Model-Based Testing Design for Embedded Automotive Software

Anila Mjeda, Pat McElligott, Kevin Ryan, Steffen Thiel
Lero – The Irish Software Engineering Research Centre, University of Limerick

ABSTRACT

The ever increasing complexity of embedded automotive software is not matched by the current development and test processes of automotive embedded software and the latter have become the limiting factor.

A model-based software development and testing approach has the potential to reduce software development times, to produce executable specifications very early in the process as well as facilitate automatic code generation. Not surprisingly, the above are regarded as highly beneficial for the automotive industry. The automotive industry is increasingly using model-based testing techniques. Despite this, model-based testing tends to be done in a bespoke and non-systematic fashion [1] and easy to use, high quality, formalised, model-based testing methodologies that cater for the specific needs of in-vehicle software are hard to find.

This paper proposes a systematic model-based testing design approach which builds on previous work on systematic model-based testing for embedded automotive software [2], [3], [4]. The testing design is based on the functional requirements for the system under test and the test data are generated via two different and independent routes.

INTRODUCTION

In-vehicle software is becoming ever more complex and is increasingly taking over more functionality, fuelling up the need for “safe/infallible” automotive software. In the automotive domain, while the driver can assume part of the fault, software defects can have quite drastic consequences.

Established standards such as IEC 61508\(^1\), address the quality issue in a prescriptive manner and provide guidelines on software development processes. The IEC 61508 standard classifies the degree of rigor required in software development, verification, validation and testing of a system/module in terms of Safety and Integrity Levels (SILs). The standard mentions a generic list of methods that could be used as part of the software development process, but it gives no description of the actual processes needed to have in place in order to produce software that would adhere to a specified SIL. Furthermore, there is no mechanism to quantitatively measure that the software actually achieves a SIL target.

While testing alone can not possibly address all the quality aspects of software, sophisticated and powerful domain specific testing methodologies can certainly give a valuable contribution. Current trends for standardization such as AUTOSAR [5], open the automotive market for new suppliers and reliable, confidence inspiring testing technologies are needed to facilitate this process.

MODEL-BASED TESTING

In the automotive domain, the need to deal with the complexity of hybrid systems is often targeted via

---

\(^1\) The IEC 61508 standard was originally developed by the process and automation industries and it is composed of seven parts (IEC 61508-1 to IEC 615098-7). IEC 61508-3 focuses on software.
building functional models based on the functional specification for the system and, to date, there is considerable research gone into this [6] and its benefits [7]. In fact, model-based development is well established and increasingly becoming a favorite approach in the automotive industry [8], [9], [10], [11].

Typically, model-based testing relies on specification based behavior models of the system to build executable test cases. Aiming to strike the right economic balance, the problem always comes down to choosing an adequate representation of tests from the complete set of possible tests and this selection will be the deciding factor on the quality of the testing process. While the manual derivation of test cases is not excluded, the real focus is into automatically generated test cases. This implies that the models upon which this type of testing is based should have a level of formality which allows automatic test generation.

Despite the affinity of model-based testing with the automotive industry, it tends to be done in a bespoke and non systematic fashion [1] and online, fully automated, model-based testing technologies are still missing.

**RELATED WORK**

Available model-based-testing technologies used in the automotive domain to date have been described in [12], [13], [14], [15], [16], [17], [2], [3], [4] and will be reviewed here.

TTCN3 (Testing and Test Control Notation) [12] is a standardized testing language, designed with testing and certification in mind. It has been used in a number of automotive projects [18], [19], [20], [21] and has been adopted by the AUTOSAR standard. TTCN3 lends itself easily to technical testing and test automation but does not go as far as to provide a design and generation methodology for tests.

The Matlab/Simulink/Stateflow [13] environment is widely used in the automotive industry throughout the automotive software development process. This environment, allows the user to model and simulate a wide range of linear, nonlinear, continuous, discrete and hybrid dynamic systems. Due to its formality it lends itself easily to automation. The Simulink Design Verifier [14] is an add-on based on formal mathematical proving methodologies that can be used to generate tests for Simulink and Stateflow models that satisfy model coverage and user-defined objectives. It also caters for validation of design properties and generates examples of violations.

Time Partition Testing [15] was developed in the DaimlerChrysler research department and is based on considering test data, semantically, as signal courses over time. The temporal course of each test data is partitioned down into sub-phases and formal semantics are developed for the method. Test evaluation is not automated but violations of the requirements are detected via watchdogs. The temporal model uses dense time which can potentially inject errors during the discretisation of time needed for execution.

Reactis Validator [16] is a popular test-generation tool that generates tests based on user-defined objectives. The user can express expected behavior of the system in the form of assertions. The tool then performs an automated search for a violation of such assertion. If a violation occurs, it returns a test which executes a sequence of events that led to the violation. Reactis Validator also can generate test data to execute specific test scenarios defined by the user. It is based on random test generation techniques and it does not include a strategy to guide the systematic design of testing.

The Classification Tree Method for Embedded Systems (CTM$_{EMB}$) [17], [2] provides a systematic approach for designing test cases for embedded software based on the functional specification of the system under test (SUT).

In the CTM$_{EMB}$ method, the input domain of the SUT is partitioned in equivalence classes which correspond to the different data sources (inputs). Each of these classes is further partitioned based on the possible range of values the input data can have. The methodology gives a graphic representation of time-variable test scenarios and includes techniques such as testing of specific values and specific value courses. The methodology was updated in 2006 [2] to describe event-based test scenarios.

