



Build-Infrastrukturen mit marktgängigen Tools



Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Björn Feustel

Steffen Schluff

Version: 1.0

Gliederung

- Einleitung
- Issue-Tracker
- Integrierte Entwicklungsumgebung (IDE)
- Source-Code Management System (SCM)
- Continuous Integration Server (CI-Server)
- Zusammenfassung und Ausblick

- **Einleitung**
- Issue-Tracker
- Integrierte Entwicklungsumgebung (IDE)
- Source-Code Management System (SCM)
- Continuous Integration Server (CI-Server)
- Zusammenfassung und Ausblick

- Ein guter Entwicklungsprozess ist einfach, flexibel und praxisorientiert, d.h.:
 - Reibungslose Arbeit im Team
 - Schnelle Entwicklungszyklen
 - Inhärenter Qualitätsanspruch
 - Gute Planung und Steuerung
- Eine Build-Infrastruktur muss das unterstützen, z. B. durch:
 - Bereitstellen gemeinsamer, integrierter Entwicklungswerkzeuge
 - Automatisieren von wiederkehrenden Prozessen
 - Vorgeben und Prüfen von Konventionen, z.B. Metriken
 - Vereinfachen der Projektkontrolle
- Und wen betrifft es?

Rollenverständnis

- Rollenbegriffe sind abhängig von Projektgröße / -struktur, Organisation
 - Developer, Architect
 - Tester / QA
 - Release Engineer & Manager
 - Product & Project Manager
 - Product Owner
 - Scrum Master
- Im Kontext der Präsentation
 - Team
 - Entwickler, Spezialisten
 - Ändert den Sourcecode
 - Erstellt Tests/sichert die Qualität
 - Kennt (und verbessert) den Build-Prozess
 - Controller
 - Scrum Master
 - Pflegt und optimiert das Projekt
 - Überwacht und steuert den Projektfortschritt
 - Stakeholder
 - Product Owner und Interessenten
 - Bestimmen die Ziele und Prioritäten



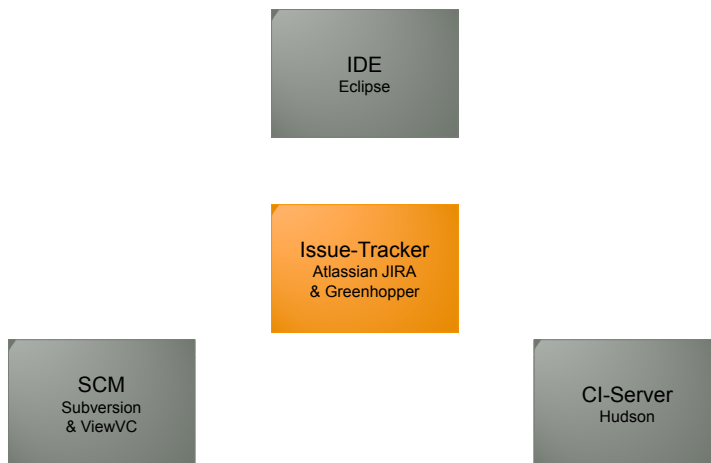
Bausteine einer modernen Build-Infrastruktur



Gliederung

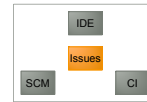
- Einleitung
- **Issue-Tracker**
- Integrierte Entwicklungsumgebung (IDE)
- Source-Code Management System (SCM)
- Continuous Integration Server (CI-Server)
- Zusammenfassung und Ausblick

