SIMPL

-

SimTech: Information Management, Processes and Languages

Softwareinfrastruktur für SIMPL

René Rehn

Agenda

- **1. Die Datenbank – IBM DB2 9**
- **2. Die Workflowumgebung**
- **3. Entwicklungstools**
- **4. Fazit zur Softwareinfrastruktur**
- **5. Große Softwareprojekte**

1. Die Datenbank - IBM DB2 9

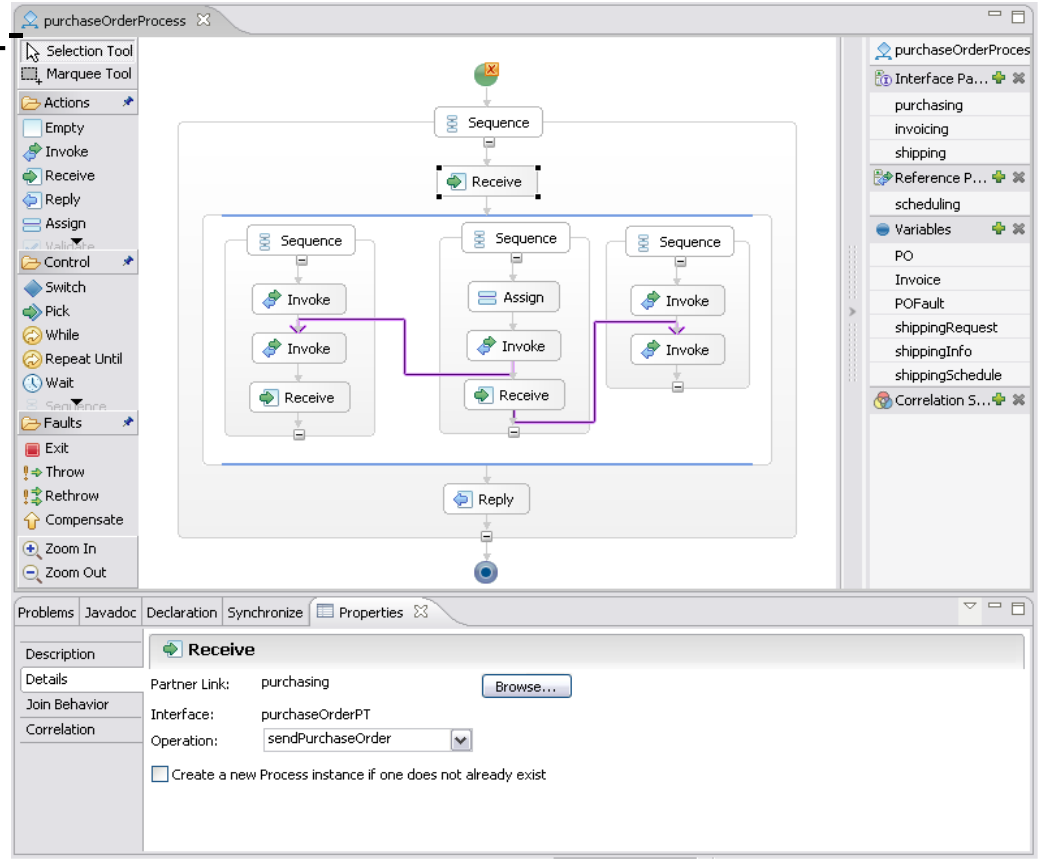
- Hybriddatenserver zur Verwaltung von relationalen und XML-Daten
- XML -Daten
  - XML-Dokumente werden direkt im XML-Format abgelegt
  - Integrität wird überprüft und bleibt bestehen
- relationale Daten (SQL)
  - neue SQL Funktionen wurden eingeführt :
    - Mengenoperation und Merge-Statements

2. Die Workflowumgebung (1)

- Apache ODE
- BPEL-fähige Workflow-Engine
  - Unterstützung von WS-BPEL 2.0 als auch BPEL4WS 1.1
  - Ausführung von Prozessen in SOA
- Hot deployment von Prozessen ist möglich
- Analyse und Validierung von Prozessen
- Eclipse BPEL Designer
- Eclipse Plugin zur Arbeit mit BPEL Prozessen
  - Leicht verständliche GUI
  - Fehlererkennung für BPEL Prozesse
  - Schritt für Schritt Ausführung von Prozessen

Die Workflow Umgebung (2)

