Embedded Selforganizing Systems

Issue Topic: “Towards Robust Situation Awareness in Autonomous Vehicles“

Towards Cloud-supported Automotive Software Development and Test

René Bergelt
University of Technology Chemnitz
Professorship of Computer Engineering
E-mail: rene.bergelt@informatik.tu-chemnitz.de

Norbert Englisch
University of Technology Chemnitz
Professorship of Computer Engineering
E-mail: norbert.englisch@informatik.tu-chemnitz.de

Abstract[1]—The development of automotive software has been an evolving process for the last decades. As a result, the paradigm of software development which is independent of the target hardware platform has been adopted in almost all parts of the automotive industry. Deploying software to a hardware platform is now controlled by an enormous parameter set stored in a mapping configuration. This led to the creation of numerous vendor-specific tools for electronic control unit (ECU) development. While this approach simplifies and supports the re-usability of vehicle functions it also increases the complexity as well as the difficulty for integration tests and error localization. In this paper, we present a conceptual platform which allows to establish references between different development and test phase items in a developer-friendly way. It revolves around two self-developed tools supported by an extensive AUTOSAR knowledge base. The system creates inter-connectivity so that it becomes easier to locate the actual origin of a misbehavior or to find a test error manifestation in the actual end system for developers and testers alike.

Keywords— automotive, software development, software test, cloud storage

I. Introduction

Automotive software development is an ever changing and evolving process. As a result, nowadays vehicle functionalities are developed based on the V-model, but in a hardware and communication technology independent way [1]. Consequently, a configuration which consists of large parameter sets defines the necessary constraints to deploy the functionality to an actual target hardware platform. This methodology resulted in a large number of diverse tools and a fragmented development environment for electronic control units (ECUs). In Europe, the de facto standard for platform-independent automotive software development is AUTOSAR [2], which in itself is a rather complex architecture and platform. By adhering to this standard, the development of reusable vehicle functions can be greatly improved both with regard to time and effort. However, this flexibility comes with a cost, namely a much higher difficulty for testing as well as major hurdles for error localization in the actual end system. Furthermore, the rising number of ECUs and functionalities per vehicle results in an increase of the number of communication messages as well.

When developing complex systems, it is vital that both functional and non-functional errors and their manifestation on the actual target platform can be found as early as possible in the development process or are prevented in the first place. This also holds true for configuration errors [3]. Therefore, there is a strong need to resolve behavioral relationships between the functionality and the required communication data of a single ECU under test as well as a large set of integrated ECUs in pre-series vehicles to allow efficient testing and validation at any stage of the development process.

II. Related Work

The usage of cloud-based platforms in the Automotive industry has seen a rising trend in the last years. In general, there are three different areas where they are applied: market analysis, cloud-based real-time driving services and product development. While the first is mainly a business-related topic and as such not in the scope of this paper, the last two are in fact technological challenges. The research field of autonomous driving has led to numerous approaches and use cases where access to cloud platforms plays a central role in the customer’s car [4], [5]. Moreover, it has been advised that big data and thus cloud-based computing can also be applied as early as during the development and test phases of automotive software [6]. A number of commercial solutions has been devised in this direction such as Elektrobit Assist Test Lab[2]. Mostly, such tools focus on automating test recording and evaluation through a cloud-computing platform. Unfortunately, no publicly available tool is able to guide developers between code base and

