Software Based Test Automation Approach Using Integrated Signal Simulation

Andreas Kurtz
BMW Group
Integration Electric/Electronics, Software
Munich, Germany
andreas.kurtz@bmw.de

Bernhard Bauer
University of Augsburg
Institute for Computer Science
Software methodologies for distributed systems
Augsburg, Germany
bauer@informatik.uni-augsburg.de

Marcel Koeberl
BMW Group
Integration Electric/Electronics, Software
Munich, Germany

***Abstract*—Test automation in distributed systems requires new methods in signal simulation for the stimulation of the distributed system. Increasing complexity of electric electronic (E/E) systems increases the testing-effort. The main challenge is to reduce the time spent on manual stimulation of input signals in favor of automated testing. The systems currently used for test automation have to be adapted to each hardware and software version of the system to be tested. The approach shows a software-based automation solution through the integration of a simulation service in the AUTOSAR architecture. By integrating a generic software-based simulation module with an interaction point at the basic software driver layer, the execution of tests can be automated and improved in terms of adaptively and reproducibility.**

Keywords–Automotive; distributed systems; model based testing; simulation; system model; test model.

I. INTRODUCTION

Mastering complexity and customer orientation are challenges in the development of electric electronic (E/E) and software functions in the automotive industry. In current and future vehicles, the increase of the distribution and the networking of functions demands new ways of automation for testing customer features. Software bugs are the main reason for malfunctions in new developed cars [1]. In the automotive industry, the safety requirements are extremely important because of their implications. Therefore, there is a need to err on the side of caution.

This paper focuses on developing a method for test automation of the automotive system model. The system model stands for the total system deemed to be a distributed system with its networked hardware (HW) and software components (SWC). The long-term goal is a solution for automating system model test, at a total system level, with an integrated distributed software-based solution.

The rest of the paper is structured as followes. Section II presents the State-of-the-art model based testing approach and the current realisation level in the automotive industry. Section III describes the over all approach, followed by Section IV with a detailed description of a concrete implementation. In Section V a related approach is shown to point out the difference to the new approach, concluded in Section IV with a short outlook.

A. Problem Statement

The increasing complexity of developed functions with shorter developing time leads to exceeding use of methods. Figure 1 shows the raising demand of using methods when reducing development time to handle equal development effort. In addition to an increase of system complexity the conventional methods have to change to virtual development.

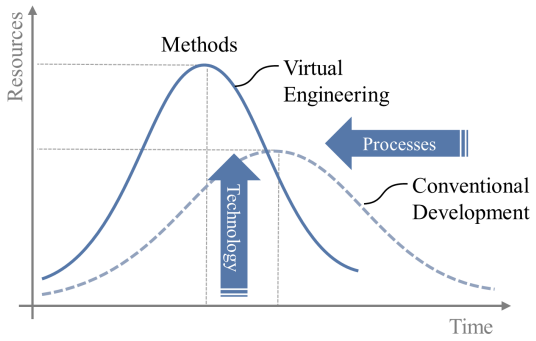

Figure 1. Reducing time demands increase of virtualisation [2]

State-of-the-art software-based automation methods, used in automotive software development, use additional software functions integrated in the software components to be tested. This kind of interaction affects the customer function itself.

These functions, used for virtual input stimulation, interact with the customer functions and require specific solutions for each type of implementation. Using this kind of automation does not reflect the functional behaviour of the system model like in customer usage. Additional signals and interfaces are used to get access to the implemented customer functions with the goal to compare the system model towards specification. Another aspect is the additional software code, needed to realize the interaction. Last, but not least, crosslinked functions are not detected because of the high-level interaction point. Therefore malfunctions in close-by or connected SWCs are not noticed.

A physical simulation of sensor signals at the hardware interface does represent the customer usage but is too expensive to build a specific solution for each HW variation. The main challenge for physical signal simulation is to find

 ISBN: 978-1-61208-458-9

a generic solution for all types of hardware interfaces. The developed method shall use the test model in combination with the AUTOSAR description files, to generate data for stimulation of the system model.

This paper focuses on a method for distributed systems, extracting the needed data out of the test model and the AUTOSAR configuration files. AUTOSAR supports the approach because of the standardised software architecture (Figure 2), giving the chance of developing a generic method.

