Agiles Konfigurationsmanagement mit Open-Source-Werkzeugen

Gunther Popp

Gunther Popp …

- arbeitet als freier Softwarearchitekt, Berater und Autor in München
- Schwerpunkte:
  - Konzeption und Implementierung von Java EE Systemen
  - Konfigurationsmanagement
  - Coaching und Architekturberatung
- Erreichbar unter
  gpopp@guntherpopp.de
  www.guntherpopp.de

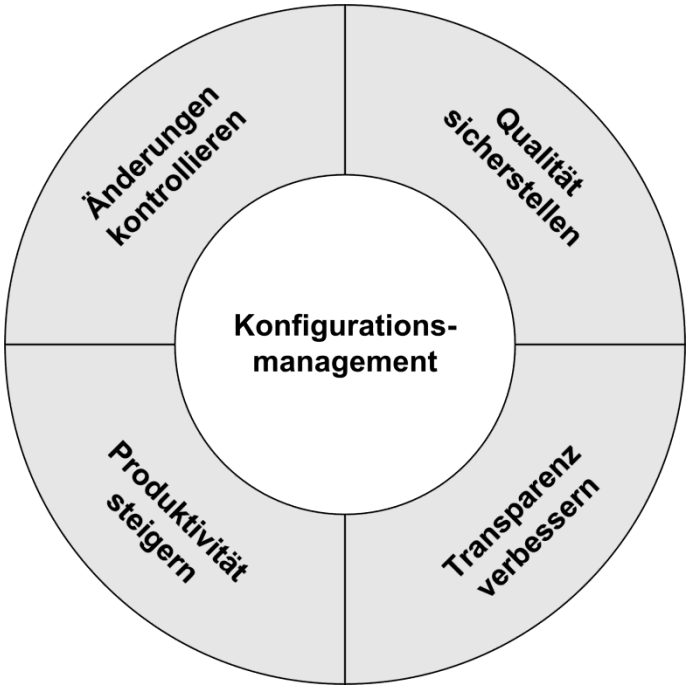
Was ist Konfigurationsmanagement?

- **Wikipedia:** „ Konfigurationsmanagement (KM) ist eine Managementdisziplin, die organisatorische und verhaltensmäßige Regeln auf den Produktlebenslauf einer Konfigurationseinheit von seiner Entwicklung über Herstellung und Betreuung anwendet“
- **Besser:** Konfigurationsmanagement hilft Teams, ihr Projektziel effizient, sicher und mit hoher Qualität zu erreichen.

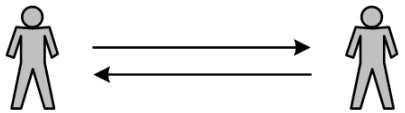
Bausteine eines agilen KM-Prozesses

- Typische Probleme in Software-Projekten
  - **Kommunikation**: Missverständnisse, keine gemeinsame Linie
  - **Effizienz**: Langes „Einschwingen“, hohe Anlaufverluste
  - **Qualität**: Willkürliche Änderungen im System, keine stabilen Releases
  - **Transparenz**: Wer arbeitet an was? Wo stehen wir?

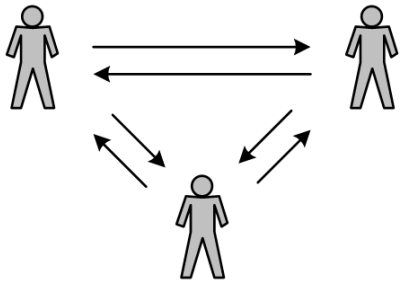
Die Bausteine eines KM-Prozesses helfen bei der Lösung dieser Probleme