[1] Copyright © 2020 by ESS Journal

[2] https://www.elektrobit.com/products/automated-driving/eb-assist/test-lab/

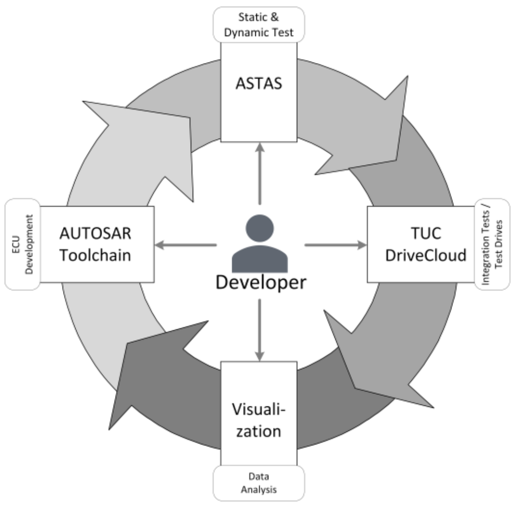


Figure 1 Development process with ASTAS and TUC DriveCloud connected

test assessment through sophisticated cross-references. This means, that much effort has to be put into finding either the actual location of an implementation problem which led to an error in the actual end system or vice versa. Additionally, static testing and validation results are not taken into account either. Furthermore, much of the complexity of testing AUTOSAR-compliant software stems from the fact that many parameters, which would have been present directly in code in the past, are now buried in a large number of configuration files - in standard-compliant as well as vendor-specific file formats. Thus, locating the actual origin of an error is a major problem. This is further complicated by tool-specific differences as the AUTOSAR standard explicitly encourages competition between vendors with regards to the actual implementation. Consequently, the research challenge considered in this paper is to connect different development and test phases in the automotive software development cycle by extended tooling support. The main objective is to make it easier for developers and testers to find problematic coding locations and test failures or misbehavior caused by such code through cross-referencing objects and test cases between different systems which are usually used at different project stages.

III. Cloud-supported Automotive Software Test

This paper presents an approach to cloud-supported automotive software test by coalescing the already existing, separate systems ASTAS (application specific test of AUTOSAR Systems) and TUC DriveCloud into a new, combined platform which supports developers with automotive software development and test at all stages of a project. The realization is backed by an extensive AUTOSAR Knowledge Base. It helps developers in testing complex AUTOSAR-driven software architectures by cross-referencing source code origins, static and dynamic test case failures or warnings and their manifestation in recorded test drive data. Consequently, the platform attaches to all phases of V-model based AUTOSAR-compliant development. This encompasses system specification, functional testing as well as system integration testing. The targeted solution is shown in Figure 1. First and foremost, we see the developer at the center of the process where they are able to step into different stages of development and testing easily. It can be seen that the output of each step serves as input to the next step. The developer may assume any of the following roles during a project step and is guided in a comprehensible way by our platform:

Software Engineer Implementation of AUTOSAR compliant ECU software

Test Engineer Setting up test requirements based on the specification & test evaluation

Data Analyst Identifying unwanted behavior (i.e. error manifestations) in data traces and finding the corresponding origins in code

A major advantage is that our system is able to produce and visualize results in different levels of detail, so that users can focus on the information they need and which suits their knowledge level of the target system.
The next section will give an overview of the actual conceptional combination of the existing tools and the requirements which have to be fulfilled.

IV. Conceptual Challenges

This section lists the main research challenge to face during realization and all subsequent challenges which have to be solved with regard to the integration of the final platform. The targeted architecture of the platform is shown in Figure 2. The car for the test drive has a data recorder which is connected to a bus system in the car, for example by OBD. A smartphone is connected to this data recorder by a wireless connection and enables the driver to label special events during the test drive. The data recorder has a wireless connection to the network

