# Automatic Consequence Analysis of Automotive Standards (AUTO-CAAS)

[Position Paper]

Thomas Arts
Quviq AB
Gothenburg, Sweden
thomas.arts@quviq.com

Mohammad Reza Mousavi[*]
Center for Research on Embedded Systems
Halmstad University
Halmstad, Sweden
m.r.mousavi@hh.se

## ABSTRACT

This paper provides some background and the roadmap of the AUTO-CAAS project, which is a 3-year project financed by the Swedish Knowledge Foundation and is ongoing as a joint project among three academic and industrial partners. The aim of the project is to exploit the formal models of the AUTOSAR standard, developed by the industrial partner of the project Quviq AB, in order to predict possible future failures in concrete implementations of components. To this end, the deviations from the formal specification will be exploited to generate test-cases that can push concrete components to the corners where such deviation will result in observable failures. The same information will also be used in the diagnosis of otherwise detected failures in order to pinpoint their root causes.

## Categories and Subject Descriptors

C.3 [**C.3 Special-purpose and Application-Based Systems**]: Real-time and embedded systems; D.2.5 [**Testing and Debugging**]: Diagnostics

## General Terms

Verification, Reliability

## Keywords

AUTOSAR, Model-Based Testing, Finite State Machines, QuickCheck, Diagnostics

[*]Mousavi's research has been partially supported by the Swedish KK Foundation (Stiftelsen för Kunskaps- och Kompetensutveckling award number(s): 20140302 (AUTO-CAAS) and the Swedish Research Council (Vetenskapsrådet) award number(s): 621-2014-5057 (Effective Model-Based Testing of Parallel Systems).

*WASA'15,* May 4, 2015, Montréal, QC, Canada.
ACM 978-1-4503-3444-0/15/05.
http://dx.doi.org/10.1145/2752489.2752495.

## 1. INTRODUCTION

The Automotive Open System Architecture (AUTOSAR) standard is gaining momentum with several automotive manufacturers (such as Volvo) and there is a growing trend towards new vehicle platforms based on the latest versions of this standard. The standard enables manufacturers to allow Tier-1 suppliers to contract arbitrary Tier-2 software developer for ECUs, as long as the developed software conforms to the specified behavior according to AUTOSAR. This is in clear contrast to earlier situation, in which a preferred Tier-2 developer was appointed to develop software for all Tier-1 hardware suppliers. This paradigm shift brings about economical and financial benefits (both for suppliers and manufacturers). However, it also introduces certain risks and challenges.

The AUTOSAR standard is complex and does leave room for interpretation and optimizations. In order to be competitive, Tier-2 developers strive after implementing several optimizations and utilizing room for interpretation of the standard to make their product out-perform in the competition.

The AUTO-CAAS project, for Automatic Consequence Analysis of Automotive Standards, aims at exploiting the technology of model-based testing in order to detect deviations from the AUTOSAR standard and furthermore trace the consequences of such deviations into visible deviating behaviors (failures). The project has started as of March 2013 and will continue for 3 years. It is funded by the Swedish Knowledge Foundation and involves 3 partners: Halmstad University, Quviq AB and ArcCore AB. Halmstad University is the knowledge provider in this project. It provides the necessary knowledge and research skills in order to co-develop a novel, yet practical, approach for the research problem and co-develop it within the toolsets provided by QuviQ and ArcCore. The role of QuviQ in this project is to provide both the tools and the models for the conformance testing according to the AUTOSAR standard. The role of ArcCore in this project is to provide the implementation and the implementation platforms for the AUTOSAR standard, respectively, based on their Arctic Core and Arctic Studio products, which provide support for various levels of AUTOSAR development and integration.