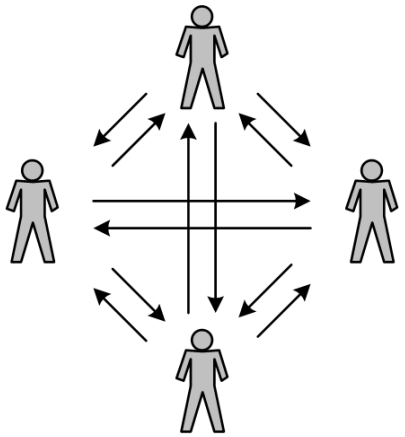
Änderungen kontrollieren



2 Personen im Team, 2 Kommunikationswege



3 Personen im Team, 6 Kommunikationswege



4 Personen im Team, 12 Kommunikationswege

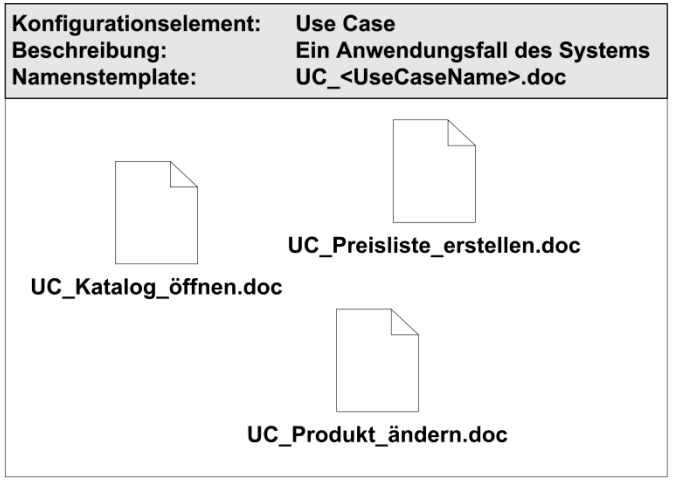
- Der Abstimmungsaufwand steigt überproportional mit der Teamgröße
- Dies führt beinahe zwangsläufig zu Kommunikationsproblemen und unkontrollierten Änderungen am System

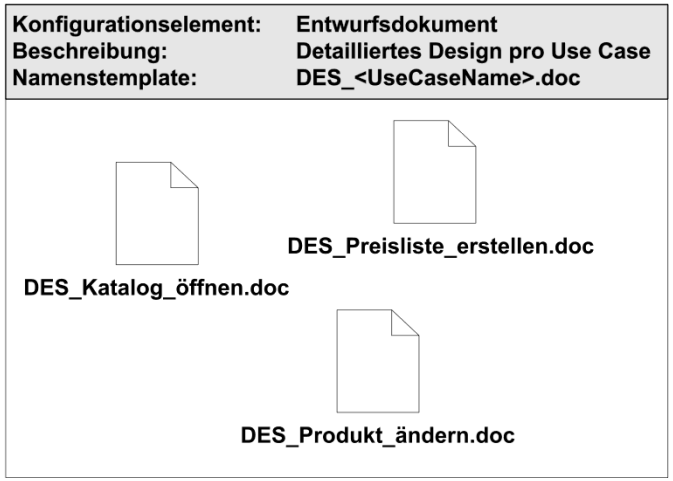
Lösungsansatz des KM-Prozesses

- Zentrale Verwaltung der Konfigurationselemente (Source-Code, Test-Cases, etc.) in Repositorys
  → Nachvollziehbarkeit von Änderungen
- Einführung eines Änderungsmanagementprozesses schon in frühen Projektphasen
  → Verhinderung von Änderungen „auf Zuruf“
- Einführung eines Collaboration-Werkzeuges (Wiki,…)
  → Verbesserung der projektinternen Kommunikation