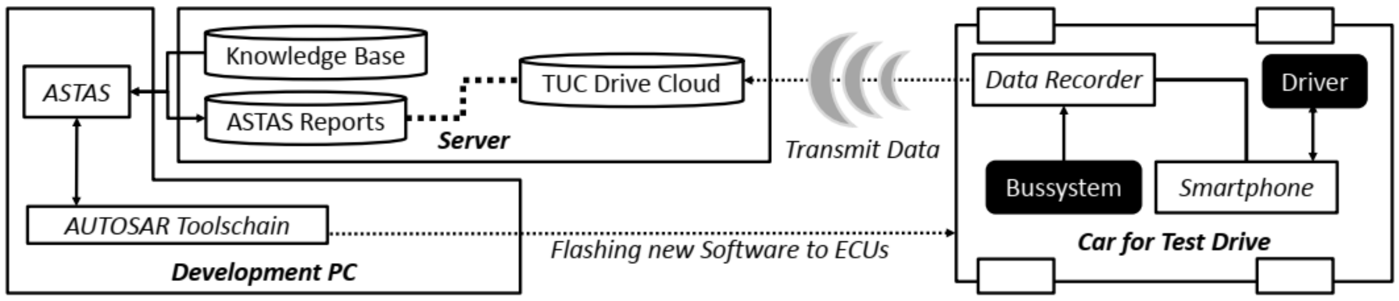


Figure 2 Architectural overview of the proposed platform

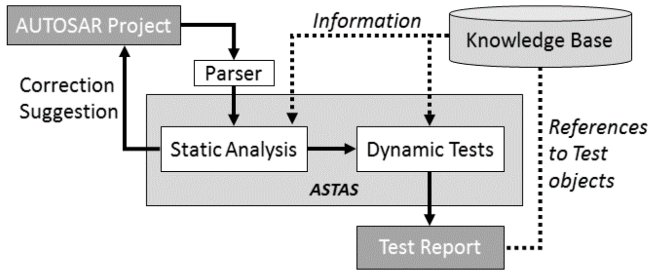


Figure 3 Abstract Workflow of ASTAS

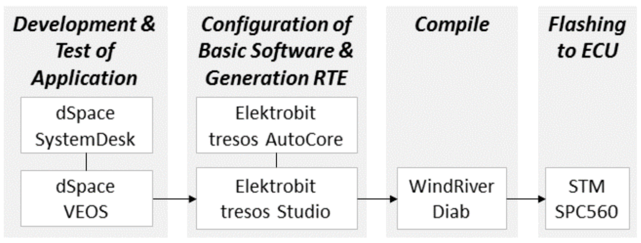
infrastructure which provides access to TUC DriveCloud as well as the AUTOSAR Knowledge Base and the database for ASTAS reports. ASTAS itself is executed on a Development PC where the AUTOSAR toolchain for the development of AUTOSAR ECUs is installed. It has to be ensured that hardware and software for the recording of data in the test drive car are able to record all data from the start to the end of a test drive. The computing platform, memory and processor have to be appropriately selected to be able to process the data. Because of different data sources with different sample rates, the recording needs to fuse the data within a common time stamp. The recording of a test drive should not have any influence on the behavior of the test object. The recorded data is transmitted to the TUC DriveCloud in an online or offline manner. When in online mode data is submitted during the test drive and can be viewed as soon as it has been uploaded. Offline transmission stores the data in local memory and submits it after end of a test drive and when connectivity to the TUC DriveCloud is given, e.g. when the test drive car enters a garage with wireless LAN access. The following sections provide a more detailed look into the different aspects of the platform.

A. The ASTAS platform

The testing tool ASTAS supports AUTOSAR specific static analysis for the three main horizontal architecture layers – application, Runtime Environment (RTE) and basic software. Moreover, it generates application specific test cases for basic software modules or clusters and RTE functions. The static analysis and dynamic tests are executed during a test run, which generates test reports, as shown in Figure 3. Before the test run, the test preparation phase processes the source code files as well as the configuration files of the given AUTOSAR project. Each test run can be defined by a set of different test modules, realizing a specific test aspect [7]. Each test module defines its compliance to a set of AUTOSAR versions. By that, ASTAS test reports represent results of static analysis [8] and dynamic test from AUTOSAR components [9]. ASTAS was mainly developed to improve the quality of software projects in the automotive domain. By that, the ASTAS test report can contain:

Violation of AUTOSAR code compliance given by standard or extra guidelines (static test)

Misconfigured architecture on any architectural layer (static test)



Development & Test of Application
dSpace SystemDesk
dSpace VEOS
Configuration of Basic Software & Generation RTE
Elektrobit tresos AutoCore
Elektrobit tresos Studio
Compile
WindRiver Diab
Flashing to ECU
STM SPC560

