



Razorcat Development GmbH

Email: support@razorcat.com

Phone: +49 - 30 - 536 357 0

New features in TESSY v5.1

Redesigned icons

TESSY v5.1 comes with a reworked icon design for all existing perspectives and views as well as for the new Test Cockpit view. The new layout of the coverage icons requires less space so that all coverage columns within the Test Cockpit and Test Project view became narrower.

Test Cockpit view

The new Test Cockpit view provides an overview of all source files located within the project root or source root directory of a TESSY project. Both the results of executed tests as well as the achieved coverage results are summarized on source file level.

Source Files / Test Objects	▶	☑	CA	HC	C1	MC DC
Project Root	✓	—	🟡	🟡		
source	✓	—	🟡	🟡		
example2.c	—	—	🟡	—		
#1af7d92	—	—				
getAcceleration	🎯	—			—	—
#aafa429	—	—				
#d77bbbd	—	—				
example4.c	✓	—	✓	🟡		
Base	✓	✓				
func2	🎯	✓			✓	🟡
func3	🎯	✓			✓	✓
Variant	✓	—				
func1	🎯	✓			✓	—
func2	🎯	✓			🟡	—
func3	🎯	—			—	—

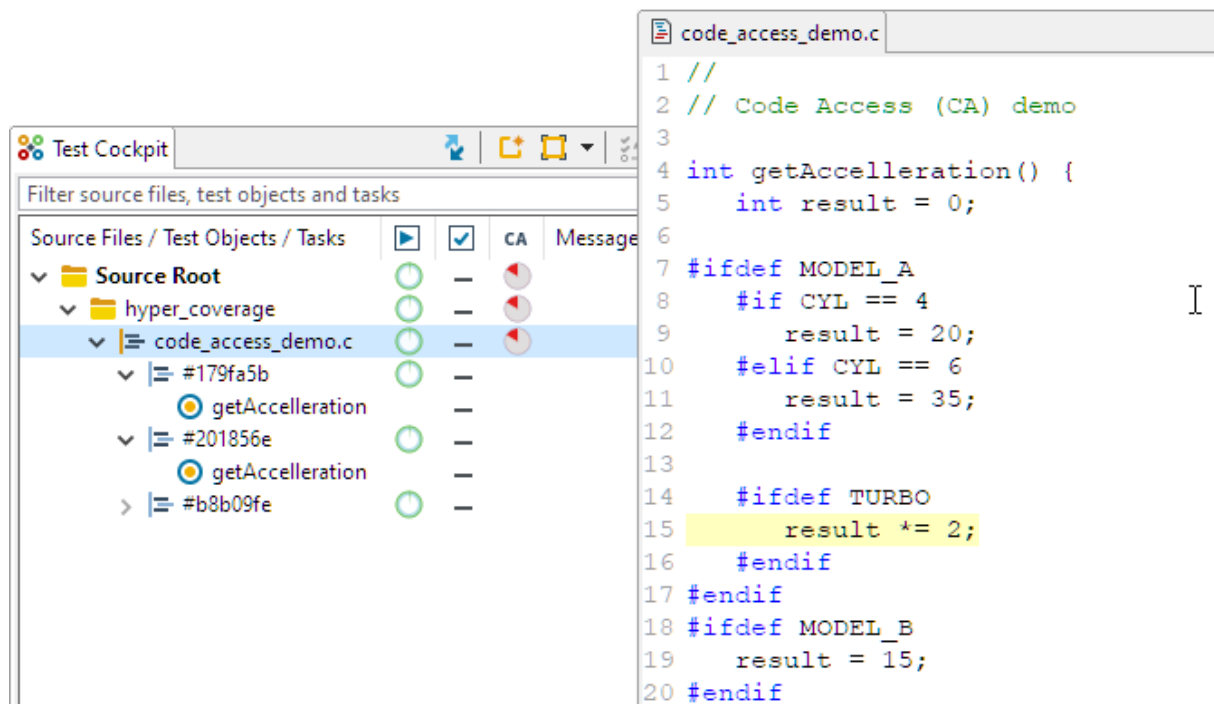
Also the test progress will be available within the Test Completion Rate column which lists the number of test objects that need to be executed, either for the first time or due to changes of tests or source code.