Issue-Tracker



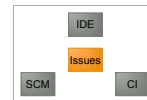
Issue-Tracker – Synopsis

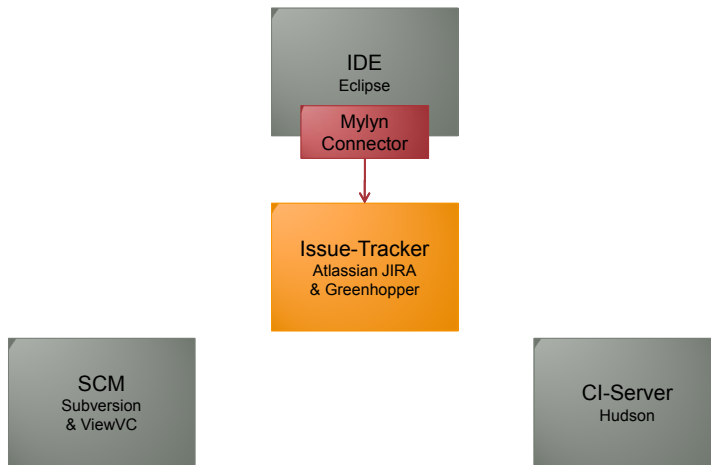
- Aufgabe
 - Erfasst **alle** Änderungen und Aktivitäten
 - Bug Tracking vs. Issue Management vs. SCRUM
 - Ermöglicht die Projektplanung
 - Features, Versionen, Fix-Termine, Kapazität
 - Gibt verbindliche Auskunft über den Projekt(zu)stand
 - Nächste Aufgaben, Versionsfortschritt, Arbeitsauslastung
 - Entkopplung der Entwicklung von ablenkenden Prozessen
 - Requirements Management, Change Management
- Rollen und Verwendung
 - Alle: Ermitteln und Pflege des Projektstatus
 - Alle: Projektplanung
 - Team: Bereitstellen des Arbeitskontexts (Myllyn / Eclipse)
- Produkte
 - **Atlassian JIRA**, Bugzilla, Roundup, FogBugz, Trac



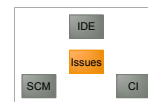
Demonstration

- Organisation der Issues / Release-Notes
- Anbindung an IDE per Mylyn (Atlassian IDE Connector)



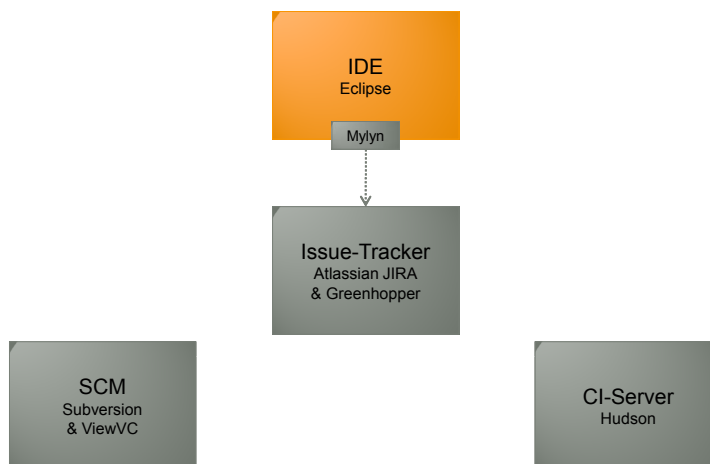


- Nachvollziehbarkeit / Reproduzierbarkeit
 - Arbeiten immer im Kontext eines Issues
 - Issues nach Versionen erfassen
- Aktualität
 - Issues immer auf Personen zuordnen
 - Änderungen unmittelbar dokumentieren
 - Medienbruch für den Entwickler vermeiden (z. B. mit Mylyn)
 - Organisation der Issues optimieren (z.B. mit Greenhopper)
- Als Single Point of Truth etablieren
 - Berührungsgängste bei allen Beteiligten abbauen
- *Aber: Individuals and interactions over processes and tools*



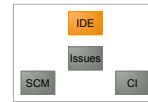
(Agile Manifesto)

- Einleitung
- Issue-Tracker
- **Integrierte Entwicklungsumgebung (IDE)**
- Source-Code Management System (SCM)
- Continuous Integration Server (CI-Server)
- Zusammenfassung und Ausblick



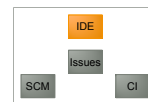
IDE – Synopsis

- Aufgabe
 - Zentrales Arbeitswerkzeug der Entwickler
 - Maximierung der Entwicklerproduktivität
- Rollen und Verwendung
 - Team: Entwicklers Habitat
 - Team: Allgemeiner Zugriff auf SCM (Subversive)
 - Team: Kontextbezogener Zugriff auf Issue-Tracker (Mylyn)
- Produkte
 - **Eclipse**, NetBeans, IntelliJ IDEA

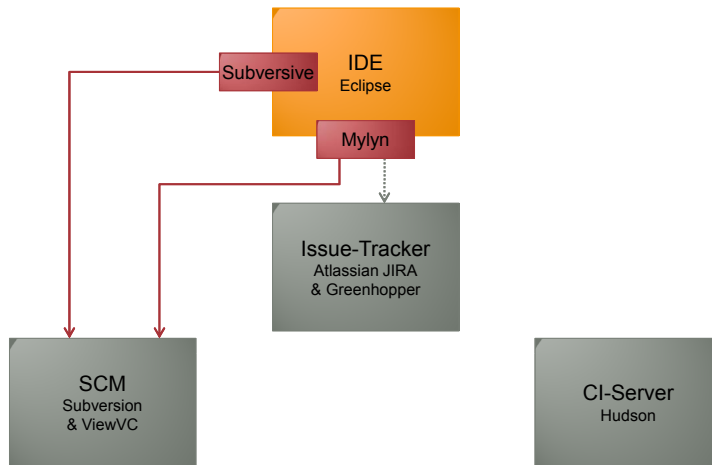


Demonstration

- SVN Integration
- Changesets verwalten mit Mylyn

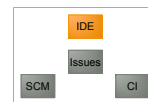


IDE – Das Build-System wächst...



