview

SMArDT supports a systematic step-by-step model-based requirement and function specification [15, 16] due to the four abstraction layers. In consequence, on each level the provided requirements and concepts are reevaluated and detailed, which - in terms of frontloading - ensures a very early verification and validation on the ongoing activities. Of course these steps are quite time-intensive but provide also high quality artifacts which can be used to significantly speed up the overall development process. The mentioned abstraction layers are applied for the structured documentation whereas the applied process is meant to be agile [17].

In the context of this paper we will focus on one of several possible new opportunities, which are provided by a high quality set of semi-formal models: the semi-automated generation of test cases [18, 19].

2.1 Semi-automated Test Case Generation

The function models on the second layer realized by activity diagrams, state charts and internal block diagrams provide already enough information to generate test cases for verification purposes automatically.

A corresponding process is illustrated in Fig. 3. Based on customer models, like use case and context diagrams, defined in the first layer, activity diagrams and state charts are defined during the function specification. These function models are defined in cooperation of specifier and tester to ensure that functional aspects, like correct failure handling, are also included. Based on these models from the function layer test cases can be generated automatically to fulfill the path coverage criteria C_{2c} [20]. In addition, the tester can configure specific aspects of the model, like specific input parameter or decisions, to manipulate the test case generation for context-related needs.

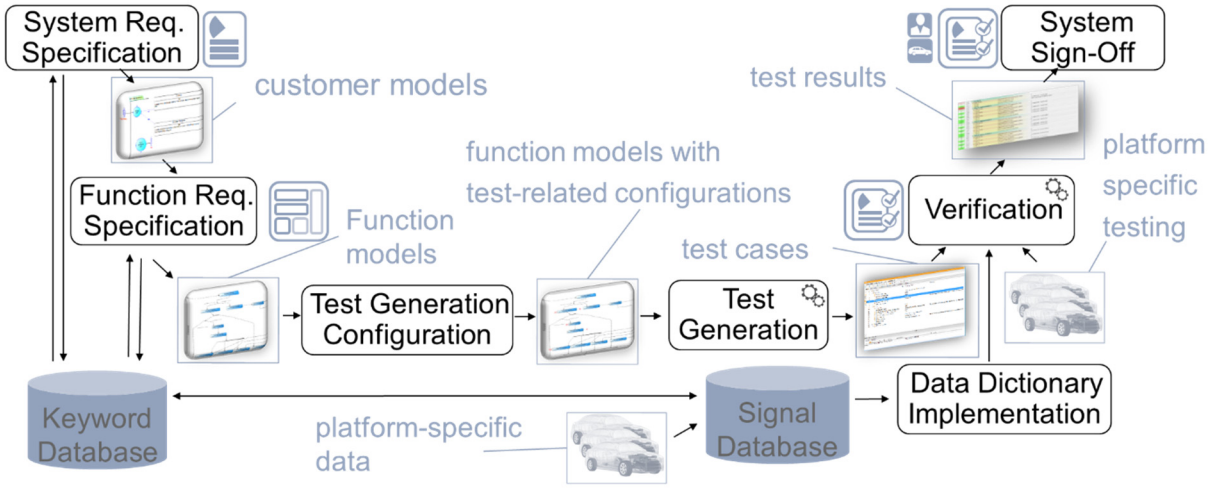

Fig. 3: Overview test case generation

Exemplary activity diagrams, which are used for test case generation, are shown in Fig. 4. Different activities or decision node can be classified to adjust the test case generation for specific needs (e.g. test step execution time / cost, requirements, decision coverage). As the activity diagram and internal block diagram of the function layer are refinements of the use case and context diagrams of the customer value layer (see Fig. 2) and the activity diagram is based on the functional architecture described by the block diagram, the generated test cases represent all the aspects defined on both layers.

To be able to generate test cases from activity diagrams or state charts, besides a correct use of the SysML language, the explicit definition of expected output needs to be defined on an abstract level. While this is common for state charts the activity diagram language has been adjusted to fulfill these needs. Besides these technical requirements, high quality semi-formal models, representing an abstract but distinct functional description, are necessary to generate test cases, which can be used to verify a system based on defined requirements. These models are provided by the process and guidelines provided by SMArDT.

