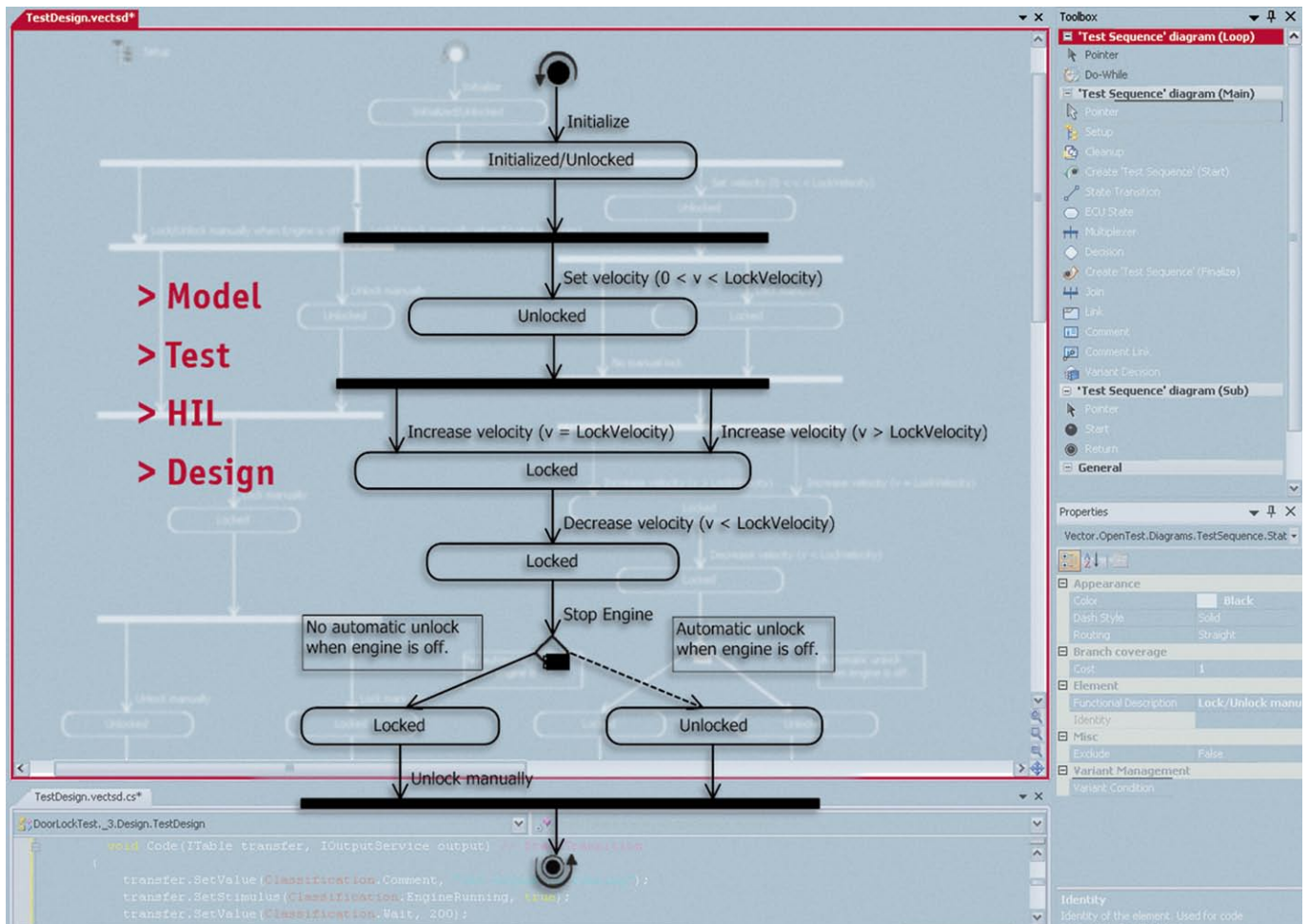


Model-based design simplifies ECU tests

Mastering Variation



HIL systems are playing an ever increasing role in automated ECU validation and testing. Structured test creation methods, standardized by the use of models, improves both test efficiency and quality. The OpenTest tool from Vector provides test engineers with the necessary infrastructure to develop model-based test scenarios. Tasks may be distributed among several specialists using a uniform system of graphic test notation, independent of the specific HIL test bench or test automation system.

