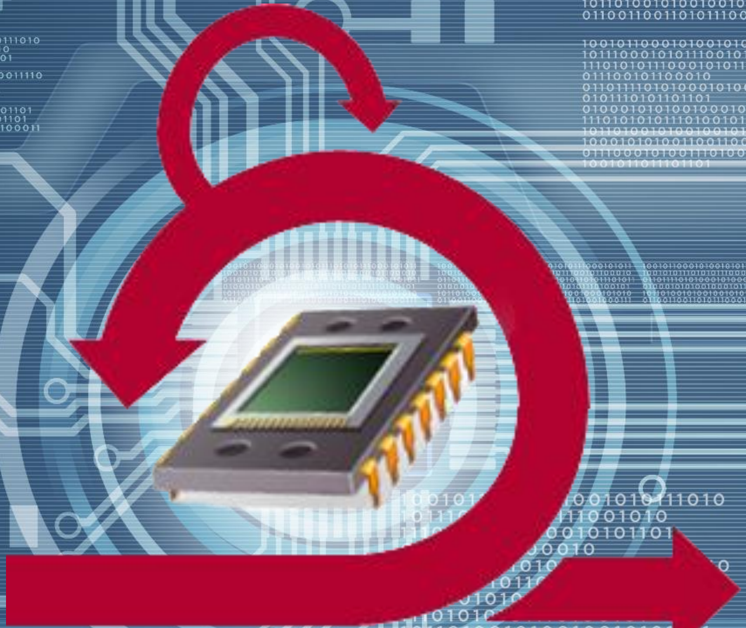
Agile Testing Techniques for Embedded Development

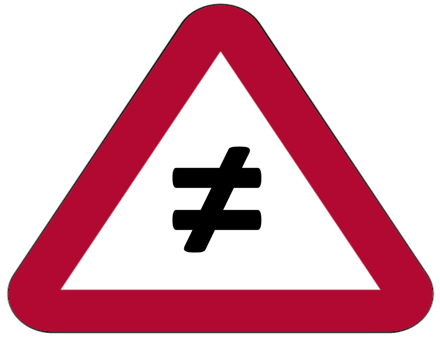


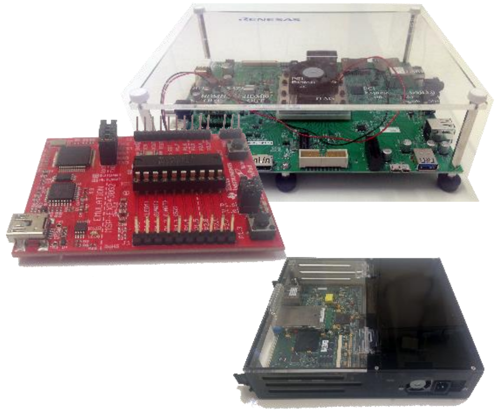
Challenges for Agile Embedded Testing

Challenge: Embedded Target Differences

Why do differences matter for agile testing?

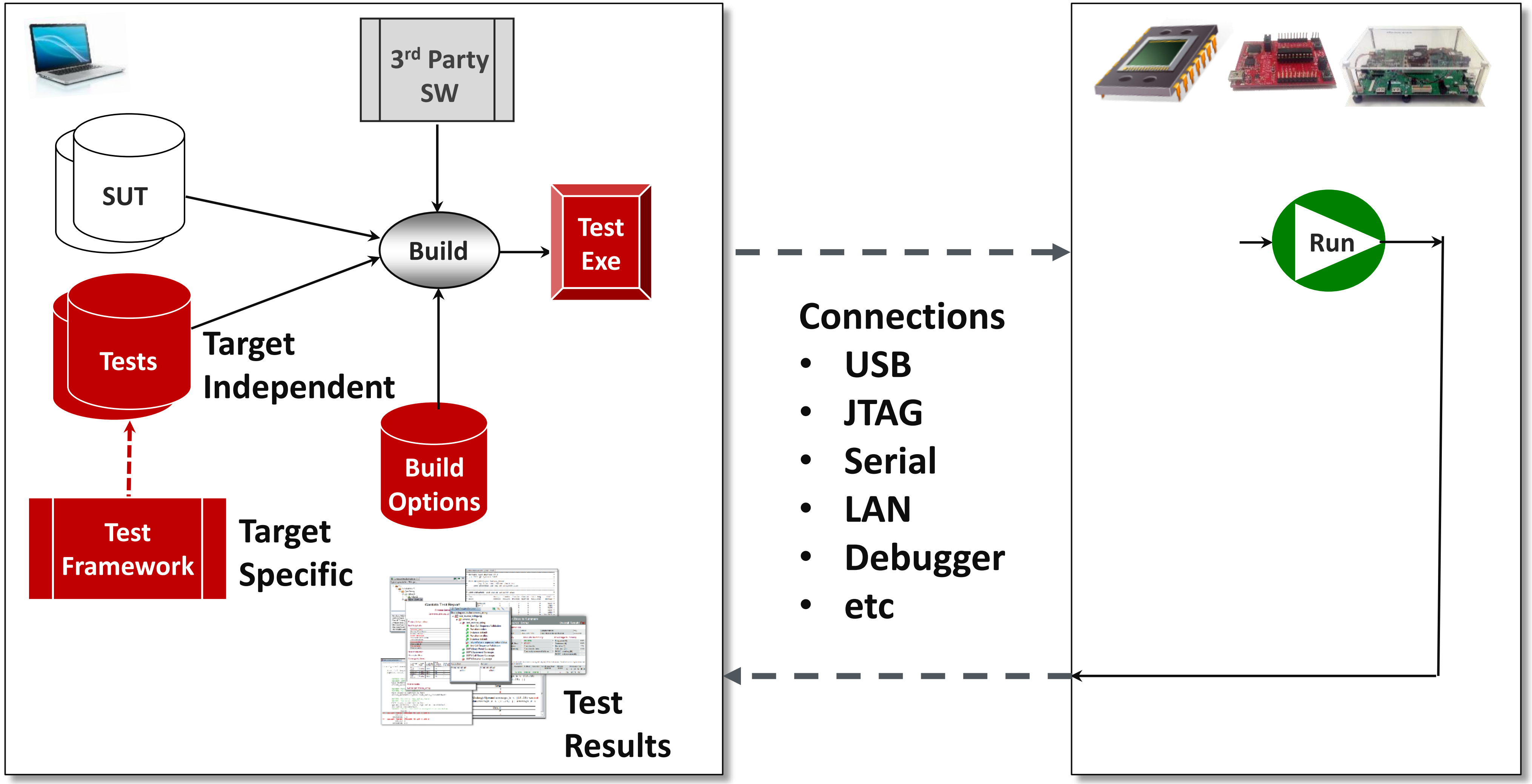
- **Dangers in Host / Target environment differences**
  - Cross-compiler bugs
  - Supposedly standard library functions (printf, scanf, etc)
  - Correct interface operation to target input/output devices
  - Interaction with the RTOS or real time kernel
  - Ordering of bytes within words
  - Word length
  - Structuring, packing of compound data (arrays, records)
  - Data representation
  - Memory constraints
  - Timing errors
- **Agile needs tests using target environment**





