



Testest Du noch oder entwickelst Du schon (wieder)?



Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Björn Feustel

Steffen Schluff

Version: 1.0

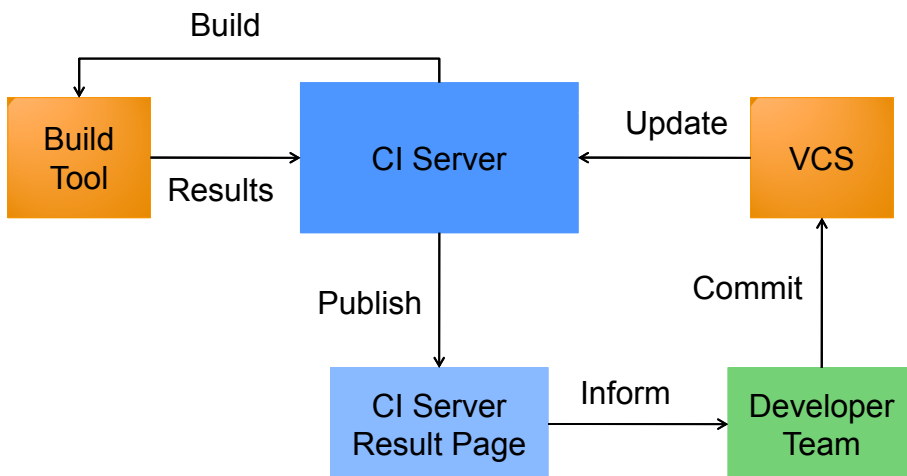
Gliederung

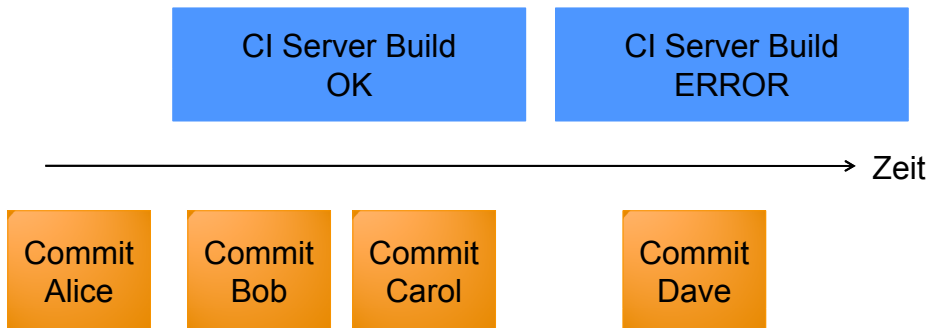
- Ist-Situation
- Gateway Build
- Testoptimierung
- Demo
- Fazit

Gliederung

- Ist-Situation
- Gateway Build
- Testoptimierung
- Demo
- Fazit

Been there, done that





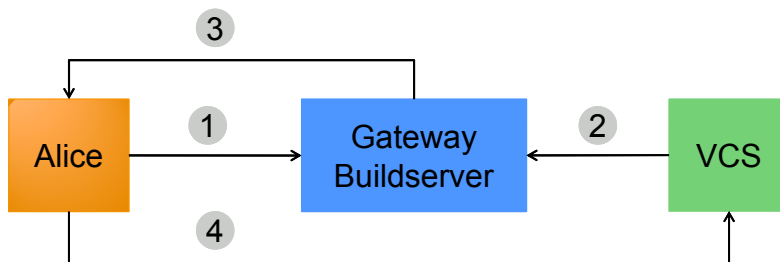
Elementary my dear Watson

- Buildzeiten beschleunigen
 - Ist hilfreich, beseitigt aber das Grundproblem nicht vollständig
 - Potentiell schwierig in der praktischen Umsetzung
- Für jeden Commit einzeln nacheinander bauen
 - Braucht viel (parallele) Rechenleistung für Build Agents
 - Leichtes Opfer des „Slow Machine“ Antipattern
- Änderungen vor Commit als „persönlichen“ Gateway Build prüfen
 - Gewährleistet eine saubere Hauptentwicklungslinie
 - In diversen CI Servern bereits verfügbar („pre-tested commit“)

Gliederung

- Ist-Situation
- **Gateway Build**
- Testoptimierung
- Demo
- Fazit

Pre-tested commit (1)



- Benutzer schickt Änderungen zum Testen an Gateway
- Gateway testet VCS Stand mit Änderungen
- Gateway sendet Testergebnis an Benutzer
- Commit der Änderungen durch Benutzer wenn Testergebnis OK

