Model-Based Testing for Better Quality

Advantages of test case generators in model-based development processes

Software engineering is a discipline of computer science that has great innovative potential. Great software complexity and the overabundance of resulting information raise the quality assurance issue of how to effectively guarantee consistency of the model and code. Test strategies and processes used during the early stages of development will reveal technical and design errors early, when they can be corrected much more cost-effectively.

Model-based development is a software engineering method which – in addition to documenting the system – also utilizes automatic generation to produce a large share of the test documentation. In practice, most test cases are still created manually. In a classic document-based approach, test cases are derived from requirements and described in a test specification. From this specification, tests are implemented to verify proper software implementation in the ECU (Figure 1).

Over the past five years, automotive OEMs have been applying more and more model-based methods to develop embedded software. One possible approach that integrates these model-based methods into the development process is one in which the software is manually implemented in the ECU while functional models are created in parallel (Figure 1). The development of executable functional models represents a quality improvement in requirements analysis and development. First, the functional models define functions and then validate them with respect to the given requirements. The test specification should be used again to verify the functional model. This process reveals errors and inconsistencies in early development phases, which can then be corrected at a fraction of the cost of correcting them later, after system or acceptance testing. Broad tool support also promises minimal time to develop the ECU software, e.g. by using automatically generated production code (Figure 1).

Stepwise development and verification of complex functional models assumes efficient test methods. A check must be made at each step in the development process to verify that no errors have
been introduced. In a process known as model-based testing, test case generators take functional models and automatically generate test cases, which can then be automatically executed, evaluated and documented.

Other issues involve where it makes sense to apply automatically generated test cases and which advantages they offer in the development of embedded vehicle systems. The new approach can be used during iterative development of functional models to automatically generate test cases for regression tests, for example. Another possibility is to re-use the test cases generated from this development phase in later testing phases of the development process, such as in ECU acceptance tests.

Test case generators simplify regression tests

Requirement definitions are often subject to change. Any errors discovered in the requirement definitions are corrected, and new functions may be added to extend the functionality of the system being developed. Later in the development process, the implementation is often restructured or simplified – all while retaining its functionality, of course. Functional models that have already been developed are then adapted to the new conditions. During this process, the developer must ensure that changing a functional model does not introduce any new errors or undesirable effects on functionality. This can be accomplished by using regression tests [1]. Three steps are necessary to attain this goal:

- A suitable Model-in-the-Loop (MiL) test environment must be developed for the regression tests
- Test cases must then be automatically executed in this MiL test environment
- Regression tests will be automatically evaluated

A key aspect of regression testing is the quality of the comparison of a modified functional model with its previous version. One measure of this quality is code coverage, which is why comparing different versions of a functional model requires the greatest possible coverage. In test cases related to requirements, greater code coverage entails immense effort, because if there are coverage gaps, the related code sections must be analyzed, and other requirement-based test cases may need to be manually created until the necessary code coverage is attained. Because of this immense effort, it is no longer advisable to manually create test cases for the regression test.

One answer lies in the use of new commercially available test case generators, such as Simulink Design Verifier from The Mathworks or Telelogic Rhapsody ATG from IBM. Test cases are automatically generated after specifying a specific code coverage to be attained. The test cases then increase the quality of the comparison. In addition, the tools can often simultaneously generate an MiL test environment, execute test cases and automatically generate a test report. This results in a working method that can be used throughout the system which has quite short throughput times. This in turn further simplifies the execution of regression tests for functional models, which results in cost and time savings.

Figure 2 shows an implementation of a regression test in a Model-in-the-Loop test environment. Functions are modeled in

![Figure 1: Comparison of different approaches to developing ECU software.](image-url)
Re-using test cases in ECU acceptance tests

The goal of the acceptance test is to verify that software functions that suppliers implement in ECUs behave as defined in the specification. To perform an acceptance test, suitable test cases must be created which test the specified functionality. Clearly, one potential way to reuse the test cases generated from model-based development is in a Hardware-in-the-Loop test environment. The advantage here is that the test cases do not need to be manually re-created.

A tester wanting to re-use the test cases may find that changing the test environment from Model-in-the-Loop (MiL) to Hardware-in-the-Loop (HiL) also shifts test boundaries. Test cases in the MiL test environment refer back to the logical input and output signals of the functional model. Test cases in a HiL test environment, on the other hand, require physical signals, e.g. CAN signals, to stimulate the system and evaluate system behavior. It is therefore necessary to transform the test cases, which involves mapping logical signals to associated CAN signals (Figure 3).

Also, when performing a transformation it must be remembered that test cases in the modeling tool do not necessarily have the same data structure or data format as test cases in a tool for HiL tests, e.g. in CANoe.