Addressing: Embedded Target Differences

Differences can be verified by running tests on both host + target(s)



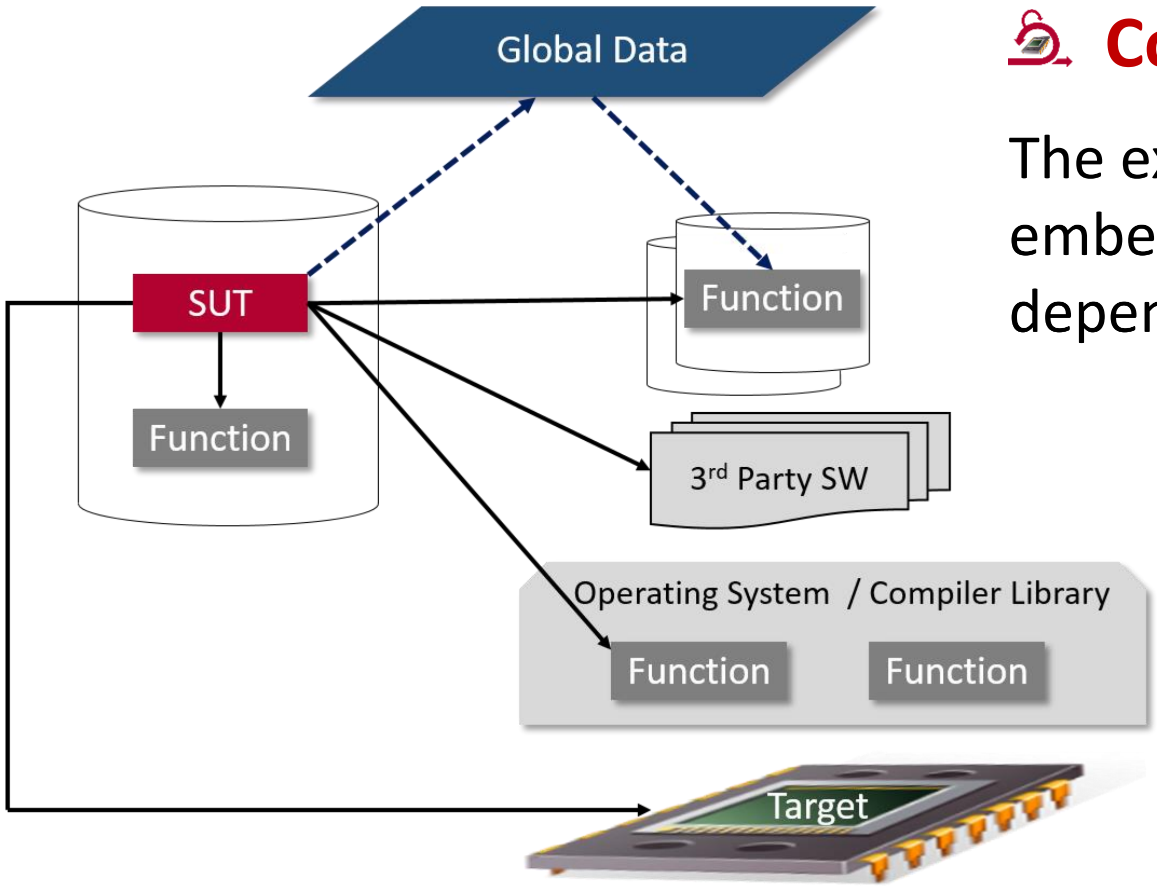
Connections
- USB
- JTAG
- Serial
- LAN
- Debugger
- etc

Embedded Environment Challenges

Embedded Software Under Test (SUT) dependencies

Availability

Whether an embedded environment dependency is available to be tested with.



Complexity

The extra difficulty with an embedded environment dependency in the loop.