TESSY provides automated analysis of tested source code variations already after the setup of tests. Any untested code line will be revealed without further effort of the tester even before starting any test run.

After test execution, the coverage measurement results will be accumulated for each function or method within the source files in order to reveal any unreachable source code lines.

Code Access analysis

The new Code Access feature automatically detects hidden or untested code in all variants in the source code under test. While analyzing a module, TESSY calculates checksums for source files and preprocessed source files in order to detect variations of source code. As a result, the Test Cockpit view shows source files with all their tested variations.



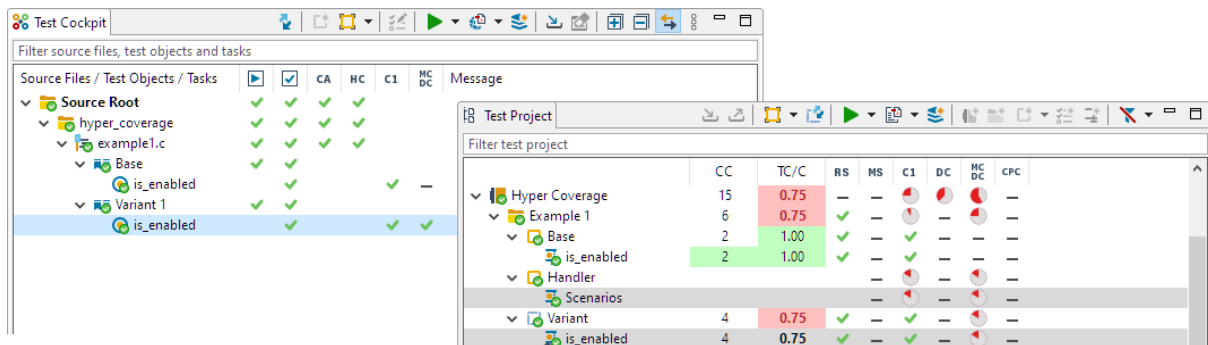
The screenshot displays the Test Cockpit interface on the left and a code editor on the right. The Test Cockpit shows a tree view of source files and test objects. The 'code_access_demo.c' file is selected, showing three variations: #179fa5b, #201856e, and #b8b09fe. Each variation has a green circle icon and a 'CA' column with a red circle icon. The code editor shows the source code for 'code_access_demo.c' with the following content:

```
1 //
2 // Code Access (CA) demo
3
4 int getAcceleration() {
5     int result = 0;
6
7 #ifdef MODEL_A
8     #if CYL == 4
9         result = 20;
10    #elif CYL == 6
11        result = 35;
12    #endif
13
14    #ifdef TURBO
15        result *= 2;
16    #endif
17 #endif
18 #ifdef MODEL_B
19     result = 15;
20 #endif
```

After analysis of all modules, the source code view within the Coverage Viewer perspective highlights any source code lines that are not accessed within any of the source code variations being tested with the existing modules (i.e. due to preprocessor directives hiding them within the preprocessed code). This result is already available after creation and analysis of modules, so that testers have a quick overview after setting up tests if there are any untested parts of the source code.

Hyper Coverage

The new Hyper Coverage features provides accumulation of coverage results across different tests, testing levels and test tools. The Hyper Coverage applies the normal coverage measurements (e.g. branch or MC/DC coverage) to create a relation between the measured coverage results with respect to the different code variations. The existing bounds of coverage measurements for different code variants were overcome which allows an accumulation of coverage based on the original source code lines.