ECU tests are most often created at the end of the module’s development cycle. As such, they must be created efficiently and modified in a timely manner to fit new and changing test requirements. Such a challenging environment requires a structured approach to test creation. An increasing amount of tests is performed automated on Hardware-In-The-Loop (HIL) test benches. Each HIL system has its own test description method ranging from general programming languages to proprietary scripting languages and graphic notation. For complex systems tests have to be performed on different test benches depending on the test requirements. This fact reinforces the need for a test creation methodology that offers

an uniform test description format independent of the HIL system used.

Structured Test Creation Process

Typically, tests are developed in the context of larger test teams. These teams may work directly in software development, or they may represent independent organizational departments. Along with the actual testers - who execute, monitor and report test results via a test automation system on the HIL test bench - the roles of the test designer, HIL programmer and test framework

specialist are gaining in importance. Today, test designers create the test descriptions in HIL-system-specific description formats, or scripting languages, such as TCL, Python or CAPL. This means however, that all test designers need to master all of the scripting languages used. For efficiency reasons, it makes sense to formulate test designs independent of the test system and then have specialized programmers write the software interfaces just once. This means that only one specialized programmer is needed for each supported HIL system who adapts the test descriptions to the specific system.

ECU development is increasingly being conducted in the framework of what are referred to as software product lines. A software product line is a set of ECUs that are all based on a common set of re-usable functions. The individual implementations, or ECU variants, consist of both common functions and variant-specific functions. In testing software systems, common components should be identified and be made available to all designers in the form of a test framework.

OpenTest Methodology

OpenTest is a methodology and framework from Vector that supports structured test creation. Its test design notation is independent of the specific HIL system thus enabling the distribution of user roles. The executable scripts used on the test benches are automatically generated by HIL-specific generators (**Figure 1**).

The requirements represent the starting point in the test creation process. Frequently, these requirements only exist in text format. The first task is to derive the individual test cases for the system under test (SUT) from these requirements. Each requirement must be covered by one or more test cases. In OpenTest, this

is accomplished by a graphical notation that is used for modeling test sequences and for reviews.

In a structured test creation process, one or more specialists create a basic framework for one or more systems under test. This framework consists of these re-usable components:

- > A formalized description of input and output variables to stimulate and test the expected reactions of the SUT
- > Functions for basic test steps or entire test sequences
- > Cross-platform and variant-specific parameter sets

This cross-variant basic framework is provided to individual test designers in the form of a library. Test creation that is based on the basic framework increasingly leads to test designs with the same type of structure. This increases the share of re-usable individual designs, and it makes them more comprehensible for test reviews. OpenTest takes test coverage strategies such as branch and path coverage into consideration in automatically generating the test cases and HIL system specific test scripts from the test models.

OpenTest Notation

Tests are modeled using UML state diagrams (**Figure 2**). In OpenTest the UML notation is enlarged by a set of elements for describing tests. Thus it provides a graphic form of expression that is specialized for ECU testing, a so-called Domain Specific Language.

A state in the diagram represents a state of the ECU function to be tested. Each transition between states defines events that stimulate the SUT and elicit defined reactions. The individual test cases are defined with the help of multiplexer elements. The test cases can be specialized at any desired point in the test flow. The advantage

of this method is that shared test steps and sub-sequences only need to be modeled once for multiple test cases.

Each path taken from the start node to the end node describes an independent test case. The sum of test cases should, at a minimum, cover the requirements to be tested. Also modeled are test cases that are not explicitly based on requirements in the Requirements Specification. Instead, they are based on the test designers' experience or the results of previously conducted tests. Explicit modeling of the tests makes this possible and further enhances testing depth.

