

Flash Technology

How to perform the

Unit Testing

and

Integration Testing

required by

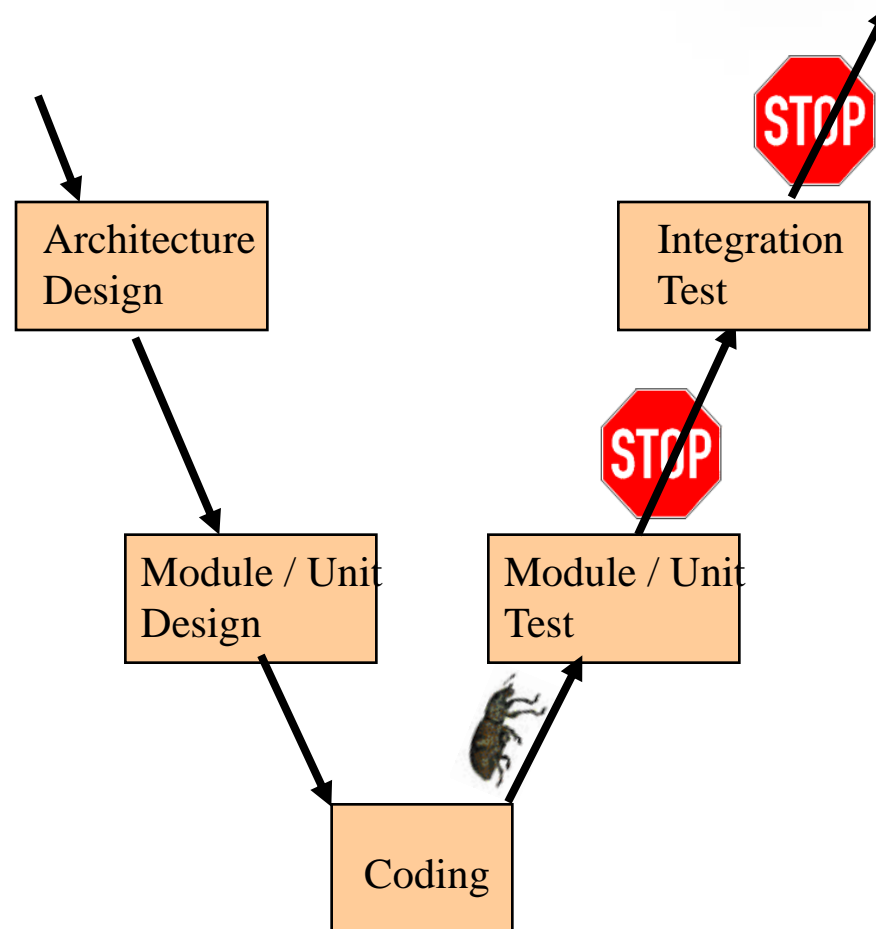
ISO 26262



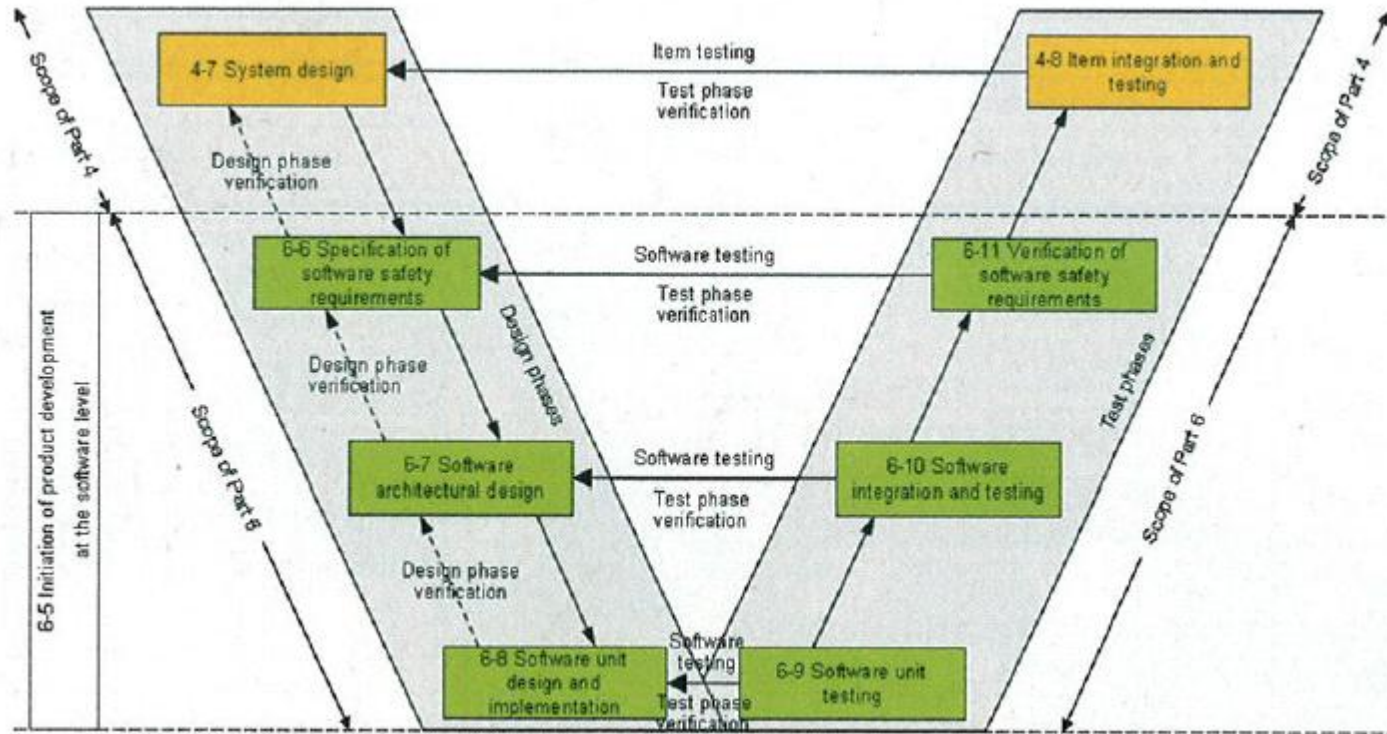
FLASH TECHNOLOGY

One Stop Tools Solution

General V-Model of System Engineering



26262 V-Model of System Engineering



ISO 26262 Requirements

Table 11 — Methods for deriving test cases for software unit testing

Methods		ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Generation and analysis of equivalence classes ^a	+	++	++	++
1c	Analysis of boundary values ^b	+	++	++	++
1d	Error guessing ^c	+	+	+	+

^a Equivalence classes can be identified based on the division of inputs and outputs, such that a representative test value can be selected for each class.

^b This method applies to interfaces, values approaching and crossing the boundaries and out of range values.

^c Error guessing tests can be based on data collected through a "lessons learned" process and expert judgment.

The Classification Tree Method (tool included in Tessy) supports 1a, 1b, 1c and even 1d.



ISO 26262 Requirements

ISO/FDIS 26262-6:2010(E)

