TESSY V3.2 Features

Frank Büchner, April 2015 --- 002

New features in TESSY V3.2 (compared to TESSY V3.1)

Contents

1 Static Analysis

TESSY V3.2 can perform static analysis of the source code, e.g. to check for compliance with the MISRA rules. TESSY uses external tools for this purpose; currently PC-Lint by Gimpel and the open source tool CppCheck are supported.

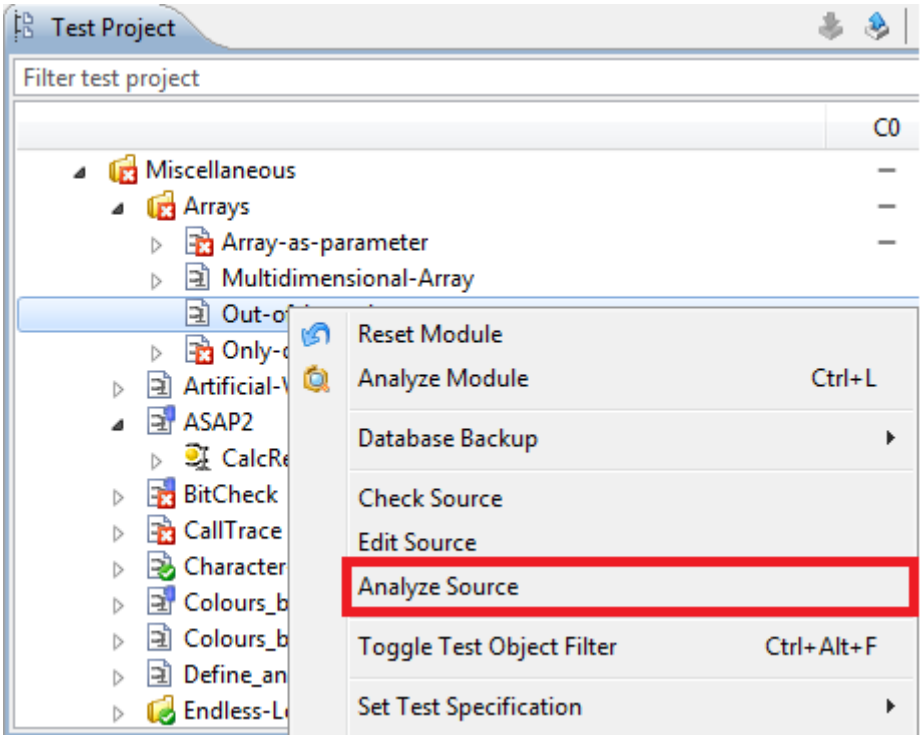

Fig. 1: This command initiates a static analysis either by PC-Lint or by CppCheck

The messages of the tools for static analysis are displayed in the Console view of TESSY. The messages include a link to the respective location in the source file and by a simple click of the mouse this location is displayed in the C/C++ perspective in TESSY. There it can be reviewed and corrected easily, if appropriate.

2 Report Format Word and HTML

Additional to the hitherto supported report formats PDF and XML, TESSY V3.2 can create test reports in Word format and in HTML format. The latter allows you conveniently viewing test reports in a browser, enabling easy resizing of the test reports according to your needs.