CTM$_{EMB}$ can be used for structural testing at a model level, and it has been used in a number of automotive software development projects [1], mainly for testing of in-vehicle software developed via a model-based approach [22]. The methodology has got good tool support (for example test scenarios built in tools such as CTE/XL [23] and MTest [1] can be automated, checked for model coverage and linked to the Matlab/Simulink/Stateflow environment).

In the current form, the CTM$_{EMB}$ methodology does not lend itself easily to design tests which are reactive to the outputs of the system.

Zander-Nowicka [3], [4], proposes a testing framework that designs testing very early in the development process and focuses on a systematic test evaluation design for a model-in-the-loop approach. The framework is based on the concept of Automotive Validation Functions (AVFs) that are assertions with “pre” and “post” conditions which evaluate online both discrete and continuous signals and deliver the test results. These preconditions and assertions, connected with temporal, quantitative, or logical dependencies, are built based on the functional requirements of the SUT.
The preconditions block, as shown in the template of Figure 1, activates the assertions block which compares the signals and their expected values. The framework also takes in consideration the nature and features of the signals and different types of AVFs are applied to evaluate different types of behavior. The framework, while not being tied to any particular language, has a natural affinity with the Matlab/Simulink/Stateflow environment.

In this framework, both the test design and the test data generation use the same common point i.e. the conditional statements extracted manually from the functional requirements of the SUT. Hence, if an error is introduced at the beginning when the functional requirements are transcribed, it will propagate throughout the whole process and potentially lead to checking an erroneous assumption with data that has been generated based on that assumption.

Despite the above mentioned promising developments, a fully automated reactive testing technology is still missing.

**PROPOSED APPROACH**

Our research aims to deliver a framework for an automatable reactive systematic model-based testing methodology for embedded automotive software.

As the first step of our research we propose to combine the capabilities of the CTM<sub>EMB</sub> method in generating test data with the AVFs testing framework.

For clarity of the argument, the same example from an adaptive cruise control system used by Conrad [2] and later by Zander-Nowicka [3] is used to illustrate the proposed approach. Figure 2 depicts an excerpt from the requirements for the interpretation of the accelerator and brake pedal positions in this adaptive cruise control system.

![Figure 1: A template of the structure of an AVF[3] designed in the Matlab/Simulink/Stateflow[13] environment.](image)

Figure 2 gives an example of tests generated via the CTM<sub>EMB</sub> method (this is described in detail in [2]).

![Figure 3: Data generation using the CTM<sub>EMB</sub> method](image)

In the AVFs testing framework, the functional requirements of the SUT are written in the form of conditional statements and Figure 4 depicts an example of requirement 02.2 transcribed as per this method (explained in detail in [3]).
- IF phi.Acc increases AND v is constant THEN T.des.Drive increases.
- IF phi.Acc decreases AND v is constant THEN T.des.Drive decreases.
- IF phi.Acc is constant AND v is constant THEN T.des.Drive is constant.
- IF phi.Acc is constant AND v increases AND T.des.Drive is non-negative THEN T.des.Drive does not increase.

**Figure 4: Requirement 02.2 transcribed as per the AVFs testing framework**

In the approach proposed in this paper, in addition to the test evaluation and test data generated according to the AVFs, test data are generated according to the CTEMB approach. The concept of the approach is depicted in Figure 5.

**Figure 5: Proposed Test Data Generation Approach**

Hence, an error introduced while extracting the test design from the functional requirements via the AVFs testing framework, is unlikely to be replicated while generating the test data via the CTEMB methodology.

For clarity of the argument, let us consider the scenario that during the transcription of the functional requirements in the form of “IF preconditions set THEN assertions set”, some preconditions has been omitted by mistake from the assertions set (say a precondition stating that the lateral acceleration of a vehicle should be kept under 9 m/s²). This would lead into an inadequate test design. Yet, since the test data generated via the CTEMB methodology, is based in the value ranges of the input parameters for the SUT, in a scenario when the vehicle is meant to drive comfortably for lateral accelerations under 9m/s², tests with the lateral acceleration equal to 9m/s², just under 9 m/s² as well as above 9 m/s² will definitely be designed.

Future work will include expanding the approach such as the execution of test cases is reactive to the outputs of the system under test. The high level view of this work in progress test architecture is depicted in Figure 6.

**Figure 6: Proposed Test Architecture**

**CONCLUSION**

This paper reports on the first phase of our research into reactive systematic model-based testing design for embedded automotive software. It makes a review of the state of the art and proposes an approach for model-based testing design for embedded automotive software that combines the test data generation from the CTEMB method with the AVFs testing framework.

In the proposed approach, testing based on the functional requirements for the system under test, can be designed very early in the process when only a model of the system exists. In order to reduce the risk of injecting human generated errors into the testing design, two different and independent routes are taken to generate the test data.

In the current approach, the execution of the test cases is not reactive to the outputs of the system under test. Future work will be focused into enhancing the proposed methodology such as to cater for such reactive behavior.

**ACKNOWLEDGMENTS**

The authors would like to acknowledge the contribution and support of Lero – The Irish Software Engineering Research Centre and to thank the SFI – Science Foundation Ireland for funding this research (SFI grant 03/CE2/I303_1).

**REFERENCES**