Table 12 — Structural coverage metrics at the software unit level

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

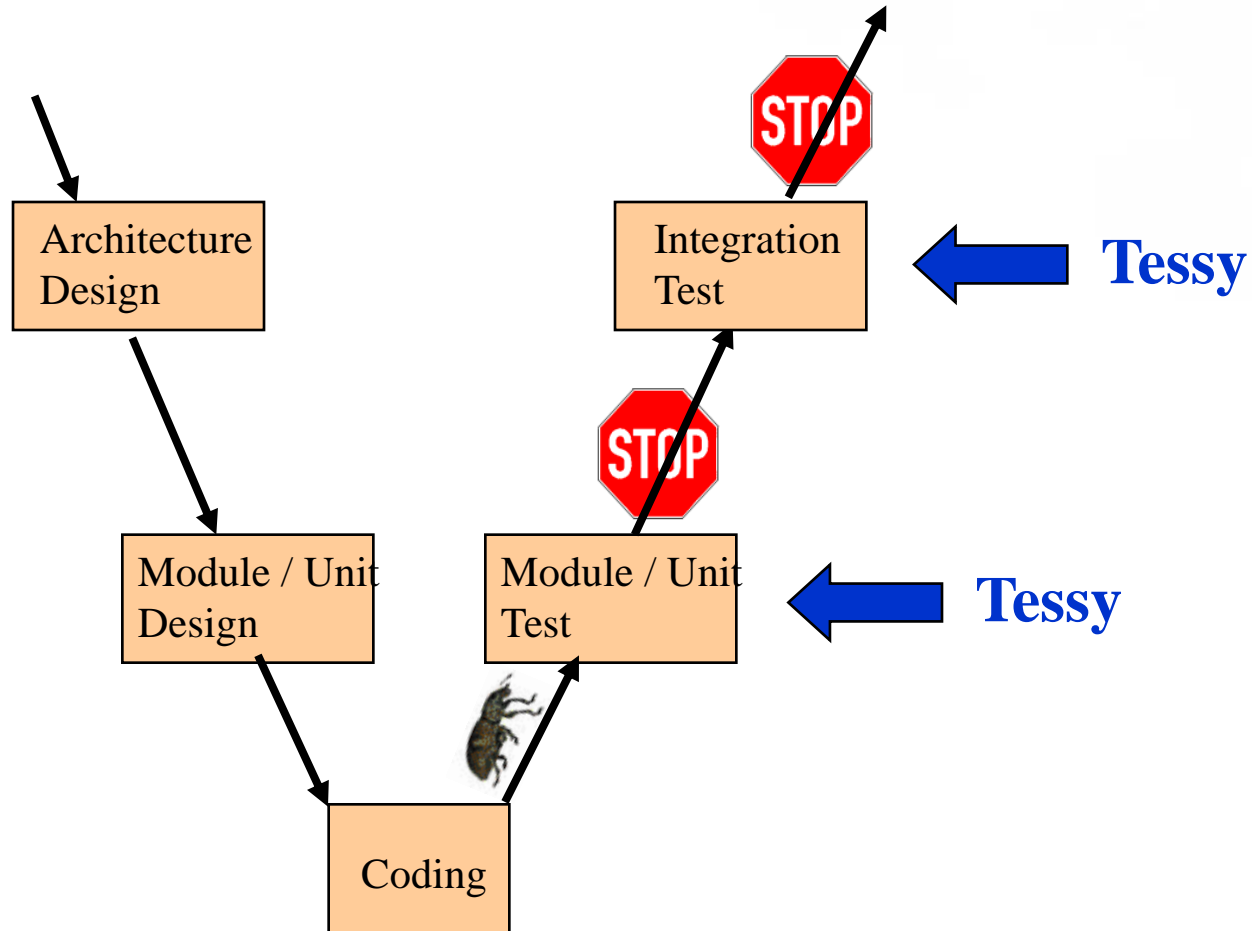
NOTE 1 The structural coverage can be determined by the use of appropriate software tools.

NOTE 2 In the case of model-based development, the analysis of structural coverage can be performed at the model level using analogous structural coverage metrics for models.

NOTE 3 If instrumented code is used to determine the degree of coverage, it can be necessary to show that the instrumentation has no effect on the test results. This can be done by repeating the tests with non-instrumented code.

9.4.6 The test environment for software unit testing shall correspond as closely as possible to the target environment. If the software unit testing is not carried out in the target environment, the differences in the source and object code, and the differences between the test environment and the target environment, shall be analysed in order to specify additional tests in the target environment during the subsequent test phases.

Tool in use - Tessy



So What is Tessy ?