- Hauptfenster des Eclipse-BPEL-Designers

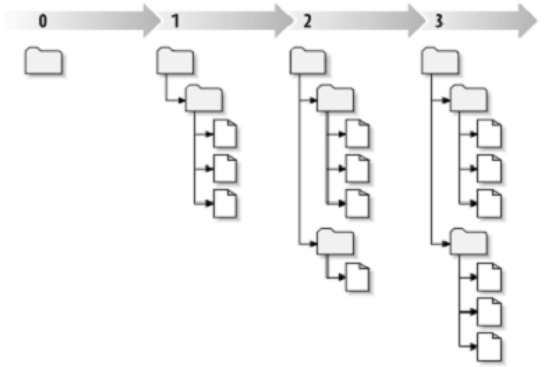


3. Die Entwicklungstools

- **3.1 Subversion**
- **3.2 Maven**
- **3.3 Hudson**

3.1 Subversion

- **Was ist Subversion**
  - Software zur Versionskontrolle
  - Weiterentwicklung von CVS
- Das Repository – Das Herzstück von Subversion
  - Ähnlich einem normalen Filesystem
    - Allerdings Speicherung aller Versionen der Daten
    - Zugriff auf alte Versionen möglich
- Ziel von Subversion:
  - Nutzer arbeiten an den selben Daten ohne sich gegenseitig zu behindern
  - Zwei Prinzipien dafür:
    - Lock-Modify-Unlock
    - Copy-Modify-Merge

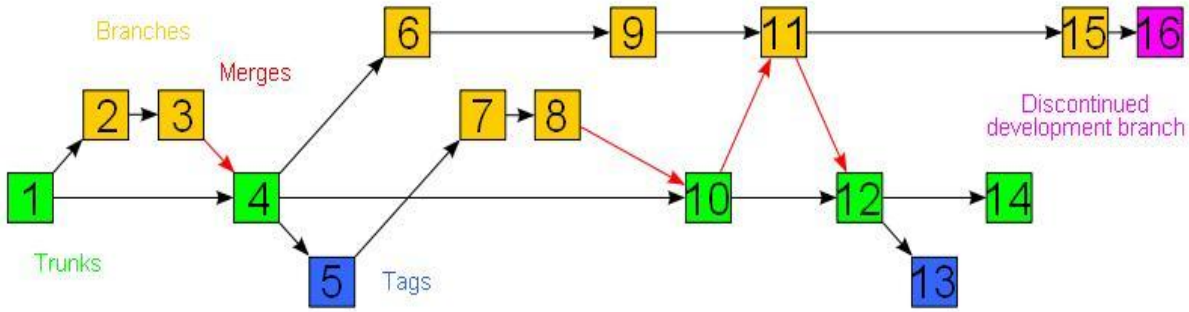


Subversion (2)

- **Tags, Branches, Trunks**
  - Trunk: Hauptentwicklung des Projektes
  - Branch: Alternative Entwicklungslinien
    - Unabhängig von der Hauptentwicklung
  - Tags: Markierte Revisionen
    - Möglichkeit auf stabile Versionen zurückzugreifen

- Die „billige Kopie“
  - Nutzen: Anlegen von Branches und Tags
  - Kein Duplikat, nur eine Verknüpfung

- Merges: Zusammenführung von Entwicklungslinien



Subversion (3)

- **Features von SVN durch Vergleich mit CVS**

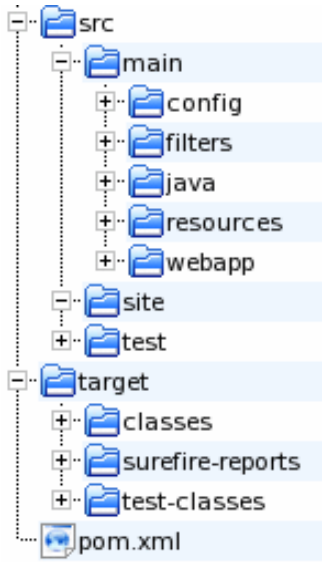
- Kennzeichnung der Repository-Inhalte
  - CVS = Datum + Uhrzeit, SVN = Revision X
  - einfacherer Zugriff auf bestimmte Revisionen
- Atomare Datenübertragung
  - Keine inkonsistenten Zustände des Repositorys

- Hinzufügen, Löschen, Kopieren und Umbennen von Daten und Verzeichnissen ist möglich
- Beim Erstellen, Aktualisieren und beim Übertragen der Änderungen in SVN wird eine Kopie der lokalen Daten angelegt
  - Enthält Zeitpunkt der Erstellung und der letzten Änderung
  - Anzeigen lokaler Änderungen ohne Repository Zugriff
    - Dadurch in SVN: Nur Änderungen müssen ins Repository übertragen werden
    - CVS: die kompletten Daten müssen übertragen werden