In the remainder of this paper, we provide an overview of the roadmap for the AUTO-CAAS project. The rest of this paper is organized as follows. In Section 2, we provide an overview of the background technologies (theories and tools)

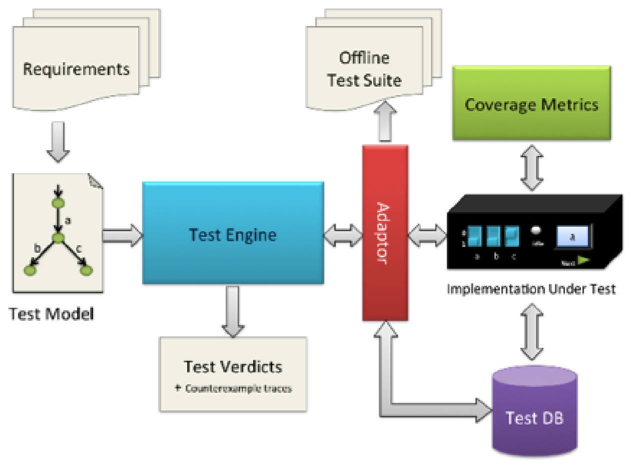

**Figure 1: A typical eco-system for Model-Based Testing**

that are to be used throughout the project and in Section 3, we sketch our expected results and our research approach.

## 2. BACKGROUND THEORIES AND TOOLS

In this section, we provide an overview of the theories and tools that form the foundations of the AUTO-CAAS project.

### 2.1 AUTOSAR

Standardizing the conformance testing process has been mentioned as a key goal of the AUTOSAR standard [7]. The recent releases of the AUTOSAR standard have partially achieved this goal by providing a standard for interfaces, behaviors and configurations for basic software [1]. According to AUTOSAR, conformance and inter-operability tests take place at various levels in the development of vehicle functions: starting from testing individual components, going on with (micro-) integration to modules, integration into ECUs and single vehicle functions after integration into the network. A final step of testing is performed in the operational environment involving multiple vehicle functions. The scope of this project is from conformance testing of module micro-integration to testing single vehicle functions. In particular, we exploit the results of model-based conformance testing to predict and diagnose inter-operability failures at the vehicle function level. Conformance test results are very helpful in identifying signature faults as well as behavioral faults. However, there are "gray areas" [8] in the results of conformance testing, which are non-conclusive. The main reason for the existence of these gray areas is due to the under-specification of the context in which the component interfaces are being used. This leaves some room for various, at times conflicting, design decisions by the suppliers and the OEMs. Such conflicting design decisions can give rise to later failures when composing modules into ECUs and ECUs into vehicle functions.

### 2.2 Model-based testing

Model-based testing (MBT) [4] is a rigorous and structured technique to test computer systems. A schematic view of a typical MBT ecosystem is given in Figure 1. The figure refers to the ecosystem used in our earlier experience with model-based testing of an embedded system in the financial domain [3]. The MBT process starts with making test models from the requirements and standards. Then a conformance test engine is in charge of generating test cases from the test models. Subsequently, the generated test cases are executed in order to interact with the implementation under test and to establish whether it conforms to the specification.

Examples of MBT test engines include Microsoft SpecExplorer [17], UPPAAL TRON [11], UPPAAL Yggdrasil, RT-Tester [15] and QuickCheck [2, 12]. We refer to [3, 12, 18] for our prior experience with industrial application of some of these tools.

In the context of this project, we will use the property-based testing approach as implemented in the QuickCheck tool. In particular, we will use the existing rigorous specification of the AUTOSAR standard in QuickCheck. The property-based approach improves upon traditional model-based testing techniques by allowing for programming-language abstractions for specifying data generators. It also allows for incrementally building finite-state machine models by gradually learning from the interaction with the system under test. The shrinking approach implemented in the QuickCheck tool provides an effective means for generating the minimal test-case, hence, defining a clear focus in the process of fault detection and diagnosis.

Our industrial partner ArcCore provides both examples of experimental correct implementation of components and modules, as well as examples of components and modules that demonstrate non-conforming behavior. We also plan to use standard fault injection and mutation techniques for generating a wide range of possibly incorrect components and modules. In addition, ArcCore provides a development environment for AUTOSAR in order to modify and develop new examples at will.

When composing such components and modules, a summary of detected (non-conforming) behavior will be used to predict possible failures. Moreover, in case of actual failures, the behavioral summary model built through interaction with the implementations will be used to designate the root cause of failure.