Konfigurationselemente auswählen

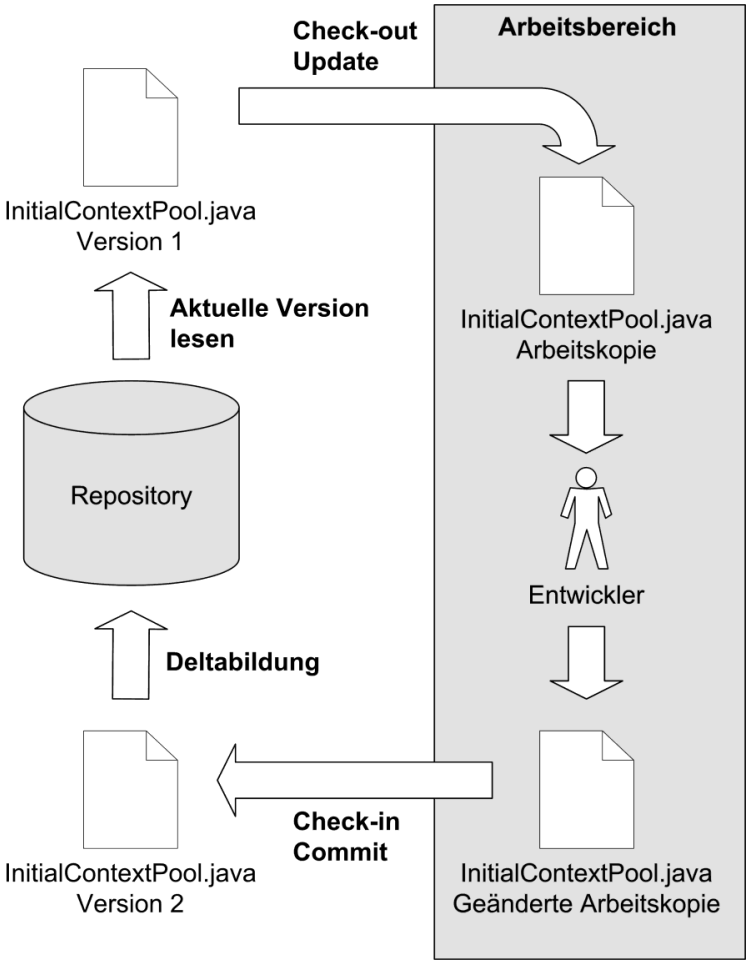
- Konfigurationselemente (KE) legen Typ und Eigenschaften einer zusammengehörenden Gruppe von Artefakten in einem Projekt fest
- Beispiele:
  - Use Cases
  - Quelltext
  - Modultests
- Ein KM-Prozess legt Richtlinien zum Umgang mit Konfigurationselementen fest:
  - Namenskonventionen
  - Gliederung (für Dokumente)
  - Coding Guidelines (für Quelltext)





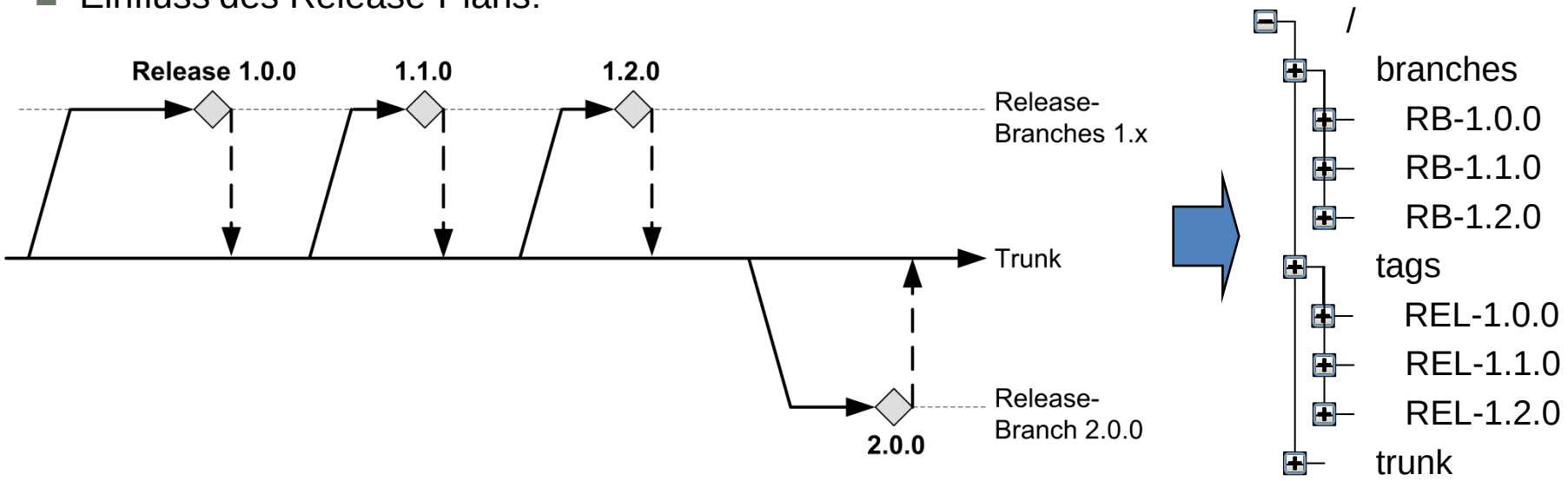
Konfigurationselemente verwalten

- Konfigurationselemente werden in einem Repository verwaltet
- Das Repository garantiert:
  - Verfügbarkeit der KEs
  - Integrität der KEs
  - Verhinderung unberechtigter Zugriffe
  - Nachvollziehbarkeit aller durchgeführten Änderungen.
- Jede Änderung an einem KE im Repository erzeugt eine neue Version
- Mit der Zeit entsteht dann eine Versionshistorie, die alle jemals an einem KE vorgenommenen Veränderungen dokumentiert