IDE – Best Practices & Konventionen

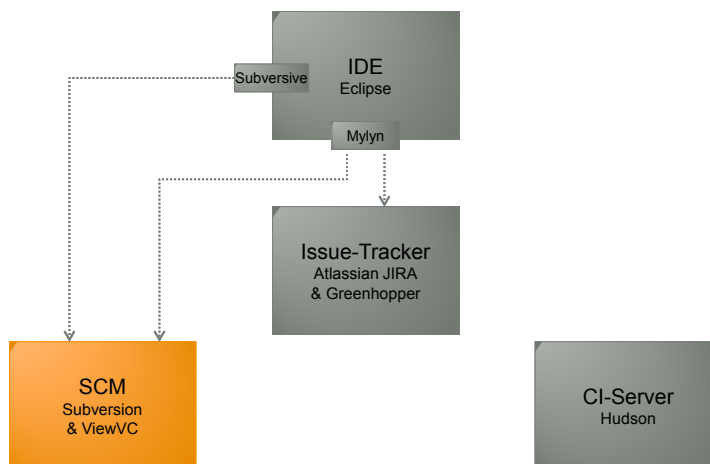
- Projektweite Vorgaben für alle
 - Die gleiche IDE (Produkt, Plugins)
 - Das gleiche Vorgehen (Handling der Issues)
 - Die gleichen Einstellungen (Code Formatter, Code Syntax, ...)
- Vermeiden von Tool-Brüchen
 - Integrierter SVN Client
 - Integriertes Deployment in lokale Testserver (pre-tested Commit)
- Optimieren des Arbeitsflusses
 - Task/Context Management (z. B. Mylyn / Eclipse oder Cube'n / NetBeans)
 - Automatische Prozesse (z. B. „Save Actions“ / Eclipse)
- Aber: Build-Prozess muss außerhalb der IDE funktionieren



Gliederung

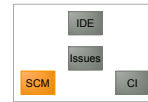
- Einleitung
- Issue-Tracker
- Integrierte Entwicklungsumgebung (IDE)
- **Source-Code Management System (SCM)**
- Continuous Integration Server (CI-Server)
- Zusammenfassung und Ausblick

SCM – Software Configuration Management



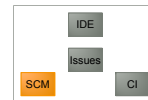
SCM – Synopsis

- Aufgabe
 - Verwaltung sämtlicher Quellartefakte
 - Sourcen, Konfiguration, Dokumentation
 - Zusammenarbeit im Team ermöglichen
 - Versionsverwaltung / Baselining
- Rollen und Verwendung
 - Alle: Zugriff auf alle Artefakte und Dokumentation (ViewVC / SVN)
 - Alle: Nachvollziehen von Änderungen (JIRA Subversion Plugin)
 - Team: Grundlage für parallele Entwicklung (Branch/Merge)
- Produkte
 - **SVN**, Git, Perforce, Mercurial, CVS

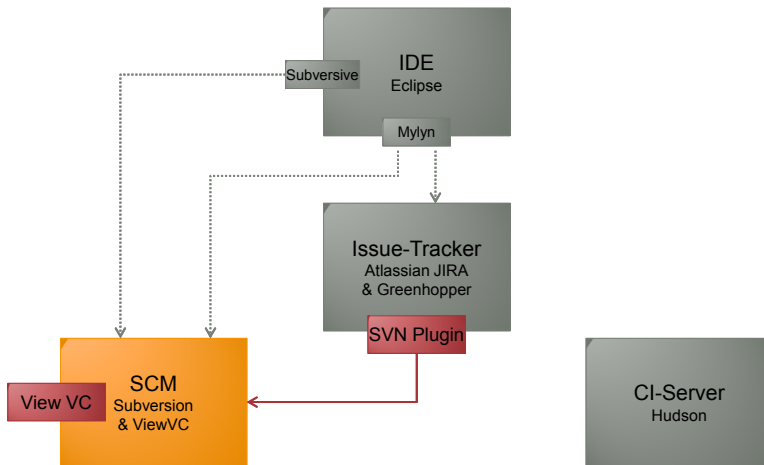


Demonstration

- Nachvollziehbarkeit in JIRA
- Repository Zugriff mit ViewVC

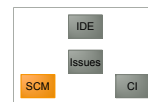


SCM – Das Build-System wächst...



