FEABHAS

TDD for Embedded C

Course category	Agile for Embedded
Training area	Agile for Embedded
Course code	TDDC-201
Duration	2 days
Course date	25th November 2020
Price exc VAT	£1275.00

Agile for Embedded Training Course

One of the core Agile Practices is Test-Driven Development (TDD). For most software engineers TDD challenges the traditional approach to software development and testing (test-after-construction). TDD changes this model by using a Test-First approach.

However, almost all current books and training on TDD ignore the challenges of applying it in an embedded systems environment. Aspects such as testing on the target, hardware dependency and managing both host and cross-compiler toolchains are generally ignored or over-simplified. In addition almost all are based on an object-oriented language, such as Java or C++ rather than C.

The course covers testing embedded C software across host, emulated and target environments.

Attendees perform practical TDD exercises using both host (Linux / gcc) and cross-development (Cortex-M / arm-none-eabi-gcc) toolchains supported by a modern build system (SCons) and configuration management (Git) tools.

The course utilises the C based test framework Unity and CMock for test doubles.

Course objectives:

- To understand the principles of Test-Driven Development
- How to select a suitable test framework for C development
- To identify the required toolset to support TDD beyond the test framework
- The boundary between host-based testing and target-based testing
- The challenges and pitfalls of applying TDD to an embedded system

Delegates will learn:

- How to practice Test-Driven Development
- The use of C based test harnesses for embedded systems
- API and Integration testing using test doubles (CMock)
- The importance of architectural design to practical TDD
- What TDD doesn't address

Pre-requisites:

- Attendees should have a good working knowledge of the C language.
- Understand the build process (preprocessor-compiler-linker-executable)

Who should attend:

The course is designed for software engineers working on an embedded C project.

The target audience is engineers working in, or looking to move to, an Agile project environment (e.g. Scrum) and who want practical experience of using TDD for a real embedded system.

Duration:

Two days

Course materials:

- Delegate manual
- Delegate workbook
- Delegate bootable Linux datakey with all tools installed

Course workshop:

Attendees perform a mixture of hands-on programming for both host and embedded targets.

The initial exercises show how to test embedded code on a host system (Linux), then move to emulator based testing (QEMU) and the final exercises include integration testing so as to apply TDD to a target hardware.

Approximately 50% of the course is given over to practical work.

The host code is developed on Linux using GCC and Scons supported by Git versioning. The embedded code is targeted at an ARM Cortex-M based MCU using arm-none-eabi-gcc. This combination gives attendees a real sense of the challenges of applying TDD to embedded application development.

Agile Techniques and TDD

- Goals
- Where does TDD fit in?
- Agile onion
- Embedded TDD Strategy

TDD Foundations

- Unit tests
- Red-Green-Refactor (RGR)
- 3 rules of TDD
- Failing tests
- Mindset
- Mechanics

Test Construction

- Tests are FIRST
- File organisation
- Fixtures
- Assertions
- Failure based tests
- Breaking encapsulation (Inspecting privates)
- Running a subset of tests
- Testing vs. Test Driving
- Arrange-Act-Assert(AAA)/Given-When-Then
- Weaknesses

Design Principles

- Abstraction and encapsulation
- What can we learn from OO's SOLID Principles

Unit isolation

- Simple test doubles
- Dependency Challenges

Testing APIs

- UML Sequence Diagrams

Advanced Test Doubles

- Mocks
- Getting Mocks in place
- Design and change
- Strategies
- Issues

Incremental Design

- Walking skeleton
- Importance of architecture
- Version control and branching (git-workflow)

Target TDD

- Building for the target
- Testing using emulation (QEMU)
- Testing on the target

Acceptance TDD

- Containers and Docker
- Continuous Integration (CI) (Jenkins / Bamboo)
- Cloud-based CI builds (Codeship, Travis-CI, Bitbucket)

Quality Tests

- non-functional tests
- test anti-patterns
- fuzzing

Legacy Code

- Legacy Code Change Policy

TDD and Threading

- GPOS threading vs. RTOS tasking
- Separate threading logic from application logic