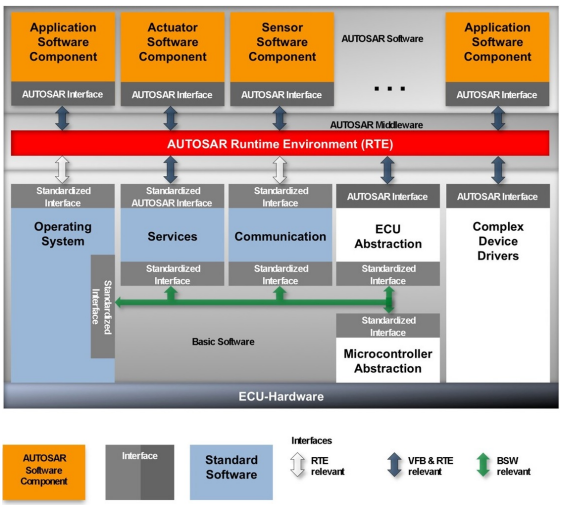

Figure 2. AUTOSAR Architecture, Components and Integration [3]

II. State-of-the-Art

The start scenario to get test cases in model based testing (MBT) is presented in Figure 3(A), with a test designer and its mental test model. The tester, expecting to have the entire test cases, executes the textual test cases. The prospective goal is to reach scenario (C) shown in Figure 3. This enables to compute the test cases for a test automation. By a change, from difficult to understand textual test cases to formal models, we can enable automation methods.

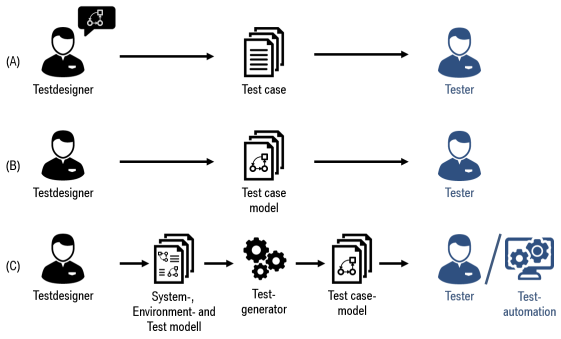

Figure 3. Textual vs. model based specification of tests [4]

With the mentioned focus on the total system as a distributed system, the test case models become complex. Due to the many options of partitioning the variable number of participants (SWCs) to the different hardware options, regarding the standardised architecture framework, the complexity of the models grows.

Model based testing methods are [4]:

- *Component Test* or so called 'unit test', in which the individual program components are tested in isolation.
- *Integration Test* is testing the interaction of several componentes.
- *System Test* is to ensure the operability of the entire system according to the requirements
- *Acceptance Test* is the test under real operating conditions, as well as the interaction of several systems.

At *System Test*, the aim of the approach, the distribution of functions increases automation complexity. In addition, the simulation of input signals is more difficult, due to less accessible interfaces without disturbing the system's behavior.

Figure 4 shows a simplified software architecture with the main layers of AUTOSAR. From bottom up, above the HW there is the basic software (BSW) allocated. The basic software contains modules shown in Figure 2. All communication to the SWCs is distributed via the runtime environment (RTE). Figure 4 shows the same architecture for both cases with two SWCs: SWC1 (and lower layers) representing the human machine interface (HMI), iDrive Controller (ZBE), and SWC2 (and lower layers) representing the Navigation System (Navi) with symbolised tree structure of the menu.

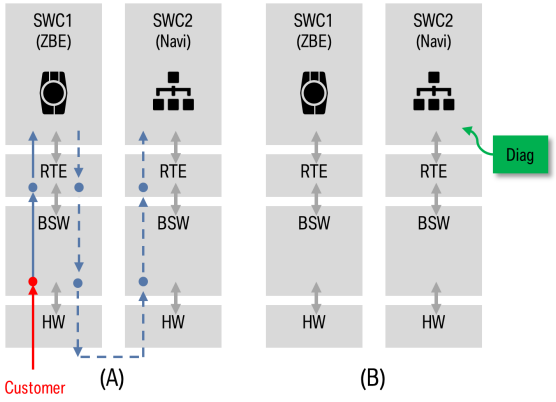

Figure 4. Use Case - SW-Architecture; (A) Customer Use; (B) Simualtion state of the Art; simple draft

With reference to the example, we look closer to the software architecture according to AUTOSAR. Conventional software-based automation methods interfere at application SWC layer, shown in Figure 4(B) named Diag. Usually these functions are integrated for software-based internal error detection and setting data trouble codes (DTC).