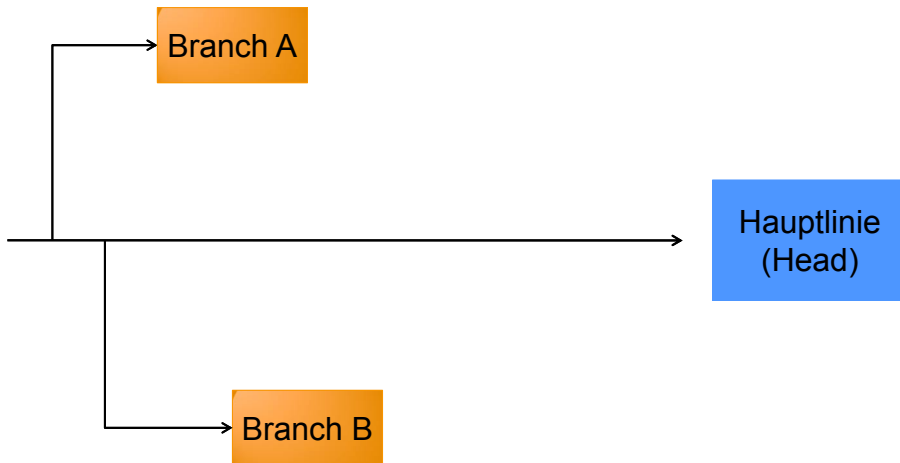
Pre-tested commit (2)

- Saubere Hauptentwicklungslinie wird gewährleistet
- Aber: Großer Ansturm auf CI Server kann Feedbackzeit vergrößern
- Aber: Konkrete Umsetzung häufig abhängig von IDE bzw. Toolstack
- Weiterhin keine echte Isolation größerer Entwicklungsaufgaben
- Gibt es eine andere Lösung?

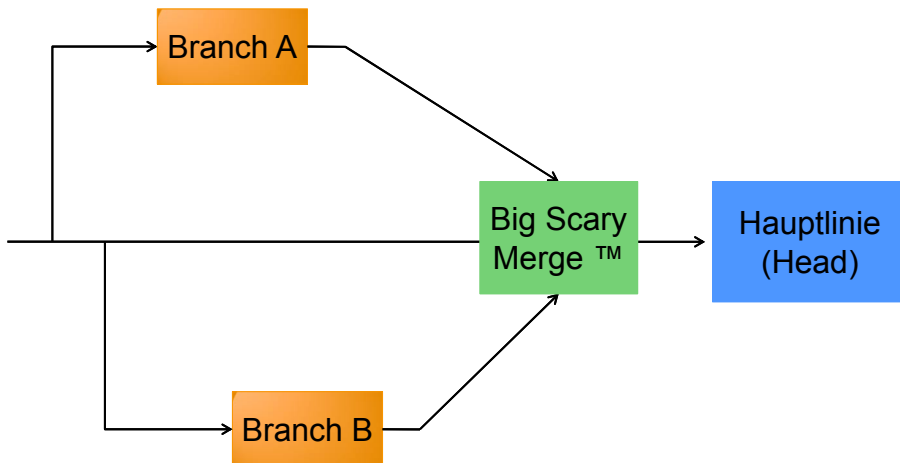
Feature Branches (1)

- „The key which opens real [...] parallel development are branches.“
- Branch ist ein sich abspaltender neuer Entwicklungsweig
- Verteilte Versionsierungstools (DVCS) erleichtern Branches deutlich
- Branches sind Grundlage der meisten DVCS Workflows

Feature Branches (2)



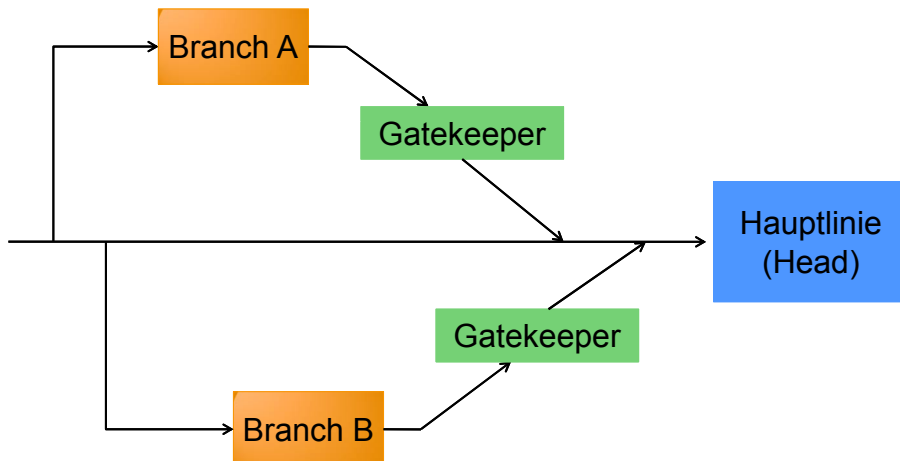
Here be dragons (1)



Here be dragons (2)