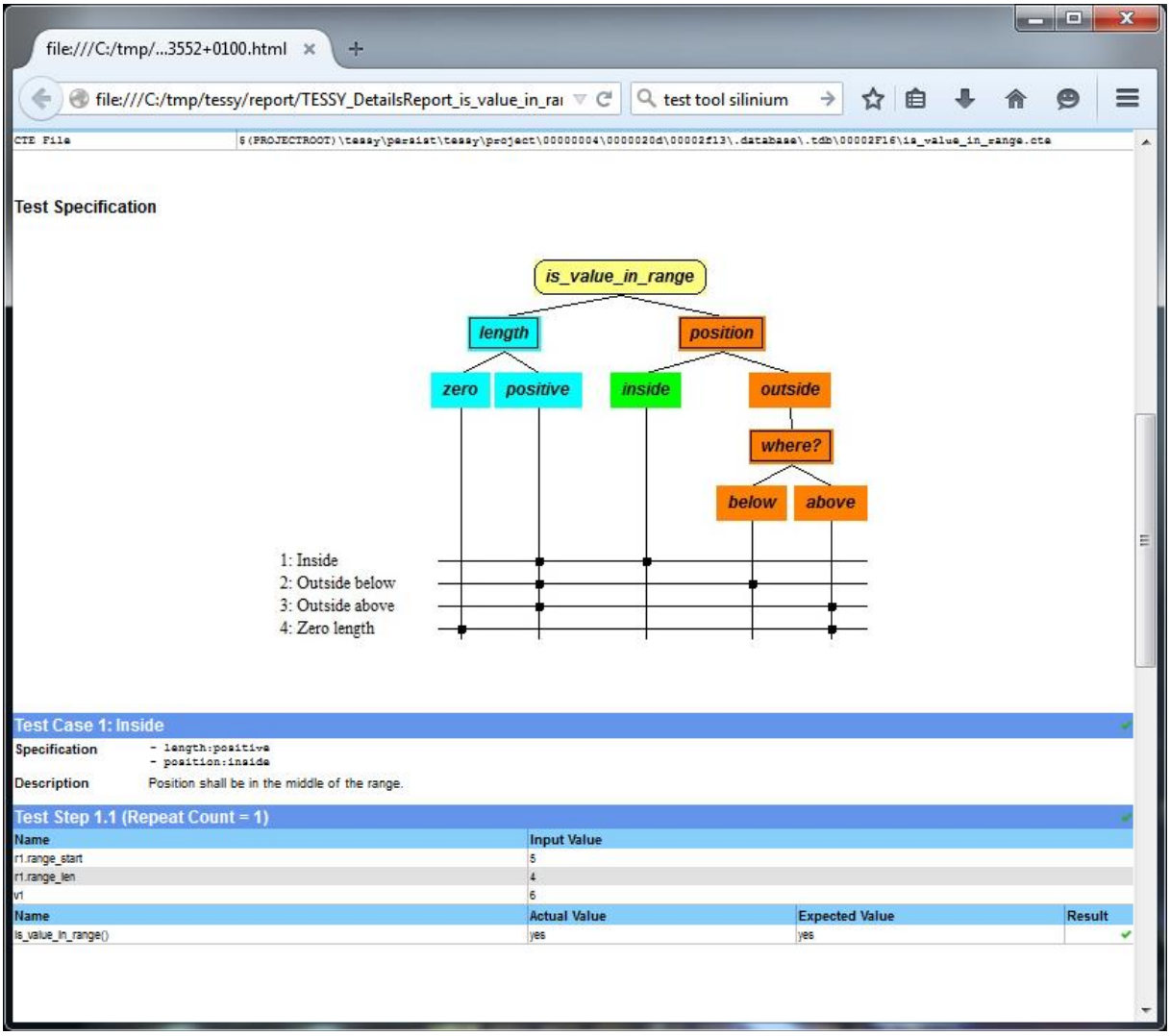

Fig. 2: Excerpt of a test report in HTML format viewed in a browser

3 Improved Graphical Display of Test Data

TESSY V3.2 can display test data with additional data types graphically, e.g. test data of arrays or test data of components of structures. For improved management of additional settings for the graphical display, the hitherto Plot view of TESSY V3.1 was divided in two views: The actual Plot view for the graphical data and the Plot Definitions view for management purposes. This allows easily suppressing the visualization of the test data of certain test cases and provides a comfortable way to select which combination of expected, actual and deviating values are to be displayed.