Figure 4(A) shows a customer interaction with the total system via the ZBE, e.g., the customer enters navigation destination. Figure 4(B) shows the conventional method with a so-called diagnosis job (Listing: 1) to hit the right menu items. The conventional virtual interaction via diagnosis job interacts in SWC2 and tends to a different behaviour in the total system.

```
1 apiJob(ECU, "steuern\_routine", "ARG;MENU;
    ↳ STR", \%i;\%i;\%i;\%i)
```

Listing 1. Example Diagnosis-Job

 ISBN: 978-1-61208-458-9

This function (Listing: 1) uses a unique identifier (ID) to hit a menu item. A series of these jobs is necessary to reach the same goal (in this case an entered navigation destination), but shows a different total system behaviour. For both cases, we see the same appearance but in Figure 4(B) the communication between SWC1 and SWC2 is missing. The result is that errors in the data communication that occur in the customer case are not detected.

III. Related Work

Model based testing is the basic approach in the automotive product development. Methods for data-flow analysis can help to improve the explained method by computing realistic software-paths and the corresponding data sets. With the help of this information, the test model can be improved to get near 100% test coverage in consideration of the customer use cases. Domain knowledge is the key for computing user realistic software paths. Other paths only have to be checked referring their impact on customer functions.

A existing approach with focus on 'automation, modularization and compatibility of all equipment to do measurement, calibration and diagnosis' [8] is the Can Calibration Protocol (CCP) [8]. The Protocol is used for calibration and data acquisition. Realised as a driver with access to the internal ECU memory this part of the protocol causes additional CPU load. During a session using CCP a 'continuous logical connection' [8] is established to transfer data from the ECU to the master device (off board test automation). This approach interacts at the driver layer. CCP has the main goal of data acquisition in contrast to data simulation.

IV. The Approach

The goal is to compare system- and test model with an integrated distributed software solution to get the system's behavior closer to the system's behavior in customer's use. Figure 5 shows the approach of comparing system- and test model.

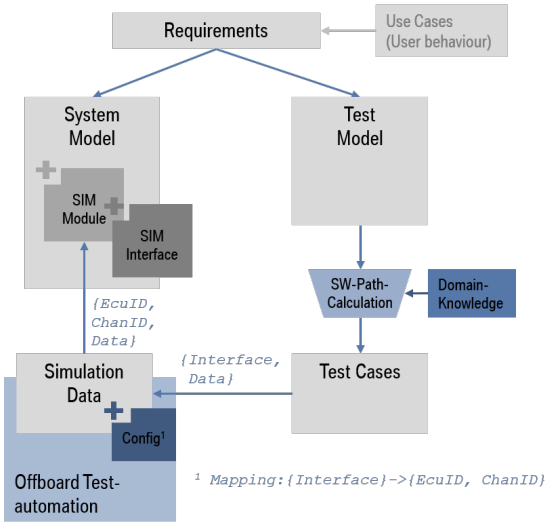

Figure 5. Approach of comparing system- and test model based on a standard model based approach

The methodological approach is to integrate a generic software-based simulation module (SIM Module) and a simulation interface (SIM Interface) in the system model. The novelty of the approach is the layer of intervention (AUTOSAR driver layer) with the associated reduced data complexity. The reduced data complexity is due to the focus on system input signals. In summary, it is a distributed simulation of input signals in a distributed system with an engagement in the basic software driver layer.

– How does this work? – These distributed SIM modules can receive test cases from the off board test automation system and execute the test cases, individually or jointly, by order of the off board system. The data for the test cases is computed out of the test model and transferred in abstract test data and a mapping table (Config). The separation has the advantage of using the test case for different hardware configurations.

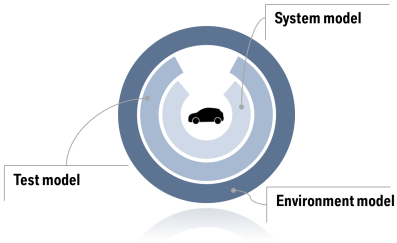

Figure 6. System-, test and environment model [4]

Figure 6 is intended to show that the test model is the combination of the specified system model functional behaviour and the environment model. The environment model includes all external factors e.g., temperature, light or physical characteristics to the system model as well as the customer.

