Razorcat Development GmbH

Email: support@razorcat.com

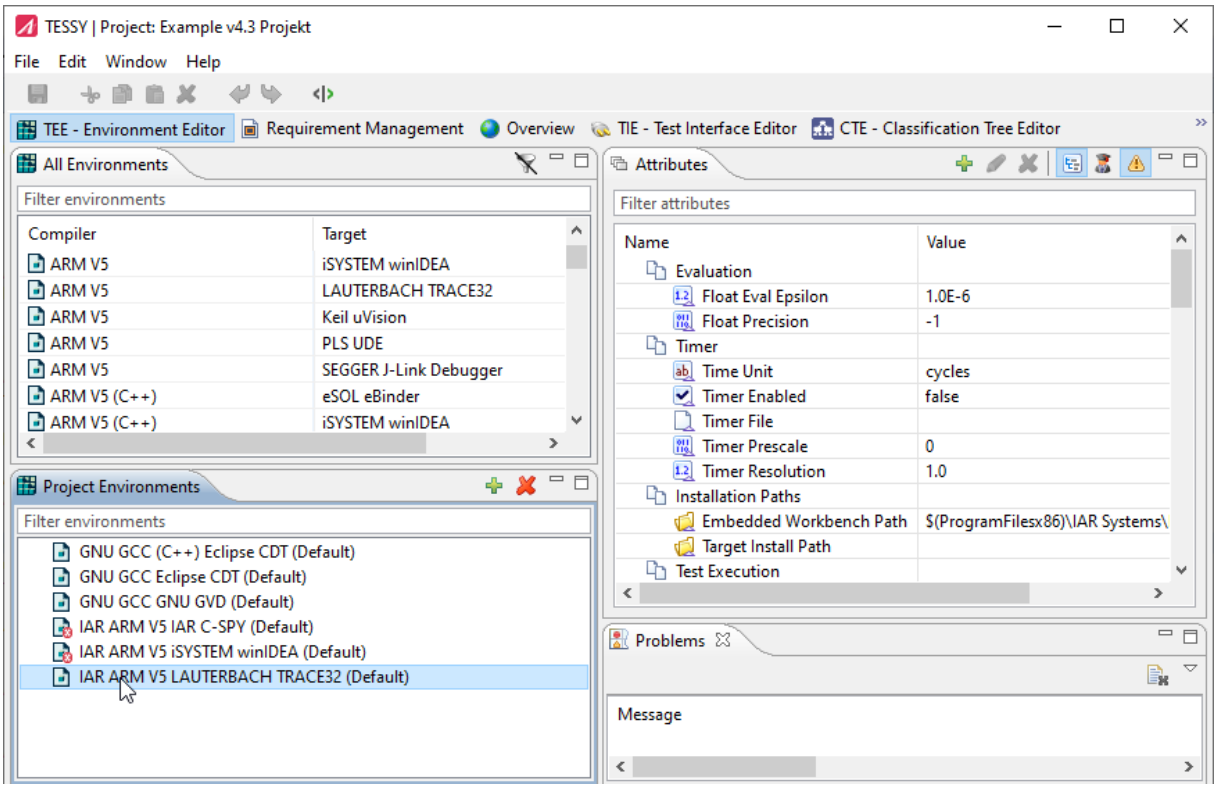
Phone: +49 - 30 - 536 357 0

New features in TESSY v4.3

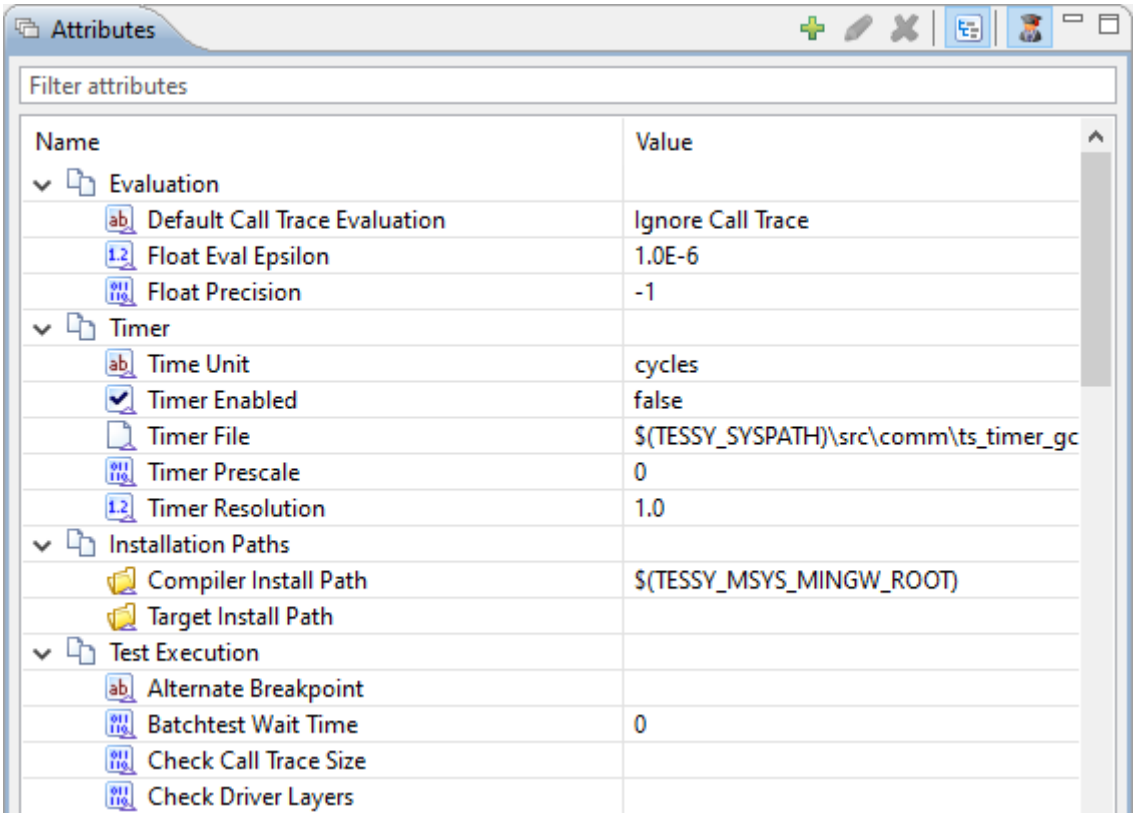
TEE perspective

The new environment editor (TEE) perspective provides editing of the project configuration which is stored within the project configuration file. It contains three views:

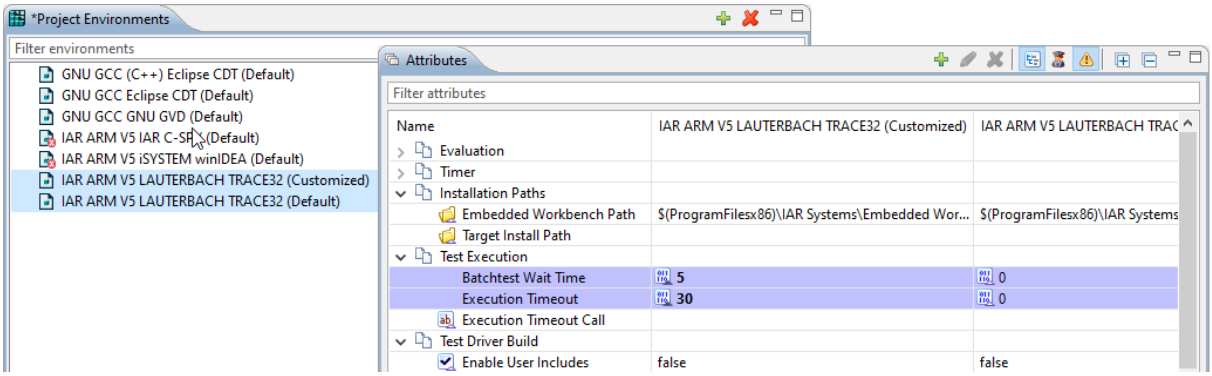
- The “All Environments” view which contains all available system configurations supported by TESSY.
- The “Project Environments” view contains the environments that are selected for the current project and stored within the configuration file.
- The “Attributes” view which shows the attribute settings for one or several selected environments within the “Project Environments” view.



All views have filters to easily find desired elements. The “Attributes” view shows the list of attributes within groups or as plain list. The groups are defined within the system configuration file which is part of the TESSY installation.



Attribute settings can be compared and assigned between different project environments to facilitate merging of related or different compiler/target combinations.