Projektstruktur festlegen (1)

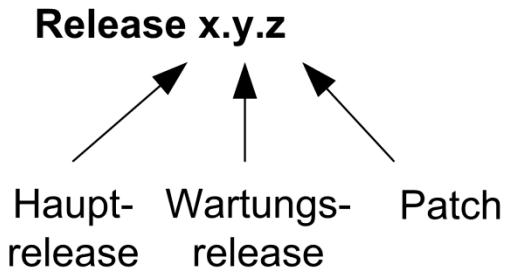
- Die Projektstruktur im Repository sollte nicht ad hoc, sondern auf Basis der im Projekt verwendeten Konfigurationselemente und der Release-Planung erstellt werden.
- Einfluss des Release-Plans:



- Um die geplanten Releases ausliefern zu können, müssen in der Projektstruktur Verzeichnisse für **Branches** und **Tags** vorgesehen werden.
- Der Hauptteil der Entwicklungsarbeit findet immer im so genannten **Trunk** (Hauptentwicklungspfad) statt.
- In den Branches werden Releases vorbereitet und Bug-Fixes bzw. kleinere Änderungen durchgeführt.

Projektstruktur festlegen (2)

- Aufbau einer Release-Nummer:

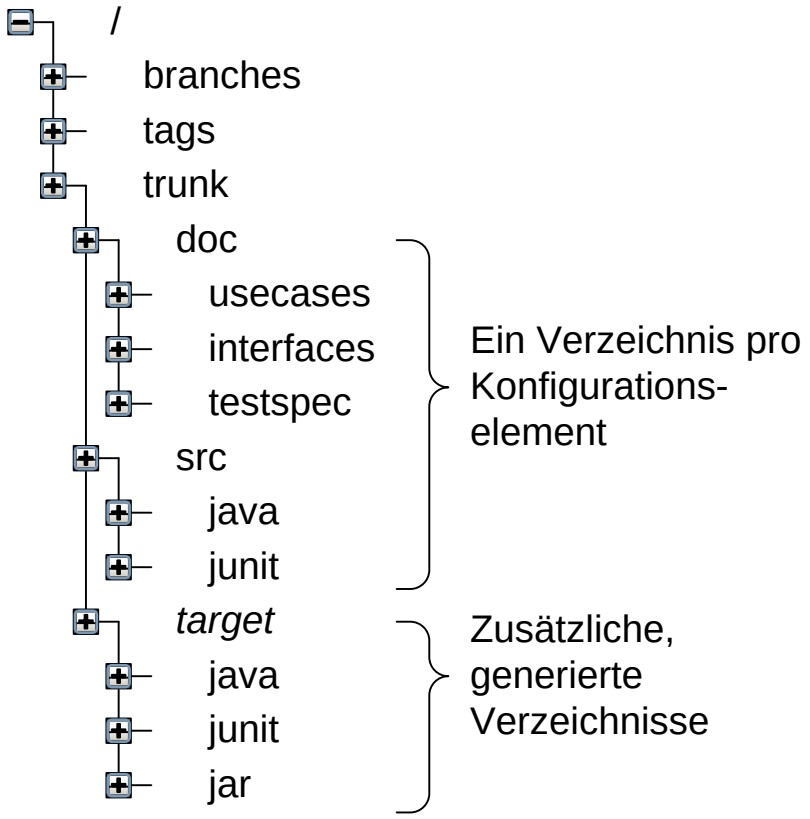


- Namenstemplates für Tags und Branches

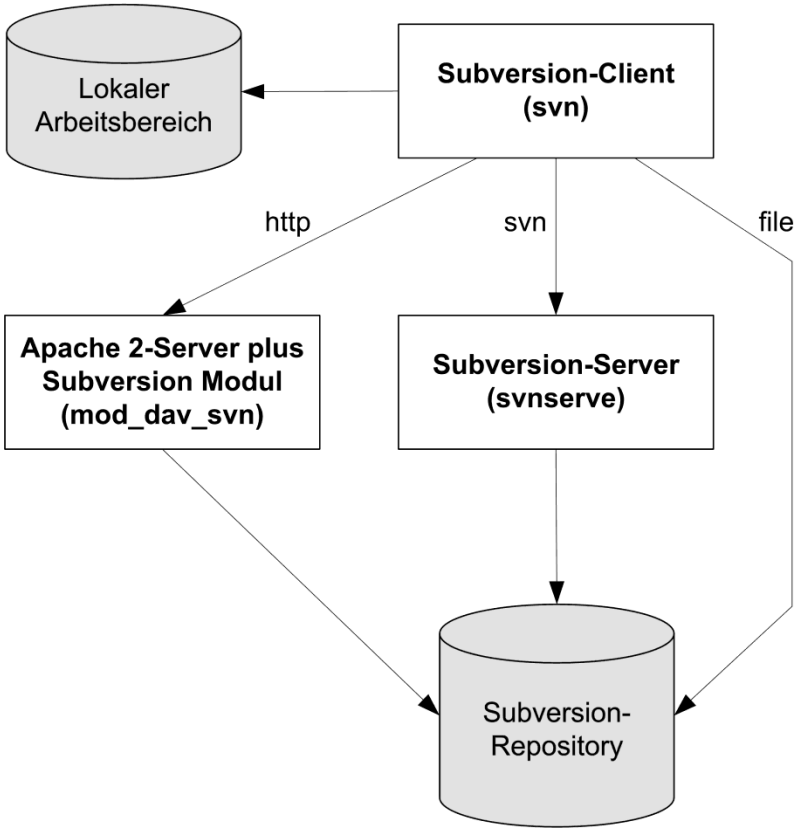
Template	Beschreibung	Beispiel
trunk	Bezeichnung des Hauptentwicklungspfads	trunk
RB-<*Release*>	Bezeichnung des Release-Branches	RB-1.0.0
REL-<*Release*>	Tag zur Fertigstellung des Releases	REL-1.0.0

Projektstruktur festlegen (3)

- Unterhalb der Verzeichnisse *branches* / *tags* / *trunk* beginnt die detaillierte Projektstruktur
