hitex
EMBEDDED TOOLS & SOLUTIONS

Building a safe and secure embedded world

Unit Testing for Software Quality

Test in the real environment and document the results

> TESSY and CTE

The quickest route to a safe application

TESSY: the invaluable test tool for safety critical applications

TESSY – The invaluable test tool

TESSY performs automated dynamic module/unit and integration testing of embedded software and determines the code coverage along the way. This kind of test is required for certifications according to standards such as DO-178, IEC 61508, ISO 13849, IEC 62279, or ISO 26262.

Integrated with TESSY is the Classification Tree Editor (CTE), a tool to specify test cases systematically according to the Classification Tree Method (CTM). TESSY is based on Eclipse Rich Client Platform (RCP) thus featuring the well-known Eclipse user interface with views and perspectives.

TESSY automatically executes the tests, evaluates the test results, and generates the test reports. TESSY can eliminate manual testing and therefore saves the embedded development engineer a tremendous amount of time. A faster development process ensures the tool recovers its costs quickly, and what's more, produces better quality software and documented tests.

TESSY is successfully used in large projects with dozens of users in multiple locations across the world. TESSY is used extensively in automotive, aerospace, avionics, railway, medical, military, and industrial applications. Ask for a testimonial!

TESSY can be operated without user intervention which is useful for regression testing and continuous integration.

Key features

- ✓ Automated test execution / regression testing
- ✓ Test report generation
- ✓ Code coverage without extra effort
- ✓ Integration testing
- ✓ Traceability of requirements to test cases
- ✓ Qualified to be used in safety-related software development
- ✓ Testing on host or actual hardware
- ✓ Global, permanent license
- ✓ Supports C and C++
- ✓ Special features for fault injection and testing of software variants

Tested for the unexpected: systematic, rigorous, isolated testing

Module / unit testing eliminates errors early on and prevents them from showing up in later stages of the development process.

What Is Module/Unit Testing?

During unit testing the test object is tested rigorously and in isolation from the rest of the application. The test object is a C-level function or a method (in C++). Often unit testing is also called module testing.

Rigorous means that testing is tightly focused on the unit in question; the test cases are specific for the unit, demanding, comprehensive, using boundary values and values that are likely to reveal a problem in the unit under test.

Isolated means that the test result does not depend on the behaviour of the other units in the application. It can be achieved by directly calling the unit under test and replacing calls to other units by stub functions.

What are the benefits?

1. Find errors early

Unit testing can be conducted as soon as the unit to be tested compiles successfully. Therefore errors inside the unit can be detected very soon after their creation.

2. Save money

It is generally accepted that errors detected late in a project are more expensive to correct than errors that are detected early. Hence unit testing saves money.

3. Give confidence

Unit testing gives confidence. After the unit testing, the application will be made up of fully tested units. A test for the whole application will then be more likely to pass.

4. Specific for the unit

Instead of trying to create test cases that test the whole set of interacting units, the test cases for unit testing are specific to the unit under test (divide-and-conquer). Test cases can easily comprise of input data that is unexpected by the unit under test, something which may be hard to achieve during system testing.

5. Easy defect isolation

If the unit under test is tested in isolation from the other units, detecting the cause of a failed test case is easy. The fault must be related to the unit under test.

A tour around the testing workflow

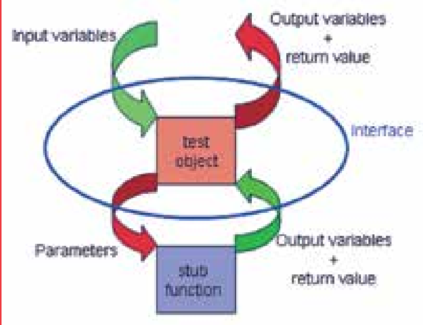
How TESSY works

Unleash it for a test run

> TESSY starts off by analyzing the source module and then lets the user select the test object. This can be a function or a method (in C++). TESSY then identifies the interface of the test object, such as global variables, parameters, and functions/methods called by the test object.

> TESSY determines whether a variable of the interface is input, output, or both.

> TESSY can be directed to allocate memory to be used as target for pointers in the interface.



The interface separates the test object from the rest of the application

State your case

TESSY provides several ways to specify values for the test cases:

> Interactive test data input is accomplished in the tabular Test Data view

> Test data import in various file formats (including Excel)

> Test cases specified systematically by use of the Classification Tree Method can be imported from the Classification Tree Editor (CTE)

> TESSY can generate test cases from value ranges

> TESSY can generate random test data

Meet your test driver

TESSY then generates source code for the test driver, which calls the test object. If it calls another function/method, TESSY is able to create a stub function as replacement. This is necessary for unit testing in its strictest sense and useful if the called function/method is not implemented yet.

TESSY provides two types of stub functions: One type allows the user to specify expected values for the input variables of the stub function which then are compared with the actual values by TESSY. Furthermore this stub function type allows you to specify the inputs from the stub function to the test object, e.g. the return value. The other type of stub function allows the user to provide source code for the body of the stub.