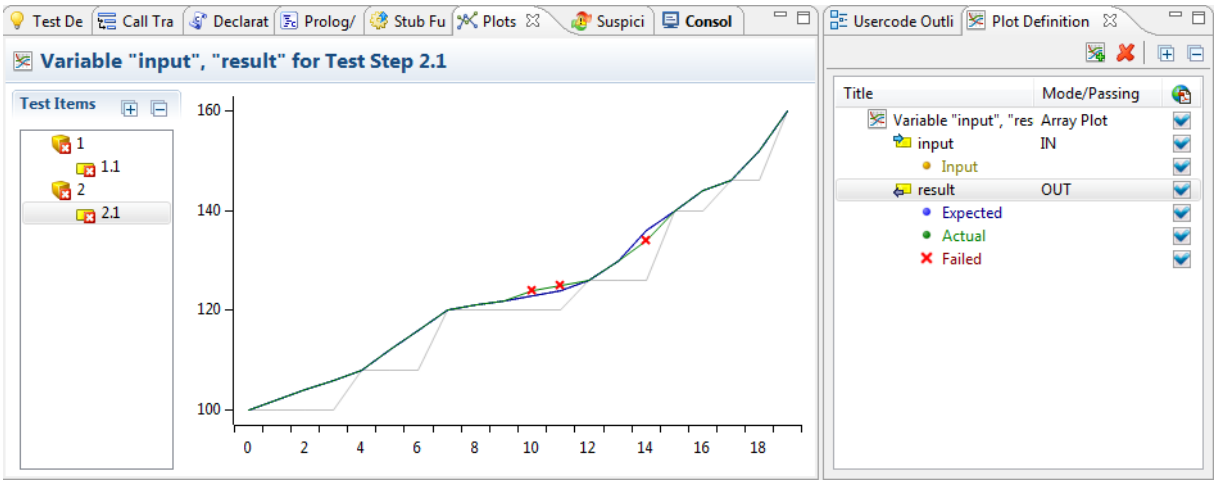

Fig. 3: The displayed test data is an array of 20 elements, both for input and output

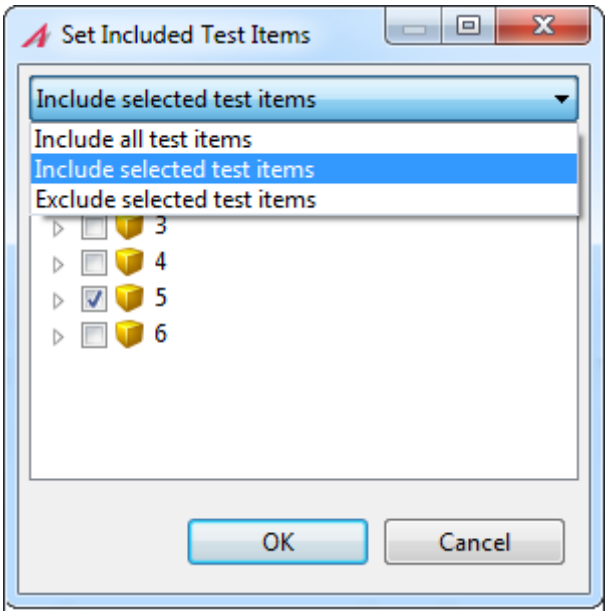

Fig. 4: Test cases can be included / excluded from the Plot view

4 Reuse of Existing Test Data

Existing test data from other test objects can now easily be reused for the current test object. This is even possible for test objects from other TESSY modules. The data types do not need to be identical. This allows easy testing of variants of test objects.