SCM – Best Practices & Konventionen

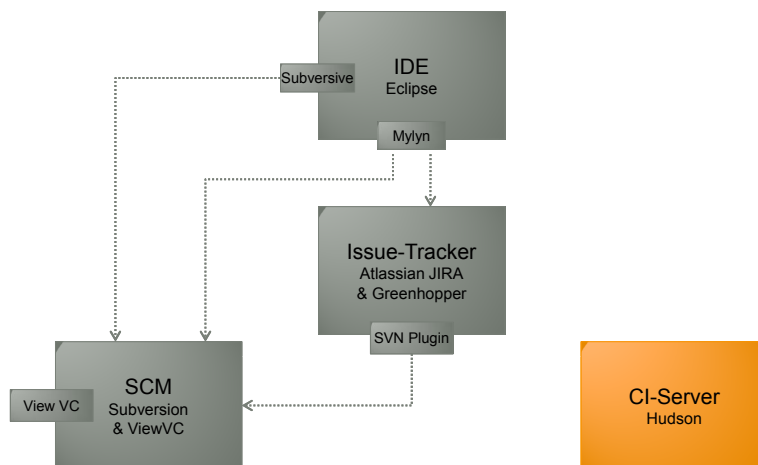
- Optimieren des Projektflusses
 - Tooling beherrschen (Merging)
 - Häufige Check-ins & Merges (aber: Head stabil halten)
 - Check-in immer auf ein Issue bezogen (z.B. SVN-Hook)
 - Atomare Check-ins mit aussagekräftigen Kommentaren
- Mechanismen zur Projektverfolgung nutzen
 - Zugriff für alle ermöglichen: ViewVC, Tortoise
 - Benachrichtigungen (z.B. automatischer Mailversand oder RSS-Feeds)
- Konsistenz, Vollständigkeit und Ordnung wahren
 - Jede Version der Software ist aus dem SCM reproduzierbar
 - Zentrales Repository bei DVCS verwenden
 - Alte Daten löschen
- Aber: Bei SCM gibt es kein „aber“!



Gliederung

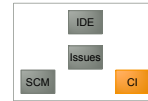
- Einleitung
- Issue-Tracker
- Integrierte Entwicklungsumgebung (IDE)
- Source-Code Management System (SCM)
- **Continuous Integration Server (CI-Server)**
- Zusammenfassung und Ausblick

CI-Server – Continuous Integration Server



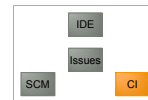
CI-Server – Synopsis

- Aufgabe
 - Qualitätssicherung
 - Automatische Integration
 - Ausführen von Tests, Erstellen von Reports
 - Qualitätshistorie und Trends aufzeigen
 - Gewährleisten der Reproduzierbarkeit
 - Ausführen von Referenz-Builds
 - Automatisches Markieren im SCM
 - Automatisches Erstellen und Ausliefern des Produktes
 - Instanzieren der Deployment Pipeline
- Rollen und Verwendung
 - Team: Integrations- und Qualitätsfeedback
- Produkte
 - **Hudson**, CruiseControl, Bamboo, TeamCity, Go

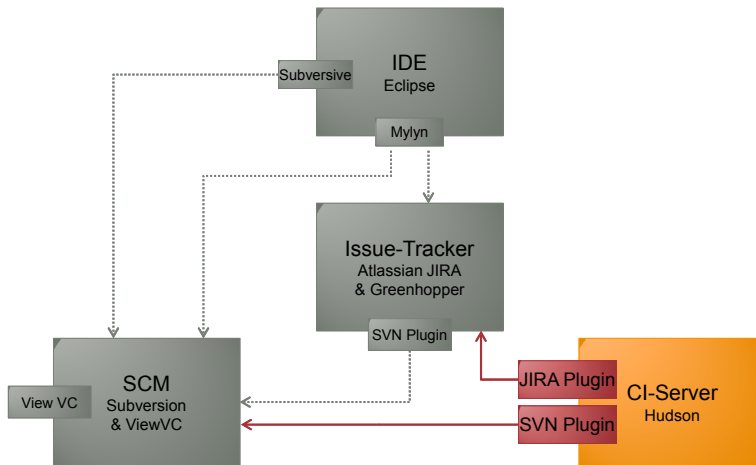


Demonstration

- SVN Anbindung
- JIRA Plugin für Hudson

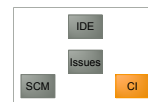


CI-Server – Das Build-System wächst...



