

STRICT TDD

DIE UNTERSCHÄTZTE WAFFE DES ENTWICKLERS

David Völkel – Stuttgarter Testtage 2013

ÜBER MICH...



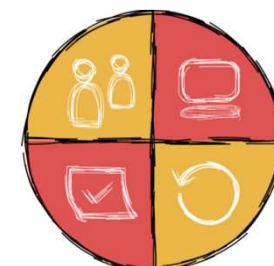
David Völkel

IT-Consultant für codecentric

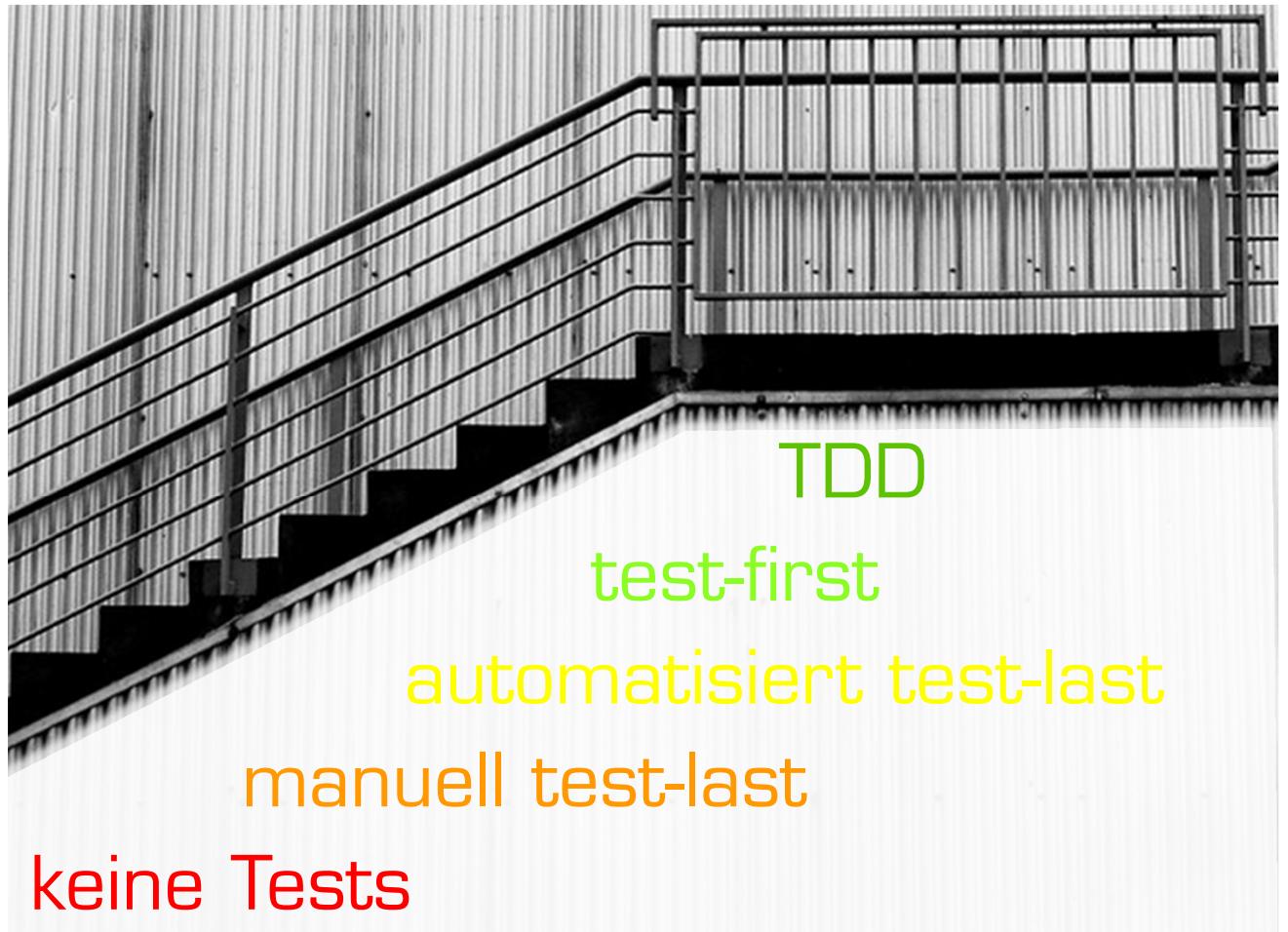
Twitter: @davidvoelkel

Schwerpunkte:

- Test Driven Development
- Softwaredesign
- Refactoring von Legacy-Systemen
- Software Craftsman @softwerkskammer



QUALITÄTSSTUFEN



KEINE TESTS

⇒Keine Qualität

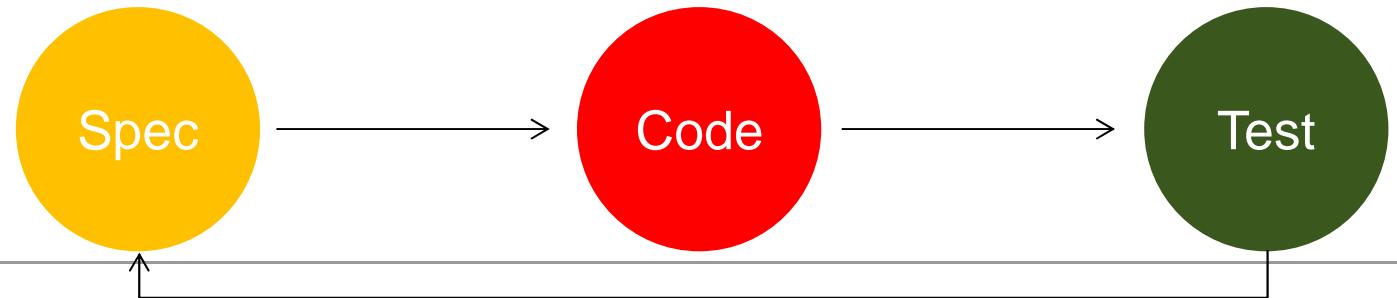
Passend für Experimente zur Klärung kritischer Fragen

- Machbarkeit
 - Prototyp
 - Technischer Durchstich
- Sinnhaftigkeit
 - (Lean) Startup
 - Hat Produkt Nutzen für Kunden?



Bundesarchiv, Bild 194-0665-20
Foto: Lachmann, Hans I 1953 c3.

TEST-LAST



- Fokus auf **Validierung**
- Langsames Feedback bzgl.
 - Fehlern in Anforderungen
 - Programmierfehlern



TEST-LAST MANUELL

- Probleme
 - Skaliert nicht bei Verkürzung der Releasezyklen
 - Entweder leidet Qualität oder Kosten



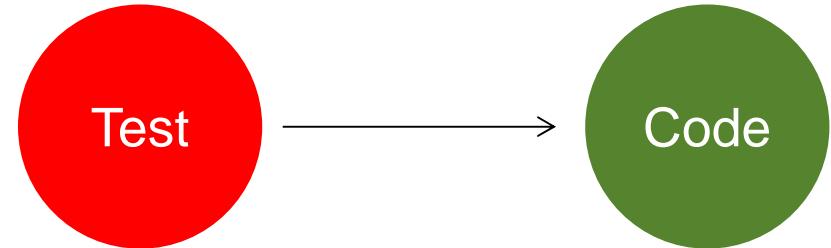
TEST-LAST AUTOMATISIERT

- Kurze Releasezyklen möglich
- Feedback langsam
- Problem Test Disziplin beim Entwickler



