



Von Continuous Integration zu Continuous Delivery



Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Steffen Schluff

Version: 1.0

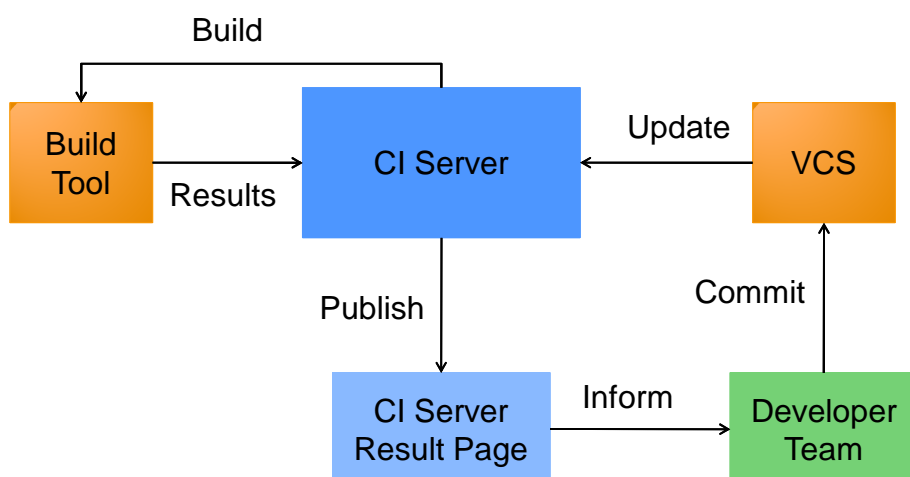
Gliederung

- Einleitung
- Continuous Delivery
- DevOps
- Zusammenfassung

Gliederung

- Einleitung
- Continuous Delivery
- DevOps
- Zusammenfassung

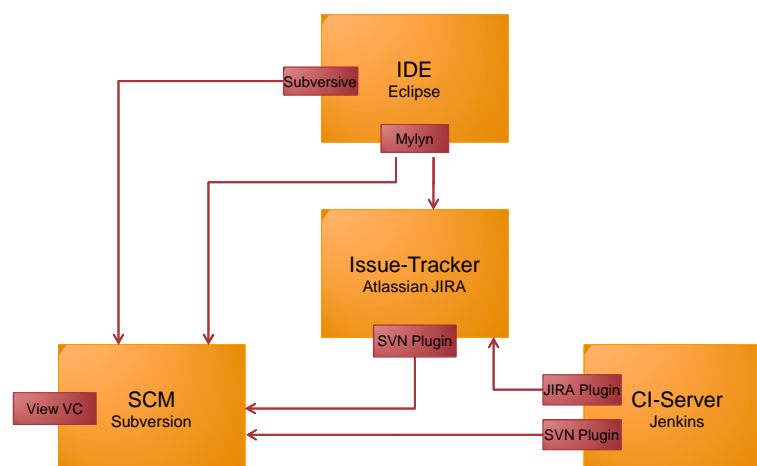
Been there, done that (1)



Been there, done that (2)

- „Daily Build and Smoke Tests“ sind schon ein alter Hut
 - Erste Veröffentlichung von Steve McConnell im Jahre 1996
 - Thema war bereits davor schon bekannt
- „Continuous Integration“ Artikel von Martin Fowler im Jahre 2000
 - Themenbereich bekam einen klingendem Namen
 - Definition „Key Practices“ (Automate the build, Make it self-testing, ...)
 - Erste Bereitstellung von „fertigen“ Tools (CruiseControl)
- „Continuous Integration“ gehört heute zum guten Ton
 - Wahlfreiheit zwischen diversen Servern (Jenkins, Hudson, Bamboo, ...)
 - Definition von Best Practices, Patterns und Anti-Patterns
 - Probleme der zweiten Generation: Testlaufzeiten, Virtualisierung, ...

