Toolunterstützung

Es gibt verschiedenste Tools zur Unterstützung der Konfigurationsprozesse.

Hier werden einige Open Source Tools vorgestellt, die weit verbreitet sind:

- **Subversion, GIT** - Versionskontrolle
- **Maven , Nexus** - Build Prozess
- **Jenkins** – Continuous Integration
- **Redmine** – Kollaboration

Integration

Mehrere Leute im Team, mehrere Komponenten

➔ Notwendigkeit, die Arbeit zusammenzuführen

Integrations Probleme

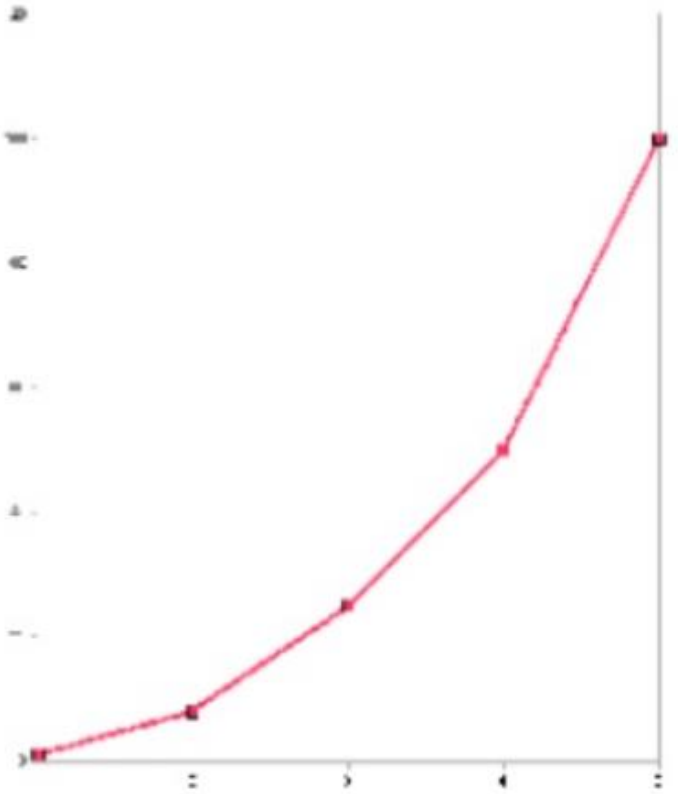
- **Merge Konflikte:**
  Das selbe File wird von verschiedenen Leuten bearbeitet.

- **Compile Konflikte**
  Unterschiedliche Files werden so bearbeitet, dass das System nicht mehr kompiliert.

- **Test Konflikte**
  Unterschiedliche Files werden so bearbeitet, dass das System zwar kompiliert, aber nicht mehr korrekt läuft.

Integration

- Integration ist aufwändig.
- Der Aufwand steigt exponentiell mit
  - Mit der Anzahl der Fehler
  - Mit der Anzahl der Komponenten
  - **Mit der Zeit seit der letzten Integration**



Idee der Continuous Integration

- Anstelle von großen und langdauernden Integrationsphasen: Häufige, kurze Integrationen.
- Integrationen sollen den Entwicklungsprozess nicht stören.
- Herkunft: Extreme Programming
  Martin Fowler:
  http://www.martinfowler.com/articles/continuousIntegration.html

Annahmen

- **“Assumption is the mother of all screw-ups.”**
  - *Wethern's Law of Suspended Judgment*
- *CI reduziert die Annahmen, indem bei jeder Änderung die Software neu gebaut wird.*

http://www.javaworld.com/article/2077731/build-ci-sdlc/introducing-continuous-integration.html

CI Workflow

- Auschecken
- Kodieren
- Automatisierter Build auf lokaler Maschine bis Tests grün sind.
- Merge (lokal) mit den letzten Änderungen bis Tests grün sind.
- Commit
- **Automatisierter Build auf sauberer Integrationsmaschine.**
- Ggfs. sofortiger Bugfix

Vorteile von CI

Drei ganz wesentliche Vorteile:

- Bugs werden schneller gefunden
- Risiko minimiert (keine technischen Schulden während der Entwicklung)
- Häufiges Deployment möglich➔ schnelles Benutzer Feedback

Weitere Vorteile der CI

- Automatisierte CI schafft Ressourcen für die wirklich kreative Arbeit.
- Weniger Widerstand gegenüber Änderungen, weil wiederholbare Tätigkeiten automatisiert sind.
- Deploybare Software zu jeder Zeit generierbar.
- Bessere Sichtbarkeit des Projekts
  - Just in Time Informationen zum letzten build und zu Qualitätsmetriken
  - Erkennbare Trends
- Vertrauen in das entwickelte Produkt

Was gehört zu CI?