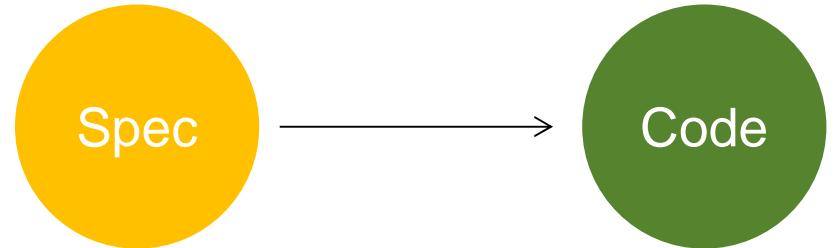
TEST-FIRST PROGRAMMING

- Ursprung eXtreme Programming (XP)
- Prozess
 - Test schreiben, der fehlschlägt
 - gerade so viel implementieren, bis Test erfolgreich durchläuft
- Nutzen
 - Testdisziplin
 - Anforderungen
 - Hinterfragen
 - Fokussierung
 - "you ain't gonna need it"
(YAGNI) eliminieren



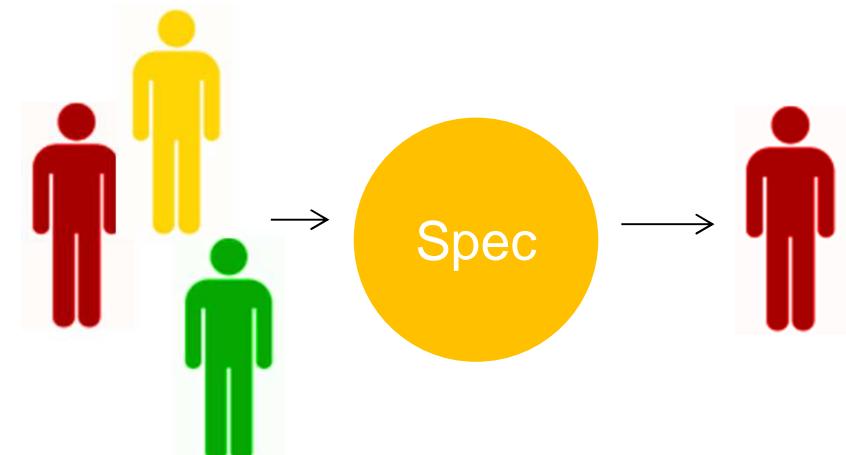
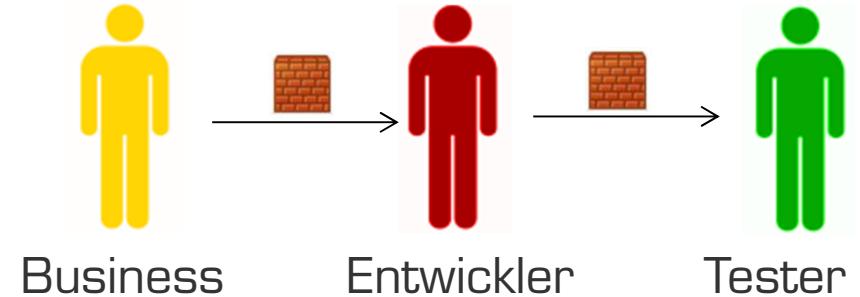
BDD

- Behaviour Driven Development
 - "TDD done right"
 - test-first-Charakter
 - "spec" statt "test"
- Fokus auf Anforderungen
 - Finden
 - "specification workshop"
 - Dokumentieren
 - "living documentation"