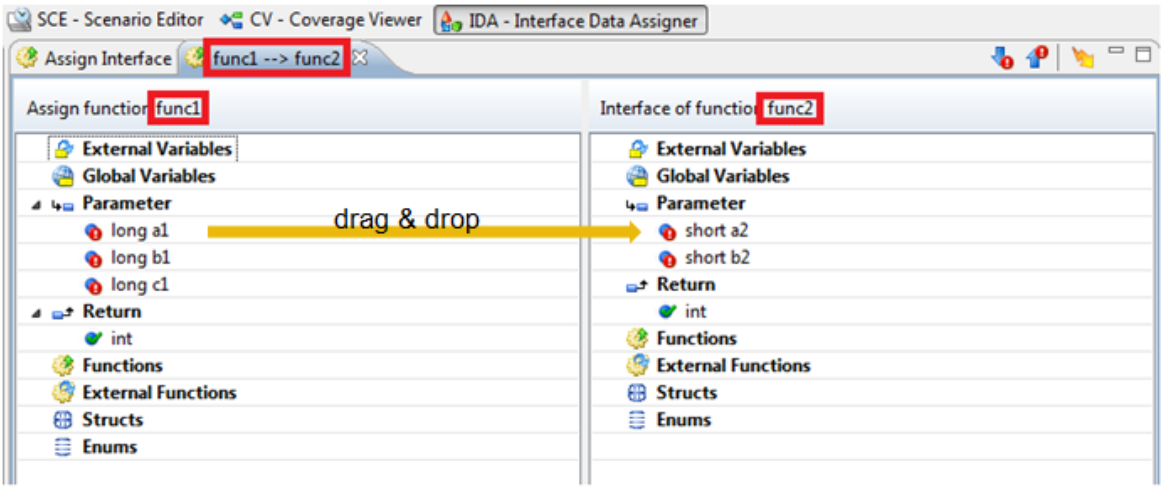

Fig. 5: Reuse test data from other test objects

Also the test data of global variables which are used by several test objects of a C source module can now be reused (e.g. after a name change) by a single operation.

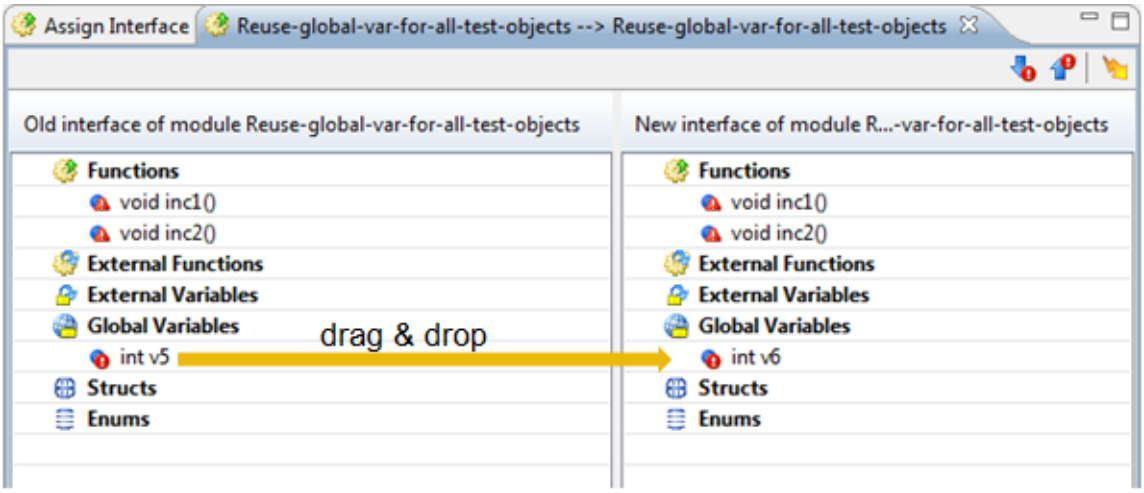

Fig. 6: Reuse test data for several test objects at the same time

In the figure above, both inc1() and inc2() use a global variable, originally named v5. When the name of the variable is changed (e.g. to v6), you can assign the test data of v5 to v6 for both inc1() and inc2() at the same time by a single drag&drop operation. In TESSY V3.1, a additional activity was necessary for inc2(), even if you had assigned v5 to v6 for inc1() already.

5 Hide Irrelevant Test Objects

If a test project contains test objects which are not relevant for the current testing objective, TESSY V3.2 allows hiding such unwanted test objects. They will not be mentioned in the test report and they will normally not be displayed in the test project view any more.