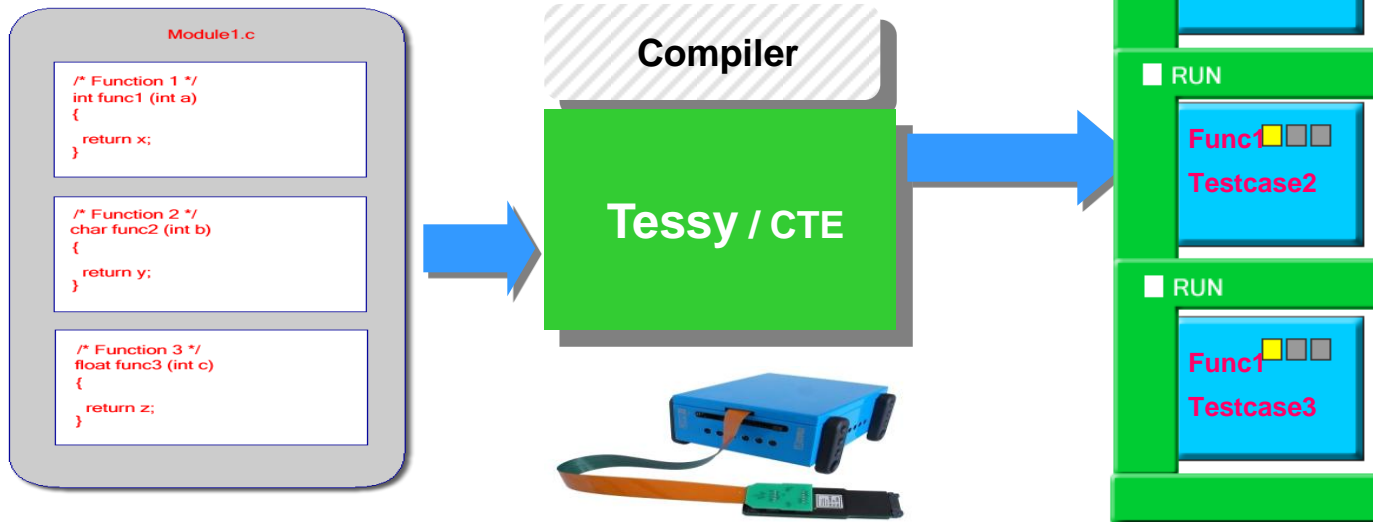


- ◆ Tessy Provide **Module tests** and **Integration tests** according to below standards:
 - **IEC 61508 (ISO 26262), all SIL levels**
 - **DO178B, all levels**
 - **Automotive SPiCE (2005)**
- ◆ **Test coverage**
 - Branch coverage
 - MCC, MC/DC coverage

	Methods
1a	Statement coverage
1b	Branch coverage
1c	MC/DC (Modified Condition/Decision Coverage)

Tessy supports All relevant

- ◆ Microcontroller Architectures
- ◆ Cross-compiler for embedded systems
- ◆ Simulators
- ◆ OCDS/JTAG/BDM
- ◆ In-circuit emulators



9.4.6 The test environment for software unit testing shall correspond as closely as possible to the target environment. If the software unit testing is not carried out in the target environment, the differences in the

CERTIFICATE

No. Z10 11 12 78930 001

Holder of Certificate: Razorcat Development GmbH
Witzlebenplatz 4
14057 Berlin
GERMANY

Factory(ies): 78930

Certification Mark:



Product: Software Tool for Safety Related Development

Model(s): Tessy

Parameters: Tool Classification: T2 (acc. to IEC 61508)
TCL3 (acc. to ISO 26262)

The verification tool fulfills the requirements for support tools according to IEC 61508-3.
The tool is qualified to be used in safety-related software development according to IEC 61508 and ISO 26262.

The test report is a mandatory part of this certificate.