Embedded Environment Availability

Availability Challenges

- **Hardware unavailable or changing**
  e.g. being developed in parallel
- **3rd Party SW is SOUP / unavailable**
- **Limited Hardware simulation**
- **Limited Memory on target**
- **Kit provision costs for testing**

How can be addressed

- ✓ Minimal Valuable Hardware,
  Host, then target testing
  Use of Simulators
  Conditional compilation (#define)
- ✓ I/O Interception or Simulation
- ✓ Host, then target testing
- ✓ Supplemented memory / Tools
- ✓ Maximise test automation on embedded platform

Embedded Environment Complexity

Complexity Challenges

- **Embedded test set-up**
- **Root cause analysis**
- **Isolation from Hardware & non-SUT software**
- **Integration with Hardware & software SW**
- **Test automation**

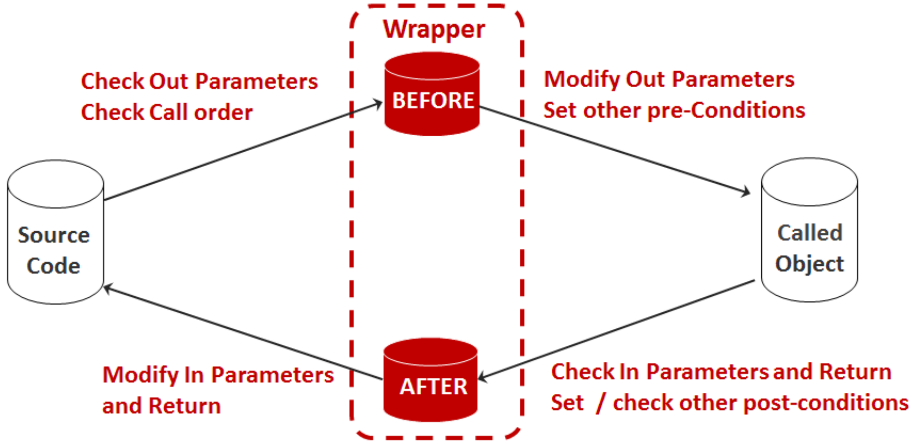
How can be addressed

- ✓ Configurable test framework
- ✓ Limiting test scope boundary, Tests run under debugger
- ✓ Unit testing + simulations
- ✓ Integration tests + interceptions
- ✓ Tools!!

Embedded Interface Dependencies

Conscious Un-coupling ?

Interface Interception ?



Abstract & Segregate

Abstract hardware from logic in implementation (e.g. layered software design for embedded testing)

Segregate by simulating SUT interfaces to hardware / software (e.g. stub, mock, fake, etc)

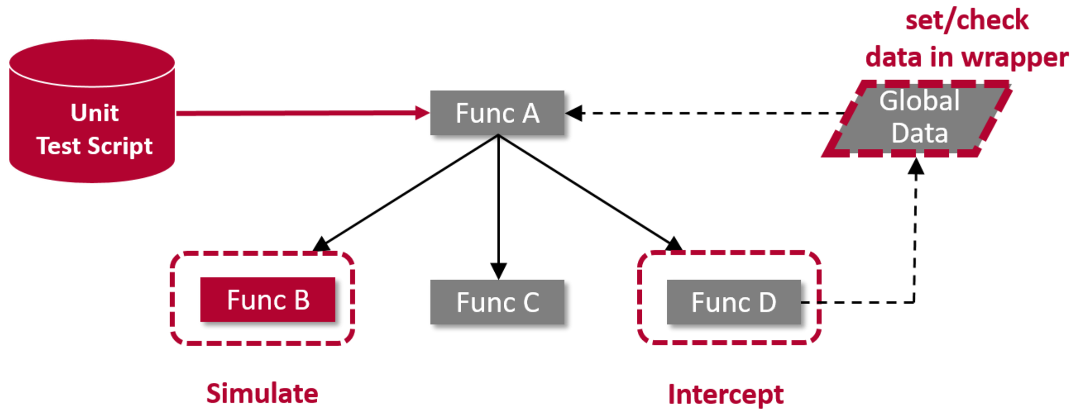
Test Integration Interfaces

Use real Hardware in Loop (HiL) testing for less 'pure' unit / integration / system tests.

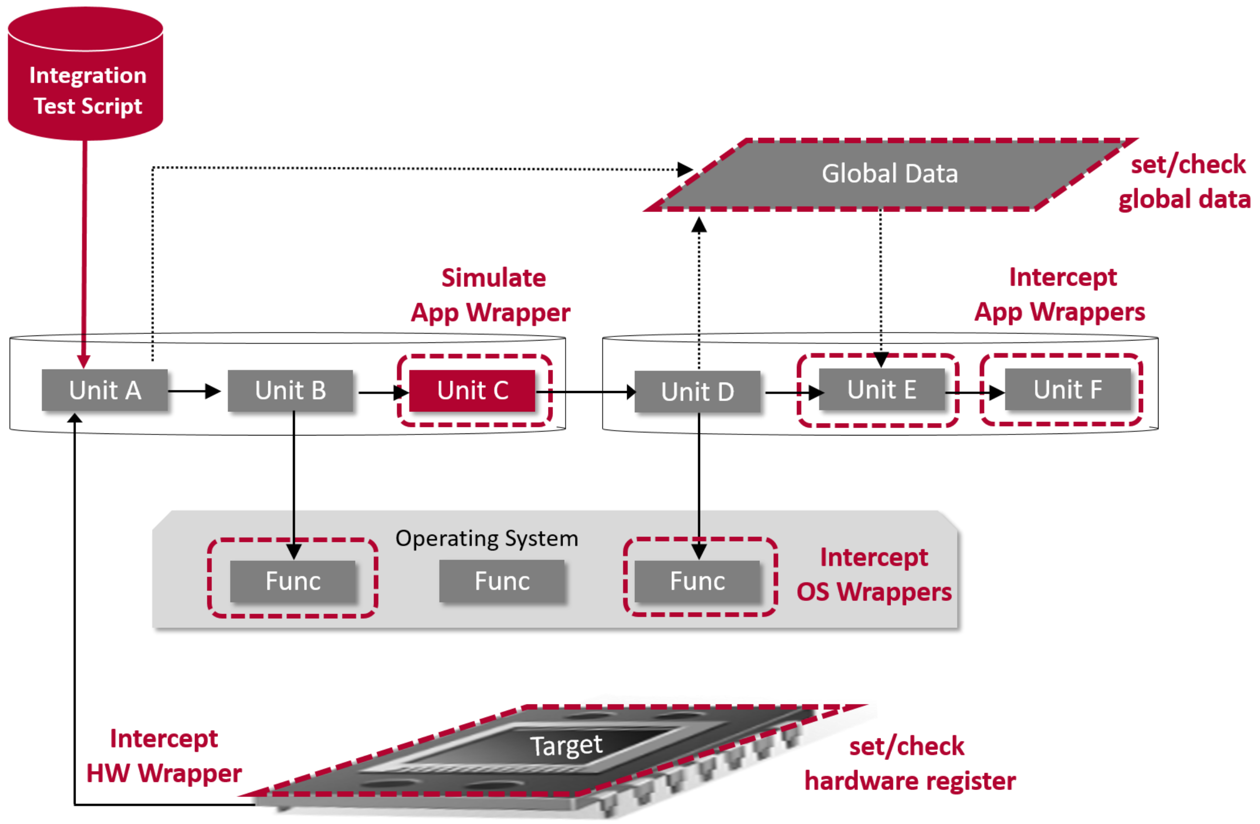
Use real code called by the SUT, but intercept calls to control them.