- Die detaillierte Projektstruktur sollte auf folgender Basis entworfen werden:
  - Vorgaben des verwendeten Build-Tools (z.B. Maven)
  - Liste der Konfigurationselemente im Projekt
- Zusätzlich sind immer temporäre Knoten in der Projektstruktur notwendig, die nicht im Repository abgelegt werden
- Die Projektstruktur wird im KM-Handbuch dokumentiert



Subversion als Repository



- Das Werkzeug Subversion ist ideal geeignet als Repository:
  - Versionierung von Dateien und Verzeichnissen
  - Versionierung von allen KEs (auch binäre Formate)
  - Flexible Client-/Server-Architektur, auch für große Projekte geeignet
  - Gute Unterstützung für Branches und Tags
  - Viele unterschiedliche Front-Ends für die unterschiedlichen Benutzergruppen (Analysten, Entwickler, etc.)

Produktivität steigern

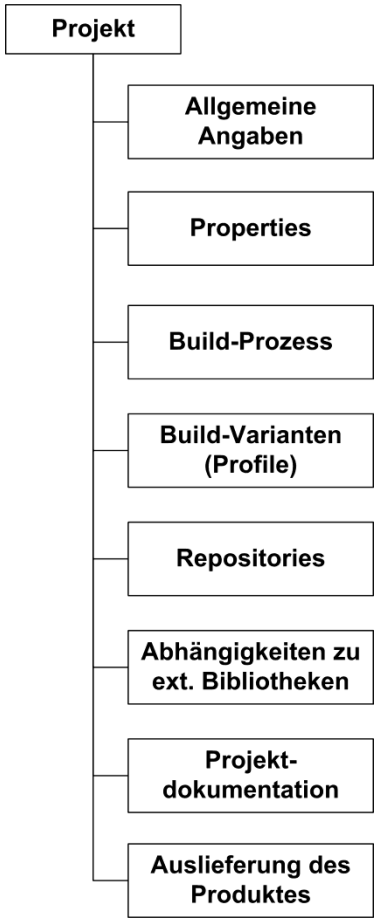
- Entwicklungsteams brauchen eine gewisse „Einschwingphase“, die sich ohne Vorgaben auch länger hinziehen kann
- Viele Aufgaben werden im Projekt manuell durchgeführt, obwohl sie automatisierbar sind
- Selbst die Produkterstellung (Build-Prozess) erfolgt oft zumindest teilweise manuell