Tested according to: IEC 61508-3:2010
ISO 26262-8:2011

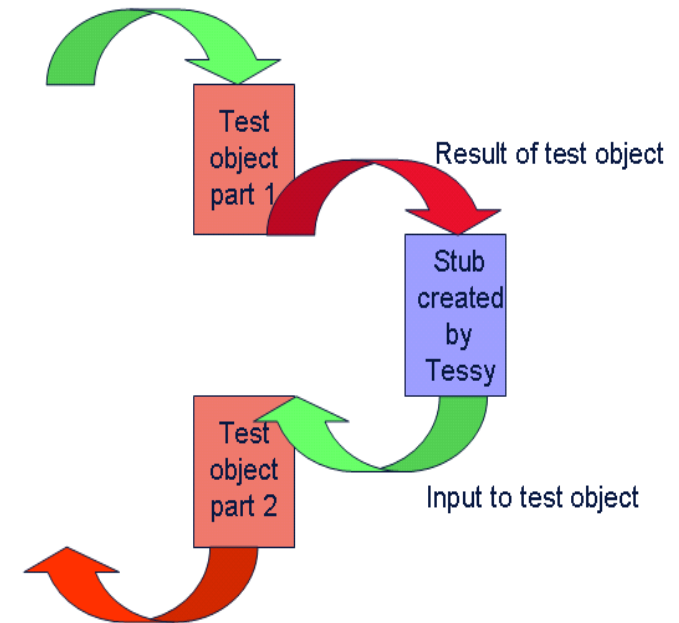
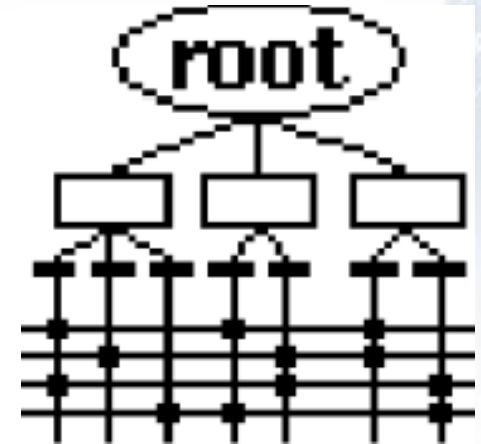
Tessy's Features

◆ Systematic Test Case Design

- According to the Classification Tree Method
- Intuitive, Graphical representation of test specifications.
- Easy to learn

◆ Reproducible tests

- ◆ Automatic test driver generation
- ◆ Test execution **on the target** and evaluation of test results
- ◆ Standardized test reporting and documentation
- ◆ Powerful **regression testing**



Test Planning-Classification Tree Editor (CTE)

The screenshot displays the Classification Tree Editor (CTE) interface. The window title is "building2.cte". The main area shows a classification tree for "building_blocks". The tree structure is as follows:

- building_blocks
 - size
 - small
 - large
 - colour
 - red
 - green
 - blue
 - shape
 - circle
 - triangle
 - shape of triangle
 - equilateral
 - isosceles
 - scalene
 - square

The bottom left panel lists 10 test cases:

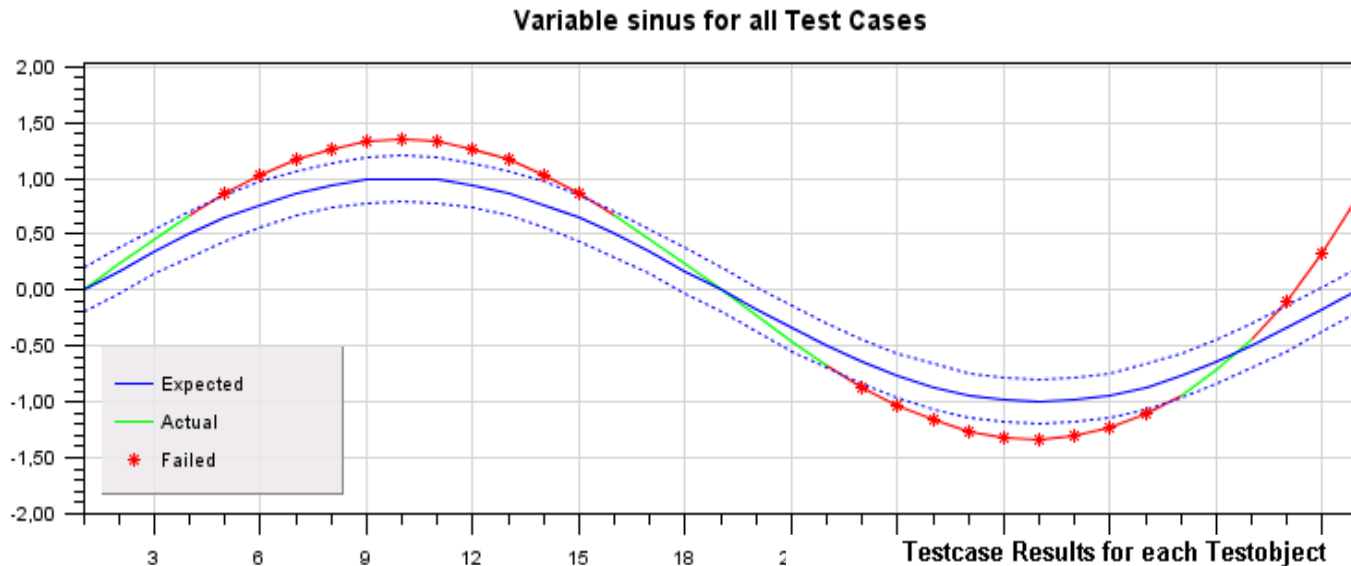
- 1: LargeRedCircle
- 2: SmallGreenSquare
- 3: SmallBluelisosceles
- 4: LargeRedEquilateral
- 5: LargeGreenScalene
- 6: SmallBlueCircle
- 7: LargeBlueSquare
- 8: SmallRedScalene
- 9: SmallBlueEquilateral
- 10: LargeGreenIsosceles

The bottom right panel shows a grid of test cases mapped to the tree nodes, with black dots indicating which nodes are active for each test case.

Various Testing Report

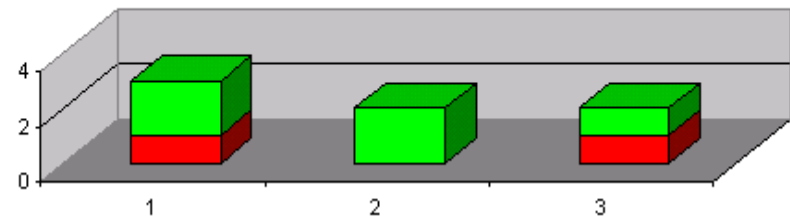
◆ Excerpt from test reports

Testobject Result Plot:



Testcase 2

Teststep 2.1	
Name	Input Value
r1.range_start	5
r1.range_len	2
v1	7



Name	Actual Value	Eval	Expected Value	Result
is_value_in_range	yes	==	no	×

Coverage Result Viewer

The screenshot displays the TESSY Coverage Viewer interface. The main window is titled "CV - Tessy Coverage Viewer" and contains three panes:

- Control Flow Graph:** A graph showing the execution flow of the code. It consists of several nodes (circles) and decision points (diamonds). Red dashed lines indicate the path taken during execution, starting from the top node, passing through a decision point, and then through several other nodes and decision points.
- Code Editor:** A window titled "is_val_in_range_corrected.c" showing the source code:

```
if (r1.range_len == 0)
    return no;

if (r1.range_len > 0)
{
    if ((r1.range_start <= v1)
        &&
```
- Coverage Data:** Two tables are visible on the right side of the interface.
 - MC/D... Table:**