Entwickler Kosmos



Da war doch noch was? (1)



Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

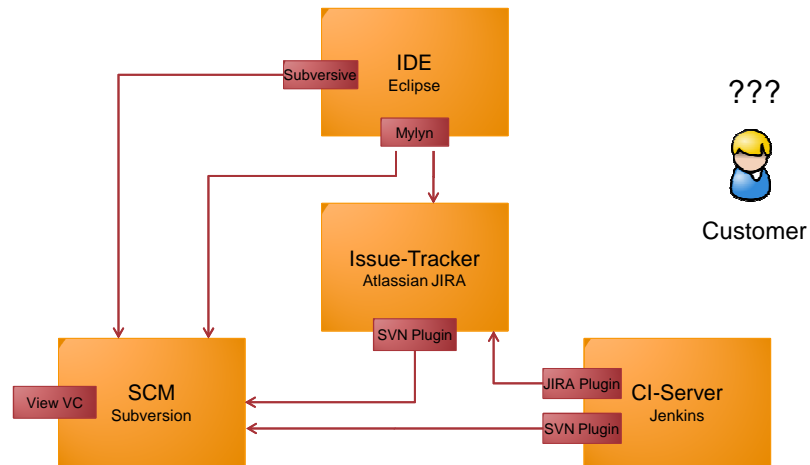
Da war doch noch was? (2)



Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.

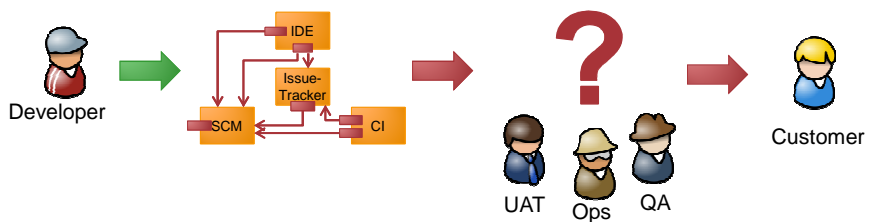
First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

Ha Ha Only Serious



Continuous Integration – und wie weiter?

- Die Build-Infrastruktur steht, ...
- der CI Server zeigt grün, ...
- und der Kunde wartet.
- Erfolgreiches Commit != Auslieferung in Produktion
- CI ist fokussiert auf Entwicklung ...
- und nicht auf Bereitstellung zum Testen oder Produktivsetzung

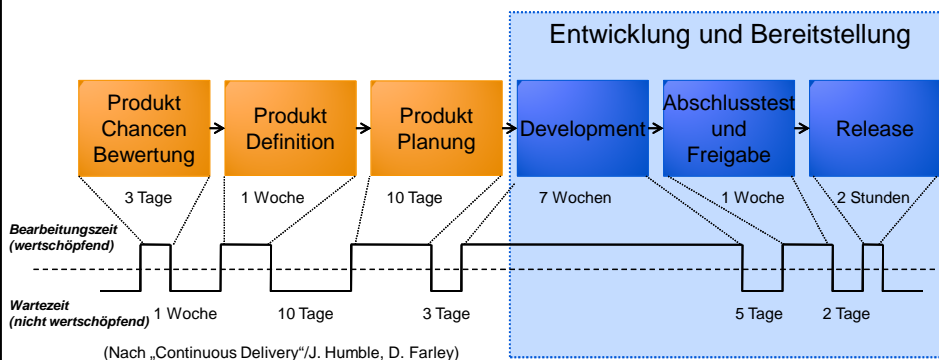


Don't do that then!

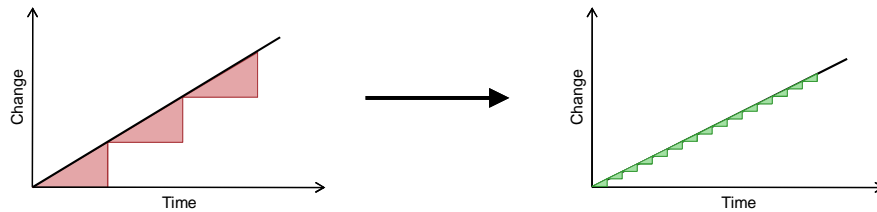
- Kunde möchte Verfügbarkeit seiner Funktionalität
 - Kein Interesse, ob CI Server rot, grün, gelb oder blau ist.
- Zwischen gutem CI Build und Kunden Verfügbarkeit liegt Release
 - Release Schritt oft nicht so gut beherrscht wie CI Ökosystem
- Release Modell „Big Bang“™ (alias der Klassiker)
 - Manuell, Zeitintensiv, Kompliziert, viele Beteiligte, Fehleranfällig
- Seltene Releases als Konsequenz
 - Frustrierend für den Kunden
 - Zugleich hoher Stressfaktor bei ungeplanten Releases (Hotfixes)

Gut Ding will Weile haben

- Dauer einer Kunden Idee bis Produktivsetzung („Concept to Cash“)
- Visualisierung als „Value Stream Map“



Stay on target



"[...] the ability to rapidly, reliably and repeatedly push out enhancements and bug fixes to customers at low risk and with minimal manual overhead."