Lösungsansatz des KM-Prozesses

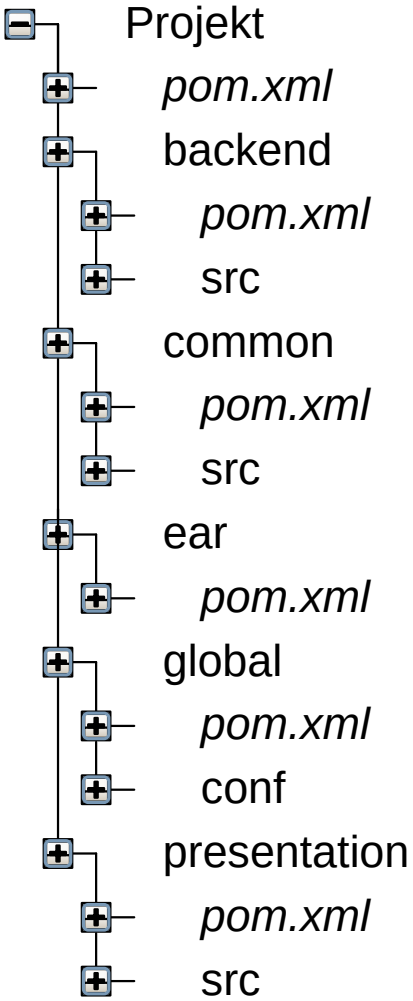
- Projektautomatisierung (→ Maven)
- Templates für Projektstrukturen (→ Maven-Archetypes)
- Erstellung eines KM-Handbuches (→ Word, Wiki)

Projektautomatisierung mit Maven

- Maven ist ein modellbasiertes Werkzeug zur Projektautomatisierung
- Zentrales Element ist das sog. POM, in dem das eigene Projekt beschrieben wird
- Es wird kein Build-Script implementiert! Maven interpretiert das POM und „generiert“ einen optimalen Build-Prozess für das Projekt
- Funktionsumfang:
  - Durchführung des Build-Prozesses
  - Verwaltung der Abhängigkeiten zu anderen Produktkomponenten
  - Erstellung der Projektdokumentation



Maven-Archetypes



- Maven implementiert von Haus aus einen einheitlichen Build-Prozess
  - Vorteil: Nur einmalige Einarbeitung der Entwickler
  - Nachteil: Hoher Aufwand für die initiale Konfiguration eines Projektes
- Ein Maven-Archetype ist ein Template, dass die grundlegende Projektstruktur und die dafür passende Maven-Konfiguration enthält
- Mögliche Inhalte eines Archetypes:
  - Vorschlag für eine bewährte Projektstruktur
  - Lauffähige Testanwendung für einen bestimmten Applikationstyp
  - Konformität zu Architekturvorgaben
  - Vorkonfigurierte technische QS (z.B. CheckStyle)

Erstellung eines KM-Handbuches

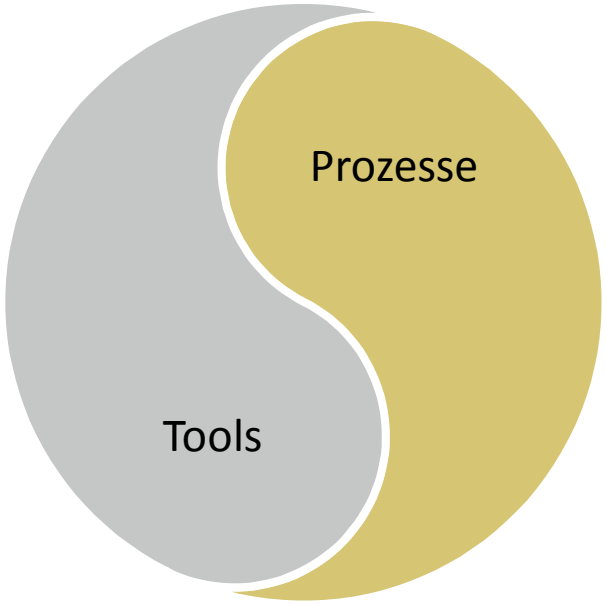
- Die Installation, Konfiguration und der Umgang mit den im Projekt verwendeten Werkzeugen werden in einem KM-Handbuch dokumentiert
- Das KM-Handbuch kann ganz klassisch als Word-Dokument erstellt werden. Alternativ ist auch ein Wiki ein gutes Medium.
- Typische Inhalte:
  - Doku der Projektumgebung (Verzeichnisse, Werkzeuge, etc.)
  - Beschreibungen und Templates für alle Konfigurationselemente (z.B. Use-Cases, Quelltextdateien)
  - Prozessbeschreibungen (z.B. Umgang mit Changes, Defects)
  - Qualitätsmanagement-Vorgaben (z.B. Durchführung von Audits, Einsatz von Metriken)

