Test-First Programming

Agile vs. klassische Methoden der Software-Entwicklung

Tobias Reinhard

13. Januar 2004

Probleme mit dem Testen

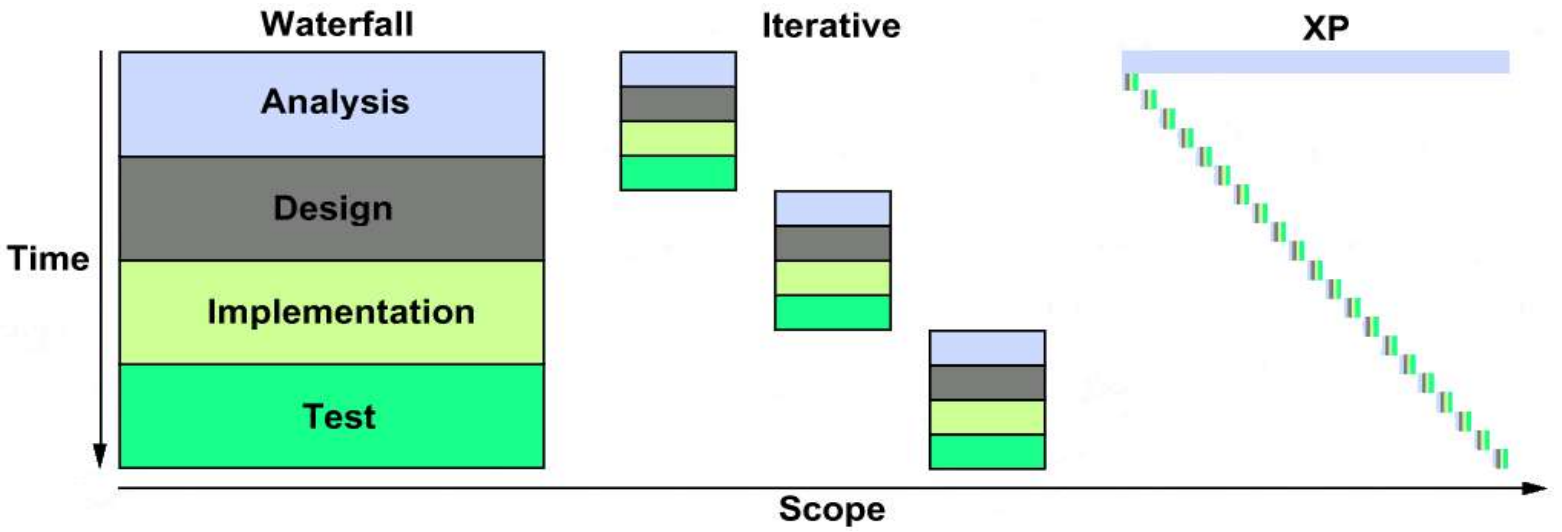
- Letzte Phase der klassischen Prozesse
  - Zuerst über Bord geworfen
- Feedback erfolgt erst sehr spät
  - Lerneffekt geht verloren
- Bei den Entwicklern unbeliebt
  - Erfolgt selten systematisch

XP Grundprinzipien

- Schnelles Feedback
- Einfachheit
- Inkrementelle Änderungen
- Änderungen willkommen heissen
- Qualitätsarbeit

Schnelles Feedback

- Testen möglichst zeitnah zur Programmierung
- Modultest vor dem eigentlichen Code schreiben
- Tests automatisiert und isoliert



