Stand 16.05.2025

Embedded-Software-Test: Best Practices für den Unit-/Modul-/Komponenten-Test - Live-Online-Training

**Ziele -
Ihr Nutzen**

Lernen Sie den Entwicklungs- und Testprozess im Zusammenhang mit all seinen Abhängigkeiten, Ergänzungen und Wechselbeziehungen kennen, um durch die Nutzung von Synergien effizient und effektiv zu testen.

In zahlreichen praktischen Übungen mit Software und Hardware wird das Gelernte umgesetzt.

Teilnehmer

Software-Entwickler, Hardware-Entwickler, Testingenieure

Voraussetzungen

Grundkenntnisse einer höheren Programmiersprache (z.B. C/C++) sind von Vorteil.

Live Online Training

21.07. – 24.07.2025	2.800,00 €	4 Tage
12.01. – 15.01.2026	2.800,00 €	4 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-EMBTEST

Präsenz-Training - Deutsch

Termin	Dauer
20.10. – 23.10.2025	4 Tage
23.03. – 26.03.2026	4 Tage

Live-Online - Englisch

Dauer
4 Tage

Präsenz-Training - Englisch

Dauer
4 Tage

Embedded-Software-Test: Best Practices für den Unit-/Modul-/Komponenten-Test - Live-Online-Training

Inhalt

Der Entwicklungs- und Testprozess im W-Modell (erweitertes V-Modell)

- Entwicklungsstufen: Analyse, Design, Implementierung
- Teststufen: Komponententest, Integrationstest, Systemtest, Abnahmetest

- Testarten: funktionaler, nichtfunktionaler, strukturorientierter Test
- Fehlernachtest, Regressionstest, Wartungstest
- Testrelevante Standards
- Entwicklung von testbarer Software
- Begriffsklärung Unit, Modul, Komponente
- Unterschied zwischen Debuggen und Testen

Statische Tests

- Review-Prozess: Dokumentenreview, Codereview, Inspektion, Walkthrough
- Werkzeuggestützte statische Code-Analyse

Dynamische Tests

- Blackbox-Verfahren: Äquivalenzklassenbildung, Grenzwertanalyse, Entscheidungstabellentest, zustandsbasierter Test, anwendungsfallbasierter Test
- Whitebox-Verfahren: Anweisungstest/-überdeckung, Entscheidungstest/-überdeckung, Bedingungstest/-überdeckung
- Erfahrungsbasierte Verfahren: Error Guessing, exploratives Testen
- Systematische Vorgehensweise bei der Entwicklung von Testfällen
- Kriterien zur Auswahl von Testverfahren
- Bewertung der Testverfahren

Codemetriken

- Lines of Code, zyklomatische Zahl nach McCabe, Halstead-Metrik
- Anwendung der Metriken im Testprozess

Design for Test

- S.O.L.I.D. Prinzipien
- Single Responsibility, Open Closed
- Liskov Substitution
- Interface Segregation, Dependency Inversion

Testen von objektorientierter Software

- Testen von Klassenhierarchien
- Testen von Methoden einer Klasse
- Testen von Klassenbeziehungen
- Testen einer strikten und nicht-strikten Vererbung
- Testen von polymorphen Klassenhierarchien

Test Driven Development, TDD

- Vorteile von TDD
- Embedded TDD Strategien
- TDD Beispiel

Integration von Komponenten zu Systemen im Überblick

- Integrationstest, Systemtest, Abnahmetest

Integration von Hardware und Software

- Testdurchführung auf der Hardware

Weitere Aktivitäten im Testprozess im Überblick

- Testmanagement, Planung, Steuerung
- Risikomanagement, Fehler- und Abweichungsmanagement
- Konfigurationsmanagement und Versionskontrolle
- Softwarequalitätsmerkmale nach ISO 9126
- Testdokumente nach IEEE 829
- Testwerkzeuge Typen, Auswahl, Einführung

Praktische Übungen

- Übungen zu Testanalyse, Testentwurf, Testrealisierung, Testdurchführung, Testbericht
- Durchführung eines Code-Reviews
- Durchführung von Blackbox- und Whitebox-Tests mit Tessy, mit und ohne Hardware
- Ermittlung von Testdaten nach der Classification Tree Methode mit CTE
- Bestimmen von Codemetriken mit den Tools cccc und CMT++
- Google Test und Google Mock im Einsatz
- Für die Tests auf der Hardware werden die Arm/ Keil µVision und ein Cortex™-M Evaluierungsboard eingesetzt.