During the system requirement and function specification phase labels from a keyword database are reused to define interfaces and conditional aspects. These keywords are mapped to concrete platform-specific signals and their specific test execution. During the data dictionary implementation step for each keyword (and related signals) a concrete test sequence is implemented to be able to perform the initialization and evaluation automatically.

Combining the mapping between keyword and signal database (and their test implementation) with the semi-automated test case generation, the effort to define, setup and execute verification tests is reduced significantly.

In addition, because of the neutral nature of customer and function models the generated test cases can be reused for different platforms.

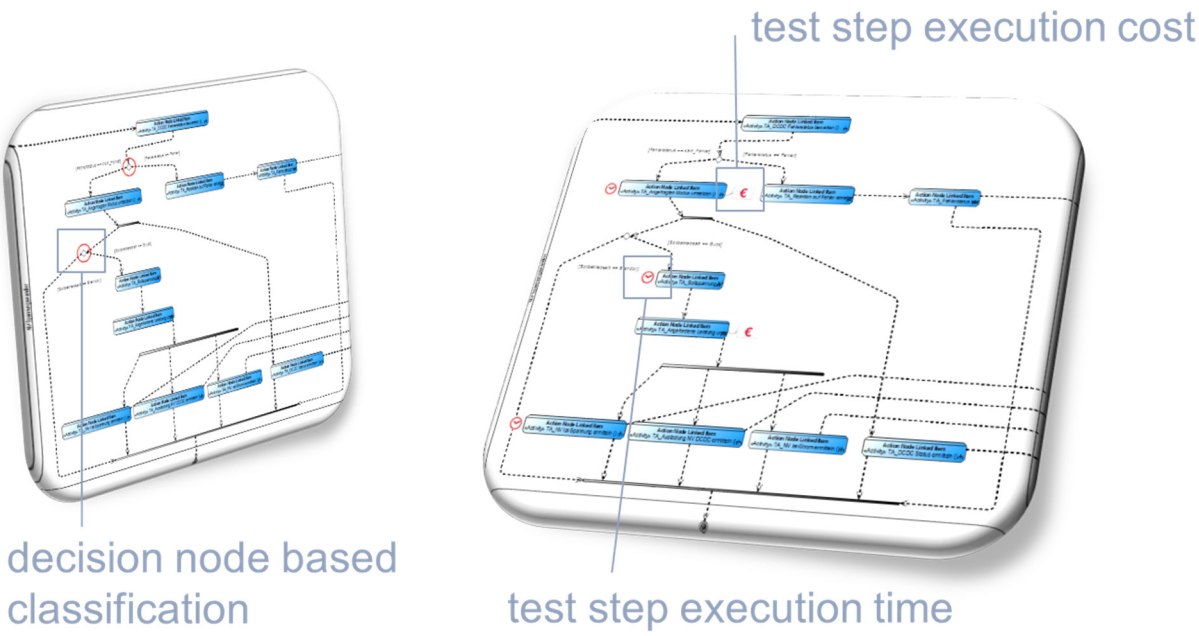

Fig. 4: Examples for test case generation configuration

3 Application: Project „MTSF“

SMArDT and the approach for semi-automated test case generation is currently used in an ongoing series development project to define BMW's upcoming generation of electric drives. In collaboration with the FEV Europe GmbH and the Software Engineering Chair of the RWTH Aachen University additional modeling guidelines for function models have been identified to allow a semi-automated test case generation, called "Model-based Testing of Software-based Functions" (MTSF).

During the development, efforts have been monitored systematically to evaluate if the proposed expectations are fulfilled. In Fig. 5 a first evaluation on five different functions is shown, which highlights the additional effort on function specification level for specifier and tester to be able to derive test cases. In relation to the complexity of the function the amount of effort, but also the amount of generated test cases, is increasing.

Function A is the first measured function including diagnosis and safety aspects and thereby highlighted the additional benefits in the context of these areas. As a path-wise test case generation requires logical branches to increase the amount of test cases, error detection and handling mechanisms highly increase the potential for automated test case generation.

