Automatisierte Build-Prozesse in Java-Projekten

- Einführung
- Continuous Integration
- Sechs Prinzipien
- Benefit
- Werkzeuge
- Fazit

Automatisierte
Build-Prozesse in
Java-Projekten

Einführung

Referent

- Olaf Kossak
- Freiberuflicher Informatiker
- Studium an der Universität Hamburg
- Java-Entwickler
- Teamleiter
- Qualitätsingenieur
- Banken, Versicherungen, Großhandel, Telekommunikation, Medizintechnik
- olaf.kossak@hamburg.de

Einführung

Heutige Situation und ihre Folgen

- Komplexe Software-Architekturen
- Komplexe Software-Entwicklungsumgebungen
- Komplexe fachliche Anforderungen
- Hohe Arbeitsbelastung der Entwickler
- Projektqualität leidet
- Schwierige Integrationsphase
- Produkt reift beim Kunden
- Hohe Mitarbeiterfluktuation

Einführung

Moderne Antworten

- Schnelle Ergebnisse:
  -> Iterative Entwicklungsprozesse,
  Inkrementelles Wachsen der Funktionalität
- Unvollständige und wechselnde Anforderungen:
  -> Agile Prozessmodelle
- Wartbarkeit erhalten:
  -> Permanentes Refactoring
- Plattformunabhängige Entwicklung, frühe Formalisierung:
  -> Modellgetriebene Entwicklung MDA, MDSD

CI

Continuous Integration

- Entwickler verwalten Projekte im Quellcode-Repository
- Scheduler horcht am Repository
- Build-Server baut Projekte
  - kompiliert
  - führt Regressionstests aus
  - baut Komponente
- Im Fehlerfall
  - Benachrichtigung der Verantwortlichen
- Im Erfolgsfall
  - Veröffentlichung der Komponente im Artefakt-Repository

CI

Integrierte Qualitätssicherung

- QS-Server sichert Qualität
  - Testabdeckung
  - Dokumentationsabdeckung
  - Verstöße gegen Programmierrichtlinien
  - Verstöße gegen Architekturrichtlinien
  - Metriken zur Wartbarkeit
  - Performance-Tests
  - Lasttests
  - veröffentlicht Qualitätsbericht auf Dokumentations-Server

Sechs Prinzipien des Java-Qualitätsmanagements

1. Standardisieren
2. Automatisieren
3. Integrieren
4. Messen
5. Visualisieren
6. Wissen verwalten

Sechs Prinzipien

1. Standardisieren

- Referenzarchitekturen
- Technologieportfolio
- Entwicklungsprozess
- Entwicklungsumgebung in Produkten und Konfigurationen
- Richtlinien und Empfehlungen für Programmierung, Dokumentation, Tests, Architektur, Metriken

Sechs Prinzipien

2. Integrieren

- Integration der Java-Qualitätssicherung in IDE
- Aufbau einer integrierten Prozesskette
  - Code-Repository
  - Build-Server
  - Artefakt-Repository
  - Test-Server
  - Report-Server
- Integration der Qualitätssicherungsberichte

Sechs Prinzipien

3. Automatisieren

- Unattended Setup der Entwicklungsumgebung
- Quellcode, Projektressourcen und Tests generieren
- Projekt auf Build-Server aus-checken, bauen, testen und archivieren
- Tests auf Test-Server ausliefern und ausführen
- Qualitätsmessungen auf QM-Server ausführen
- Build-, Test- und QM-Reports auf Report-Server veröffentlichen
- Anwenderdokumentation in verschiedenen Formaten generieren

Sechs Prinzipien

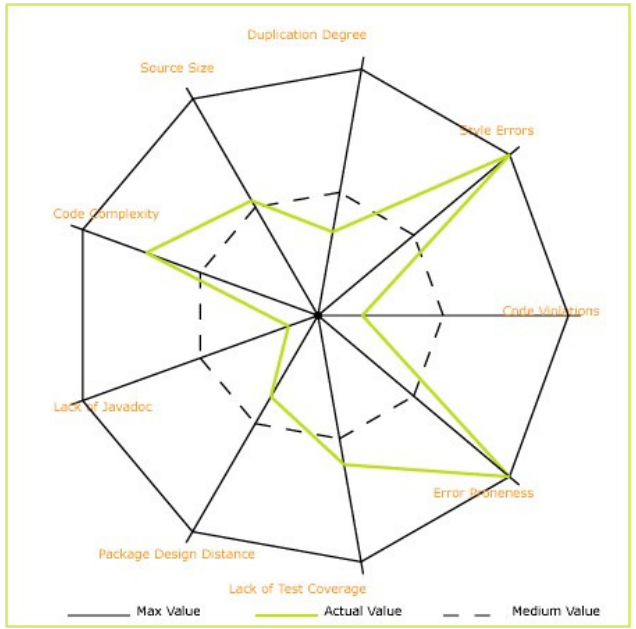
4. Messen

- Einhaltung der Architekturrichtlinien und –metriken
- Grad der Testabdeckung
- Grad der Dokumentationsabdeckung
- Einhaltung der Programmierrichtlinien
- Projektfortschritt