- Besserer Umgang mit Binärdaten
  - Auch hier nur Speicherung der Änderunge

3.2 Maven

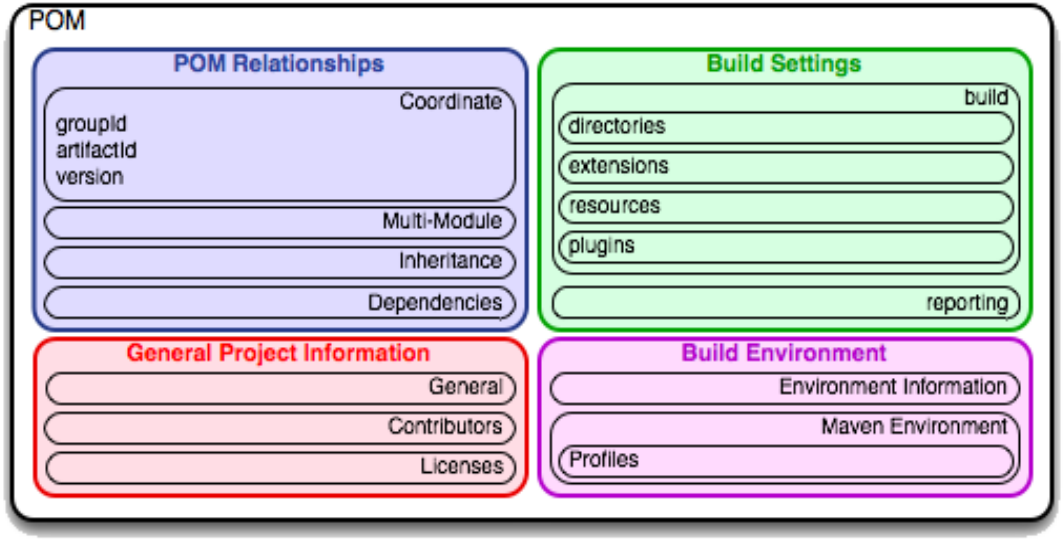
- **Was ist Maven**
  - Ein Projekt-Management-Tool
  - Enthält:
    - Ein umfassendes Projektmodell
    - Einen definierten Projekt-Lebenszyklus
    - Ein Abhängigkeits-Management-System
- Maven basiert auf einer Plugin-Architektur
  - Build eines Projektes durch Plugin-Goals
    - Plugin-Goal: Anwendung eines Plugins auf das Projekt
- Maven-Archetypen
  - Grundgerüst für ein Softwareprojekt
  - Unterschiedliche Typen: für Java, Hibernate, Portlet-Anwendungen
  - Festgelegte Verzeichnisstruktur



Maven (2)

- **Die POM.xml (Project Object Model)**
  - Enthält alle Informationen über das Projekt