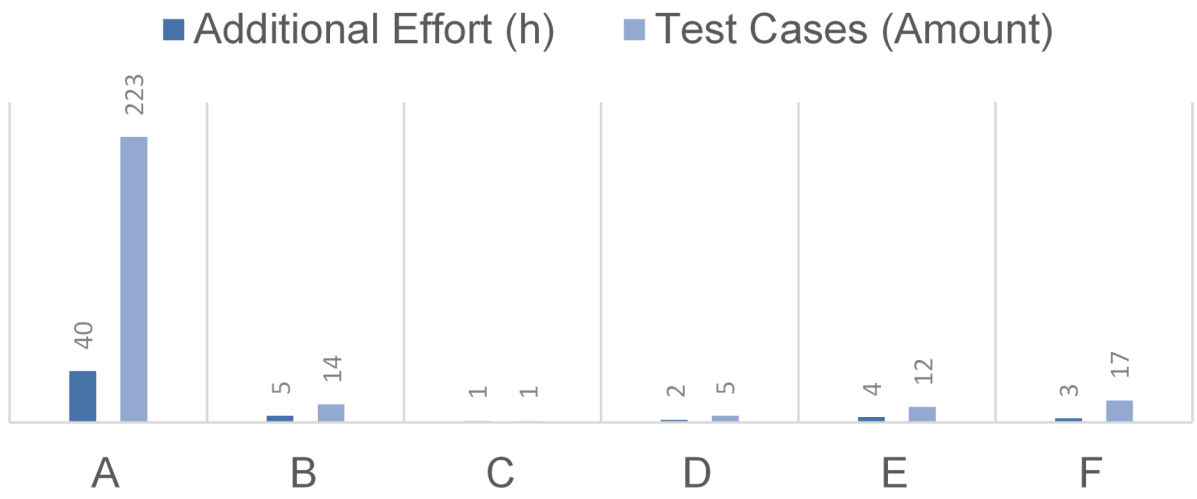

Fig. 5: Additional effort for automated test case generation

Regarding the measured efforts it needs to be considered that the additional guidelines for test case generation have been applied the first time in a serious development context. Thereby, reduced efforts can be expected once the approach is established. In addition, not only the effort could be reduced, but also the quality of models is increased, because of the semi-formal nature of the introduced guidelines.

Nevertheless, comparing the measured efforts with ongoing traditional development, the expectations can already be matched, as reduced efforts on requirement interpretation and verification tasks are already negating the additional efforts on function specification level as shown in Fig. 6.

Based on the current experiences further improvements on the test generation configuration are already identified to increase the quality of resulting test cases even more.