### 2.3 Fault diagnosis and automated debugging

Fault diagnosis and model-based fault diagnosis have a long tradition in dynamical systems and supervisory control [13, 6, 14]. Fault diagnosis ideas have subsequently been exploited in computer systems, e.g., in the form of spectrum-based diagnosis of software systems [10]. In spectrum-based diagnosis, passed and failed executions are scrutinized, and annotated with information about the execution of each line (block or module) of program code. Note that for diagnosis, one does not differentiate whether in a particular run a block caused the failure or not; it is just checked whether the block is part of the whole execution. An alternative approach to spectrum-based fault diagnosis, which uses more semantic information, is delta debugging [21]. It uses a set of passing and failing conditions in order to efficiently uncover a small failing execution. Subsequently, the comparison of system states in passing and failing runs can reveal the root cause of failure in this approach.

To exploit these approaches in our context, we exploit the extra information obtained in the process of conformance testing to narrow down the search process when applying fault diagnosis and debugging techniques. We envisage that exploitation of these extra pieces information, which are made available through conformance testing will result in faster and more accurate diagnosis. We have tried both spectrum-based diagnosis and delta debugging in our past research [20] and hence, do have in-house knowledge about both.

## 2.4 Symbolic execution and concolic testing

Symbolic execution has been successfully applied to test and verify computer (particularly software) systems in the past ten years [19, 5]. To apply symbolic execution in software testing, one usually starts by running the system under test (symbolically or concretely with random seed values) and following the execution trace until reaching decision points. Conditions at decision points are accumulated along the execution and by using constraint solvers (such as powerful satisfiability-modulo-theory solvers), the obtained conditions are turned into concrete valuations for parameters. Hence, new concrete test cases are obtained, leading to maximum coverage of the code. This technique is often called "concolic (a combination of concrete and symbolic techniques in) testing". In our context, concolic testing can be particularly useful in producing summaries of models and implementation during the conformance testing process, similar to the approaches reported in [9, 16]. In this context, we will exploit the results obtained from a companion project (EFFEMBAC), which focuses on the integration of model-based testing with concolic testing. While EFFEMBAC focuses on the establishing a fundamental link between these two techniques, AUTO-CAAS will apply these techniques on the concrete case studies at hand.

## 3. EXPECTED RESULTS AND APPROACH

We develop an automated diagnosis approach that exploits the information from the conformance testing process. In this case, the goal is to predict whether the integration of concrete realizations will lead to any failure. Additionally, we use the information gathered during the conformance testing process in order to diagnose the later observed integration and vehicle-function level failures and find the root cause of failure. This will resolve a contemporary problem in the application in the practice of automotive software and systems. Particularly, use of the popular AUTOSAR standard in our research forms the basis of our model-based approach and enables its wide-spread application in industrial practice. Next, we give the steps involved in achieving these goals in more detail.

## 3.1 From conformance testing to summaries

The first technical milestone of the project is to create model and implementation summaries that compactly describe the variation points between the AUTOSAR model and the implemented component / module. These symbolic models also include possible parameterizations of the implementation. To this end, we define a formal framework that is expressive enough to act as the common semantic domain both for AUTOSAR models and implementations. Moreover, we shall specify the semantic properties and define composition and reduction techniques for such models and implementations and prove them correct.

Next, we produce summaries of deviating behavior during the process of conformance testing for individual components and modules. We envisage to use ideas from concolic testing to make this extraction possible and efficient. We first provide a mathematical description of this process and apply it to small examples. After successful application of the mathematical definitions to small examples, we will implement these definitions and integrate them with QuickCheck.

## 3.2 Aligning failures with summaries

Next, we use the summary of mismatching behaviors gathered in the previous phase in order to both predict possible failures, as well as exploit an existing failure trace in order to find the root cause for failure. Both goals boil down to a guided (respectively, forward and backward) search in the composition of summaries in order to reach a failure (a particular failure in the second case). In this task, we will exploit and integrate the existing bodies of knowledge in the (symbolic) model checking literature and the automated debugging and fault diagnosis approaches. Similar to the previous step, we first focus on the mathematical definition of an abstract framework for predicting failures from composition of component models and deviation summaries, as well as analyzing observed failure traces into deviations of individual components. Subsequently, we apply simple examples distilled from realistic models and implementations to validate our models and algorithms. After having gained sufficient confidence (both by having proven formal properties as well as successful application to small examples), we implement the developed techniques and apply them to larger case studies.