Interface Testing by Type of Test

✓ 'Isolation' Unit Test



✓ Integration Test



Example Approach for HiL Testing

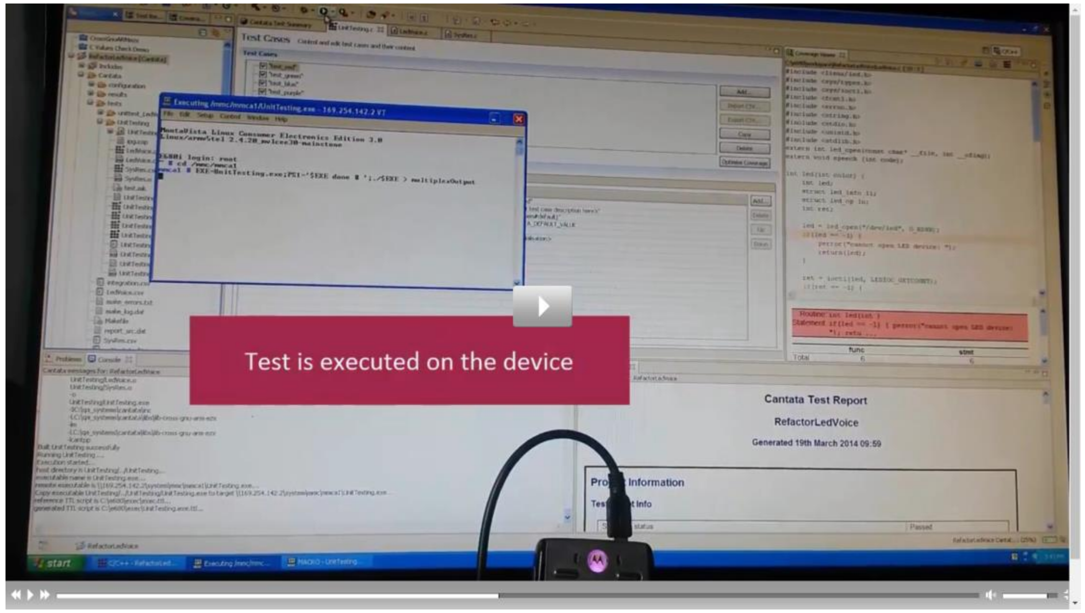
Unit Test HiL with Wrapping

Example controlling colours of an LED

Low-level calls to read / write operations on the LED port

Automatically intercepting return from LED to modify the call behavior at run time (HiL)

Injects faulty 'error conditions' back to controlling function to achieve desire code coverage

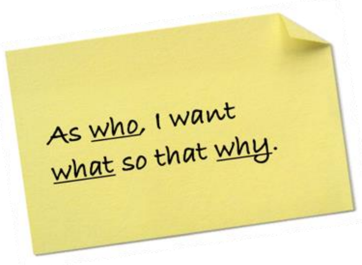

Test is executed on the device
Cantata Test Report
RefactorLedVoice
Generated 19th March 2014 09:59

Impact of Changing Requirements 1

Requirements Management Tools in Agile

Suitability

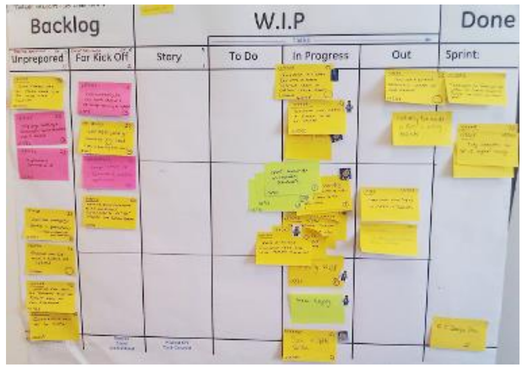
Whole team availability & use

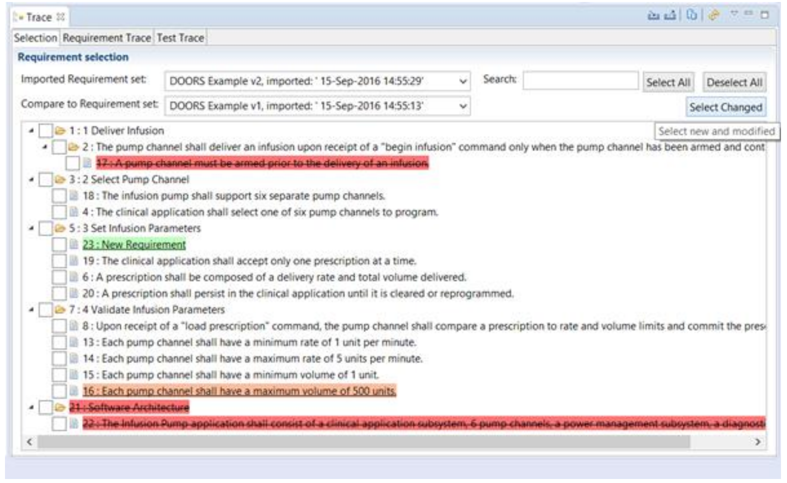

As who, I want what so that why.