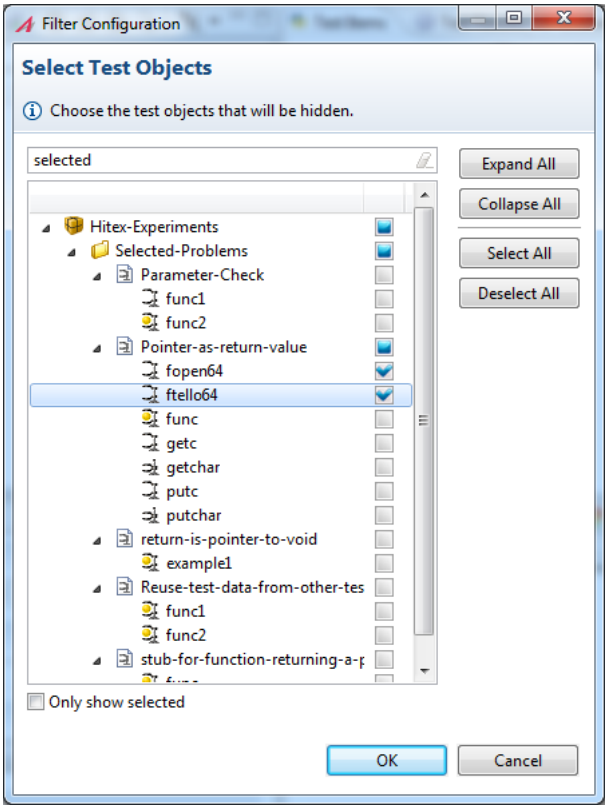

Fig. 7: Selected test objects will be hidden

In the figure above, the test objects fopen64() and ftell64() show up, because stdio.h is included. (Also getc(), getchar(), putc(), and putchar() stem from stdio.h). To hide fopen64() and ftell64(), we select the two test objects in the figure above.

No.	Name	Test Cases	Result
	Hitex-E1	1 of 3 failed	
	Hitex-Experiments	1 of 3 failed	
	Selected-Problems	1 of 3 failed	
	Pointer-as-return-value	1 of 3 failed	
1	func	1 of 3 failed	
2	getc	-	
3	getchar	-	
4	putc	-	
5	putchar	-	

Fig. 8: The hidden test objects are excluded from the test report

6 Extended Use of Views

With TESSY V3.2 selected views can now be displayed additionally in other perspectives than the perspective they originate from. This allows you to decide which information you want to see side by side.