Figure 4 Integrated AUTOSAR Toolchain

Incorrect functionality of AUTOSAR software modules in the basic software or RTE (dynamic test)

Timing information of AUTOSAR software modules [9] for application specific values on the target platform (dynamic test)

To be able to map these results to data from test drives, these test results should have some relation to data which can be measured in a test drive. This leads to a relation of an ASTAS test result to bus messages or I/O data. This relation can be evaluated by the data stored in the AUTOSAR configuration of a project. For that, official file formats as defined by the AUTOSAR consortium have to be analyzed, as well as product specific configuration files.

For each change in an AUTOSAR project which is flashed to the car used in the test drive, a new ASTAS test report needs to be generated. This enables to focus on the correct bus messages or I/O information for the test drive.

To be able to localize problems in the huge number of parameters in an AUTOSAR project, each single test result should contain a link to a software module within the AUTOSAR architecture. This can be determined during the test case generation which accesses the AUTOSAR Knowledge Base [10]. By that, a set of sensor data from a test drive can be associated to a set of AUTOSAR software modules.

The AUTOSAR Knowledge Base is the major database for the test generation and test execution in ASTAS. It contains information for each AUTOSAR version combined with the tool environment and the hardware platform. This enables data handling independent of the tool provider. Internally, the Knowledge Base handles a set of items. For example, such an item can represent a basic software module, a function of a basic software module or a layer of the basic software. Additionally, each item can represent a test object for the dynamic test or static analysis in ASTAS.

Currently, the ASTAS platform controls three different AUTOSAR versions and various tools. However, in our laboratory the most used version up to now is the toolchain of AUTOSAR version 3.2. As shown in Figure 4 this toolchain contains the tool dSpace System Desk for development of an AUTOSAR application. This application can be tested in the environment of dSpace VEOS. After that, tresos AutoCore, which is a product from Elektrobit can be configured by tresos Studio. Additionally, tresos generates the RTE, the middleware between application and basic software. The source code of our AUTOSAR projects is compiled by the Diab compiler of WindRiver and flashed to our hardware platform STM SPC560.

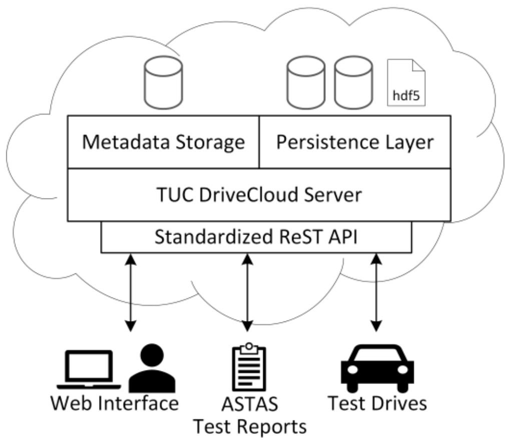


Figure 5 Architectural overview of TUC DriveCloud

B. TUC DriveCloud

TUC DriveCloud is an online, big data storage platform for automotive test drive data. Basically, it serves as a structured data container which can only be assessed through a single, standardized API. This ensures data consistency and integrity across all connected devices. Additionally, it allows to enforce security features at a single gateway. This basic structure is shown in Figure 5. One main feature of TUC DriveCloud is that metadata storage is completely separated from the storage of the actual sensor measurement recordings (persistence layer). Consequently, the platform is easily scalable in choosing the right storage technology based on the sensor data format. For example, basic single value measurements can be stored in a traditional database system for fast access, but complex sensor values such as image sequences, point clouds or audio data can be stored in more appropriate formats, such as HDF5. However, this is completely transparent for users or services due to the standardized API. Sensor data is always uploaded and retrieved in the same way, independent of its actual storage technology and location. In order for meaningful references to be created between different systems TUC DriveCloud stores a large set of metadata associated with test drives and vehicles, where each dataset is associated with a *globally* unique identifier (UUID, Universally Unique Identifier). This enables a high level of non-conflicting parallel processing and distributed storage as the probability for identifier collisions is neglectable [11]. Thus, different datasets can be created by concurrent systems at the same time and later be merged easily. Furthermore, stable cross-references with ASTAS can be created based on these unique identifiers. Currently, the main meta data entities stored in TUC DriveCloud are:

Vehicle Definition of an object for which sensor data can be recorded (e.g. a real car, a virtual car)

Sensor Definition of a measurement object which delivers timestamped data values as part of a vehicle

Stream Metadata for an actual, recorded sequence of sensor values

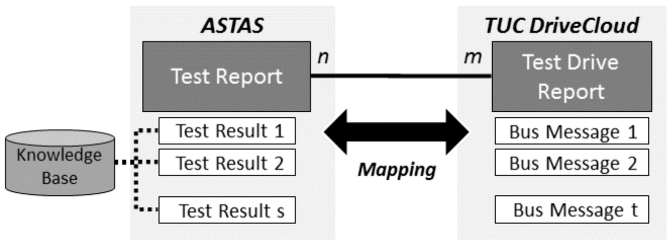


Figure 6 Mapping between a set of test results and events from test drive

Test Drive Information about a test drive, such as start and end time, recorded streams

Label Keywords which can be added to a test drive (manually or automatically)

In general, the TUC DriveCloud provides functionality to map test drive data to vehicles, which are or contain the test objects [12]. Due to different development levels and debugging loops, multiple sets of functionalities are definable for a vehicle. Furthermore, the automatic detection of critical or erroneous situations based on the input data from ASTAS is a vital feature to support developers and testers.

C. Mapping of Test Drive and Test Reports

A huge improvement potential for debugging and optimization is held by the connection of results from ASTAS test reports and test drive reports from the TUC DriveCloud through cross-references. As shown in Figure 6 an ASTAS Test Report contains a set of Test Results and each of these results has a reference to the Knowledge Base. For example, a Test Result can indicate a misconfiguration of a basic software module or a timing problem of the whole communication stack. In contrast to this, a Test Drive Report from the TUC DriveCloud contains a set of bus messages and sensor values. Usually, the main challenge and most time-consuming activity during the evaluation of data from a test drive is the localization of the implementation problem and deciding on the necessary changes in configuration or programming. The time needed for the localization can be greatly improved by using cross-references. For communication processes based on the AUTOSAR configuration, the mapping can be realized as analyzed by ASTAS. As a result, each test object from ASTAS is associated with concrete bus signals inside the configuration. In the case of a failed test, this bus signal can be linked to the bus messages in the Test Drive Report by referring to the Knowledge Base. Furthermore, a cloud-supported test platform, such as realized by the connection of ASTAS and TUC DriveCloud, is capable to improve the efficiency of automotive development cycles in many more aspects. For example, TUC DriveCloud can automatically identify signals which are most likely relevant to the test case, by observing bus traffic and sensor data fluctuation over time. If ASTAS detects fundamental problems in a project under test, TUC DriveCloud can inform test engineers and drivers that the test drive should probably be postponed. Additionally, when misbehaviors or errors are detected during a test drive, a corresponding test module for ASTAS can be generated through TUC DriveCloud. The

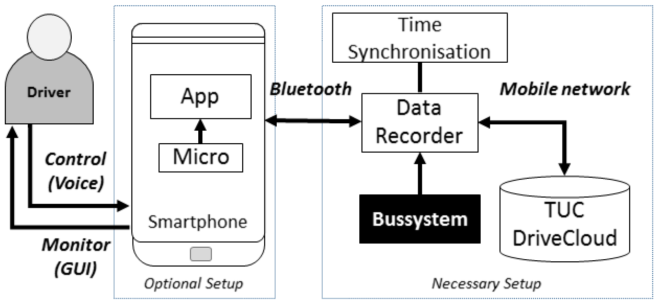


Figure 7 Smartphone Integration

module will check for the existence of exactly this fault in future versions of the software project. This poses an effective way at supporting automatic regression testing, once the actual error has been fixed in the future.