Managing Changes

Requirements Definitions

Testing work allocation





Requirement selection

Imported Requirement set: DOORS Example v2, imported: ' 15-Sep-2016 14:55:29'

Search:

Compare to Requirement set: DOORS Example v1, imported: ' 15-Sep-2016 14:55:13'

- 1 : 1 Deliver Infusion
  - 2 : The pump channel shall deliver an infusion upon receipt of a "begin infusion" command only when the pump channel has been armed and cont
    - ~~17 : A pump channel must be armed prior to the delivery of an infusion.~~
- 3 : 2 Select Pump Channel
  - 18 : The infusion pump shall support six separate pump channels.
  - 4 : The clinical application shall select one of six pump channels to program.
- 5 : 3 Set Infusion Parameters
  - 23 : New Requirement
  - 19 : The clinical application shall accept only one prescription at a time.
  - 6 : A prescription shall be composed of a delivery rate and total volume delivered.
  - 20 : A prescription shall persist in the clinical application until it is cleared or reprogrammed.
- 7 : 4 Validate Infusion Parameters
  - 8 : Upon receipt of a "load prescription" command, the pump channel shall compare a prescription to rate and volume limits and commit the pres
  - 13 : Each pump channel shall have a minimum rate of 1 unit per minute.
  - 14 : Each pump channel shall have a maximum rate of 5 units per minute.
  - 15 : Each pump channel shall have a minimum volume of 1 unit.
  - 16 : Each pump channel shall have a maximum volume of 500 units.
- ~~21 : Software Architecture~~
  - ~~22 : The Infusion Pump application shall consist of a clinical application subsystem, 6 pump channels, a power management subsystem, a diagnost~~

Direct Integration

codeBeamer

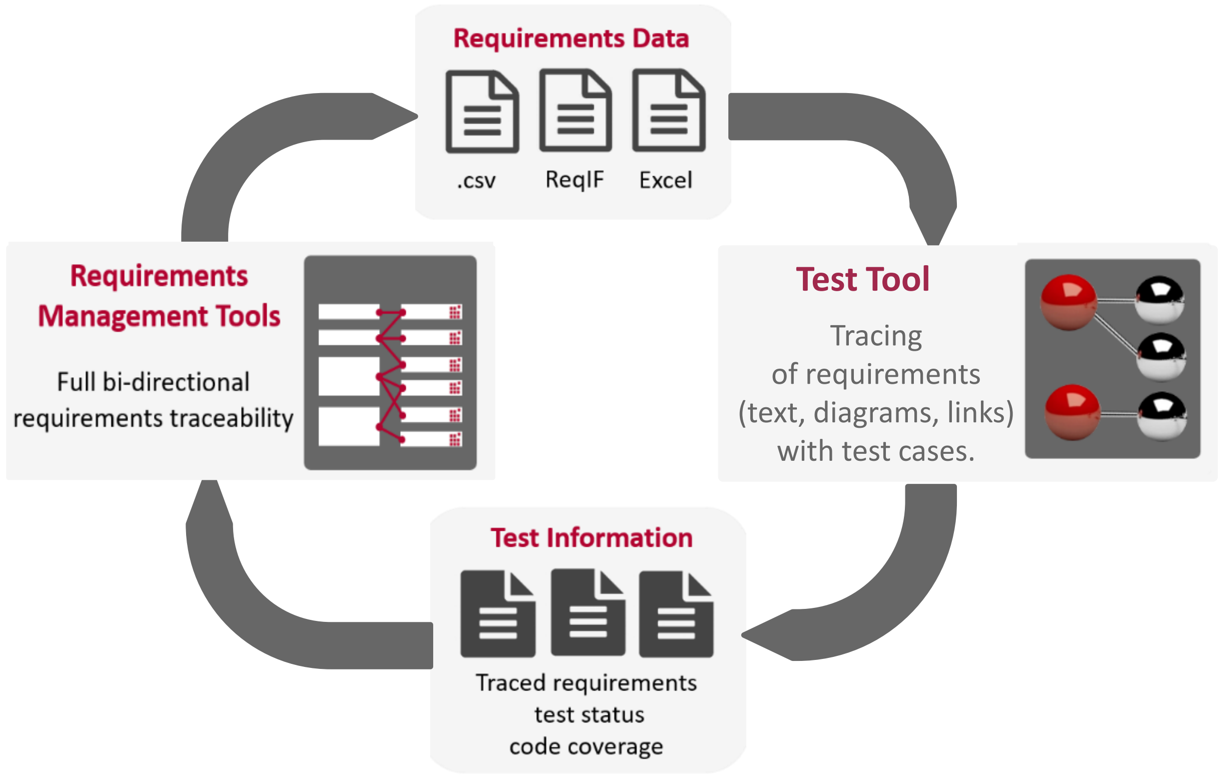
Impact of Changing Requirements 2

Traceability to Tests

Visibility during testing

What to trace and when

Control of traceability data



Requirements Data
.csv
ReqIF
Excel
Requirements Management Tools
Full bi-directional requirements traceability
Test Tool
Tracing of requirements (text, diagrams, links) with test cases.
Test Information
Traced requirements
test status
code coverage