Go TESSY go!

Using a cross compiler, TESSY compiles and links the driver source code, the function under test and any stub functions, and then downloads the resulting executable to the test system. This might be an in-circuit emulator in stand-alone mode or one connected to a target system, or a JTAG / BDM / OCDS debug system. This might also be a simulation of the target microcontroller running on the host PC. Testing can also be performed on the host PC using the native GNU compiler.

TESSY executes each test case and then determines, if it has passed or failed. A comprehensive test report can be created in pdf, word, html, and xml format.

Why TESSY eases testing

Software quality needs TESSY

Dream debugging

If a test case fails, an easy and efficient debug plan is in place. TESSY is able to re-execute a test case and direct the debugger in use to stop test execution at the beginning of the test object.

The debugger's features now can be used to reveal the culprit. After the source code is changed to fix the bug, the test case in question (and all others) can easily be re-run to verify that the correction operates successfully.

Re-use test data and save time

If any interface element of a tested function has been changed in the course of the development process, TESSY allows the user to re-use test data from the old interface, which considerably aids the regression testing process.

Regression testing

Regression testing can reveal if new errors have been introduced during further development of the application, such as bug fixes in other sections, rewriting of the tested function, switching to a new compiler version or porting the software to another microcontroller architecture.

TESSY's easy-to-use regression testing ability is an extremely helpful method of checking modified software, therefore ensuring software quality.

Non-interactive testing

TESSY features a command language which allows the user to automate almost all tasks in TESSY.

The command language enables the integration of TESSY in Continuous Integration (CI) systems.

Trace your requirements

Traceability of requirements to test cases checks if all requirements have a test case associated with them and also lets you find out which test cases may need adaptation if a requirement has changed.

Requirements can be imported / exported and created in TESSY.

Fault injection

TESSY features a fault injection mechanism allowing comfortable testing of the reaction of the test object to faults from the outside, e.g. defective RAM.

This mechanism can also be used to execute theoretically unreachable code parts, hence achieving 100% code coverage.

Testing of software variants

TESSY eases the testing of software variants. Test cases for a base software can be transferred and adapted to the software variants. If a base test case changes, the change is applied to the inherited test cases of the variants automatically.

Integration testing – check the interaction

TESSY can test the interaction of cooperating units even if the units do not call each other, e.g. push() and pop() of a stack. Several units are composed to form a bigger unit, usually called a component. A test case calls the units in a certain order and sets variables of the component. Resulting calls to other components and variable values can be checked by TESSY. Integration test cases can also include simulated time (temporal component testing).

Supported microcontrollers

TESSY is currently adapted to more than 150 combinations of microcontroller / cross compiler / debugger. This ensures that TESSY is able to handle non-ANSI-C microcontroller-specific code of some cross compilers. Since TESSY controls the debuggers, TESSY can execute the tests automatically.
The list of supported combinations is extended steadily. Please check www.hitex.com/tessy.

The Classification Tree Method

The Classification Tree Method supports a developer confronted with issues such as:
> Finding the ”right” test cases
> Minimizing a set of test cases while assuring that none are missing
> Estimating the amount of testing required
> Defining criteria needed to conclude testing without risking integrity of the test process

Transforming requirements

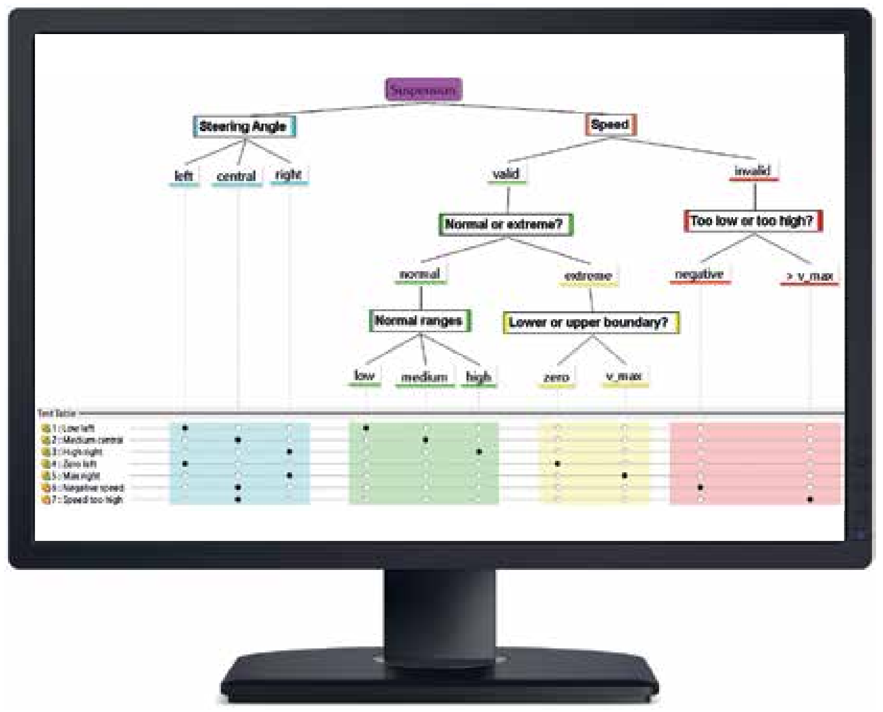
The Classification Tree Method transforms requirements systematically into a set of error-sensitive, low-redundancy test cases.