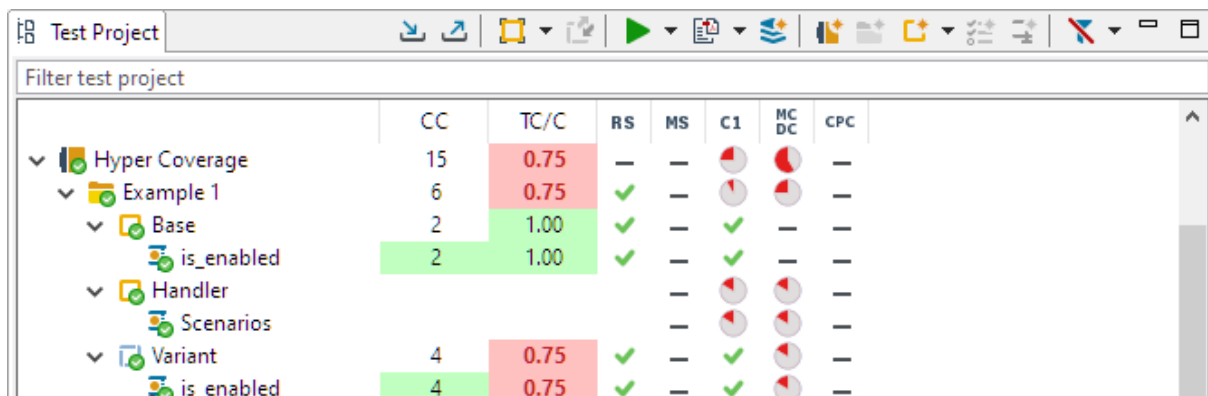


The Test Project view is now dedicated for working with test objects whereas the Test Cockpit view summarizes all results and coverage for each of the test objects within the source files. Selections in both views are synchronized so that related test objects will be revealed when selecting elements in either one of both views.

Coverage results from different unit and component tests will be combined so that it is easier to reach full coverage for all test objects being shown with their summarized results within the Test Cockpit view.

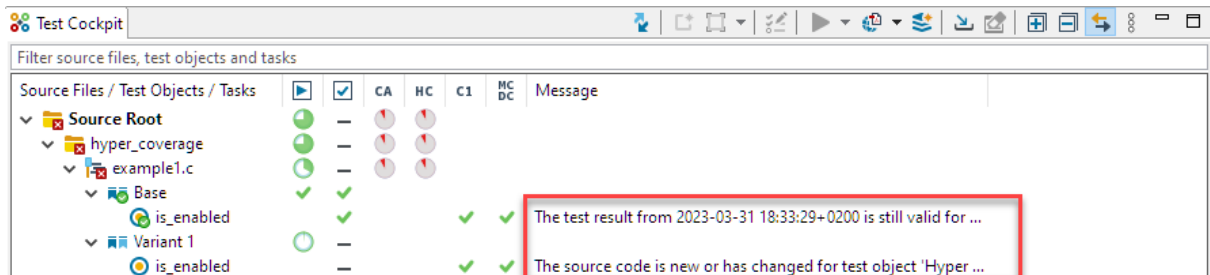
Changed behavior of Test Project view

A new default setting for the Test Project view ignores the coverage results for the test result status icons of test collections, folders, modules and test objects. The coverage results will still be summarized up to the test collection within the coverage columns but the test result excludes the achieved coverage.



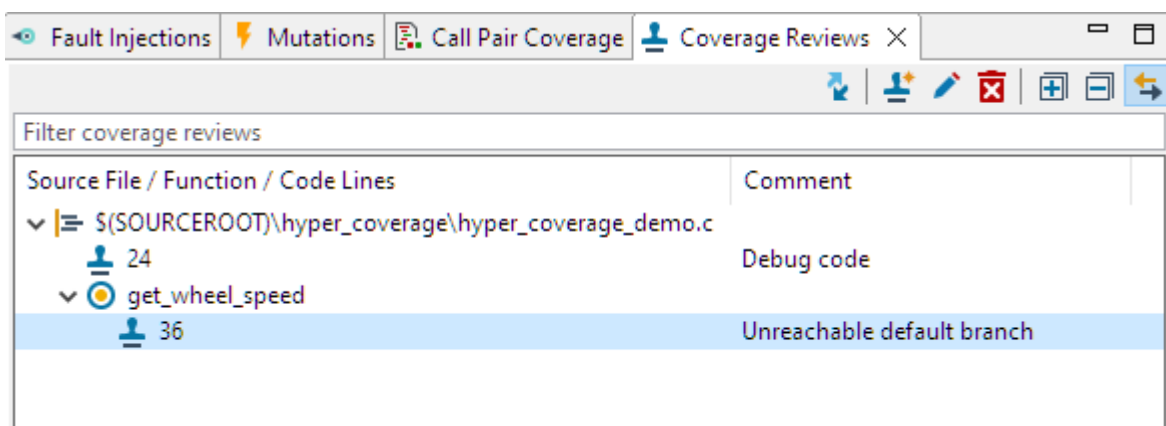
This setting can be changed within the preferences to revert to the legacy behavior.

