Test-Driven Development in der Praxis

Software durch kurze Test-Code-Refactoring-Zyklen entwickeln

01 INTRO

Test-Driven Development nutzt automatisierte Tests nicht erst zur nachträglichen Prüfung, sondern als Werkzeug während des Softwareentwurfs. Der kurze Wechsel zwischen Test, Implementierung und Refactoring zwingt dazu, Verhalten explizit zu formulieren und Designentscheidungen kontinuierlich zu überprüfen.

Der Kurs vermittelt Test-Driven Development praktisch anhand wachsender Implementierungsaufgaben. Dabei wird es weder als Garantie für gutes Design noch als Dogma behandelt, sondern als Engineering-Technik mit konkreten Stärken, Grenzen und Voraussetzungen.

DAUER

2 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Sichere Programmiergrundlagen und Grundkenntnisse von Unit Tests

02 TRAININGSPROGRAMM

Inhalte

Test-Driven Development einordnen

- Test First
- Red-Green-Refactor
- Feedback
- Design
- Regression
- Vergleich mit nachträglichen Tests
- Grenzen

Der Entwicklungszyklus

- kleinen Test wählen
- Red
- minimale Implementierung
- Green
- Refactor
- Wiederholung
- kleine Schritte

Tests formulieren

- beobachtbares Verhalten
- Arrange-Act-Assert
- aussagekräftige Namen
- eine Aussage
- Testfälle auswählen
- Grenzfälle
- Lesbarkeit

Vom Beispiel zur Implementierung

- einfachster Fall
- Triangulation
- Generalisierung
- Fake It
- Obvious Implementation
- nächste Anforderung
- Schrittgröße

Refactoring im Zyklus

- Duplikate
- Namen
- Methoden
- Klassen
- Verantwortlichkeiten
- Verhalten erhalten
- grünen Zustand halten

Test Doubles

- Stub
- Mock
- Fake
- Spy
- Isolation
- externe Abhängigkeiten
- sinnvoller Einsatz
- Overmocking

Tests und Design

- Kohäsion
- Kopplung
- Schnittstellen
- Dependency Injection
- emergentes Design
- Testbarkeit
- Design Feedback

Schwierige Abhängigkeiten

- Datenbanken
- Dateisystem
- Zeit
- Zufall
- Netzwerk
- externe Services
- Seams
- Architekturgrenzen

Testgetriebene Arbeit an bestehendem Code

- Legacy Code
- Characterization
- neue Funktionalität
- bestehende Strukturen
- Refactoring
- Risiken
- schrittweise Einführung

Typische Anti-Patterns

- fragile Tests
- Implementierungsdetails testen
- zu viele Mocks
- riesige Testfälle
- zu große Schritte
- testinduzierte Fehlarchitektur
- Dogmatismus

Zusammenarbeit im Team

- Pair Programming
- Code Reviews
- Continuous Integration
- Testkonventionen
- gemeinsame Qualität
- Testwartung
- Definition of Done

Kata und Praxisprojekt

- Anforderungen schrittweise entwickeln
- Tests zuerst schreiben
- Implementierung wachsen lassen
- refactoren
- Design diskutieren
- Alternativen vergleichen
- Retrospektive

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	2 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit Entwicklungsumgebung und Test-Framework

> **Inhouse-Schulungen**
> Die Übungen können in der Sprache und mit dem Test-Framework durchgeführt werden, die im Team eingesetzt werden.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Softwareentwickler, die Test-Driven Development praktisch erlernen oder ihren bisherigen Umgang damit systematisieren möchten.

> **Voraussetzungen**
> Sichere Programmiergrundlagen und Kenntnisse von Unit Tests.

05 LERNZIELE

Was Sie nach dem Kurs können

Die Teilnehmer können den Red-Green-Refactor-Zyklus sicher anwenden, Anforderungen in kleine Entwicklungsschritte zerlegen, Tests als Designfeedback nutzen und die Grenzen testgetriebener Entwicklung realistisch beurteilen.

06 FAQ

Häufige Fragen

Wird wirklich jeder Test zuerst geschrieben?

Im Entwicklungszyklus ja. Das bedeutet nicht, dass jede Form von Softwareentwicklung ausschließlich testgetrieben durchgeführt werden muss.

Entsteht dadurch automatisch eine gute Architektur?

Nein. Tests liefern Feedback und fördern bestimmte Designqualitäten, ersetzen aber keine Entwurfsentscheidungen.

Worin unterscheidet sich der Kurs vom Refactoring-Kurs?

Test-Driven Development ist eine Entwicklungstechnik. Refactoring ist eine eigenständige Technik zur strukturellen Verbesserung unter Verhaltenserhaltung und bildet nur einen Teil des Zyklus.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/test-driven-development/