Beispiel-Gliederung

1 Einleitung
1.1 Inhalt dieses Dokuments
1.2 Leserkreis
1.3 Projekt-Homepage
2 Konfigurationselemente
2.1 Überblick
2.2 KE: Quellcode
2.3 KE: Use Case
…
3 Projektumgebung
3.1 Repository-Struktur
3.2 Verzeichnisstruktur des Arbeitsbereiches
3.3 Subversion
    3.3.1 Installation der Clients
    3.3.2 Konfiguration der Clients
    3.3.3 Berechtigungen
3.4 Weitere Werkzeuge …
4 Verwaltung der Konfigurationselemente
4.1 Erstmaliger Check-Out des Arbeitsbereiches
4.2 Hinzufügen, Ändern und Löschen von Dateien
4.3 Durchführung von strukturellen Änderungen
4.4 Sperren von Dateien
4.5 Verwendung von Tags
4.6 Verwendung von Branches
4.7 Subversion-Knigge

Qualität sicherstellen

Softwarequalität



- Hohe Softwarequalität ist nur durch ein abgestimmtes Paket von Prozessen und Tools zu erreichen
- Regelmäßige Überprüfung der Qualitätsziele (→ Checkstyle, JUnit)
  - Ermittlung von Metriken
  - Automatisierte Regressionstests
- Änderungs- und Fehlermanagement (→ Redmine)
  - Dokumentation, Bewertung und Priorisierung bekannter Fehler und geplanter Änderungen
  - Input für die Release- und Projektplanung
- Projektautomatisierung (→ Maven)
  - Erstellung des Produktes kann jederzeit wiederholt werden (Wiederholbarkeit)
  - Bei unveränderten Ausgangsbedingungen wird immer dasselbe Produkt erzeugt (Reproduzierbarkeit)

Qualitätssicherung und Metriken

- Metriken versuchen, bestimmten Aspekt eines Softwareprojektes in eine auf den ersten Blick leicht verständliche Zahl zu gießen
- Tatsächlich sind viele Metriken kompliziert und provozieren geradezu Missverständnisse
- Folgende Metriken werden unterschieden:
  - Produktmetriken: Lines of Code, Cyclomatic Complexity, …
  - Prozessmetriken: Durchschnittliche Dauer zur Behebung eines Fehlers
  - Projektmetriken: Auftragsvolumen, Teamgröße, …
- Viel hilft bei Metriken garantiert nicht viel!
- Metriken müssen zielgerichtet erfasst werden. Geeignet ist beispielsweise das *Goal-Question-Metric* (GQM) Verfahren
- Sobald sinnvolle Metriken bestimmt wurden, können einige davon auch automatisiert ermittelt werden (→ z.B. mit CheckStyle)

GQM-Verfahren

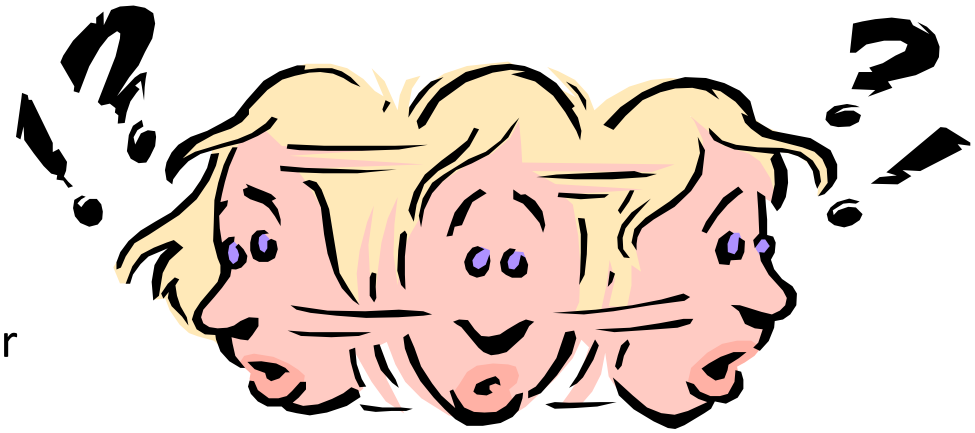
1. **Festlegung eines konkreten Ziels.**
   Beispiel: Einhaltung der *Java Code Conventions* von Sun
2. **Formulierung von Fragen zur Überprüfung des Ziels.**
   Beispiel: Wie viele Programmzeilen sind länger als 80 Zeichen?
3. **Definition einer Metrik, zur Einhaltung des Ziels.**
   Beispiel: Zähle alle Zeilen, die länger als 80 Zeichen sind.