(http://en.wikipedia.org/wiki/Continuous_delivery)

Buzzword (1)

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

Buzzword (2)

Our highest priority is to satisfy the customer through early and **continuous delivery** of valuable software.

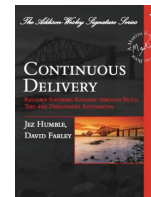
First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

Gliederung

- Einleitung
- **Continuous Delivery**
- DevOps
- Zusammenfassung

Continuous Delivery

- „*Continuous Delivery is not Continuous Integration. Continuous Delivery is being in the position to ship your product whenever you want, day or night.*“ (Neal Ford)
- Frühere Begriffsverwendung und Wurzeln
 - „Agile Manifesto“ (2001)
 - „Deployment Pipeline“ (2004/2005)
 - „Continuous Deployment“ (2009)
- Gleichnamiges Buch von Jez Humble & David Farley
 - Eigentliche Begriffsprägung
- Schwerpunktthemen „Automation“ und „Collaboration“



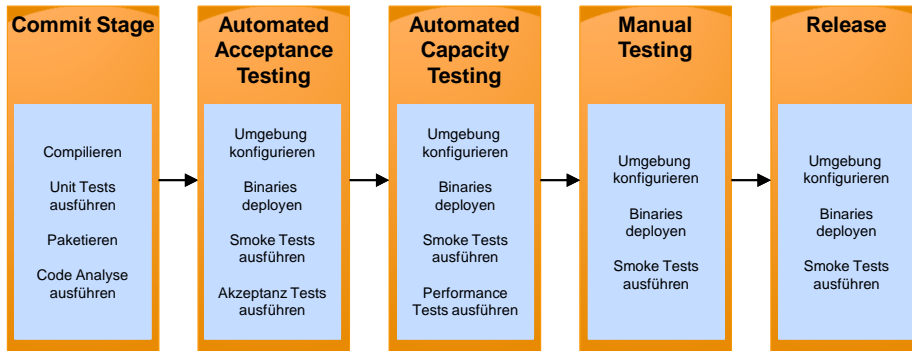
Continuous Delivery – Kerngedanken

- “Create a Repeatable, Reliable Process for Releasing Software”
- “If It Hurts, Do It More Frequently, and Bring the Pain Forward”
- “Everybody Is Responsible for the Delivery Process”
- “Keep Everything in Version Control”
- “Automate Almost Everything”
- “Done Means Released”

(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Ein erster Blick

- Zentrale Abstraktion „Deployment Pipeline“
 - Visualisierung aller Prozesssteile für alle Beteiligten
 - Verbessertes Feedback während der Ausführung
 - Möglichkeit eines vollautomatischen Releases in alle Umgebungen

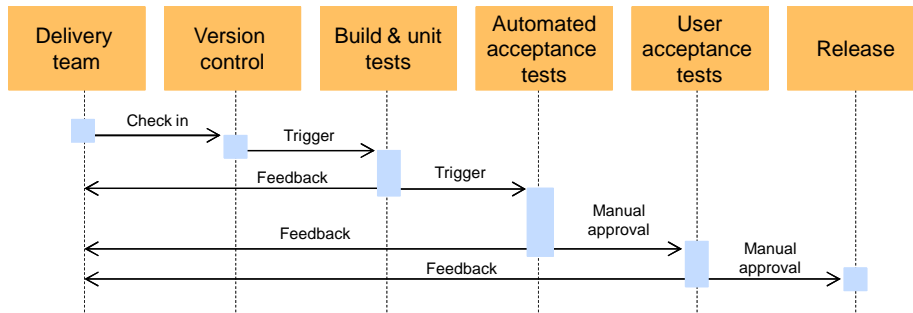


(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Bestandteile

- Die **Deployment Pipeline**
 - Macht Status der Produktentwicklung sichtbar
 - Liefert Feedback zu jeder Änderung
 - Technisch-konzeptuelle Basis des Release Prozesses
- Die Pipeline besteht aus einer Folge von **Stages**
 - Commit Stage als zentrales Eingangs-Gate
 - Typische Stages: UAT, Performance Tests, Production Deployment
 - Stages verbunden durch Trigger (automatisch oder manuell)
- **Jobs** sind die Bausteine der Stages
 - „Unit of Work“
 - Bestehen aus Tasks wie Build, Deploy, Copy, Test, ...