- Merge ist das erneute Zusammenführen von Branches
- Branches können sich textuell und semantisch unterscheiden
- „It's [...] particularly semantic conflicts that make big merges scary.“
- Stabilität der Hauptentwicklungslinie darf nicht leiden

Are you the Keymaster? (1)



Are you the Keymaster? (2)

- Gatekeeper garantiert Stabilität des Heads
 - Bezeichnung kann variieren (z.B. Integrator)
- Gatekeeper kann Mensch oder Maschine sein
 - Besitzt als einziger Schreibrechte in der Hauptentwicklungslinie
- DVCS bieten entsprechende Tool Unterstützung
 - z.B. Server für Patch Queue Management (PQM)
- Verwandtschaft zu Pre-tested commit
 - Pre-tested commit über Branches implementierbar

You're gonna need a bigger boat (1)

- „[...] each commit and merge comes with an automated build and test on the branch it's on.“
 - Jeder Branch erhöht Last auf Buildserver
- „[...] running all the tests again and again will be CPU demanding“
 - Weil alle Test in allen Branches verwendet werden
- „[...] subteams might be interested in a specific set of tests which is too slow to be run for every commit“
 - Nicht jeder Branch soll alle Tests ausführen

You're gonna need a bigger boat (2)

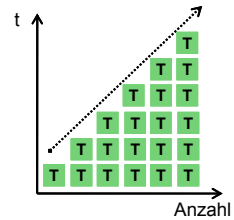
- „The gatekeeper does a merge, a compile and runs the test suite“
 - Zusammenhang zwischen Testlaufzeit und Merge Durchsatz
- „when [an] integration is performed, the integrator runs a subset (smoke tests) of the complete test suite for each integrated branch“
 - Und das sind welche?
- „[...] you can use various tools and heuristics to decide which tests it might be prudent to run“

Gliederung

- Ist-Situation
- Gateway Build
- **Testoptimierung**
- Demo
- Fazit

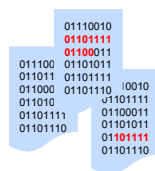
Test Optimierung – Motivation

- Idealerweise sind Tests schnell
- ∞ viele schnelle Tests sind ∞ langsam
- Die Folgen
 - Haben wir gerade gehört
- Ziel: Das Richtige in der verfügbaren Zeit testen
 - Ad-Hoc Feedback – „Continuous Testing“
 - Laufzeiten CI Tests
 - Manuelle Tests



Test Optimierung – Ansatz Test Selection (1)

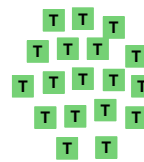
- Test Selection
 - Problem: Für jede Änderung werden alle Tests ausgeführt



Quellcode-
änderung



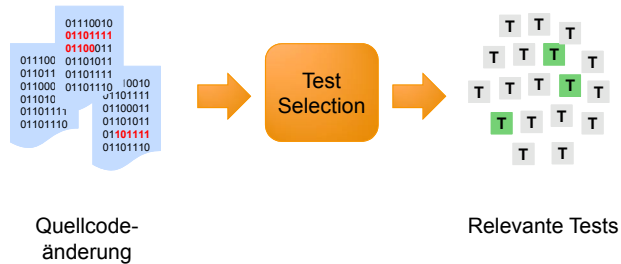
Normale
Testausführung



Alle Tests

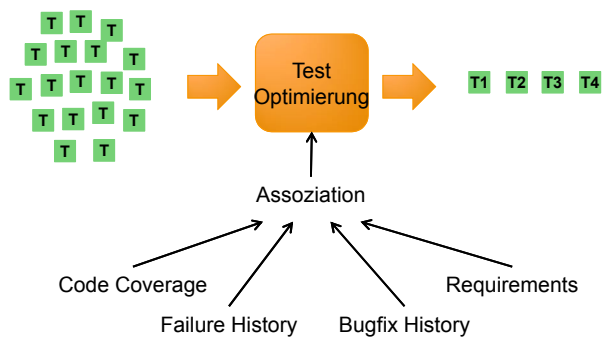
Test Optimierung – Ansatz Test Selection (2)

- Test Selection
 - Lösungsansatz: Nur relevante Tests ausführen



Test Optimierung – Relevanzkriterien

- Aber wovon Relevanz ableiten?
 - => Durch Assoziation von Tests zum Quellcode



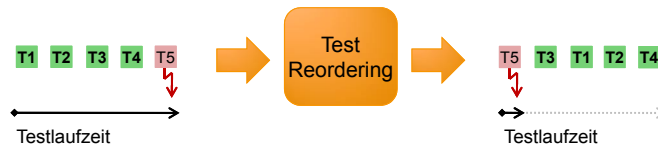
Test Optimierung – Kriterien für Selection (1)

- Code Coverage
 - *Welche Tests korrelieren mit den geänderten Quellen?*
 - Problem: Was ist mit Quellen ohne Coverage, z.B. XML?
- Failure History
 - *Welche Tests schlugen bei ähnlichen Änderungen fehl?*

Test Optimierung – Kriterien für Selection (2)

- Bugfix History
 - *Welche Tests wurden für Bugs angelegt, deren Behebung die geänderten Quellen betrafen?*
- Requirements
 - *Welche Tests wurden für ein Requirement angelegt, für das auch die geänderten Quellen erzeugt/geändert wurden?*
 - Problem: Umfassendes Requirements-Management/TDD erforderlich

- Test Reordering
 - Problem: Erst einer der letzten Tests schlägt fehl
 - Lösungsansatz: Tests gezielt priorisieren – Fail Fast!