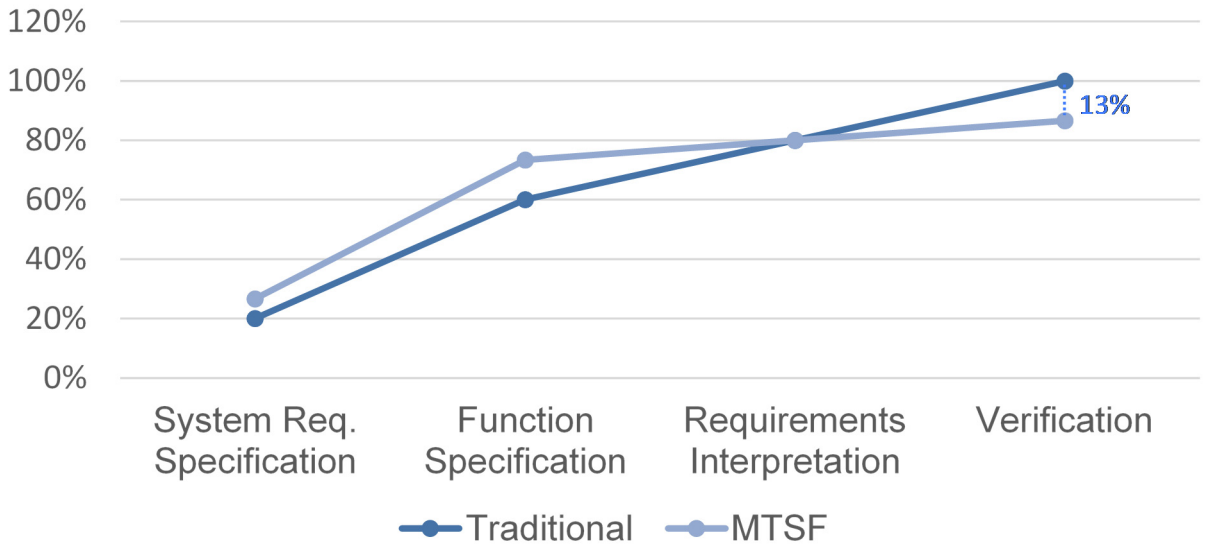

Fig. 6: Reduced effort due to MTSF

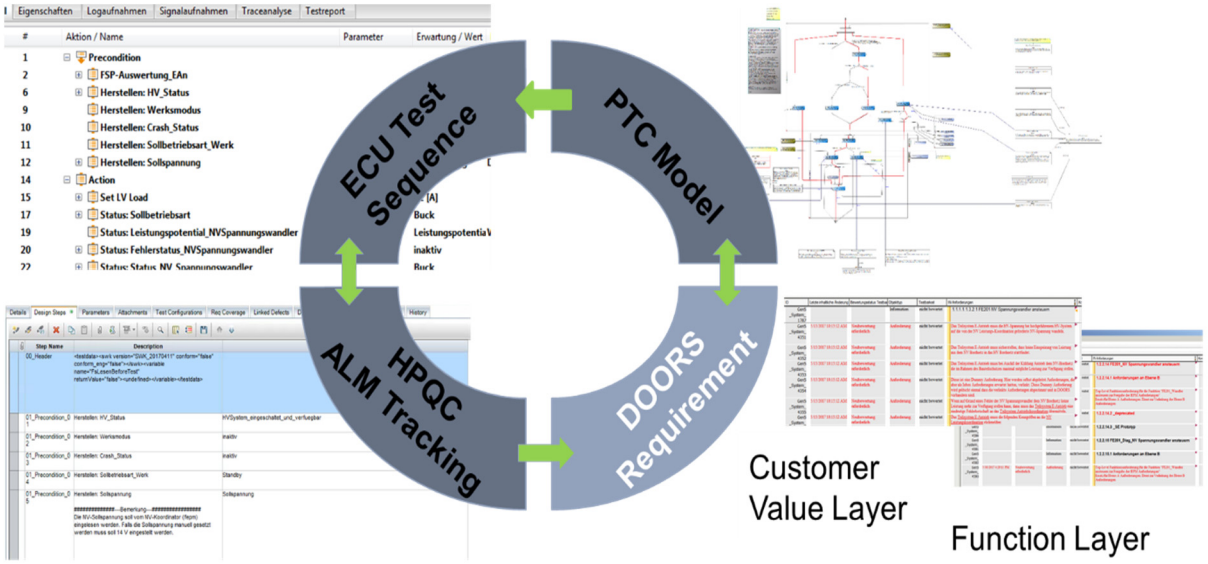

Fig. 7: Integrated toolchain from requirement to test case

The overall process is supported by an integrated toolchain as represented in Fig. 7. A fluent connection between the tools DOORS, PTC, ECU Test and HPQC is established to sustain the ongoing industrialization.

4 Summary & Outlook

The MTSF system engineering approach tackles the current conflict between solid requirements, test case and architecture definition for complex systems and cost and time pressure due to mobility market conditions. The method bases on model-based description of system functionalities through clearly defined abstraction levels. Semi-formal description enables efficient requirement alignment and semi-automated test case generation.

In this paper, the design on the higher abstraction levels was focused sharing application experiences for the next generation of BMW's electrified powertrains. For the first time, model-based specification was applied to system level development of automotive series products. Driven by a systematic function architecture definition, more than 50 functions reflect the designed behavior of energy management, charging and torque provision. Considering coverage criteria, test costs and execution time, test cases were derived semi-automatically for defined model paths. At the end of the workflow, automatically executable test sequences were produced.