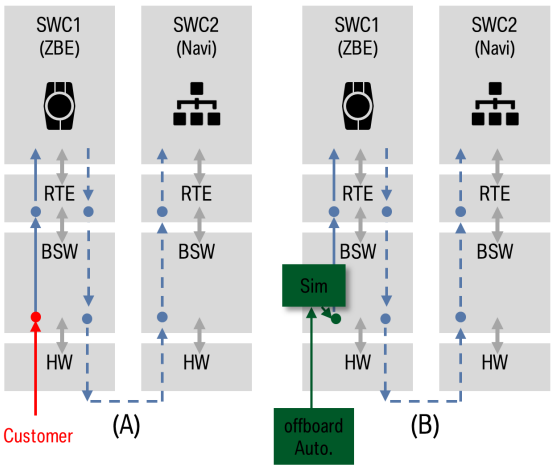

Figure 7. Automation Approach AUTOSAR view. (A) Customer use; (B) simple draw of SIM Interface integration in AUTOSAR Basic Software

The interaction of the simulation, formerly manipulation, at the AUTOSAR basic software interface (Figure 7(B)) allows the SWCs to operate closer to the customer use.

On the one hand, the customer function operates closer to the behaviour in the normal customer usage which leads to detection of errors in the functional implementation or errors between layers. On the other hand, the simulation gets easier

 ISBN: 978-1-61208-458-9

by getting closer to the hardware layer, because of reduced types of signals. There are only discrete signals, because all data processing of customer input ends up in analog digital converter. A positive effect is that in case of simulation, cross-linked functions are also triggered and show their behaviour as well as the misbehaviour.

V. The Details

The methodological approach detailed in Figure 5 shows schematically how the test case data is loaded to the system model. Based on the assumption that a test model is available test cases are derived therefrom. To get only the relevant test cases, we compute theoretical software paths of the test model with the background knowledge of the specific domain (Figure 5).

The separation of the total system in cluster e.g., power-train, power-network, infotainment and other is called domain. We need the domain knowledge to reduce the number of computed paths. It is not meaningful to test all theoretical possible software paths of a single software component at total system level testing. Looking at the distributed system with all of its participants, the software components in cooperation reduce the possible paths. The more components participate the more complex the system gets, but also the functions limit each other e.g., timing, min- and max values. For example when focusing on power-network functions the domain knowledge does have information and boundaries of the power train like the engine speed range. The speed range is decisive for the operating range of the generator and thus for energy availability.

The bigger part of software functions is developed with a model based approach and realised with state machines. The main reasons for the increase of complexity are that on one hand the conditions for triggering the transitions are built in various state machines in different software levels, and on the other hand, almost all of the conditions do have timing constrains. This expands the number of use cases by testing boundary values e.g., lower limit, upper limit, lower limit follower, upper limit follower and last but not least the time steps on the valid timing interval.

Computing the paths of the menu structure has been performed with a data-flow based model analysis [5] [6]. All paths had to start and end in the *main menu* with the requirement that every transition can be passed only once per path to avoid loops. The result is a set of paths with the information of states passed, transitions and trigger for transitions.

After computing all relevant paths of the test model, we can generate test cases, including information as named before. With this information, the idea is to compute a set of all sequences for simulating all user input hardware signals. The computed data is separated in the interface information and the data sequence. This dataset describes the overall customer function input and will be mapped to the ECU and HW-Channel allocation specified in the system model. This configuration database is stored in the off board test automation (Figure 5). With this approach the general customer input information can be mapped to each specific AUTOSAR implementation.