CI-Server – Best Practices & Konventionen

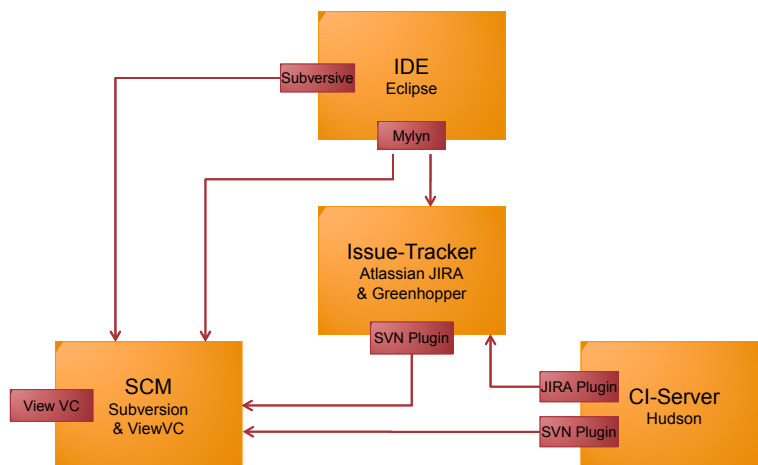
- Optimieren des Projektflusses
 - Abwarten des CI-Laufs, ggf. sofort claimen/reparieren
 - Tests lokal ausführen vor dem Check-in (Pre-Commit Test)
 - Don't commit on a broken build
 - Quantität & Qualität der Tests muss stimmen
- Optimieren des Arbeitsflusses
 - Testlaufzeiten niedrig halten (Test-Optimierung, Staffelung, Parallelisierung)
 - Status für alle sichtbar machen
- Feedback nutzen
 - Benachrichtigungen bei Fehlern (Mail, IM, IDE-Plugin)
 - Code Qualität (Metriken) auswerten, Trends beobachten
- Aber: CI-Server ist nur so gut wie man ihn gut sein lässt



Gliederung

- Einleitung
- Issue-Tracker
- Integrierte Entwicklungsumgebung (IDE)
- Source-Code Management System (SCM)
- Continuous Integration Server (CI-Server)
- **Zusammenfassung und Ausblick**