Sechs Prinzipien

5. Visualisieren

- Logische Software-Architekturen
- Kennzahlen der Qualitätssicherung
- Kennzahlen des Projektmanagements
- Wissensverteilung der Mitarbeiter



Spinnwebgrafik von XRadar

Sechs Prinzipien

6. Wissen verwalten

- Wissensmatrix der Mitarbeiter erheben
- Wissensportfolio für Technologien, Werkzeuge, Prozesse definieren
- Wissensportfolio über Fachdomänen definieren
- Schulungsdokumentation erstellen
- Wissensengpässe beheben

Benefit

- Schnelle Rückmeldung über Projektzustand
- Jederzeit verfügbare Versionen aller Komponenten
- Höhere Produktqualität bei funktionalen und nichtfunktionalen Anforderungen
- Mehr Projekttransparenz, höhere Steuerbarkeit, verbessertes Controlling
- Kein Eigentum an Quellcode mehr, Quellcode ist Material
- Aufhebung des Fachidiotentums, mehr Identifikation als Team

Werkzeuge

Ant

- Kommandoprozessor für Build-Prozesse
- Build-Prozess ist XML-Datei (DTD existiert nicht)
- Build-Prozess als Baumstruktur (project, target, task)
- Ant ist keine Skriptsprache
- Ant ist nicht orthogonal
- Ant liefert keinen Standard-Build-Prozess
- http://ant.apache.org/