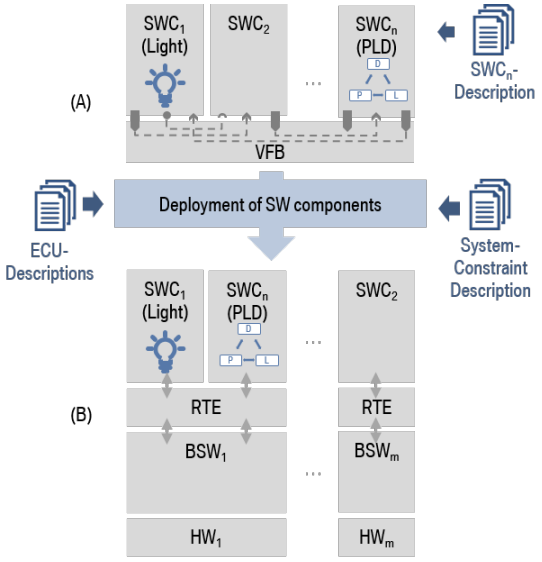

Figure 8. Schematic Diagram of AUTOSAR Configuration files [3]

The configuration *Config* (Figure 5) is an information depending on the specific implementation, computed out of the AUTOSAR description files. These files are generated in the software development process (Figure 8). These files give the specific input of which SWC is mapped to each ECU as well as channel and communication path information. In Figure 8, the shown files contain the following information:

- *Software Component Description*: Describes the functional dependencies
- *System Description*: Describes the partitioning of SWCs to ECUs
- *ECU Configuration*: Describes the signal routing to the HW-Abstraction
- *Basic Software Description*: Describes the mapping of the HW-Abstraction to the HW-Channels

These descriptions are specific for each AUTOSAR software architecture. Therefore, the information depends on the software of the system model. Errors in the configuration files are handed to the test automation.

Figure 9 shows the software architecture solution for the new methodological approach. What is new is that this SIM module is integrated in each ECU, which reads sensor signals. The SIM Module represents the merger of the following three components:

- *SimAgent* is the logical component, including a state management and is responsible for the execution of the sequences, data storage and safety requirements,
- *SimGW* does only route signal data to the Microcontroller Abstraction Layer (MCAL),
- *DioSim* is the interface to the existing driver (DIO) and its read services with the goal to replace the physical signals with the simulated in case of an active simulation.

 ISBN: 978-1-61208-458-9

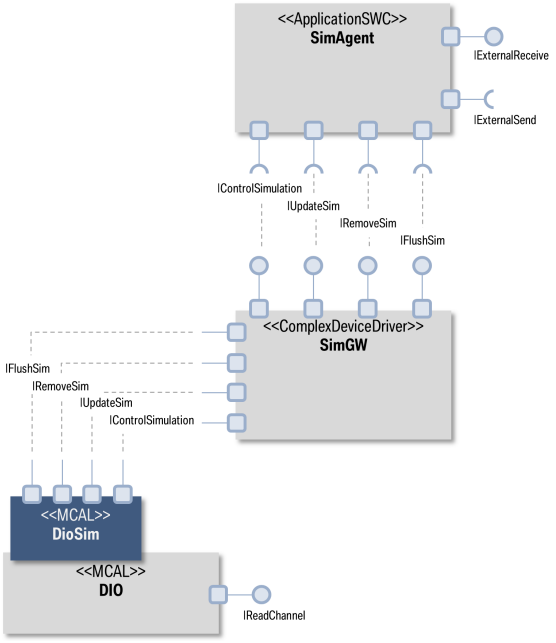

Figure 9. Software Architecture of Simulation module

The SimGW module is used to route the signals from the SimAgent to the DioSim. This is a temporary solution that we do not have to edit the I/O-Abstraction of the existing AUTOSAR basic software. Above the RTE there will be the SimAgent as a part of the *BMW System Function Software Components* next to the normal application software components (Figure 10). *BMW System Functions* are standardised software components that are integrated into each ECU, e.g Diagnosis- or DTC-Functions.

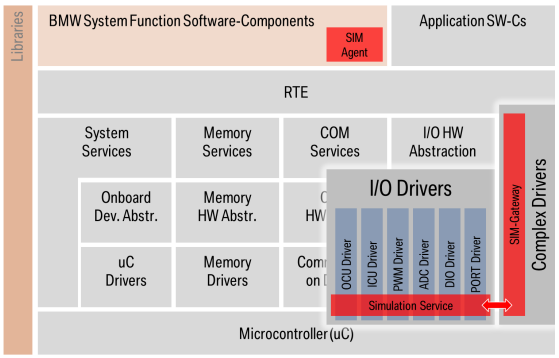

Figure 10. Integration of Simulation Interface in AUTOSAR Driver Layer

– The simulation process. – Enabling of the simulation will follow the sequence shown in Figure 11.

