Agile Testing Techniques for Embedded Development

Challenges for Agile Embedded Testing
Challenge: Embedded Target Differences

Why do differences matter for agile testing?

遵守 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

遵守 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

Test Framework

SUT

Tests

Target Independent

Target Specific

Build Options

Build

Test Exe

Run

Test Results
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.
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?

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)

Interface Interception?

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

- Unit Test Script
- Func A
- Func B
- Func C
- Func D
- Global Data

Simulate
Intercept
set/check data in wrapper

✓ Integration Test

- Integration Test Script
- Unit A
- Unit B
- Unit C
- Unit D
- Unit E
- Unit F
- Global Data
- Simulate App Wrapper
- Intercept App Wrappers
- Intercept OS Wrappers
- Intercept HW Wrapper
- Target
- Operating System

set/check global data
set/check hardware register
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
Impact of Changing Requirements 1

Requirements Management Tools in Agile

Suitability
Whole team availability & use

Managing Changes

Requirements Definitions
Testing work allocation

© All Copyright and Trademarks of their respective owners are acknowledged
Impact of Changing Requirements 2

**Traceability to Tests**
Visibility during testing
What to trace and when
Control of traceability data

**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)

Results Filtering – Additional Data

Trending history displays and Filters
How Functional Safety Impacts Agile Testing
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
Automated Agile Embedded Testing

Certified Unit & Integration Testing
Thank you

Questions?