Werkzeuge

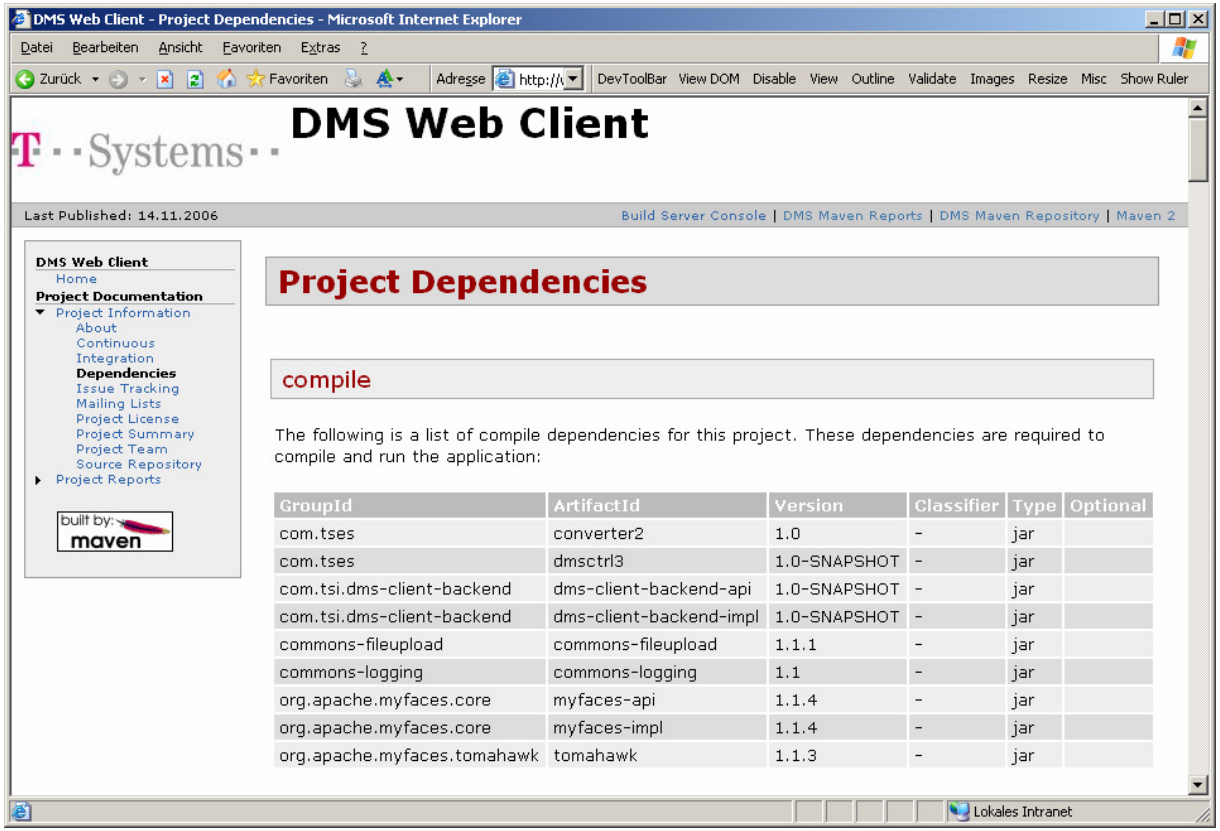
Maven

- Kommandoprozessor-Framework für Build-Prozesse
- Standard-Build-Prozess
- Alle Projekteigenschaften in einer Datei pom.xml
- Project Object Model
- Artefakt-Repository
- Verwaltung der Projektabhängigkeiten
- Build-Report als Website
- http://maven.apache.org/

Automatisierte Build-Prozesse in Java-Projekten

Werkzeuge

Maven



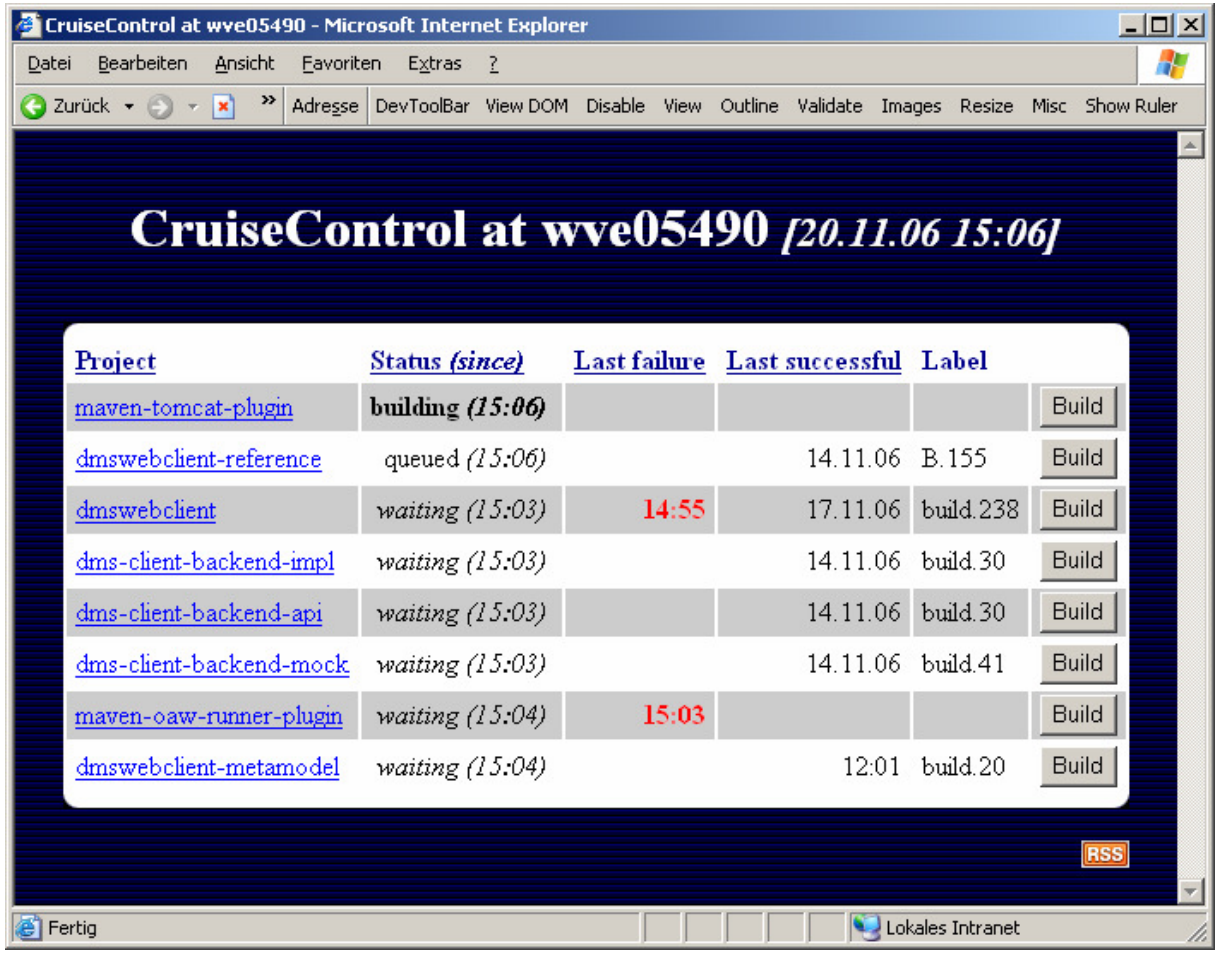
Werkzeuge

CruiseControl

- Scheduler für Continuous Integration
- Horcht am Quellcode-Repository
- Arbeitet mit ca. 20 Repository-Produkten
- Stößt Ant-, Maven- oder Batch-Builds an
- Konsole als Web-Anwendung
- Build-Log
- Meldet Build-Ergebnis über Email
- http://cruisecontrol.sourceforge.net/
- Alternativen: Anthill, Hudson, Continuum