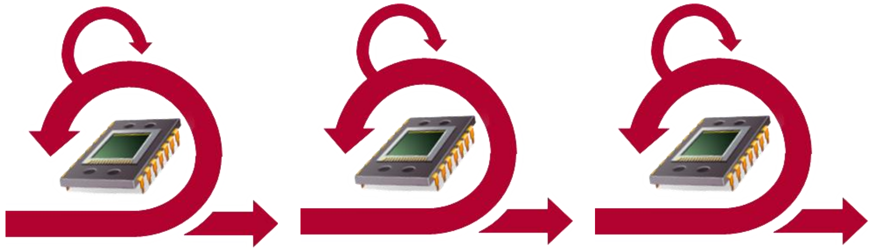
Regression Testing

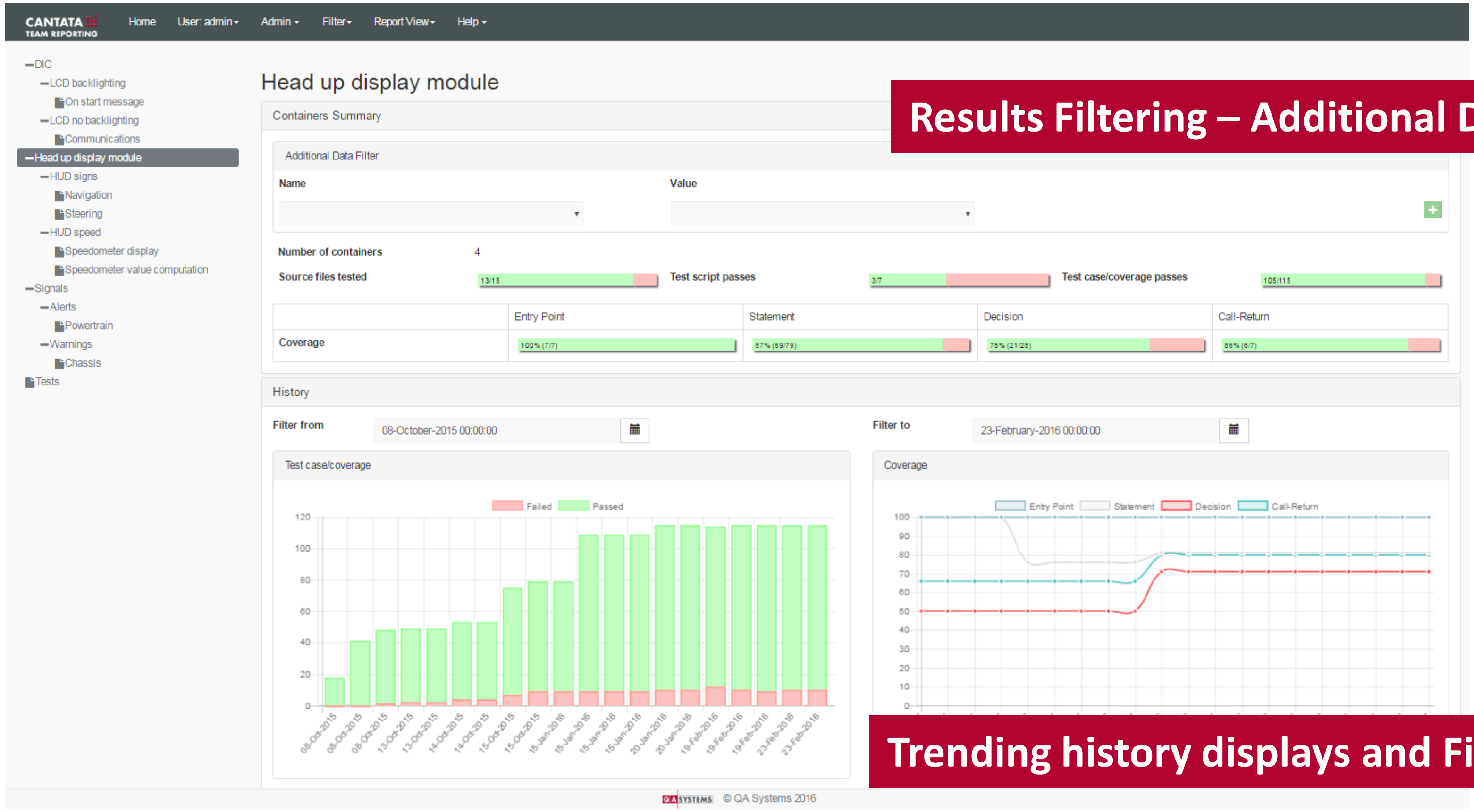
Automated test suites & Continuous Testing

Incremental / Full suite test runs

Progress & Differentiating Targets

- **Monitoring test progress**
- **Test execution on different variants (HW & SW)**





CANTATA TEAM REPORTING — Home | User: admin | Admin | Filter | Report View | Help

- DIC
  - LCD backlighting
    - On start message
  - LCD no backlighting
    - Communications
  - Head up display module
    - HUD signs
      - Navigation
      - Steering
    - HUD speed
      - Speedometer display
      - Speedometer value computation
- Signals
  - Alerts
    - Powertrain
  - Warnings
    - Chassis
- Tests

Head up display module

Containers Summary

Additional Data Filter

Name	Value

Number of containers	4				
Source files tested	13/15	Test script passes	3/7	Test case/coverage passes	105/115