The concept of software product lines is supported by variant branches

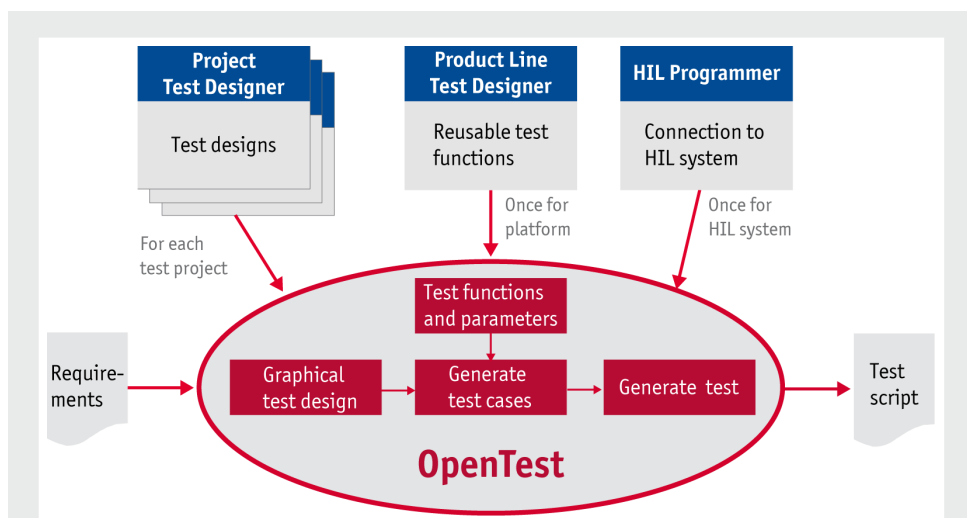


Figure 1:
OpenTest method and user roles

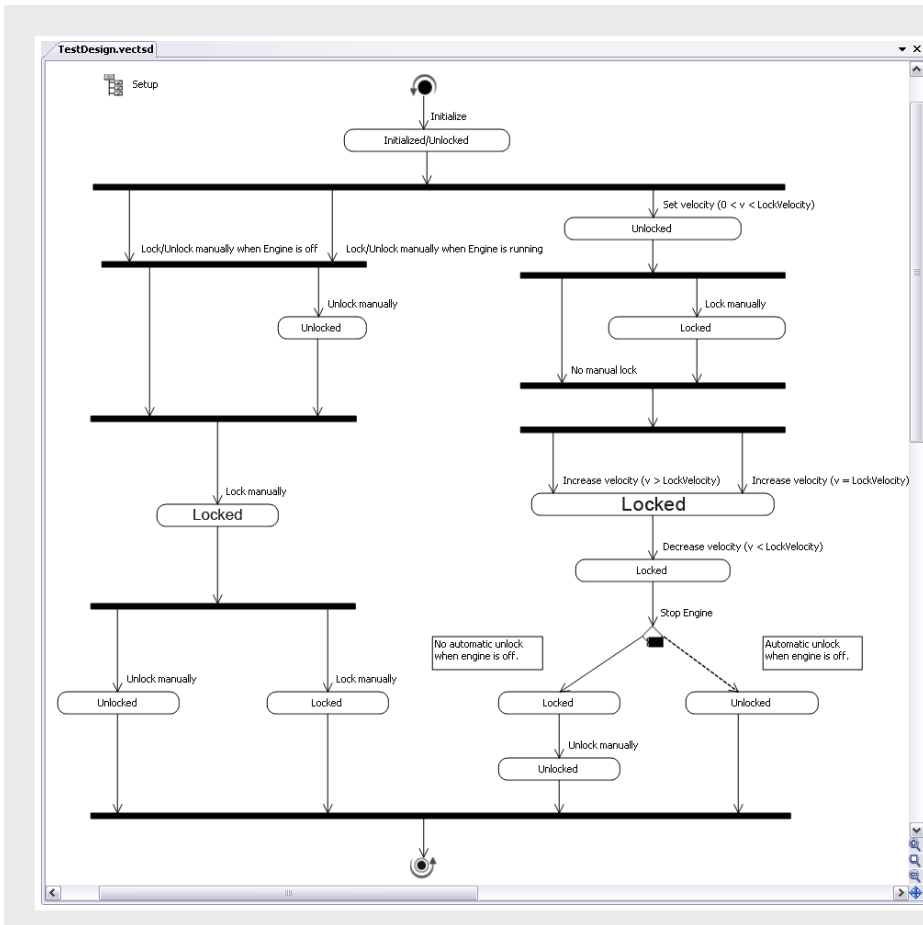


Figure 2: Test model in OpenTest notation

in OpenTest notation. Variant branching adds specialized subsequences for the individual variants within the software product line. Cross-variant test steps apply to all software product line variants and are modeled jointly within the test sequences.