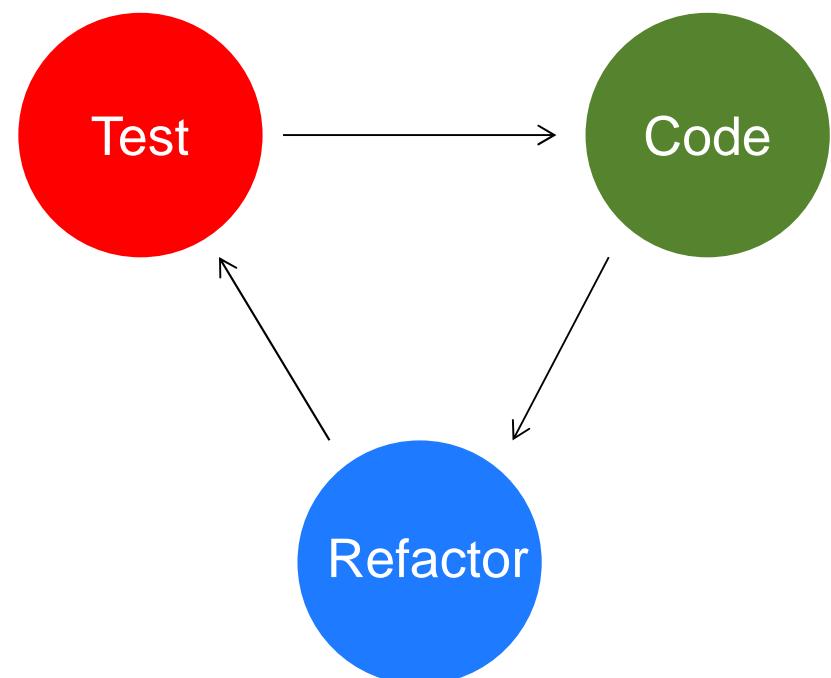
BDD

- test-last
 - Langes Feedback
- "specification workshop"
 - "three amigos"
 - Business, Tester, Developer
 - 3 Sichten
 - ⇒ "specifications by example"
 - ⇒ Schnelles Feedback durch Diskussion



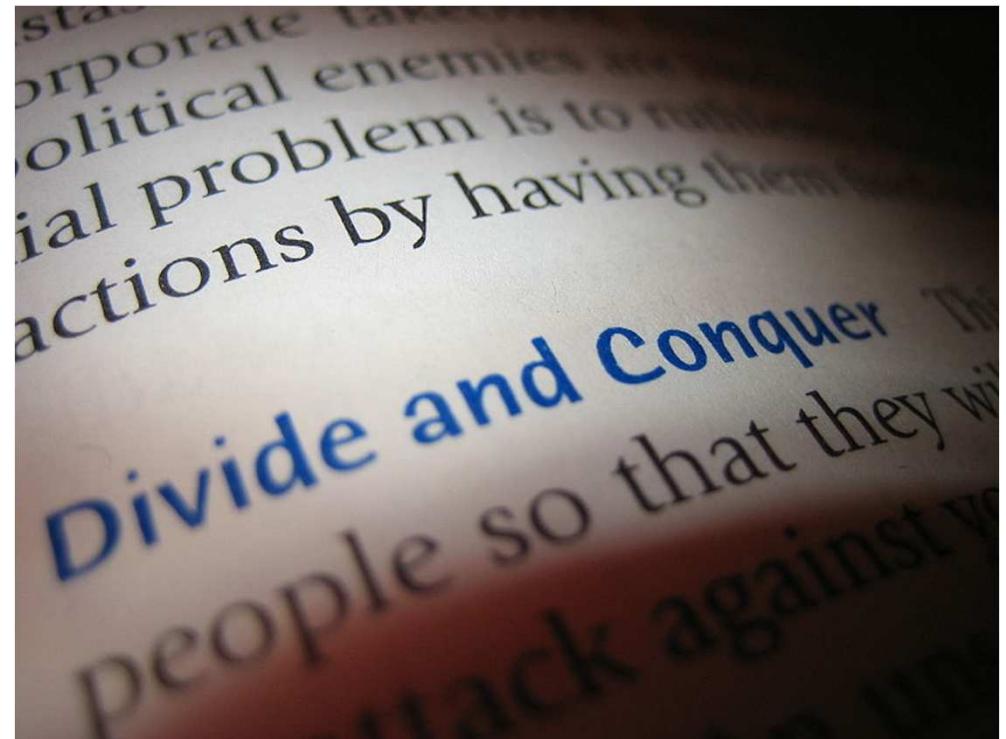
TEST DRIVEN DEVELOPMENT

- TDD = test-first + refactoring = "Test Driven Design"
- Kurze Zyklen
 - im Minutenbereich
 - Nur ein Test auf einmal
 - Schnelles Feedback
 - auf Requirements (test-first)
 - auf Design (refactoring)
 - Validierung (Test)
- Fokus
 - nur eine Aktivität gleichzeitig
 - kleine Aufgaben ("baby steps")