r1.range_l...	Teststep
0	5.1
1	1.1
 - C1-Coverage Table:**

Teststep	Coverage
1.1	30.00%
2.1	30.00%
3.1	30.00%
4.1	30.00%



Traceability Requirements/Test cases

Planning Execution

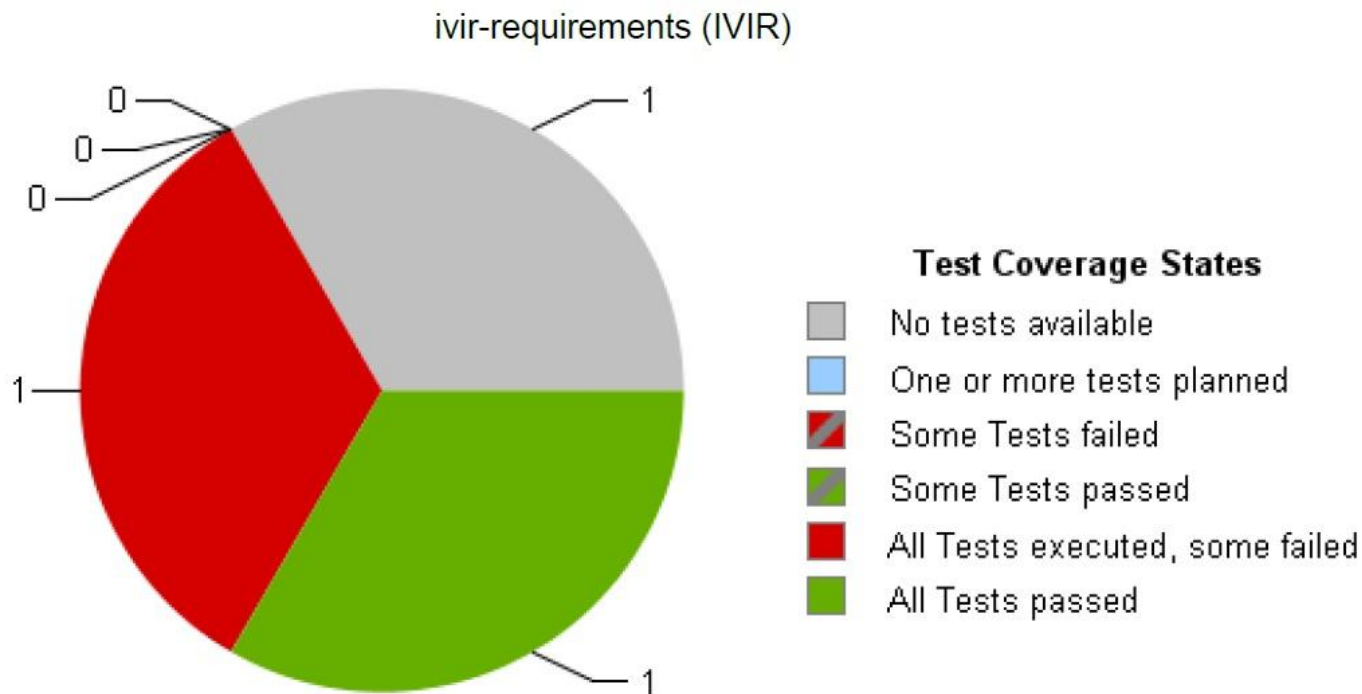
type filter text

▲ All	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2
▲ IVIR [1-1.0] Is a value in a given range? The range is given by a start value and a	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1
▲ Gnu-CDT-Req	<input type="checkbox"/>				1
▲ is_value_in_range		<input type="checkbox"/>			1
1			<input checked="" type="checkbox"/>		1
2			<input type="checkbox"/>		0
▲ IVIR [2-1.0] The end of the range shall not be inside the range.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1
▲ Gnu-CDT-Req	<input type="checkbox"/>				1
▲ is_value_in_range		<input type="checkbox"/>			1
1			<input type="checkbox"/>		0
2			<input checked="" type="checkbox"/>		1
▶ IVIR [3-1.0] If the length is 0, no value is inside the range.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