Nach der Identifikation der *sinnvollen* Metriken, müssen diese im Projekt regelmäßig ermittelt und überprüft werden. Für die o.g. Metrik kann beispielsweise das *LineLength*-Modul von CheckStyle verwendet werden.

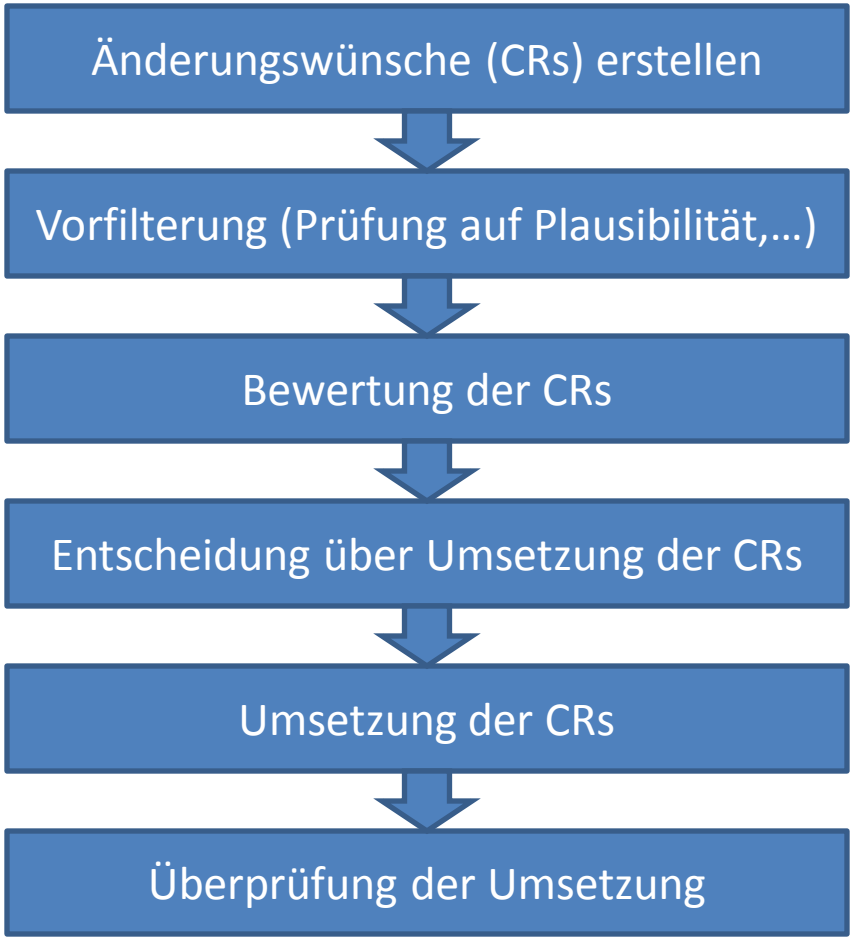
**) http://java.sun.com/docs/codeconv*

Transparenz verbessern

- Typische, banale Fragen im Projektalltag:
  - Wer arbeitet an was?
  - Wo stehen wir?
- Lösungsmöglichkeiten:
  - MS Project
  - Excel
  - Statusmeetings mit Kontrolle der zugewiesenen Tasks
  - …
- Besser: Einführung eines Werkzeuges zum Änderungsmanagement (→Redmine)



Änderungsmanagement mit Redmine



- Redmine unterstützt alle Schritte des Änderungsmanagements
- Zusätzliche Features
  - Automatische Bearbeitung von Tickets durch Analyse der Subversion-Kommentare
  - Unterstützung für die Release-Planung
  - Umfangreiche Auswertungen
  - Integriertes Wiki und Foren
  - …

Fragen ?