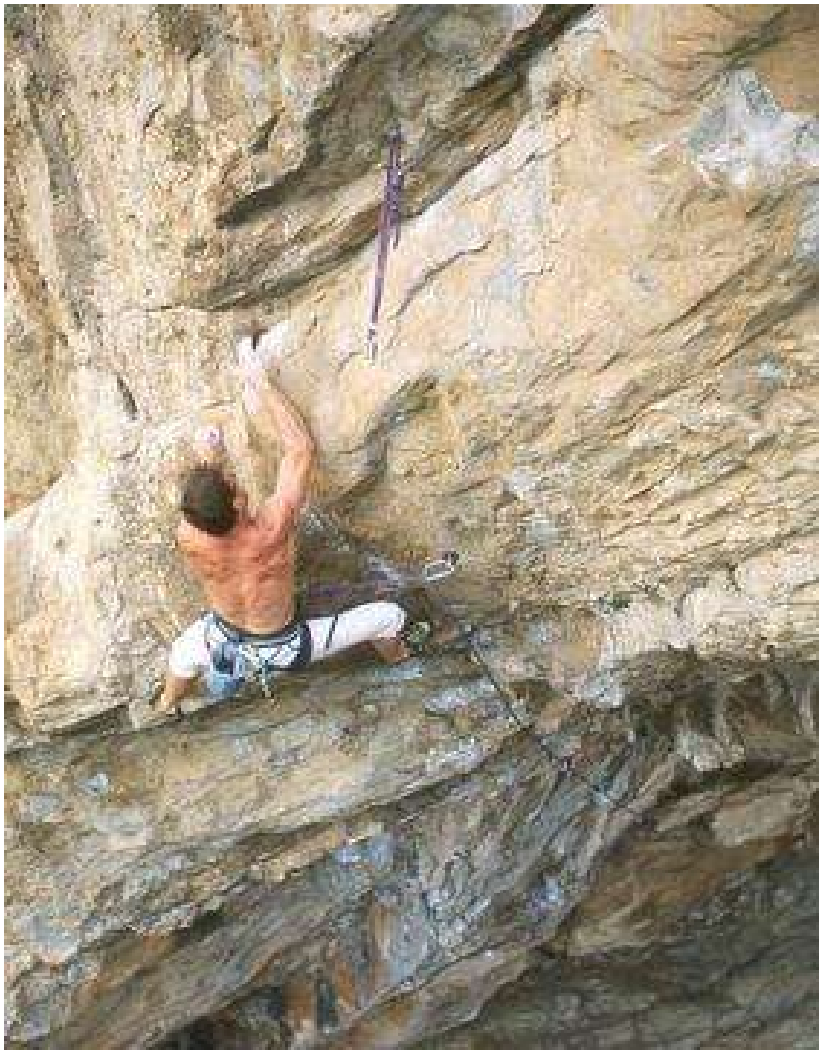
Einfachheit

- Testfall so einfach wie möglich erfüllen
- Nicht für die Zukunft arbeiten
- Code wann immer möglich vereinfachen (Refactoring)

Inkrementelle Änderungen (I)

- Jede Änderung des Programmverhaltens zuvor durch einen Test motivieren
- Möglichst kleine Änderungen
- Entwicklung wird durch Tests geleitet (Test-Driven Development)

Inkrementelle Änderungen (II)



Änderungen willkommen heissen

- Software ändert sich kontinuierlich
- Code bei Änderungen und Erweiterungen durch bereits bestehende Tests abgesichert

Qualitätsarbeit

- Funktionale Qualität:
  Bleibt durch automatische Tests erhalten
- Strukturelle Qualität:
  Fortlaufendes Refactoring
- Vertrauen in den Code
- Testbarer Code

Rolle des Entwicklers

- Testen nicht als lästige Pflicht, sollte einen direkten Nutzen bringen

„Unfortunately at least for me (and not only) testing goes against human nature. If you release the pig in you, you will see that you program without tests. Then after a while, when your rational part wins, you stop and start writing tests.“

Massimo Arnoldi

Vorgehen

1. Füge einen kleinen Test hinzu.
2. Führe alle Tests aus. Neuer Test scheitert.
3. Ändere den Code soweit, dass der Test erfüllt wird.
4. Führe alle Tests aus.
5. Vereinfache den Code und entferne Duplikate. (Refactoring)

Test-First Episode

- Miete für das Ausleihen von DVDs berechnen
- Erster Tag: 2.-
  Jeder weitere Tag: 1.50
- Testframework: JUnit

