Stand 03.07.2024

Embedded Software Test: Best Practices for Unit/ Module /Component Testing - Live Online Training

**Ziele -
Ihr Nutzen**

The embedded software test training highlights the development and test process as well as the related dependencies, extensions and interactions. This way, you can exploit synergies and perform tests efficiently.

Numerous hands-on exercises on software and hardware help you implement what you learned in the training.

Teilnehmer

Software developers, hardware developers, test engineers

Voraussetzungen

Basic knowledge of a higher-level programming language (e.g. C/C++) is of advantage.

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: LE-EMBTEST

Präsenz-Training - Englisch

Dauer
4 Tage

Live-Online - Deutsch

Termin	Dauer
08.07. – 11.07.2024	4 Tage
27.01. – 30.01.2025	4 Tage

Präsenz-Training - Deutsch

Termin	Dauer
14.10. – 17.10.2024	4 Tage

Embedded Software Test: Best Practices for Unit/ Module /Component Testing - Live Online Training

Inhalt

Development and Test Process in the W-Model (V-Model Extension)

- Development stages: analysis, design, implementation
- Test stages: component test, integration test, system test, acceptance test
- Test types: functional, non-functional, structure-oriented test
- Post test, regression test, maintenance test
- Standards relevant to testing

- Developing testable software
- Definition of terms: unit, module, component
- Difference between debugging and testing

Static Tests

- Review process: document review, coder review, inspection, walkthrough
- Tool-aided static code analysis

Dynamic Tests

- Blackbox: equivalence class construction, threshold testing, decision table testing, state transition testing, use case testing
- Whitebox: statement testing/ coverage, decision testing/ coverage, condition testing/ coverage
- Experience-based techniques: error guessing, explorative testing
- Systematic approaches for developing test cases
- Selection criteria for test methods
- Test method assessment

Code Metrics

- Lines of code, cyclomatic number according to McCabe, Halstead metric
- Using metrics in the test process

Design for Test

- S.O.L.I.D. principles
- Single responsibility, open closed
- Liskov substitution
- Interface segregation, dependency inversion

Testing Object-Oriented Software

- Testing class hierarchies
- Testing methods of a class
- Testing class relations
- Testing inheritance
- Testing polymorphic class hierarchies

Test-Driven Development, TDD

- Advantages of TDD
- Embedded TDD strategies
- TDD example

Integration of System Components - Overview

- Integration test, system test, acceptance test

Integration of Hardware and Software

- Test execution on hardware

Further Activities in the Test Process - Overview

- Test management, planning, control
- Risk management, fault and incident management
- Configuration management and version control
- Software quality features according to ISO 9126
- Test documents according to IEEE 829
- Test tools types, selection, implementation

Hands-on Exercises

- Exercises on test analysis, test design, test implementation, test execution, test report
- Execution of code Review
- Execution of blackbox and whitebox tests with Tessy, with and without HW
- Test data determination using the classification tree method with CTE
- Determining code metrics using the tools cccc and CMT++
- Using Google Test and Google Mock
- The Arm/ Keil µVision and a Cortex™-M evaluation board are used for tests on HW