- *Init()*, activating the SimAgent via diagnosis job (CAN-Message),
- *FlushSim()*, erasing the existing data in the memory,
- *UpdateSim()*, setting initial values and parameters, like start value and start time,
- *EnableSimulation()*, activating the simulation service.

After enabling the simulation service, the *UpdateSim()*-Function is used to feed the DioSim-module with the data during the simulation.

The novelty here is that a test case is temporarily stored in the ECU memory and is executed by the SIM Agent. Each SIM module has to keep only the simulation data necessary for the ECU specific simulation to low memory requirements.

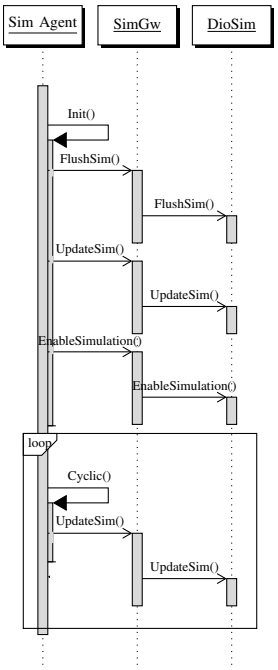

Figure 11. Process Start of Simulation Service [7]

Figure 12 shows the chain of reaction through the AUTOSAR basic software when a SWC requests data from the RTE. If there is a request of a RTE variable, the normal chain of reaction will be triggered. If the request reaches the Dio-module the Dio-module will check if there is a simulation active for the requested channel (*GetSimState()*) and will switch to the simulated data if required. In all other cases the physical state of the hardware I/O will be read.

The additional function request *GetSimState()* will be integrated in the Dio-module and will have an insignificant influence on the time response of the request. There will be no difference between the time response in normal customer use or in simulation usage, because in both cases the new additional function request will be triggered.

The worst case execution time analysis (WCET) calculates a percentage increase of the processor about 3,3%. This is a calculation for 255 simultaneously simulated channels, on an 80MHz CPU with a task cycle of 1ms and in case of the non-existence of simulation data. In this case the software reads, after failure of reading a simulated value, the physical value of the hardware I/O.

For the research an 80MHz CPU with a 12.5ns Assembler instruction execution time is used. This CPU is used to calculate execution times for 1ms tasks. The first one is with no simulation active and the second one is with simulation active without data:

 ISBN: 978-1-61208-458-9

- *DioSim_GetSimState=0* - [normal case]
  & no active Simulation
  WCET = 12,5ns according to 1 function call
- *DioSim_GetSimState=1* - [worst case]
  & no Simulation Data
  & Reading real data
  WCET = 32950ns

An impact could be a time delay being critical for a SWC. In normal case the SWC runs in tasks about 10ms and the delay will be about less than 0,01%.

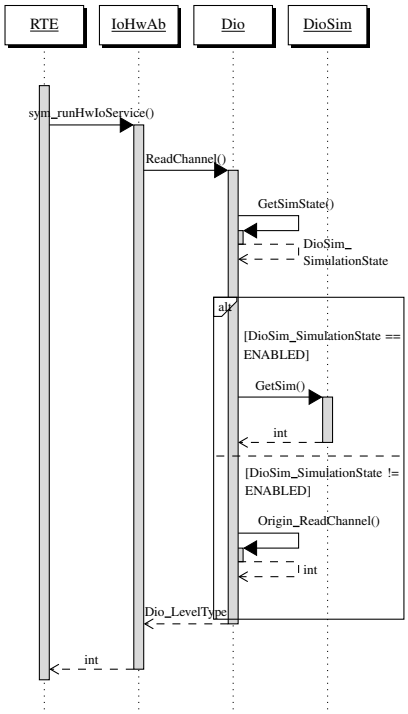

Figure 12. Simulation Service process [7]

Savety functions such as stopping and fall-back mechanisms have been integrated in the SimAgent. This allows to abort the simulation at any time in two ways: delayed or immediately. 'Delayed' for a smooth end of the simualtion. 'Immediately' in case of a system malfunction.