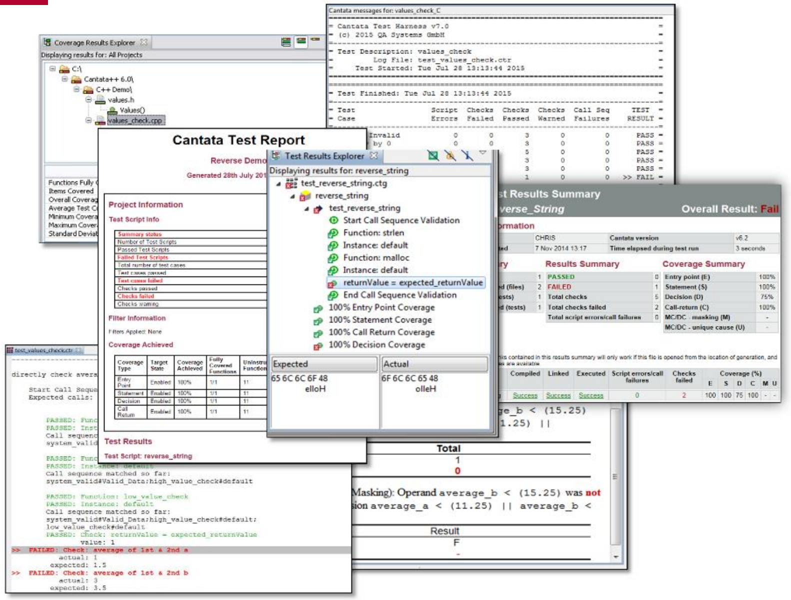
	Entry Point	Statement	Decision	Call-Return
Coverage	100% (7/7)	87% (69/79)	75% (21/28)	86% (6/7)

History

Filter from 08-October-2015 00:00:00 Filter to 23-February-2016 00:00:00

© QA Systems 2016

Results Filtering – Additional Data



Cantata messages for: values_check_C

```
= Cantata Test Harness v7.0                                          =
= (c) 2015 QA Systems GmbH                                           =
= Test Description: values_check                                     =
=          Log File: test_values_check.ctr                           =
=     Test Started: Tue Jul 28 13:13:44 2015                         =
= Test Finished: Tue Jul 28 13:13:44 2015                            =
= Test              Script  Checks  Checks  Checks  Call Seq    TEST =
= Case              Errors  Failed  Passed  Warned  Failures  RESULT =
  Invalid              0       0       3       0       0        PASS =
   by 0                0       0       3       0       0        PASS =
                                       5       0       0        PASS =
                                       3       0       0        PASS =
                                       3       0       0        PASS =
                                       1       0       0     >> FAIL =
```

Coverage Results Explorer — Displaying results for: All Projects
- C:\
  - Cantata++ 6.0\
    - C++ Demo\
      - values.h
        - Values()
      - values_check.cpp

Functions Fully / Items Covered / Overall Coverag / Average Test C / Minimum Covera / Maximum Cover / Standard Deviat

Cantata Test Report — Reverse Demo — Generated 28th July 201

Project Information

Test Script Info

Summary status / Number of Test Scripts / Passed Test Scripts / Failed Test Scripts / Total number of test cases / Test cases passed / Test cases failed / Checks passed / Checks failed / Checks warning

Filter Information

Filters Applied: None

Coverage Achieved

Coverage Type	Target State	Coverage Achieved	Fully Covered Functions	Uninstru Function
Entry Point	Enabled	100%	1/1	11
Statement	Enabled	100%	1/1	11
Decision	Enabled	100%	1/1	11
Call Return	Enabled	100%	1/1	11

Test Results

Test Script: reverse_string

Test Results Explorer — Displaying results for: reverse_string
- test_reverse_string.ctg
  - reverse_string
    - test_reverse_string
      - Start Call Sequence Validation
      - Function: strlen
      - Instance: default
      - Function: malloc
      - Instance: default
      - returnValue = expected_returnValue
      - End Call Sequence Validation
    - 100% Entry Point Coverage
    - 100% Statement Coverage
    - 100% Call Return Coverage
    - 100% Decision Coverage

Expected	Actual
65 6C 6C 6F 48 elloH	6F 6C 6C 65 48 olleH

st Results Summary — verse_String — Overall Result: Fail

rmation

	CHRIS	Cantata version	v6.2
ed	7 Nov 2014 13:17	Time elapsed during test run	3 seconds

ry		Results Summary		Coverage Summary	
	1	PASSED	0	Entry point (E)	100%
d (files)	2	FAILED	1	Statement (S)	100%
ests)	1	Total checks	5	Decision (D)	75%
d (tests)	1	Total checks failed	2	Call-return (C)	100%
		Total script errors/call failures	0	MC/DC - masking (M)	-
				MC/DC - unique cause (U)	-

ks contained in this results summary will only work if this file is opened from the location of generation, and s are available

Compiled	Linked	Executed	Script errors/call failures	Checks failed	E	S	D	C	M	U
Success	Success	Success	0	2	100	100	75	100	-	-

test_values_check.ctr

```
directly check avera
    Start Call Seque
    Expected calls:

        PASSED: Func
        PASSED: Inst
        Call sequenc
        system_valid

        PASSED: Func
        PASSED: Instance: default
        Call sequence matched so far:
        system_valid#Valid_Data:high_value_check#default

        PASSED: Function: low_value_check
        PASSED: Instance: default
        Call sequence matched so far:
        system_valid#Valid_Data:high_value_check#default;
        low_value_check#default
        PASSED: Check: returnValue = expected_returnValue
                value: 1
>>  FAILED: Check: average of 1st & 2nd a
            actual: 1
          expected: 1.5
>>  FAILED: Check: average of 1st & 2nd b
            actual: 3
          expected: 3.5
```

ge_b < (15.25)
1.25) ||

Total
1
0

Masking): Operand average_b < (15.25) was not
ion average_a < (11.25) || average_b <