In practice, XSLT Stylesheets might be used to perform such a transformation, for example; this assumes that test cases can be exported from the modeling tool in XML format. Before the test cases may be transformed to suitable executable test scripts for the Simulink/Stateflow, while the Simulink Design Verifier generates the test cases. In principle, other modeling tools could also be used, provided that they can be obtained with test case generators.

The ‘Test Cases’ block shown in the diagram (Figure 2) contains the automatically generated test cases and stimulates the ‘Test Unit’ system under test. This represents the modified functional model. The ‘Expected Results’ block contains the reference outputs from the previous version of this functional model. In the ‘Regression Test Analyzer’ block, the newly generated outputs of the modified functional model are compared to the reference outputs at each step of the simulation. After the test run, a script automatically creates the test report and adapts it to the desired output format. In this case, the output format matches a CANoe test report by Vector. If the evaluation finds a difference in the outputs, checking must indicate whether the difference is acceptable or whether it is an error.

The iterative approach presented here is an efficient way to support continuous improvement and adapt the functional model to new requirements. It also improves the quality of the functional model. In addition, verification at each step in the development process ensures that no new errors have been introduced, further improving quality. Developing functions with this model-based approach will make test cases available. The goal is to not only use these test cases for regression testing of functional models, but also for later test phases in the automotive software engineering process, such as in ECU acceptance testing.

Figure 2: Setup of a Model-in-the-Loop test environment for regression tests in MATLAB/Simulink. A script generates test reports and automatically adapts their output format.
given HiL test environment, an intermediate step is necessary: mapping the signals, e.g., using an Excel table. Visual Basic for Applications replaces the relevant items in the XSLT Stylesheet where the mapping is implemented. Finally, the transformed test cases are linked and automatically executed in CANoe. An automatically generated test report helps in evaluating the test results. All necessary steps in this tool chain can be automated, which yields time and cost savings.

Acceptance testing with reusable test cases now tests both the software integration and the hardware-software integration. Testing verifies that the software components properly interact with one another via their specified interfaces [1]. Also, the interplay of the complete software system with the ECU hardware is tested in the HiL test environment. This verifies the entire implementation in the ECU.

Summary and Outlook

Powerful new test case generators are now commercially available which make it possible to conduct efficient regression tests during model-based development of vehicle functions without any additional effort. Test cases generated can be re-used in later test phases of the automotive software engineering process, such as in ECU acceptance testing. This requires converting the test cases by means of a suitable transformation to produce test scripts for the specific HiL test environment.

A method-based approach to testing may be applied to model-based function development with functional descriptions in MATLAB/Simulink. Use of model-based test methods is being studied in ongoing research projects. For example, the goal of the VitaS3 research project at the University of Applied Sciences in Regensburg is to determine the extent to which formal and semi-formal description languages such as the Object Constraint Language (OCL) and temporal logic methods can be used to generate test cases via model transformations [2]. The use of formal description techniques is prescribed in the primary industrial standard for functional safety IEC-61508 for safety-critical systems. This approach enables virtual integration of vehicle functions.

Translation of a German publication in Elektronik automotive, 3/2009

Literature:

Links:
Homepage Vector: www.vector.com
Homepage LaS3, University Regensburg: www.las3.de/englisch/home.html
Product information VT System: www.vector.com/vt-system
Product information CANoe: www.vector.com/canoe

Wolfgang Hartig, Vector
obtained his Master of Engineering degree in Electrical and Microsystems at the University of Applied Sciences in Regensburg. He conducted his Master’s thesis work at Vector on the topic: “Study of the potential uses of automatically generated test cases starting from software models”. Today, he is employed in the area of Technical Consulting and Engineering Services at Vector.

Albert Habermann, Vector
served as an advisor for the Masters thesis of Mr. Hartig as part of his work in Technical Consulting and Engineering Services at Vector (with a focus on model-based SW development and testing). Today, Mr. Habermann is project manager for the eASEE process tool in the Customer Service area at Vector.

Jürgen Mottok, University Regensburg
teaches computer science at the University of Applied Sciences in Regensburg. His areas of teaching include: Software engineering, programming languages, operating systems and functional safety. He directs the Laboratory for Safe and Secure Systems (LaS3), is a member of the advisory board of the Bavarian Cluster of IT Security and Safety, and active in other working committees.
>> Your Contact:

Germany and all countries, not named below

France, Belgium, Luxembourg

Sweden, Denmark, Norway, Finland, Iceland
VecScan AB, Göteborg, Sweden, www.vector-scandinavia.com

Great Britain
Vector GB Ltd., Birmingham, United Kingdom, www.vector-gb.co.uk

USA, Canada, Mexico
Vector CANtech, Inc., Detroit/Novi, USA, www.vector-cantech.com

Japan
Vector Japan Co., Ltd., Tokyo, Japan, www.vector-japan.co.jp

Korea

E-Mail Contact
info@vector-worldwide.com