D. Smartphone integration

As an optional setup, a smartphone can be integrated with the data recording system in the car. During test drives, the driver should be able to label a situation, i.e. define keywords which briefly describe the driving situation. Doing this through a touchscreen or similar interface which needs to be controlled by the driver's hands either requires the test driver to stop the car beforehand or imposes an additional safety risk. Due to this we suggest a speech-controlled solution by using the microphone of a smartphone which allows the driver to give instructions by voice. The smartphone app can then be used to start and stop the recording of a test drive and to add labels to the recording. Additionally, it can visualize important data in an abstract way. By that, the driver can be informed whether the recording works or if there are problems. As visible in Figure 7, we establish a Bluetooth connection between our Data Recorder and the smartphone. Consequently, the Data Recording Unit now needs to fuse the oral information from the driver with the data recorded from the bus system of the car.

V. CONCLUSION

When developing algorithms for autonomous driving and as such situation awareness, it is vital that developers and testers can quickly cross-reference actions and behaviors between different stages of the development process. Accordingly, this paper has presented the major challenges for a cloud-based system which enables just this by storing data from development, test and test drives in a structured way. We have introduced our technical realization, consisting of a testing environment for AUTOSAR systems and a cloud-based platform for storing and referencing test drive data.

The system facilitates common debugging scenarios by providing a way to move between architectural definition, implementation and manifestation in the integrated system on one platform. Additionally, it helps with the validation of functions as well as the reproducibility of test cases since the relationship between code, found errors and corresponding test data is stored in one place. This allows developers and testers to concentrate on the task at hand and relieves them of most of the management overhead which is usually needed to keep an overview of test protocols, test data and tested code version. This holds the potential of increased productivity and an improved communication between testers and developers alike.

REFERENCES

[1] S. Mathur and S. Malik, "Advancements in the V-Model," *International Journal of Computer Applications,* no. 12, 2010.

[2] AUTOSAR GbR, "AUTOSAR," [Online]. Available: www.autosar.org. [Accessed 20 07 2020].

[3] T. Xu, X. Jin, P. Huang, Y. Zhou, S. Lu, L. Jin and S. Pasupathy, "Early Detection of Configuration Errors to Reduce Failure Damage," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016.

[4] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth and G. Wu, "A Digital Twin Paradigm: Vehicle-to-Cloud Based Advanced Driver Assistance Systems," in *IEEE 91st Vehicular Technology Conference*, Antwerp, Belgium, 2020.

[5] L. Gu, D. Zeng and S. Guo, "Vehicular cloud computing: A survey," in *IEEE Globecom Workshops*, Atlanta, USA, 2013.

[6] M. Johanson, S. Belenki, J. Jalminger, M. Fant and M. Gjertz, "Big Automotive Data: Leveraging large volumes of data for knowledge-driven product development," in *IEEE International Conference on Big Data*, Washington, DC, USA, 2014.

[7] D. Markert, "Entwicklung einer generischen Testumgebung für Automotive Software Systems," 2017.

[8] R. Mittag, "Entwicklung Statischer Analysen für AUTOSAR Steuergerätesoftware," 2017.

[9] N. Englisch, F. Hähnchen, F. Ullmann, A. Masrur and W. Hardt, "Application-Driven Evaluation of AUTOSAR Basic Software on Modern ECUs," in *Proceedings of the 13th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, 2015.

[10] N. Englisch, R. Mittag, F. Hähnchen, O. Khan, A. Masrur and W. Hardt, "Efficient Static Testing of AUTOSAR Software supported by an automatically created Knowledge Base," in *Proceedings of the 7th Conference on Simulation and Testing for Vehicle Technology*, 2016.

[11] M. Mealling and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," The Internet Society, 2005.

[12] I. Mühlmann, "Generische Anbindung von Testfahrtdatenquellen an ein Automotive-Cloud-System," 2019.