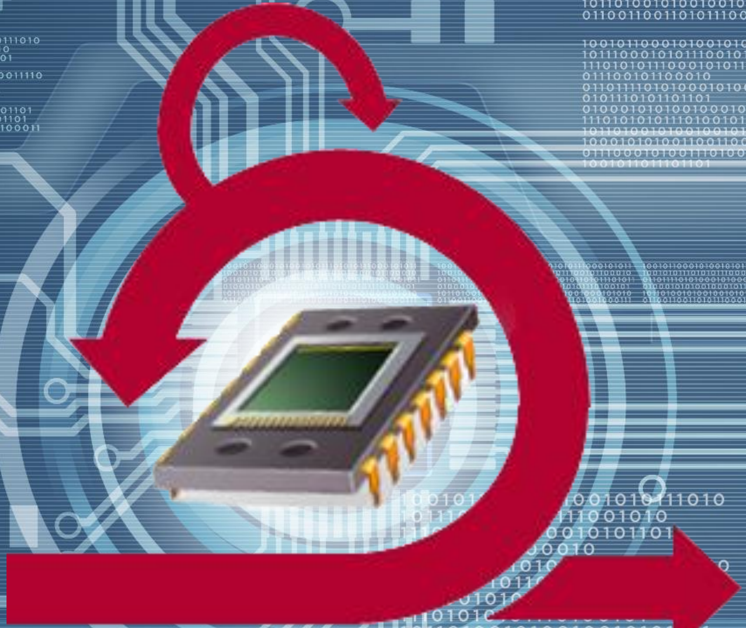
Result
F
-

Trending history displays and Filters

QA SYSTEMS The Software Quality Company

How Functional Safety Impacts Agile Testing

Agile for Embedded Conference 2017



MISRA C:2012
Guidelines for the use of the C language in critical systems
INTERNATIONAL STANDARD IEC 62304
INTERNATIONAL STANDARD ISO 26262-6
Road vehicles — Functional safety — Part 6: Product development at the software level
ISO
IEC
Space project management
Space product assurance
Space engineering
SOFTWARE DEVELOPMENT AND DOCUMENTATION
Die CENELEC-Normen zur Funktionalen Sicherheit
The CENELEC-Standards regarding Functional Safety
IEC 61508-2
INTERNATIONAL STANDARD
NORME INTERNATIONALE
DO-178B

Impacts of Functional Safety

Standards & Certifications

Industrial Specializations

Independent Certification Authorities

Standards dictate where code should be tested & by whom

As close to the running system configuration as possible

Variation by Safety Integrity Level

Role of Independent Verification & Validation

Standards dictate use of suitable test tools

The need for tool qualification / certification

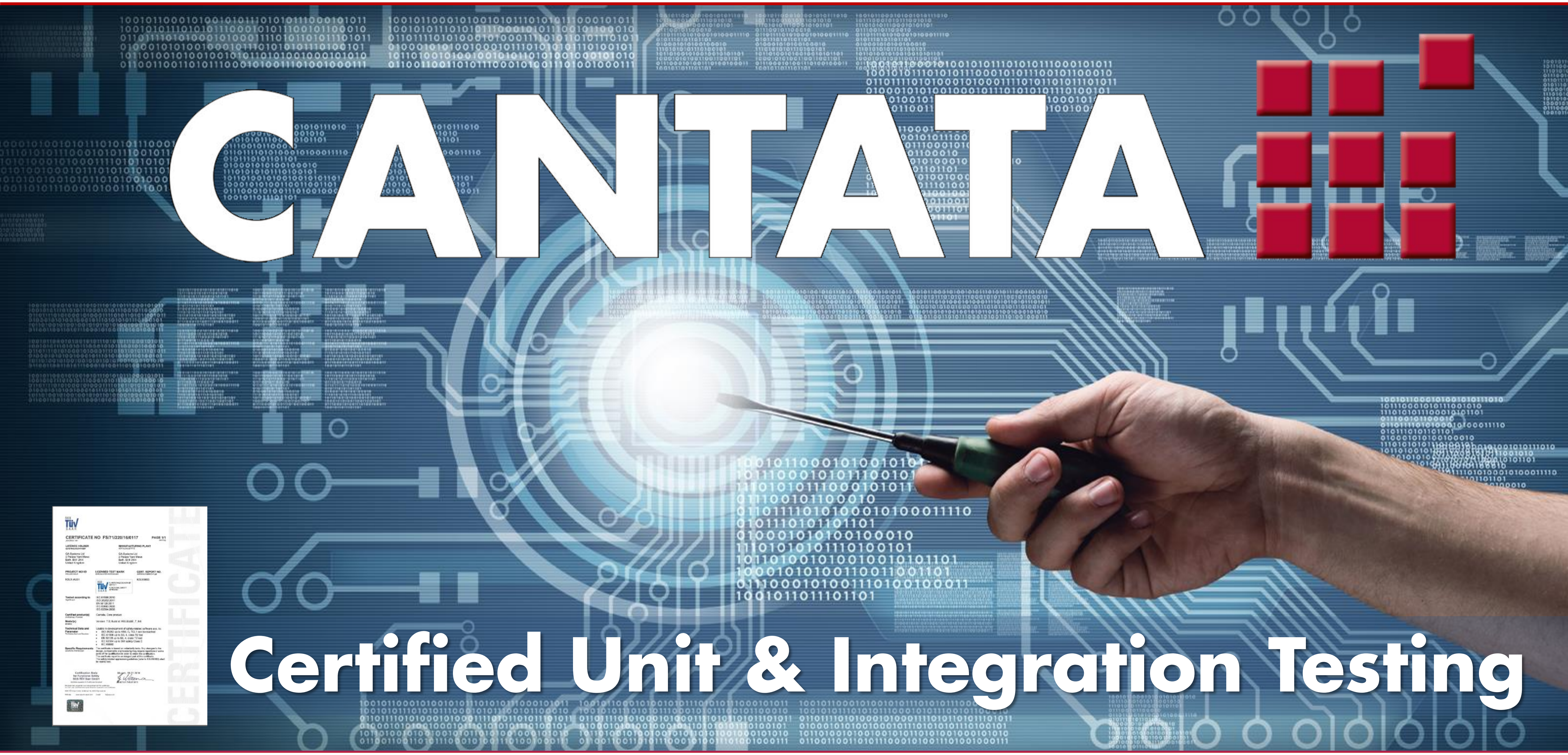
Tool configuration on embedded target test platforms

UP NEXT…

Industrial experience with Agile in high-integrity software development

Altran UK

Automated Agile Embedded Testing



Thank you

Questions?