FOKUS

- Trennung in 3 Phasen
 - Test Schreiben
 - ⇒ was soll mein Code können?
 - Implementierung
 - ⇒ wie löse ich das Problem?
 - Refactoring
 - ⇒ gutes Design?
- Baby Steps
 - Fokus auf kleinste Inkrements, da Komplexität exponentiell wächst
 - Übung "taking baby steps"
 - von Adrian Bolboaca



SCHRITT 0: AUSWAHL DES NÄCHSTEN TESTS



- Schwierigste Frage
 - Welcher Test treibt die Implementierung in richtige Richtung?
- Strategien
 - von einfach zu komplex
 - von bekannt zu unbekannt
 - "Transformation Priority Premise"
 - Von Uncle Bob

FEHLSCHLAGENDER TEST

- Warum muss Test fehlschlagen?
 - Sicherheit durch Verhinderung von "false positives"
- Starte mit Assertion, nicht Setup
- Diagnostik
 - Ziel:
 - Aussagekräftige Fehlermeldungen
 - vor allem für zukünftige Entwickler
 - Mittel
 - Assertions mit description
 - Hamcrest-Matcher



IMPLEMENTIERUNG

- simplicity
 - simplest thing that could possibly work
 - => automatisch hohe Testcoverage
- Strategien
 - fake it & triangulation
 - baby steps
 - Hilfe bei Testfindung
 - obvious implementation
 - Bei trivialer Implementierung



REFACTORING



Kent Beck's "4 rules of simple Design"

- Alle Tests laufen
- Gute Namen, verständlicher Code
- Keine Codeduplikation
- Minimale Klassen und Methoden

TEST DRIVEN DESIGN



- Test erster Client des Codes
 - Emergent
 - Entsteht kontinuierlich direkt nach Implementierung durch Refactoring
 - Gute Designskills nötig!!!
- ⇒Gutes Design
- ⇒minimal, brauchbar, kein BDUF,
kein overengineering
- ⇒SRP, testbar

MOTIVATION DURCH TDD



- Dramaturgie
 - test-last demotivierend grün->rot
 - test-first motivierend rot->grün
- Sicherheit
 - weniger Angst vor Fehlern
- Fokus
 - Genau ein Ziel pro Phase
 - Klare DoD
 - Safe Baby Steps, weniger Debugging, weniger Fehlersuchen

GRENZEN VON TDD

- Exploratives Testen als Ergänzung
- Fehlende Know-how-Tiefe
 - TDD
 - Hard-to-test Produktionscode
 - Legacy
 - 3rd-party code / frameworks
 - DB / GUI
 - Design Testcode
 - Wartungsprobleme
 - Zu viel pro einzelnen Test
 - Wahl der falschen Teststart
 - Abhängigkeiten zwischen Tests



⇒Abwehrhaltung im Team

FAZIT

QUALITÄTSSTUFEN



TDD
test-first
automatisiert test-last
manuell test-last
keine Tests

NUTZEN

Design, Fokus, Feedback
Spezifikation
Häufige Releases
Valides Verhalten
Experimente

FRAGEN UND DISKUSSION

David Völkel

codecentric AG
Landsberger Str. 302
80687 München

Twitter: @davidvoelkel
david.voelkel@codecentric.de

www.codecentric.de
blog.codecentric.de
www.meettheexperts.de



LIZENZ

- Attribution-ShareAlike 3.0 Germany
- <http://creativecommons.org/licenses/by-sa/3.0/de/>