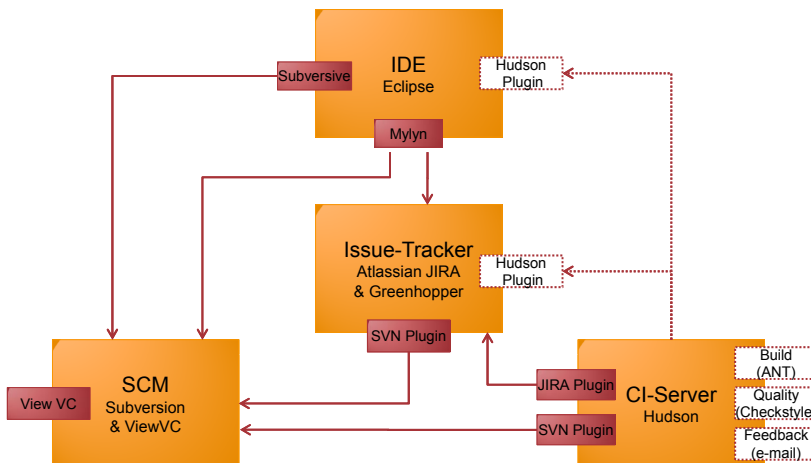
Et voilà – die Build-Infrastruktur



Et voilà – die Build-Infrastruktur



Und hier der Nachschlag

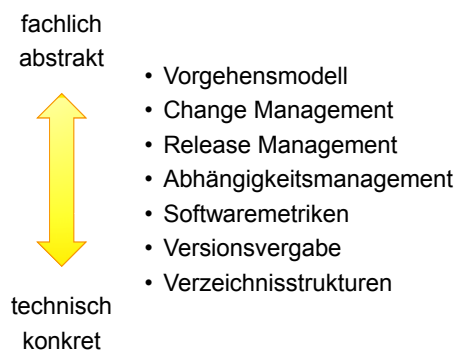


Zusammenfassung

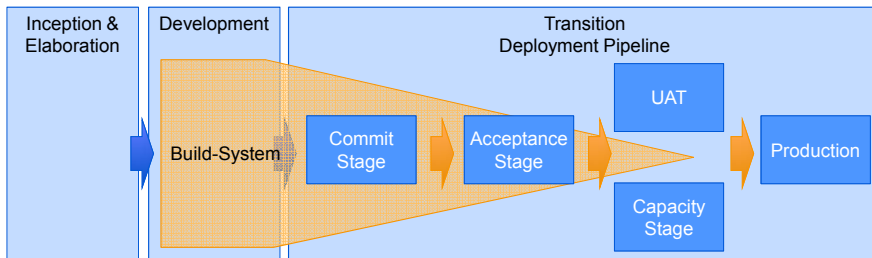
- Ausgereifte Tools existieren und sind...
 - ...rein konfigurativ integrierbar
 - Automatisierung der Kernprozesse ist möglich...
 - ...hierfür ist die Integration unverzichtbar
 - Konventionen sind wichtig...
 - ...deren Einhaltung wird vom Tooling vereinfacht...
 - ...aber auch kontrolliert
 - Projektkontrolle basierend auf Issues ermöglicht...
 - ...eine hohe Informationsdichte und –vernetzung in allen Tools
 - Projektstatus ist nicht nur für den Controller wichtig (Wallboards)
- Aber: Tooling alleine reicht nicht, der Prozess muss gelebt werden!

Implementierung von Konzepten

- Bisheriger Fokus: Die technische Infrastruktur
- Build-Systeme besitzen aber keinen Selbstzweck...
- ...sie implementieren Konzepte



- Auslieferung eines Produktes bedarf mehrerer Prozessschritte
- Ein Build-System ermöglicht nur einen Teil davon...



- ...und sollte ein elementarer Teil des Deployments sein
 - Deployment Pipeline als Fortsetzung des CI-Konzeptes – Dave Farley 2007
 - Continuous Delivery als ganzheitliche Sicht – <http://continuousdelivery.com>

- Ein Build-System muss immer an die Gegebenheiten adaptiert werden
 - Vorgehen, Teamgröße, Projektgröße, Know-How, Organisation, ...
- Es lohnt der Blick ins Detail, die Herausforderungen warten!



- *Stark verteilte Entwicklung?*
Verteiltes Versionsmanagement
Dr. Ralph Guderlei



- *Testlaufzeit zu groß?*
Die Cloud - ein perfekter Lebensraum für Hudson
Arnd Kleinbeck



- *Welches Build-Tool?*
Maven Magie für Muggels
Gunther Popp

Mehr von OIO zum Thema

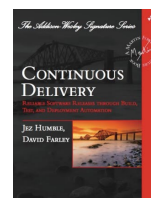
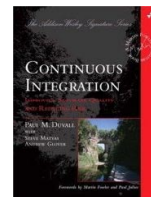
- Atlassian Jira - Issue Tracker
 - <http://www.oio.de/software-factory/tools-agile-software-entwicklung/jira/atlassian-jira-agil-preise-euro.htm>
- Schulung: Jira - Fachliche Administration
 - <http://www.oio.de/seminar/methodik-prozess-management-soft-skills/seminar-training-atlassian-jira-schulung.htm>
- Schulung: Entwicklung mit Eclipse
 - <http://www.oio.de/seminar/open-source/eclipse-schulung.htm>
- Schulung: Versionsverwaltung mit Subversion
 - <http://www.oio.de/subversion-svn-schulung.htm>
- Schulung: Das Buildtool Apache Maven
 - <http://www.oio.de/maven-schulung.htm>

Links

- **Continuous Integration**
Improving Software Quality and Reducing Risk

Paul M. Duvall, Steve Matyas, Andrew Glover
ISBN 0321336380
- **Continuous Delivery**
Reliable Software Releases through Build, Test, and Deployment Automation

Jez Humble, David Farley
ISBN 0321601912



Steffen Schluff

Trainer, Berater, Entwickler



Schwerpunkte
*Open Source Tooling
Build Management
Refactoring*

Björn Feustel

Trainer, Berater, Entwickler



Schwerpunkte
*Build- und Konfigurationsmanagement
Systemarchitekturen
Requirements-Engineering*



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



<http://www.publicdomainpictures.net/>