Also the module analysis will now only discard results shown within the Test Project view. Any results for unchanged test objects will still be available within the Test Cockpit view even after a module analysis. This setting can also be changed to the legacy behavior within the preferences. Messages within the Test Cockpit view will provide information about results being kept:

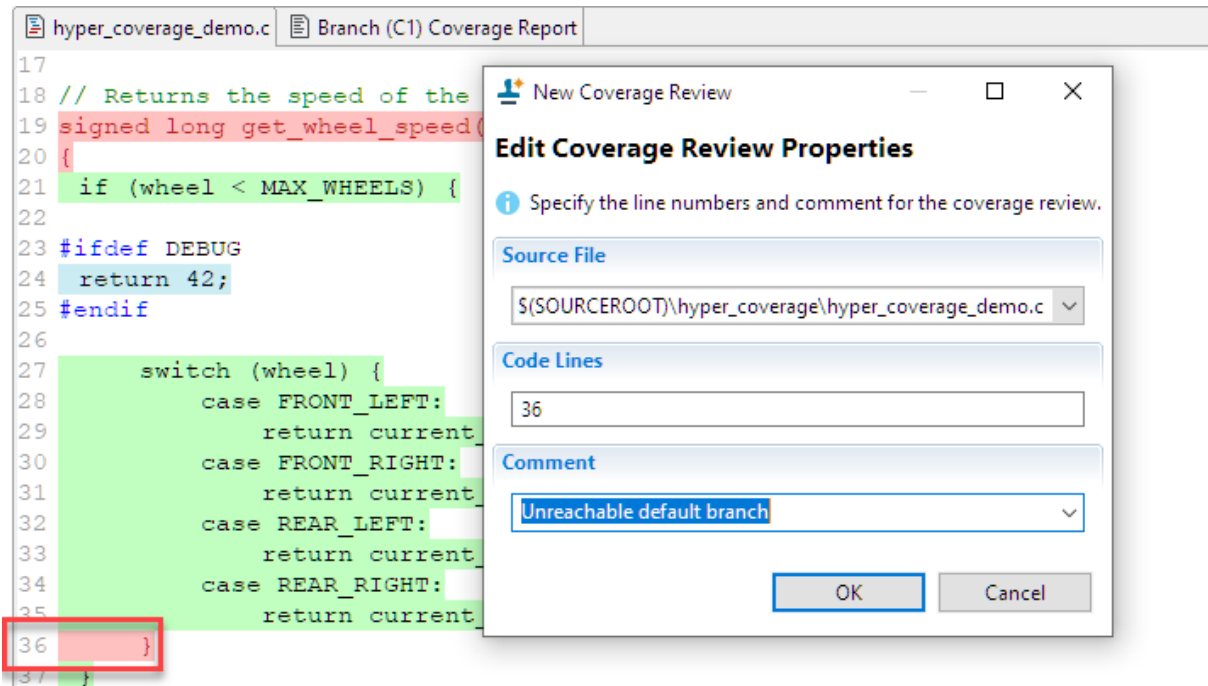


Coverage Reviews

The new coverage review feature supports handling of unreachable source code lines when measuring code coverage using the new Code Access (CA) and Hyper Coverage (HC) features. Source code lines can be marked with predefined as well as arbitrary comments for documentation of why they cannot be reached. Typical situations are hidden debug code or unreachable default branches.