Mehr als „continuous compilation“:

- Testen
- Test Metriken
- Code Metriken
- Speichern der Software Artefakte
- Publikation der Ergebnisse der Tests
- Publikation der Ergebnisse der Metriken
- Projekt Homepage

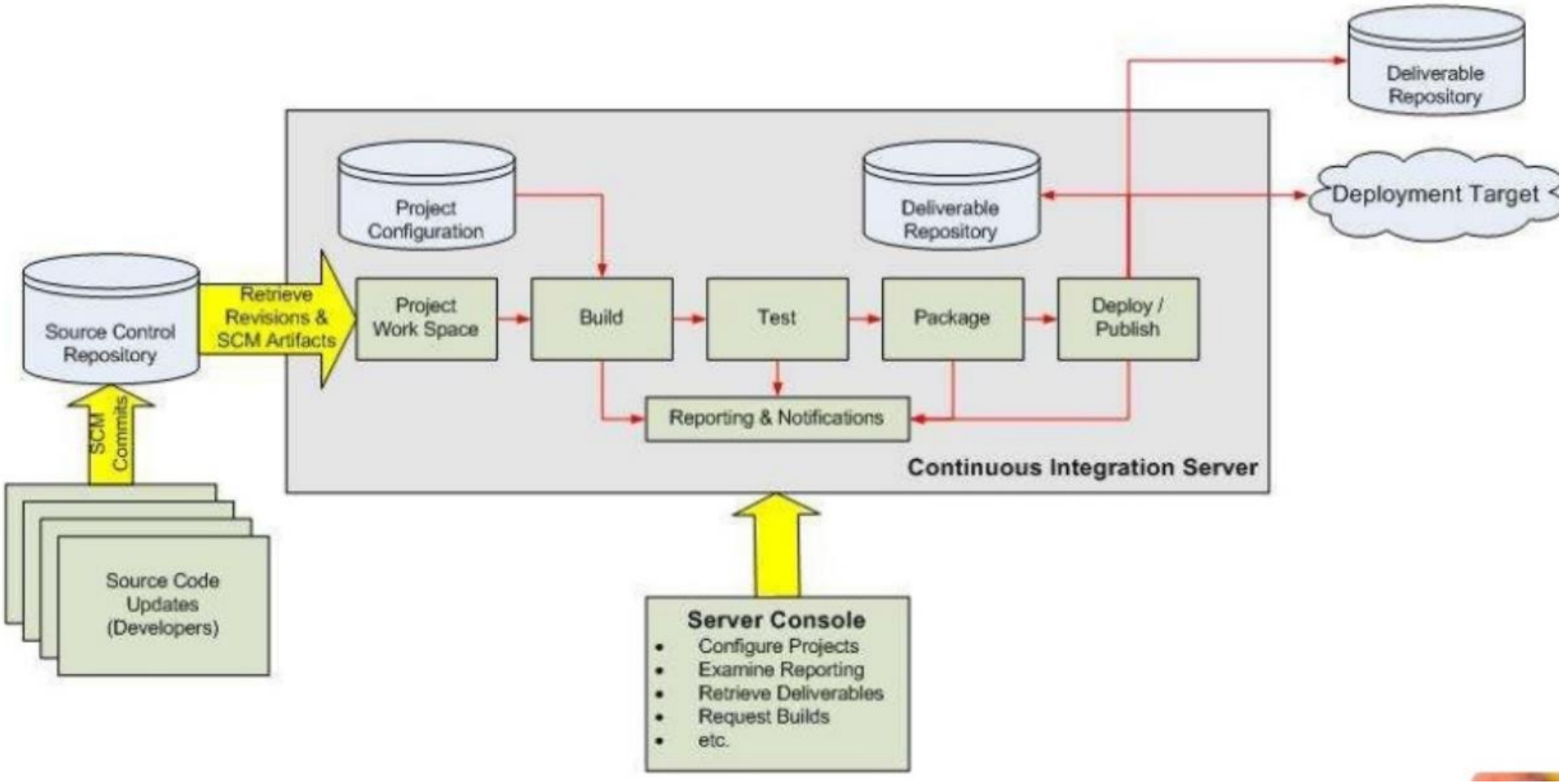
- ….

Best Practices der CI nach Fowler

1. Maintain a Single Source Repository
2. Automate the Build
3. Make Your Build Self-Testing
4. Everyone Commits Every Day
5. Every Commit Should Build the Mainline on an Integration Machine
6. Fix Broken Builds immediately
7. Keep the Build Fast
8. Test in a Clone of the Production Environment
9. Make it Easy for Anyone to Get the Latest Executable
10. Everyone can see what's happening
11. Automate Deployment

http://www.martinfowler.com/articles/continuousIntegration.html

Continuous Integration



http://www.javaworld.com/article/2077956/open-source-tools/continuous-integration-with-hudson.html

Tool Unterstützung bei der CI

- Maven könnte in einem ersten Schritt mithilfe eines Plugins die Sourcen aus dem VCS holen und dann automatisch den Build bauen.
- **Problem:** Auch das POM, das diese Aktion steuert, liegt im VCS.
- ➔ Der Abgleich mit dem VCS muss vor dem Aufruf von Maven erfolgen.

Build Server, z.B. Hudson http://hudson-ci.org/ helfen bei der CI und lösen zudem das Henne-Ei Problem von Maven.

Referenzen

- http://www.javaworld.com/article/2077731/build-ci-sdlc/introducing-continuous-integration.html
- http://www.javaworld.com/article/2077956/open-source-tools/continuous-integration-with-hudson.html