Experiment mit Test-First (I)

- Universität Karlsruhe: M. Müller und O. Hagner
- Auswirkungen von Test-first Programming auf:
  - Effizienz bei der Programmierung
  - Zuverlässigkeit des resultierenden Codes
  - Verständnis für das resultierende Programm
- Bibliothek zur Analyse und Darstellung von Graphen

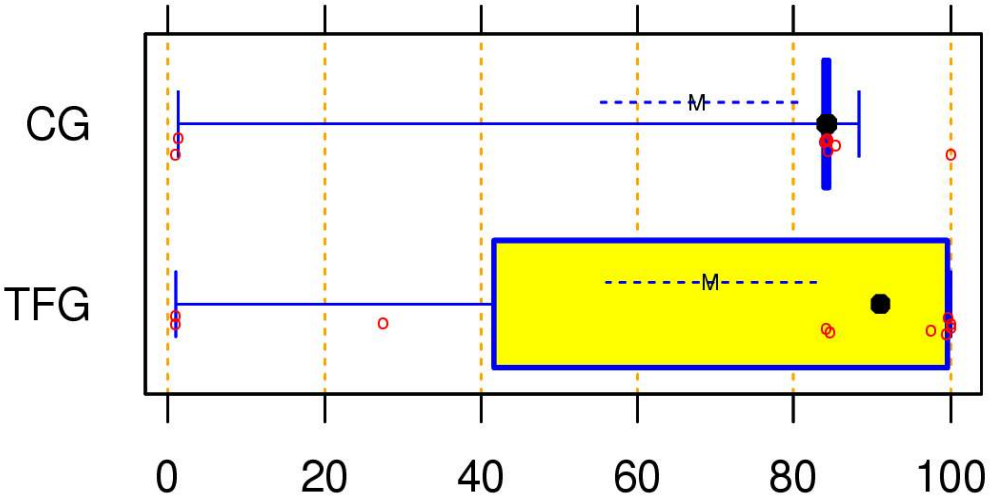
Experiment mit Test-First (II)

- Test-first Programming isoliert von XP
- 10 Studenten gemässt Test-First Ansatz
  9 Studenten als Kontrollgruppe

- Implementierungsphase
- Automatisierter Akzeptanztest

Resultate (I)

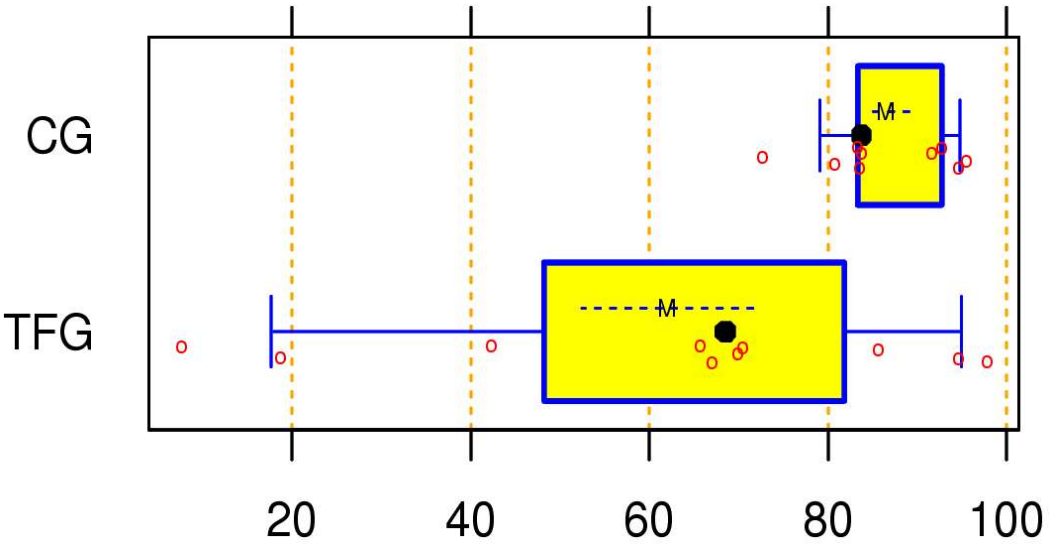
- Effizienz bei der Programmierung:
  Test-First Gruppe nicht effizienter
- Zuverlässigkeit des resultierenden Codes:



Erfüllung der Akzeptanztests des endgültigen Programms in %

Resultate (II)

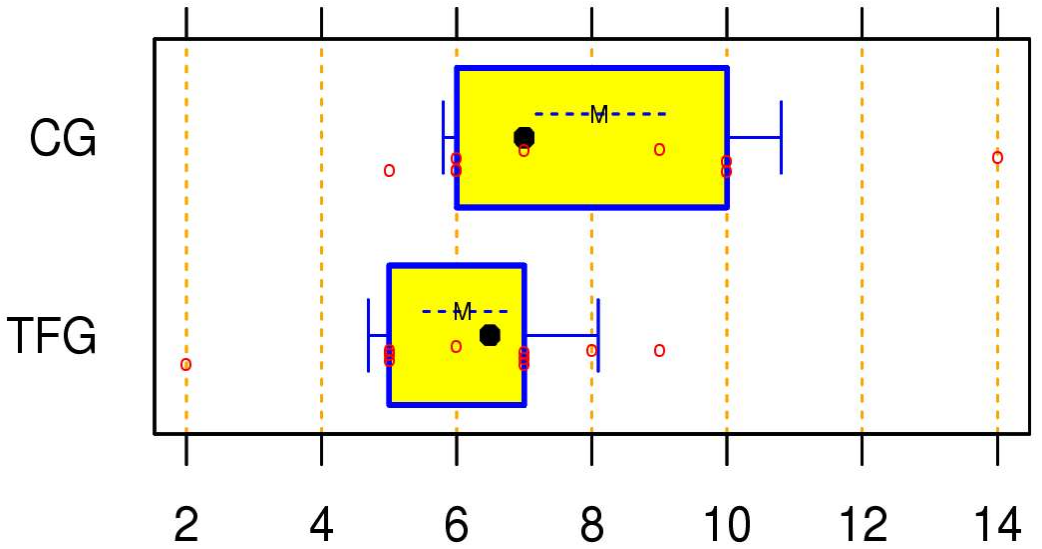
- Zuverlässigkeit nach der Implementierung: Test-First Gruppe signifikant schlechter



Erfüllung der Akzeptanztests nach der Implementierung in %

Resultate (III)

- Verständnis für das resultierende Programm



Anzahl von Methodenaufrufen, die mehrfach scheiterten

Zusammenfassung

- Test-First führt nicht zwingend zu effizienterem Programmieren
- Programme nicht zuverlässiger
- Existierende Methoden werden schneller korrekt wiederverwendet

Testfälle

- Tests können zu falschem Vetrauen führen
- Testfälle systematisch erstellen (Äquivalenzklassen)

One of the ironies of TDD is that it isn't a testing technique [...]. It's an anlysis technique, a design technique, really a technique for structuring all the activites of development."

Kent Beck

Test-First in der Praxis (I)

- CARUSO Projekt der EU:
  CRM Framework für kleinere bis mittlere Unternehmungen:

  For example, [the customer] was referring to CARUSO as their customer care dream. As is common with dreams, CARUSO was supposed to do everything; but because of the complexity of CRM, no concrete requirements were given, since nobody knew where to start.

Test-First in der Praxis (II)

- Für jede neu benötigte Methode wurde zuerst ein automatisierter Test geschrieben
- Neue Funktionalität liess sich relativ einfach hinzufügen
- Code konnte einfach und sauber gehalten werden

Fazit

- Tests werden auf jeden Fall geschrieben
- Tests sind während Entwicklung vorhanden
- Tests ermöglichen erst das Refactoring
- Anforderungen werden in den Tests beschrieben, Tests dienen als Dokumentation

- Qualität der Testfälle
- Tester vs. Entwickler