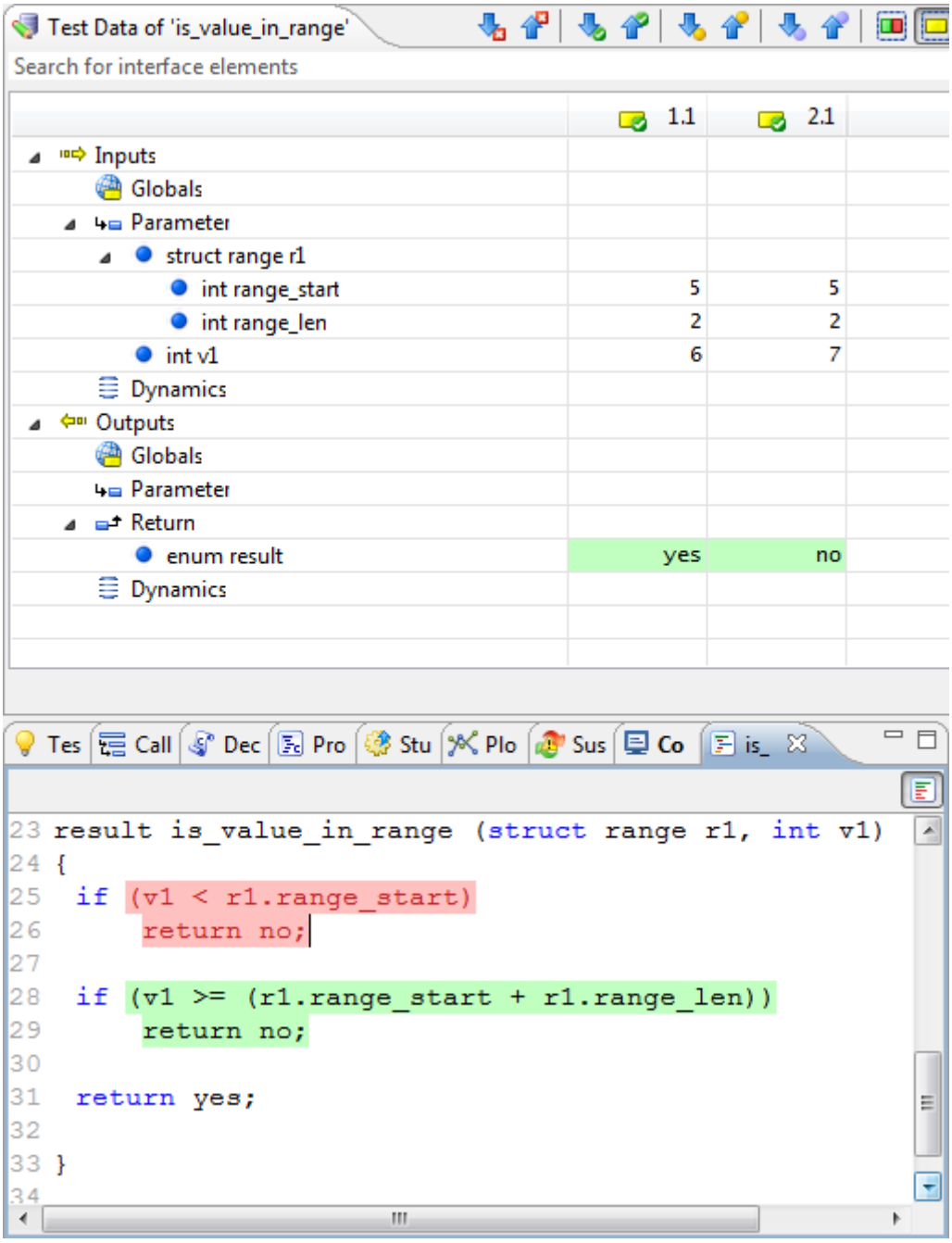

Fig. 9: The Source Code view from the Coverage perspective in the TDE perspective

7 Global User Code

Since ever it was possible to direct TESSY to execute user-supplied C source code prior and after execution of a test (and a test step). Up to now, this code needed to be specified for each test case / test step explicitly. (Of course, it could be duplicated easily.) With TESSY V3.2 it is possible to specify this user-supplied code globally one time; the code then will be passed on to all test cases / test steps automatically. Of course, the inherited code can be overwritten individually. Even additional test cases / test steps get assigned the globally defined code automatically.