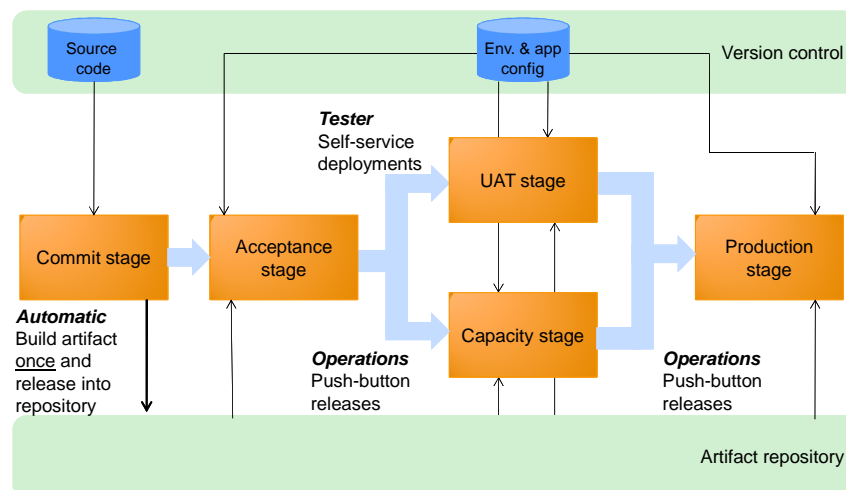
Deployment Pipeline – Sequenzdiagramm



- Jede Ressourcen Änderung startet neue Pipeline Instanz
- Erste Stage produziert alle Artefakte
- Durchlaufen aller Stages bis Fehlschlag („Stop the line“) oder ...
- Pipeline Ende erreicht ist (Letzte Stage führt Deployment aus)

(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Ein zweiter Blick



(Nach „Continuous Delivery“/J. Humble, D. Farley)

Continuous Delivery – Prinzipien (1)



- Fortlaufende Optimierung
 - In Verantwortung aller Beteiligten (Development, Operations, ...)
- Artefakte
 - Werden **einmal** gebaut und in einem Artefakt Repository verwaltet...
 - und allen Stages zur Verfügung gestellt
 - Ziel ist identisches Deployment in allen Umgebungen
 - Umgebungsspezifika durch eigene Konfigurationen
- Configuration-Management
 - Basis für einmalig erstellte Artefakte
 - Umfasst Software und Infrastruktur („Infrastructure as code“)
 - Konfigurationen werden versioniert

Continuous Delivery – Prinzipien (2)



- Automatisierung
 - So umfangreich wie möglich
 - Umfasst auch alle Aspekte der Infrastruktur (inklusive OS)
 - Prägung durch Development und Operations
- Tests
 - Basis für Automatisierung und Pipeline Processing
 - Geben Sicherheit für erfolgreiche Änderungen
 - Smoke Tests speziell für Deployment
- Monitoring
 - Basis für fortlaufende Optimierung
 - Ermöglicht Feedback für Operating (vgl. Code-Metriken für Developer)

Alle Theorie ist grau



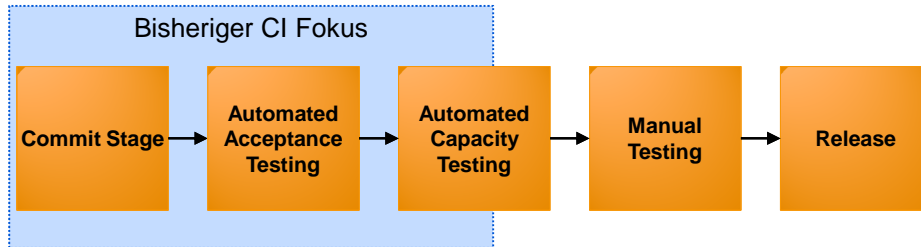
- Jedes Projekt hat in der Praxis Spezialitäten
 - Art und Größe des Produktes (Web-App vs. Standalone)
 - Komplexität des Release Prozesses
 - Technischer Rahmen (Programmiersprache, OS, Browser)
- Somit sind Continuous Delivery Implementierungen individuell
- Es gibt nicht **das** Continuous Delivery Tool
- Erster Impuls oft selbstgemachte Lösungen („Home grown“)
 - Kosten-Nutzen Verhältnis häufig schlecht
 - Veralten in der Regel nach Initialerfolg während des Betriebs
 - Oft auf Einzelpersonen ausgerichtet („Job security“)