VI. Conclusion

The method of virtual hardware signal simulation, with an integrated software approach, allows to automate the user input analogous to the customer use cases and thereby to compare system- and test model in an innovative way. The approach shows a different solution with no need of special hardware equipment because of the integration of the simulation in the distributed system as a software-based distributed system. The methodology is realised as a standard module for easy integration in the AUTOSAR BSW to get an interface for the automation. The key aspect of this approach is the point of interaction located in the BSW driver layer. The methodology uses an abstraction to specific hardware input signals via mapping to reduce data and to keep the simulation module as generic as possible.

A simple generic simulation module controls the simulation process. Because of its simplicity, the simulation has a barely measurable effect on the CPU workload. The system reaction, respectively the system interaction with the customer and environment will be evaluated with proven and tested methods already in use. Therefore, there is no need in building up new evaluation methods and systems for the system analysis.

The new approach has a substantial similarity to the CCP approach in the connection layer: both interact at the driver layer. The enormous difference between both is the cutting of the data communication to the SIM Agent (slave), during simulation. This reduces CPU workload and the new approach is enables a simulation, much closer to the customer case.

Next steps for the implementation are to check the data size of the simulation sequences, especially for long-term simulations, beacuase memory space in automotive ECUs is scarce. The memory space in the automotive ECUs is associated with high cost hardware, and therefore the main constraint on the test case steps and the number of parallel simulated channels.

References

[1] Basycon Unternehmensberatung GmbH. (2006) Softwarequalität durch verbesserte Entwicklungsprozesse. [Online]. Available: http://www.basycon.de/de/web/basycon/publ_typ_poster [Jan. 5, 2016]

[2] M. Eigner and R. Stelzer, *Product-Lifecycle-Management: Ein Leitfaden für Product-Development und Life-Cycle-Management*, 2nd ed., ser. VDI. Berlin and Heidelberg: Springer, 2013.

[3] AUTOSAR Partnership. (2014) AUTOSAR Components. [Online]. Available: http://www.autosar.org/fileadmin/images/media_pictures/AUTOSAR-components-and-inte.jpg [Dec. 15, 2015]

[4] T. Roßner, *Basiswissen modellbasierter Test*, 1st ed. Heidelberg: dpunkt.verl., 2010.

[5] C. Saad and B. Bauer, Eds., *Model-Driven Engineering Languages and Systems: Data-Flow Based Model Analysis and Its Applications*. Springer, 2013.

[6] C. Saad and B. Bauer, Eds., *Industry Track of Software Language Engineering (ITSLE), 4th International Conference on Software Language Engineering (SLE 2011)(May 2011): The Model Analysis Framework An IDE for Static Model Analysis*, 2011.

[7] M. Köberl, "Integration softwarebasierter Automatisierungsmethoden in eine Test-ECU," Master's thesis, University of Augsburg, Augsburg, 2015.

[8] H. Kleinknecht, A. Krüger, H.-G. Kunz, R. Maier, H. Schröter, and R. Zaiser. (1999) Can Calibration Protocol - Version 2.1.

[9] AUTOSAR Partnership. (2014) AUTomotive Open System ARchitecture: Enabling Innovation. [Online]. Available: http://www.autosar.org/ [Dec. 15, 2015]

[10] B. Beizer, *Software testing techniques*, 2nd ed. New York: International Thomson Computer Press, op. 1990.

[11] D. W. Hoffmann, *Software-Qualität*, ser. EXamen.press. Berlin and Heidelberg: Springer, 2008.

[12] M. Pezzè and M. Young, *Software testing and analysis: Process, principles, and techniques*. [Hoboken and N.J.]: Wiley, ©2008.

[13] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*, 3rd ed. Hoboken and N.J: John Wiley & Sons, ©2012.

[14] R. Seidl, M. Baumgartner, and T. Bucsics, *Praxiswissen Testautomatisierung*, 1st ed. Heidelberg and Neckar: dpunkt, 2011.

[15] S. Byhlin, A. Ermedahl Jan Gustafsson, and B. Lisper, "Applying Static WCET Analysis to Automotive Communication Software."

[16] Vector Informatik GmbH, "AUTOSAR Configuration Process - How to handle 1000s of parameters: Webinar 2013-04-19," 2013.

[17] H. Balzert, *Lehrbuch der Softwaretechnik/2: Software-Management*, 2nd ed., ser. Lehrbücher der Informatik. Heidelberg: Spektrum Akad. Verl, 2008.

 ISBN: 978-1-61208-458-9