The Coverage Reviews view within the Coverage Viewer (CV) perspective lists the reviews for each source file. New coverage reviews can be added using the source code view that highlights any unreachable code lines.



The reviewed source code lines will be added to the Code Access (CA) and Hyper Coverage (HC) measurement so that it is always possible to reach full coverage by using the standard coverage measurements in combination with the coverage reviews. All coverage reviews will be documented within the test summary report.


Test summary report

The new test summary report replaces the former test overview report. It provides the condensed summary of the current status of the test project based on the tested source files showing test and coverage results as well as coverage reviews.

TEST SUMMARY REPORT

Project EW 2023

2023-03-31, 16:39:36+0200



Coverage Reviews

The following table lists all coverage reviews for source code lines not reviewed contribute to the Code Access (CA) and Hyper Coverage (HC)

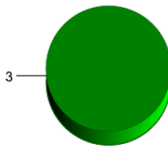
Source File / Line Number
\$(SOURCEROOT)\hyper_coverage\example1.c 24
\$(SOURCEROOT)\hyper_coverage\hyper_coverage_demo.c 24 36

Summary

Total Source Files: 3
 Successful: 3
 Failed: 0
 Not Executed: 0
 No Tests: 0

Date: 2023-03-31
 Time: 16:39:36+0200
 Source File Filter: Show all source files

Results per Source File



- Successful
- Failed
- Not Executed
- No Tests

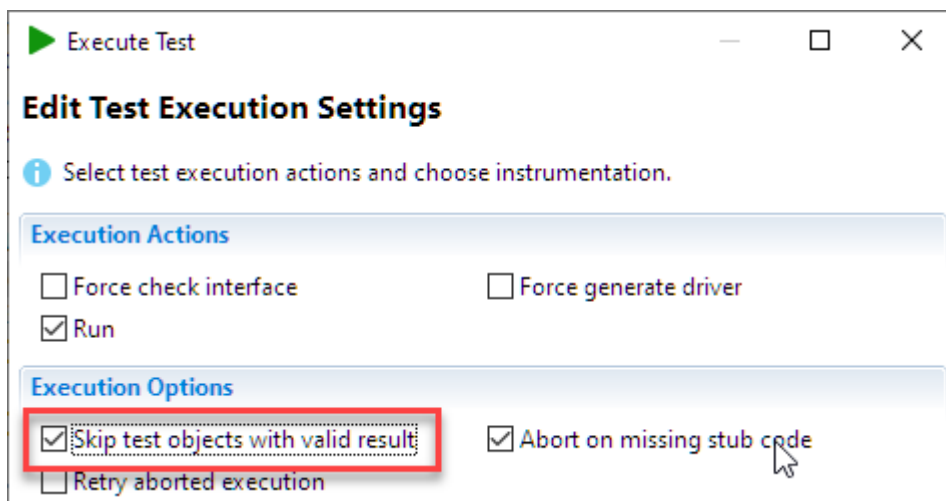
No.	Name	CA	HC	Number of Code Lines	Test Result	Overall Result
\$(SOURCEROOT)\hyper_coverage						
1	example1.c	100 %	100 %	13	✓	✓
2	example1_handler.c	100 %	100 %	8	✓	✓
3	hyper_coverage_demo.c	100 %	100 %	24	✓	✓

The test summary report XML file contains checksums of all test objects together with the current test results. Such an XML file can be loaded as base summary for subsequent test runs so that it is possible to execute only changed test objects.

Change based testing

When testing new versions of a source code, former results for unchanged source code parts will automatically be reused and displayed within the Test Cockpit view. The analysis of a module may discard the existing results within the Test Project view but they will still be applied for unchanged test objects within the Test Cockpit view.

For test execution you can decide to run only tests where the tests objects have been changed or tests that have been updated since the last execution:



The necessary information about former results is retrieved from test summary report XML files. Any old test summary report XML file can be loaded as baseline for tests and test objects. This feature can dramatically reduce the test execution time for recurring continuous testing on CI systems because only changed tests or code parts will be tested again.