OpenTest modeling enables structured and well-organized test descriptions. This is necessary since the test model is not only used to generate test cases and HIL programs, it is also used for reviews. Communication with customers, software developers and quality managers is much easier on the graphical notation level than on sequential or text-based test scripting level. This approach enables graphic highlighting of failed test cases, critical test steps or test cases for regression testing. Furthermore, each test step may be documented with comments.

Independence from the HIL System

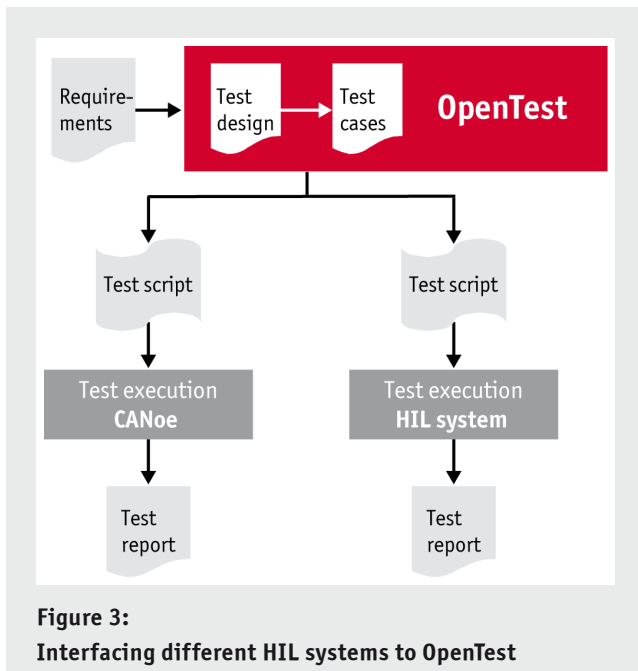
The models that test designers create in OpenTest are independent of the specific HIL system used. HIL-specific executable scripts are automatically generated. The programmers familiar with the specific system define mapping rules which convert the individual test steps to executable commands. These commands range from simple value stimuli or results comparisons to complex commands for system diagnostics or extraction of internal ECU parameters and

system states over XCP.

OpenTest currently supports such test systems as CANoe from Vector and TestStand from National Instruments. Its framework character makes it easy to integrate other HIL systems as well (Figure 3).

Practical Use

The methodology presented here and the OpenTest tools was initially developed for testing mechatronic systems by Robert Bosch GmbH, where it has been in operational use since 2006. The concept of the software product line is especially well developed in this area and leads to an intensified role for the test framework specialist. Efficiency in creating tests was significantly improved by applying a uniform methodology and implementing it with OpenTest as the software tool. Distribution of tasks within the test team – as designer, HIL programmer and framework specialist – also makes test engineers experts in specific work areas and improves identification with their respective tasks.



Translation of a German publication in *Automobil Elektronik*, April/2010

All Figures:
Vector Informatik GmbH

Links:
Homepage Vector: www.vector.com



Michel Dietrich
studied Engineering and Business Administration and works at Robert Bosch GmbH, Automotive Electronics Business Division, as Senior Expert for model-based testing. He is responsible for methodological and technical support of the Test Center for Mechatronic Systems.



Hans Quecke
studied computer science and has worked at Vector Informatik since 1994, where he is Group Leader responsible for conceptualizing and developing tools for model-based testing and Product Manager for OpenTest.

>> Your Contact:

Germany and all countries, not named below

Vector Informatik GmbH, Stuttgart, Germany, www.vector.com

France, Belgium, Luxembourg

Vector France, Paris, France, www.vector-france.com

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, USA, www.vector-cantech.com

Japan

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

Korea

Vector Korea IT Inc., Seoul, Republic of Korea, www.vector.kr

India

Vector Informatik India Prv. Ltd., Pune, India, www.vector.in

China

Vector Informatik GmbH Shanghai Representative Office, Shanghai, China, www.vector-china.com

E-Mail Contact

info@vector.com