- Most recent failures first
 - *Welche Tests schlugen als letztes fehl?*
 - Hohe Trefferquote im Entwicklungsalltag
- Most frequent failures first
 - *Welche Tests schlagen am häufigsten fehl, sind am "wackeligsten"?*
- Quick tests first
 - *Welche Tests laufen am schnellsten?*
- Random
 - *Bestehen unerwünschte, implizite Abhängigkeiten zwischen Tests?*

- Weitere Kriterien sind denkbar
 - Priorität, Komplexität, Änderungsrate von Requirements
 - Aus statischer Code-Analyse, z.B. AdHoc in IDE

- Wahl und Gewichtung der Kriterien ist abhängig vom/von
 - Prozess: Werden Requirements erfasst?
 - Projektstadium: Am Anfang eher Requirements getrieben
 - Testart: Unit Tests, Integrationstests, GUI Tests, manuelle Tests, ...

- Ist-Situation

- Gateway Build

- Testoptimierung

- **Demo**

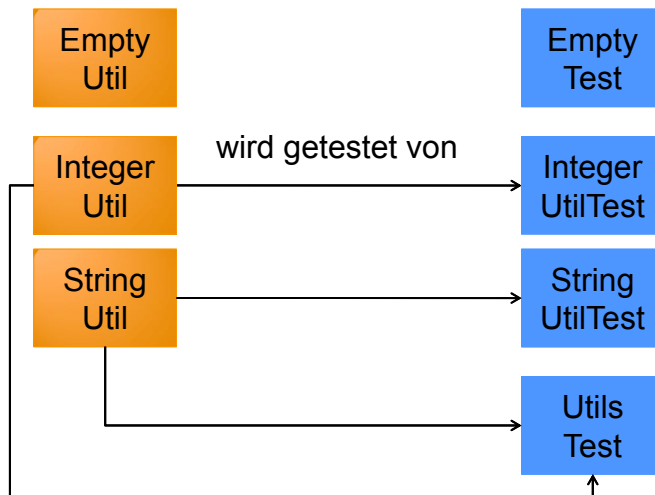
- Fazit

Demo

- Demo „Testoptimierung“



Aufbau Demo „Testoptimierung“



Gliederung

- Ist-Situation
- Gateway Build
- Testoptimierung
- Demo
- **Fazit**

Fazit

- Branches und Gatekeeper können Entwicklung deutlich stabilisieren
- Ein solcher Entwicklungsprozess erfordert aber mehr Erfahrung...
- ... und die Testlaufzeiten werden dadurch (noch) relevanter
- Abhilfe durch Testoptimierung (Selective Testing und Test Reordering)
- Optimierung basiert auf diversen Kriterien (Code Coverage, ...)
- Toolunterstützung ist noch im Entstehen

“But how will I know when I have received enlightenment?”
asked the novice.

“Your program will then run correctly,”
replied the master.

(The Tao Of Programming, Book 4.2, Geoffrey James)

Mehr von OIO zum Thema...

- Artikel: Wie man sich (test-)bettet, so liegt man
 - <http://www.oio.de/j2ee-testtools.pdf>
- Artikel: Test-Coverage für Grails
 - <http://www.oio.de/public/opensource/tutorial-grails-test-coverage.htm>
- Artikel: Continuous Integration mit CruiseControl
 - <http://www.oio.de/cruisecontrol.htm>
- Artikel: Cover your world!
 - <http://www.oio.de/testcoverage-clover.htm>
- Schulung: Testen von Java Programmen
 - <http://www.oio.de/java-testen-schulung.htm>
- Schulung: Refactoring Workshop
 - <http://www.oio.de/seminar/methodik-prozess-management-soft-skills/training-java-kurs-refactoring-schulung.htm>

Björn Feustel

Trainer, Berater, Entwickler



Schwerpunkte
*Build- und Konfigurationsmanagement
Systemarchitekturen
Requirements-Engineering*

Steffen Schluff

Trainer, Berater, Entwickler



Schwerpunkte
*Open Source Tooling
Build Management
Refactoring*



Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de



**Vielen Dank für ihre
Aufmerksamkeit !**

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de