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>org.sonatype.mavenbook</groupId>
<artifactId>my-project</artifactId>
<version>1.0</version>
</project>
```



Maven (3)

- **Features von Maven**
- Einfachstes Erstellen von Projekten
  - Halten an vorgegebene Standards: fast keine Konfigurationseinstellungen
- Verfügt über eine Abhängigkeitsverwaltung
  - Automatische Updates
  - Auflösen von Abhängigkeiten
- Packen von Projekten in vordefinierte Ausgabe Typen
  - JAR, WAR
- Erstellung von Web-Seiten oder PDFs zur Dokumentation
- Ein Plugin für Eclipse ist verfügbar

Maven (4)

- **Vergleich Maven – Ant**

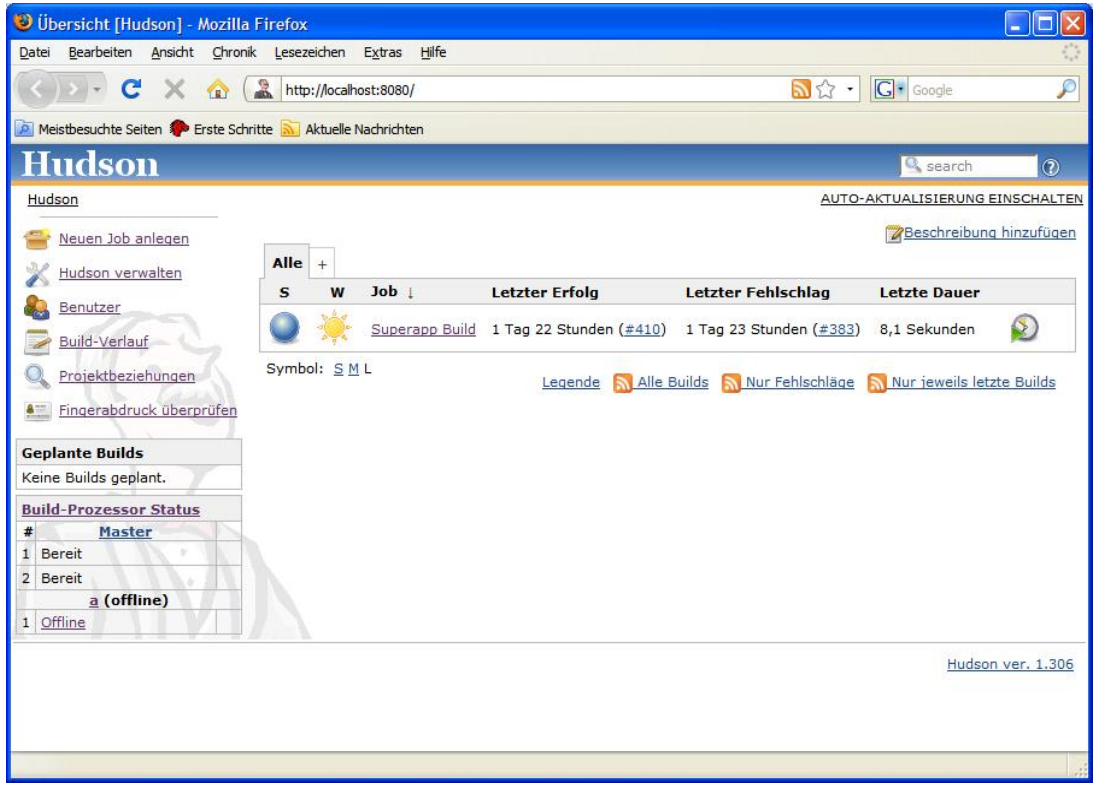
- Ant: Build System mit Targets und Abhängigkeiten
  - Target: Anweisungen in XML

- Nutzung von Ant zum Build von Projekten:
  - Es muss alles bis ins kleinste Detail definiert werden
  - Spezifische Anweisungen zum Packaging notwendig
  - Angabe wo sich der Quellcode befindet und wo Zieldaten abgelegt werden

- Maven dagegen:
  - pom.xml generieren
  - Quellcode und Ressourcen in die Maven Verzeichnisse legen
  - mvn install aufrufen

3.3 Hudson

- **Was ist Hudson**
  - Ein webbasiertes System zur kontinuierlichen Integration von Softwareprojekten
  - Kontinuierliche Integration: regelmäßige Neubildung und Testen einer Anwendung
  - KI Server benötigt: Quellcode-Repository, Build-Tool, Tests



Hudson (2)

- **Features von Hudson**

- Konfiguration direkt über die Web-GUI
- Anzeigen von Änderungen der Builds durch SVN / CVS

- Permanente Links zu den URLs „last build“ und „latest succesfull build“
- Meldungen über Erfolg oder Fehlschlag per Email, IM oder RSS

- Die Möglichkeit Abhängigkeiten im Projekt zu verfolgen
- Master / Slave Betrieb
  - Slave: anderer Rechner auf dem Builds durchgeführt werden
    - Auslastung für Builds auf mehrere Rechner aufteilen
    - Projekt in verschiedenen Umgebungen erstellen (z.B. Testumgebung)

4. Fazit zur Softwareinfrastruktur

Wir verfügen von Beginn des Projektes an über eine einheitliche Entwicklungsumgebung.

- Durch diese ist es möglich:
  - Einfaches arbeiten mit BPEL
  - Versionskontrolle in einem gemeinsamen Repository
  - Kontinuierliche und unkomplizierte Builds durch Maven und Hudson
- Keine Probleme wie uneinheitliche Entwicklungstools bei Teammitgliedern

5. Große Softwareprojekte

- **5.1 Vergleich Studienprojekt und Softwarepraktikum**
- **5.2 Probleme im Studienprojekt und Lösungen**

5.1 Vergleich Studienprojekt und Softwarepraktikum

- **Softwarepraktikum**
  - Team: 3 Mitglieder
  - Dauer: 1 Semester
  - Mittlere Programmgröße
  - → Wenig Organisationsaufwand
  - → Komplexität angemessen aber nicht zu hoch
  - → Keine Rolleneinteilung
- **Studienprojekt**
  - Team: 6 – 12 Mitglieder
  - Dauer: 2 Semester
  - Ein großes „echtes“ Programm
  - → Hoher Organisationsaufwand
  - → Komplexität ist sehr hoch
  - → Rolleneinteilung notwendig für Erfolg

5.2 Probleme im Studienprojekt und Lösungen

- Keine konsequente Rolleneinteilung im Team
- Keine klare Prozessdefinition
- Zeiteinteilung in einzelnen Phasen (z.B. Tests)

END OF DOCUMENT