Throughout the project and as a bi-product of our continuous involvement with examples of correct and incorrect AUTOSAR components, we build a benchmark library of such components. This benchmark will be made freely available online for researchers and practitioners in the area of testing AUTOSAR-based software.

## 4. REFERENCES

[1] AUTOSAR BSW & RE Conformance Test Specification, Release 4.0, Revision 2, 2011.

[2] T. Arts, J. Hughes, J. Johansson, and U. Wiger. Testing telecoms software with Quviq QuickCheck. In Proc. of ERLANG'06, ACM, 2006.

[3] H.R. Asaadi, R. Khosravi, M.R. Mousavi, and N. Noroozi. Towards Model-Based Testing of Electronic Funds Transfer Systems. In Proc. of FSEN'11, vol. 7141 of LNCS, Springer, 2012.

[4] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. (Eds.) Model-Based Testing of Reactive Systems. volume 3472 of LNCS, Springer, 2005.

[5] Cristian Cadar, Patrice Godefroid, Sarfraz Khurshid, Corina S. Pasareanu, Koushik Sen, Nikolai Tillmann, Willem Visser: Symbolic execution for software testing in practice: preliminary assessment. In Proc. of ICSE 2011, ACM, 2011.

[6] R.J. Chen and R.J. Patton. Robust Model-Based Fault Diagnosis for Dynamical Systems. Kluwer, 1999.

[7] H. Fennel et al., Achievements and Exploitation of the AUTOSAR Development Partnership, SAE Convergence Congress, Detroit, 2006.

[8] A.A. Gilberg, B.B. Kunkel, C.A. Ribault, D.P. Robin, and E.N. Spinner. Conformance Testing

for the AUTOSAR Standard, Embedded Real Time Software and Systems Conference, 2010.

[9] P. Godefroid. Compositional dynamic test generation. In Proc. of POPL 2007, IEEE, 2007.

[10] M.J. Harrold, G. Rothermel, K. Sayre, R. Wu, and L. Yi. An empirical investigation of the relationship between spectra differences and regression faults. Software Testing Verification and Reliability, 10(3):171–194, 2000.

[11] A. Hessel, K.G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou. Testing real-time systems using UPPAAL. In Proc. of FMT'08, vol. 4949 of LNCS, Springer, 2008.

[12] J. Hughes. QuickCheck testing for fun and profit. In Proc. of PADL'07, Springer, 2007.

[13] R. Isermann. Supervision, fault-detection and fault-diagnosis methods - an introduction. Control engineering practice, 5(5): 639–652, 1997.

[14] R. Isermann. Model-based fault-detection and diagnosis-status and applications. Annual Reviews in control, 29(1), 71–85, 2005.

[15] J. Peleska and W.-l. Huang, Model-Based Testing With RT-Tester. Lecture slides of the HSST'13, Halmstad University, 2013.

[16] J.H. Siddiqui and S. Khurshid. Scaling symbolic execution using staged analysis. ISSE 9(2): 119–131, 2013.

[17] M. Veanes, C. Campbell, W. Grieskamp, W. Schulte, N. Tillmann, and L. Nachmanson. Model-based testing of object-oriented reactive systems with Spec Explorer. In Proc. of FMT'08, vol. 4949 of LNCS, pp. 39–76, Springer, 2008.

[18] V. Vishal, M. Kovacioglu, R. Kherazi, and M.R. Mousavi. Integrating Model-Based and Constraint-Based Testing Using SpecExplorer. In Proc. of MoTiP'12, IEEE CS, 2012.

[19] N. Williams, B. Marre, P. Mouy, and R. Muriel. PathCrawler: Automatic Generation of Path Tests by Combining Static and Dynamic Analysis. In Proc. of the EDCC'05, pp. 281–292. Springer, 2005.

[20] M. Woehrle, R. Bakhshi, and M.R. Mousavi. Mechanized Extraction of Topology Anti-patterns in Wireless Networks. In Proc. of iFM'2012, vol. 7321 of LNCS, Springer, 2012.

[21] A. Zeller and R. Hildebrandt. Simplifying and isolating failure-inducing input. IEEE Transactions Software Eng. 28:183–200, 2002.