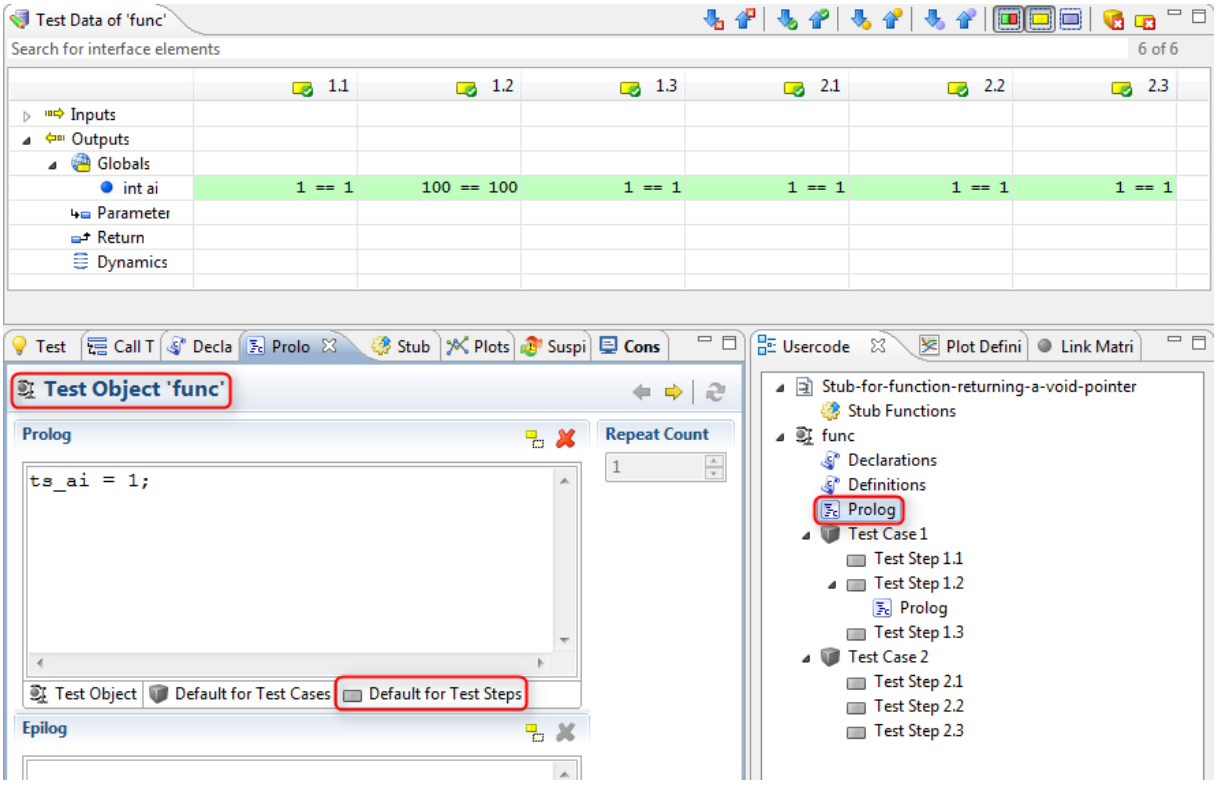

Fig. 10: User-supplied C code can be set as default for test cases and test steps

8 Notes in the Test Details Report

If enabled in the Test Details Report Settings, notes are included in the Test Details Report.

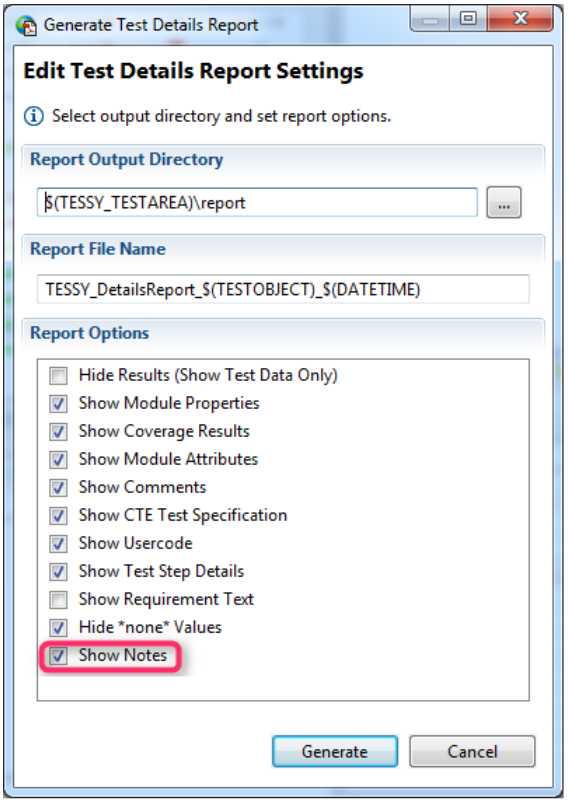

Fig. 11: Enable "Show Notes" in the Test Details Report Settings

Notes

Type	Text
[INFO] Test Object 'evaluate_2bits_w_if'	**Coverage** The else branch of the last if-statement in this test object cannot be executed. This reduces the maximum coverage reachable for this test object to 90% branch coverage.

Fig. 12: A note in the Test Details Report

This might be used to justify coverage measurements below 100%.

9 The Author

Frank Büchner, Hitex Development Tools GmbH, frank.buechner@hitex.de

Any comments or questions to this document are welcome.