Werkzeuge

CruiseControl



Werkzeuge

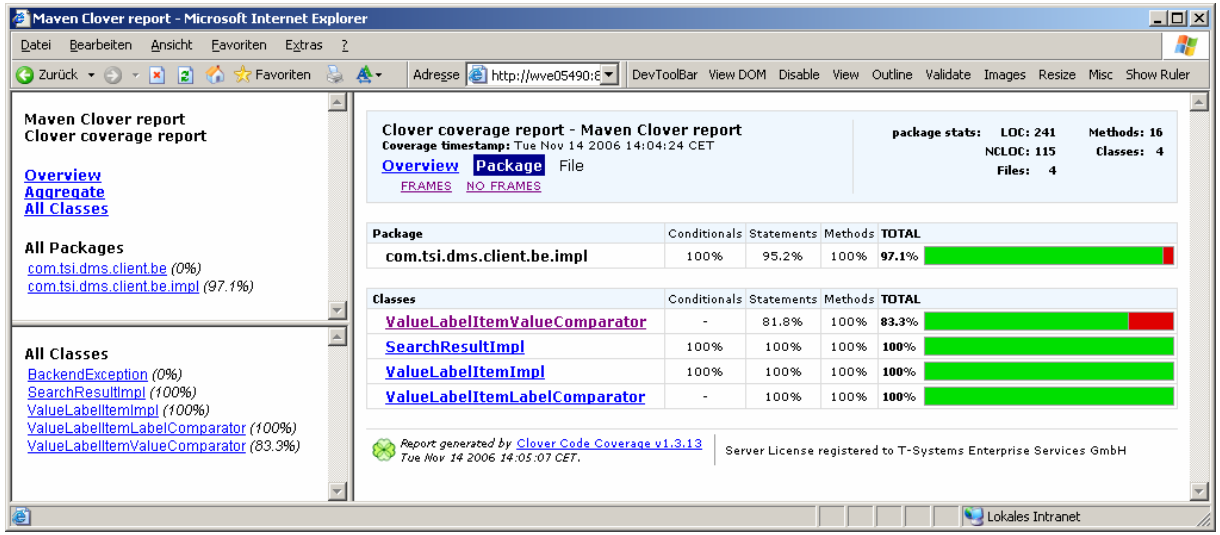
Clover

- Messung der Code-Abdeckung durch Tests
- Bytecode wird mit Zählern instrumentalisiert
- Ausführung der Tests zählt Zähler hoch
- Visualisierung á la Javadoc:
  durchlaufene Statements grün, ungetestete rot
- HTML-Testbericht mit Prozent Testabdeckung
- Drilldown von Projekt- bis auf Methodenebene
- Kommerzielles Produkt, ab 2.000 €
- http://www.cenqua.com/

Automatisierte Build-Prozesse in Java-Projekten

Werkzeuge

Clover



Werkzeuge

Jalopy

- Automatische Formatierung von Java-Quellcode
- Whitespace, Indentation, Wrapping, Sort
- Aufräumen von import-Anweisungen
- Generieren fehlender Javadoc-Kommentare
- Überprüfung von Programmierrichtlinien
- Integration in Eclipse, JBuilder, IDEA, NetBeans, Ant, Maven
- Trend geht in Richtung automatisiertes Refactoring
- Kommerzielles Produkt, ab 50,- €
- www.triemax.com

Werkzeuge

SonarJ

- Überprüfung auf Einhaltung von Architekturregeln
- Definition einer logischen Architektur aus Schichten, Schnitten und Subsystemen
- Build-Report über Ant oder Maven
- Integration in Eclipse
- Kommerzielles Produkt
- www.hello2morrow.de

Fazit

- Die Sechs Prinzipien des Qualitätsmanagements
  - ermöglichen Bewältigung höherer technischer und fachlicher Komplexität
  - erhöhen Planungssicherheit der Projektleitung
  - steigern Motivation und Knowhow der Mitarbeiter
  - sind Voraussetzung für Agile Prozesse und Modellgetriebene Architektur

Automatisierte Build-Prozesse in Java-Projekten

- Dienstleistungsangebot
  - Automatisierung des Build-Prozesses
  - Integration von Qualitätssicherung
  - Projekt-Review, Technisches Audit
- olaf.kossak@hamburg.de