Continuous Delivery – Tooling (1)



- *„The deployment pipeline has its foundation in the process of continuous integration and is in essence the principles of continuous integration taken to its logical conclusion.“*
- Was für die Prinzipien gut ist, kann für die Praxis nicht schlecht sein
- Continuous Integration Server wird zum Continuous Delivery Server
- CI Server werden bereits für alle möglichen Projekt Arten eingesetzt

Continuous Delivery – Tooling (2)



- Bestehendes Tooling für weitere Stages bereits vorhanden
 - Artefakt Repositories (CI-Server eigene Repos, Maven, ...)
 - „Infrastructure as code“ (Puppet, Chef, ...)
- Was CI Servern fehlt ist Umsetzung von „Pipeline“, „Stages“, „Jobs“

Demonstration

- Tool Beispiel



Gliederung

- Einleitung
- Continuous Delivery
- **DevOps**
- Zusammenfassung

Die Tücke im Detail

- Continuous Delivery ist Fortsetzung von Continuous Integration
 - Continuous Integration Tooling ist bekannt und etabliert
- Aber: Jetzt nicht mehr nur Development beteiligt, sondern auch...
 - Quality Assurance in Stage „Manual Testing“
 - Operations in Stage „Release“
- Problem wird „verschärft“ durch „neue“ Methoden und Technologien
 - Agile Projekte releasen häufiger als Wasserfall Projekte
 - Verfügbarkeit von Server Hardware stetig wachsend (Cloud)
 - Bisheriger Release Stress und Sockelkosten nicht mehr tragbar
- Beteiligte Gruppen müssen stärker zusammenrücken
 - Betrifft vor allem Development und Operations

First things first

- Neues Buzzword notwendig: *DevOps*
- Portmanteau (Kofferwort) aus „Development“ und „Operations“
- Erste Verwendung „DevOpsDays 2009“ in Belgien

DevOps [...] is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) professionals.

(<http://en.wikipedia.org/wiki/Devops>)

Houston, we have a problem (1)

- Zwei beteiligte Parteien mit gegenläufigen Zielen
 - Development will neue Features (Change)
 - Operations will hohe und schnelle Verfügbarkeit (Stability)
 - Konkurrenz, da häufig keine übergreifende Sicht in Unternehmen
- Trennung durch „Wall of Confusion“
 - Unterschiedliche Ziele und unterschiedliche Tools
 - „Wir werfen Operations dann den nächsten Release über den Zaun“



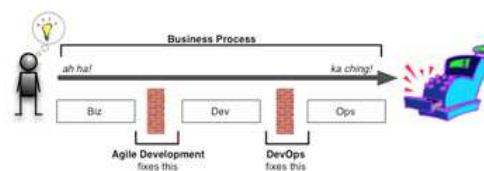
(Quelle: <http://dev2ops.org/2010/02/what-is-devops/>)

Houston, we have a problem (2)

- Nach fehlgeschlagenen Release spielen alle „The Blame Game“
 - *Ops*: Artefakte, Skripte, Config Files, ... waren fehlerhaft
 - *Dev*: Bei uns hat es funktioniert, ihr habt was falsch gemacht
 - *Ops*: Ihr müsst selber drauf schauen, was nicht stimmt
 - *Dev*: Wir kommen doch gar nicht auf die Prod Maschinen (usw.)
- In der Zwischenzeit kann niemand arbeiten
 - Fachabteilung ist es egal, ob Development oder Operations schuld
- Rollback eines teilweise durchgeführten Releases oft unmöglich
 - Releases nicht atomar, häufig im Bereich Datenbankänderungen
- Releases wird als Konsequenz „irgendwie“ lauffähig gemacht
 - Oft „Dirty Hacks“ ohne Lerneffekt dafür mit versteckten Fehlern

Déjà vu

- „Wall of Confusion“ ist nicht neu
- Frühere Trennung zwischen Business und Development
- Lösung war agile Entwicklung
- DevOps soll gleiches für Development und Operations leisten



(Quelle: <http://dev2ops.org/2010/02/what-is-devops/>)

DevOps Prinzipien – Development



- Done means released
- Infrastructure as code
- Version control everything
- Frequent releases
- Test end to end (i.e. code and infrastructure)
- Instrument operations

DevOps Prinzipien – Operations



- Automate where possible
- Instrument pervasively (to detect trends early)
- If anything fails stop the line
- If it's hard do it more often
- Enable graceful degradation (some is better than none)
- Reprovision not repair (i.e. recover to a known state)

Patterns again, ...