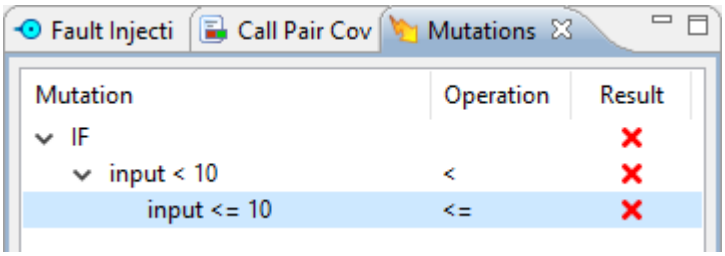


Mutation testing for test case quality analysis

The new mutation test feature automatically checks the error detection capability of existing test cases. This unique function thus improves the review of test methods and test quality, as required by the standards for functional safety (IEC 61508, IEC 62304, ISO 26262 and EN 5012) and significantly reduces the manual review efforts.

The mutation test minimally "mutates" the C/C++ source code to be tested at error-sensitive locations and thus incorporates typical programming errors. If the unit and integration tests detect the error, these tests are considered useful. Non-detection indicates weak test cases, and the mutation test gives hints where they can be optimized and a better quality in testing can be achieved. With this unique method, test quality can be evaluated easily and automatically.

The new mutation test feature automatically checks the quality of existing tests. Auditors can use it to subject all tests to quality control in the shortest possible time and to verify compliance with the standards for test creation. Especially in the development of safety-critical systems, a high verifiable quality of functional and non-functional tests is crucial for the safety of embedded applications and their functional safety certification.



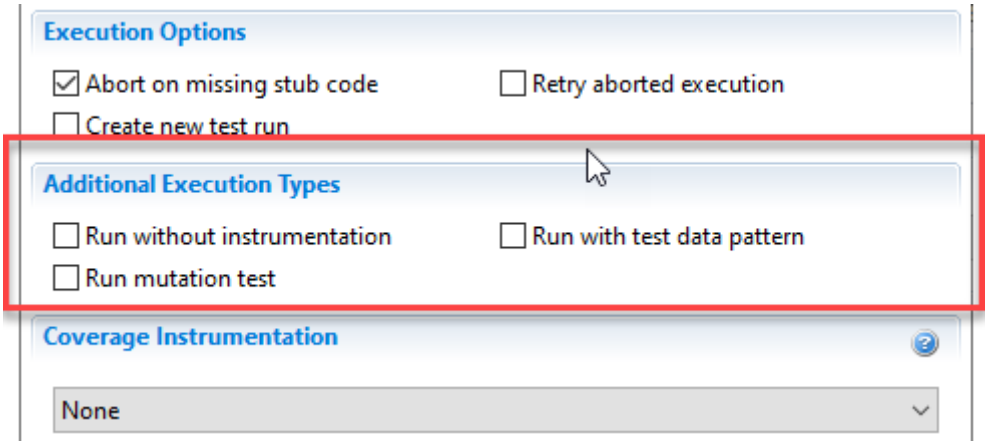
The mutation view lists all mutated code locations and shows the result (i.e. if any test has detected to mutation). It is possible to exclude mutations in order to prevent from mutations that cannot be found (e.g. equivalent mutants). For component testing, the coverage information is used to determine all code locations that are reached and though should be mutated. This significantly reduces the amount of mutations because only code locations exercised by the tests are taken into account.

Initialization of OUT variables with test data patterns

In order to check whether variables with passing direction OUT have really been written during execution of a test case, two different test data patterns can automatically be applied to initialize such variables (or components of structs/unions) without the need to change the passing direction. This feature checks the independence of test cases and ensures that all expected outcomes have explicitly been calculated.

Additional test execution types

Mutation testing as well as test data initialization patterns can be activated as additional test execution types within the normal test execution dialog:



The following optional execution types can be selected:

- “Run without instrumentation” executes and evaluates only those test cases and evaluation checks that can be applied independently of any source code instrumentation (e.g. coverage measurement, testing of static local variables or call trace as well as fault injection requires code instrumentation). This mode shall ensure that the instrumentation does not change the behavior or hides any errors within the software.
- “Run with test data pattern” initializes all OUT variables with two configurable patterns.
- “Run mutation test” executes the test cases with mutated code.

All additional executions are conducted after successful completion of the normal test execution (either with or without coverage measurement) and all of them must yield the same results as the normal test execution.