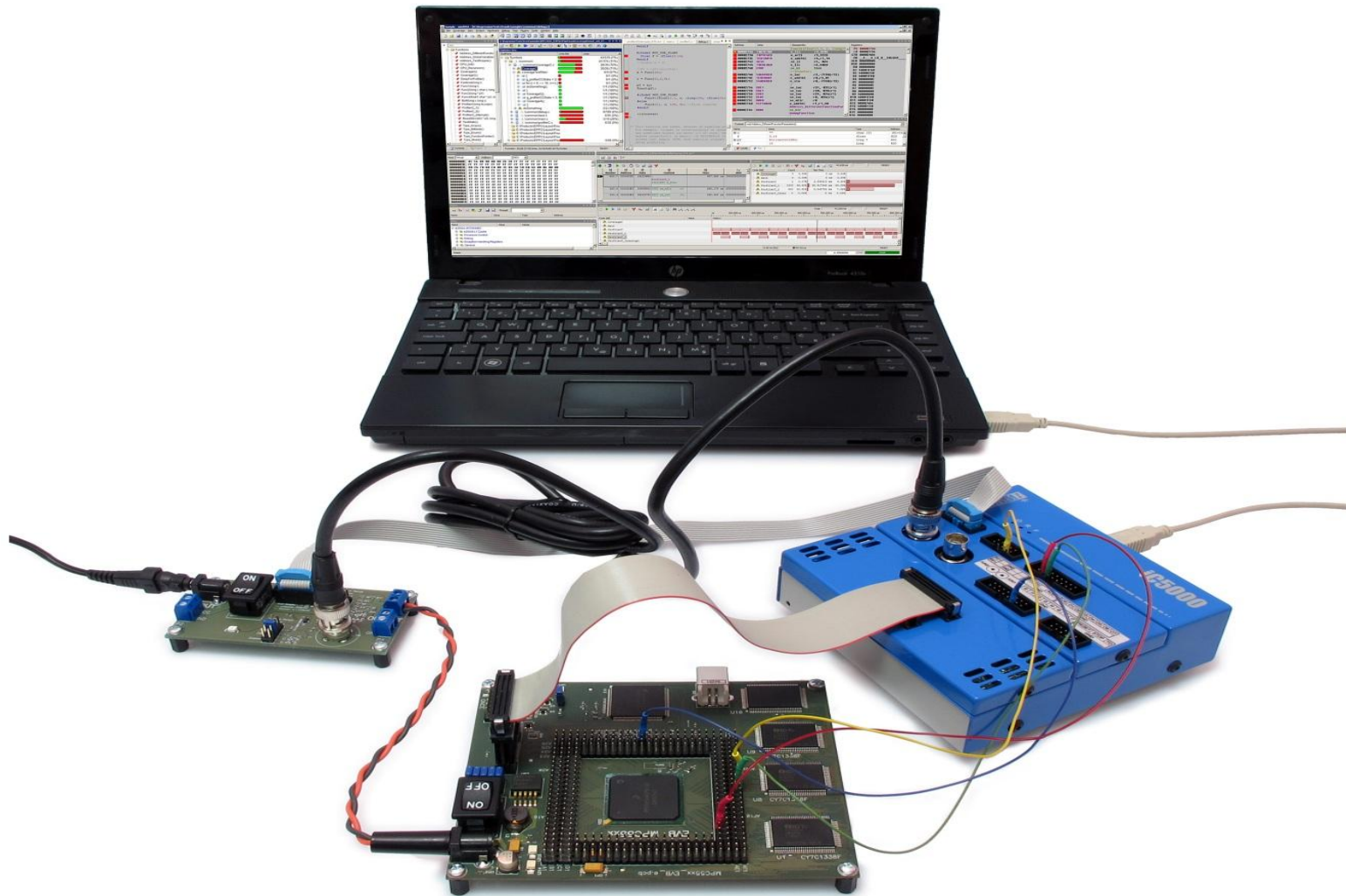


Traceability Requirements/Test cases

Requirement Test Coverage Overview



Email to sales@flashtech.com.cn for demo



FLASH TECHNOLOGY

One Stop Tools Solution