- Entstehende Pattern Ansätze im DevOps Sektor
 - „Agile DevOps“ und „Automation for the people“ Reihe von Paul Duvall
- Scripted environments
 - Automatisierte und versionierte Bereitstellung einer Server Umgebung
 - Server Umgebung ist Teil des Deployments auf Produktion
 - Mögliche Tools: Chef oder Puppet
- Test-driven infrastructures
 - Wenn Infrastruktur Code ist, dann muss Infrastruktur getestet werden
 - Testet ob alle Bestandteile der Infrastruktur verfügbar sind
 - Beispiel: Test, ob Apache in richtiger Version läuft

... again ...



- Chaos Monkey
 - „*Everything fails, all the time*“ (Werner Vogels)
 - Terminiert regelmäßig Instanzen in einer Gruppe von Systemen und ...
 - Testet indirekt, ob das System trotzdem weiterläuft
 - Schlägt in kontrollierten Zeiten zu, um für den Ernstfall bereit zu sein
- Transient environments
 - Umgebungen sind so kurzlebig wie möglich (Stunden bis Tage)
 - Keine „heiligen“ Server mehr, mit nicht reproduzierbarer Konfiguration
 - Forciert Konzept, dass alles automatisiert sein muss

... and again.

- Version everything
 - Ziel ist Etablierung einer „single source of truth“
 - Maximal ein Checkout zum Loslegen für einen neuen Entwickler
- Delivery Pipeline
 - Vergleiche Deployment Pipeline aus Continuous Delivery
- DevOps Dashboard
 - Anzeige wie Änderungen das System in welcher Stage wie beeinflussen
 - Für alle beteiligten Gruppen zugänglich
 - Häufig im Continuous Integration Server verankert

Frage des Standpunkts

- Continuous Delivery und DevOps sind eng verwandt
 - Ähnliche Prinzipien und Kernbegriffe wie Deployment Pipeline
- Keine gegenseitige Ignoranz sondern vielmehr gleiches Ziel
 - Veröffentlichungen beziehen sich auf den jeweils anderen Begriff
- „Flattening“ von Release Prozessen und Organisationsstrukturen
 - Wunsch sind schnellere, billigere und bessere Releases

Gliederung

- Einleitung
- Continuous Delivery
- DevOps
- **Zusammenfassung**

Zusammenfassung

- Kundenzufriedenheit erfordert Auslieferung von Software
 - Klassische Release Modelle passen nicht zu moderner Entwicklung
- Continuous Delivery ist logische Folge von Continuous Integration
 - CI Server unterstützen jetzt auch Deployment Pipelines
- Auslieferung erfordert Team übergreifende Zusammenarbeit
 - Vor allem zwischen Development und Operations
- Zwischen Development und Operations steht „Wall of Confusion“
 - DevOps will diese Wand mit Methoden und Verfahren überwinden

If you remember one thing



*“Computers are designed to do simple repetitive tasks.
The second you have humans doing repetitive tasks,
all the computers get together late at night and laugh
at you”*

(Neal Ford)

Mehr von OIO zum Thema...



- Schulung: Versionsverwaltung mit Git
 - <http://www.oio.de/git-schulung-versionsverwaltung-seminar-dvcs-training.htm>
- Schulung: Versionsverwaltung mit Subversion
 - <http://www.oio.de/subversion-svn-schulung.htm>
- Schulung: Hudson Grundlagen
 - <http://www.oio.de/schulung-hudson-seminar-continuous-integration-training-jenkins.htm>
- Schulung: JIRA – Fachliche Administration
 - <http://www.oio.de/seminar/methodik-prozess-management-soft-skills/seminar-training-atlassian-jira-schulung.htm>

Mehr von OIO zum Thema...



- Artikel aus dem Java Spektrum:
Optimiertes Testen (PDF)
 - http://www.oio.de/public/softwaretest/optimiertes-testen-gateways-gatekeeper-keymaster_JS_03_11.pdf
- Beratung zu Open Source Tools:
 - <http://www.oio.de/beratung-consulting/open-source-software/tools/>

Ihr Sprecher



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