This method classifies test-relevant aspects using the equivalence partitioning method and leads to test case specifications that can comprise boundary values.

Intuitive and easy to learn

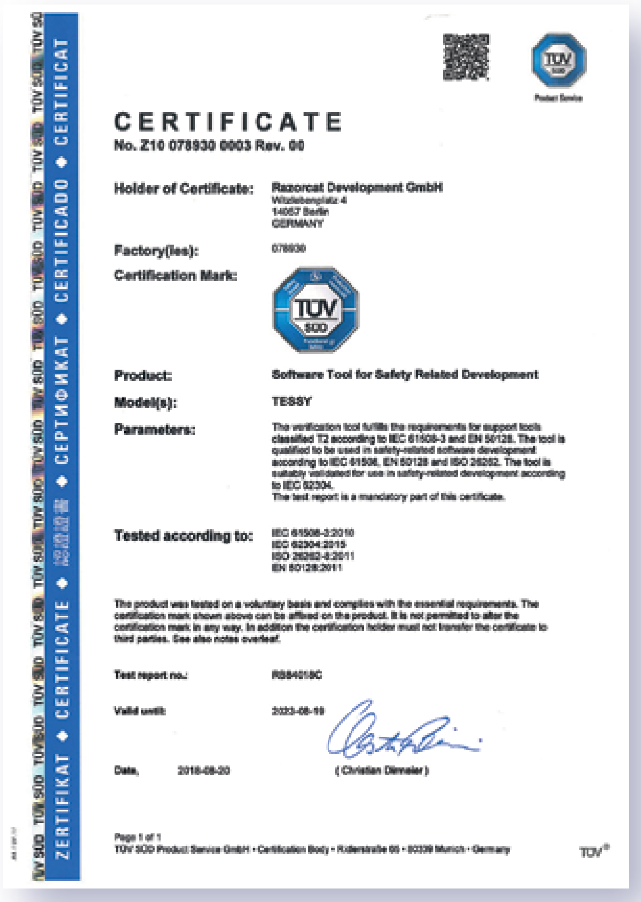
The Classification Tree Method is intuitive and easy to learn. It requires and encourages the developer to employ his creativity.

Because thinking about the problem specification is at the very beginning, the Classification Tree Method also reveals inconsistencies or omissions in the problem specification.



Test case specifcation according to the Classification Tree Method. The CTE (Classification Tree Editor), integrated in TESSY, supports this method

Building a safe and secure embedded world



ZERTIFIKAT ◆ CERTIFICATE ◆ CERTIFICADO ◆ CERTIFICAT

Product Service

CERTIFICATE
No. Z10 078930 0003 Rev. 00

Holder of Certificate:	**Razorcat Development GmbH** Witzlebenplatz 4 14057 Berlin GERMANY
Factory(ies):	078930
Certification Mark:	
Product:	**Software Tool for Safety Related Development**
Model(s):	**TESSY**
Parameters:	The verification tool fulfills the requirements for support tools classified T2 according to IEC 61508-3 and EN 50128. The tool is qualified to be used in safety-related software development according to IEC 61508, EN 50128 and ISO 26262. The tool is suitably validated for use in safety-related development according to IEC 62304. The test report is a mandatory part of this certificate.
Tested according to:	IEC 61508-3:2010 IEC 62304:2015 ISO 26262-8:2011 EN 50128:2011

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.:	RS84018C
Valid until:	2023-08-19

[illegible]

Date, 2018-08-20 (Christian Dirmeier)

Page 1 of 1
TÜV SÜD Product Service GmbH • Certification Body • Ridlerstraße 65 • 80339 Munich • Germany

TÜV®

TESSY is qualified to be used in safety-related software development according to IEC 61508, ISO 26262, EN 50128, and IEC 62304

Hitex Head Office, Germany	Hitex UK
Hitex GmbH Greschbachstraße 12 76229 Karlsruhe Germany	Hitex (UK) Ltd Millburn Hill Road University of Warwick Science Park Coventry CV4 7HS United Kingdom
Phone: +49-721-9628-0 Fax: +49-721-9628-149 Email: info@hitex.de	Phone: +44-24-7669-2066 Fax: +44-24-7669-2131 Email: info@hitex.co.uk

B-TESSY-E04.indd - DEC2018

Consulting | Engineering | Testing | Training | Tools | Software Components | Systems Manufacturing