The overall development effort could be reduced while working within given market and product milestones. Even more importantly, system quality is increased: more test cases are defined, test coverage can be determined in early stages, bugs can be identified at earlier design stages. Also, requirements, test cases and development artefacts can be aligned easily and traced doubtlessly.

Next steps will include the continuation of system development on lower abstraction levels. Quality gains will be observed - when reaching the vehicle test level, a

significant reduction of prototypes and debugging phases is expected. The MTSF method will be expanded application-wise to other vehicle development domains outside the powertrain. Also, Failure-tree-analysis for e.g. safety applications will be facilitated. Next method development steps will focus the industrialization for large, distributed and cross-enterprise development teams. Here, we especially expect further work on bridging established development cultures of computer and engineering science domains.

5 References

[1] PRASAD, B.
Analysis of pricing strategies for new product introduction
Pricing Strategy and Practice, Vol. 5 Issue: 4, pp.132-141
Bingley (UK), 1997

[2] MOHR, D. et al.
The road to 2020 and beyond: What's driving the global automotive industry?
Mc Kinsey & Company, Inc.
Stuttgart, 2013

[3] HÜBNER, H.-P.
Automatisiertes Fahren – Wohin geht die Fahrt?
Proc. 18. Kongress Fortschritte in der Automobilelektronik
Ludwigsburg, 2014

[4] STEINKAMP, N.
2016 Automotive Warranty & Recall Report: Industry Insights for the Road Ahead
Chicago, 2015

[5] KRIEBEL, S.
Economic High Quality Software for Automotive Systems
3d Congress on Real Time, INCHRON; Braunschweig
2011

[6] WYMORE, A. W.
Model-Based Systems Engineering
CRC Press, Inc.
USA, 1993

[7] ESTEFAN, J. A.
Survey of Model-Based Systems Engineering (MBSE) Methodologies
Incose MBSE Focus Group
2007

[8] GRÖNNINGER, H. et al.
View-Centric Modeling of Automotive Logical Architectures
In: Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme IV. Informatik-Bericht 2008-02, CFG-Fakultät, TU Braunschweig,
2008

[9] KRIEBEL, S.
Timing propagation in the development of software-based automotive systems
4th Symtavision NewsConference on Timing Analysis, Braunschweig
2010

[10] WEILKIENS, T.
Systems Engineering with SysML/UML: Modeling, Analysis, Design
Morgan Kaufmann
USA, 2008

[11] RUMPE, B.
Agile Modeling with UML: Code Generation, Testing, Refactoring
Springer International
Germany, 2017

[12] CUENOT, D. et al.
Managing Complexity of Automotive Electronics Using the EAST-ADL
12th IEEE International Conference on Engineering Complex Computer Systems
2007

[13] PAULK, M.
Capability Maturity Model for Software
John Wiley & Sons
2002

[14] HILLEBRAND, M.
Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen
KIT Scientific Publishing
Germany, 2012

[15] GRÖNNINGER, H. et al.
View-Centric Modeling of Automotive Logical Architectures
4th European Congress ERTS - Embedded Real Time Software, Toulouse
2008

[16] GRÖNNINGER, H. et al.
In: Proceedings of the Object-oriented Modelling of Embedded Real-Time Systems (OMER4) Workshop, Paderborn,
2007

[17] LINZ, T.
Testen in Scrum-Projekten Leitfaden für Softwarequalität in der agilen Welt
dpunkt.verlag, 2. Auflage
2016

[18] PRETSCHNER, A. et al.
Model-based testing for real. STTT 5(2-3): 140-157
2004

[19] PHILIPPS, J. et al.
Model-Based Test Case Generation for Smart Cards.
Electronic Notes in Theoretic Computer Science 80: 170-184
2003

[20] LIGGESMEYER, P.
Software-Qualität: Testen, Analysieren und Verifizieren von Software
Spektrum Verlag
2009