Vorlesung Software Engineering

Foliensatz Nr. 5 (10.11.11)

Arbeitsgruppe Software Engineering
Prof. Elke Pulvermüller

Universität Osnabrück
Institut für Informatik, Fachbereich Mathematik / Informatik
Raum 31/318, Albrechtstr. 28, D-49069 Osnabrück

elke.pulvermueller@informatik.uni-osnabrueck.de

http://www.inf.uos.de/se

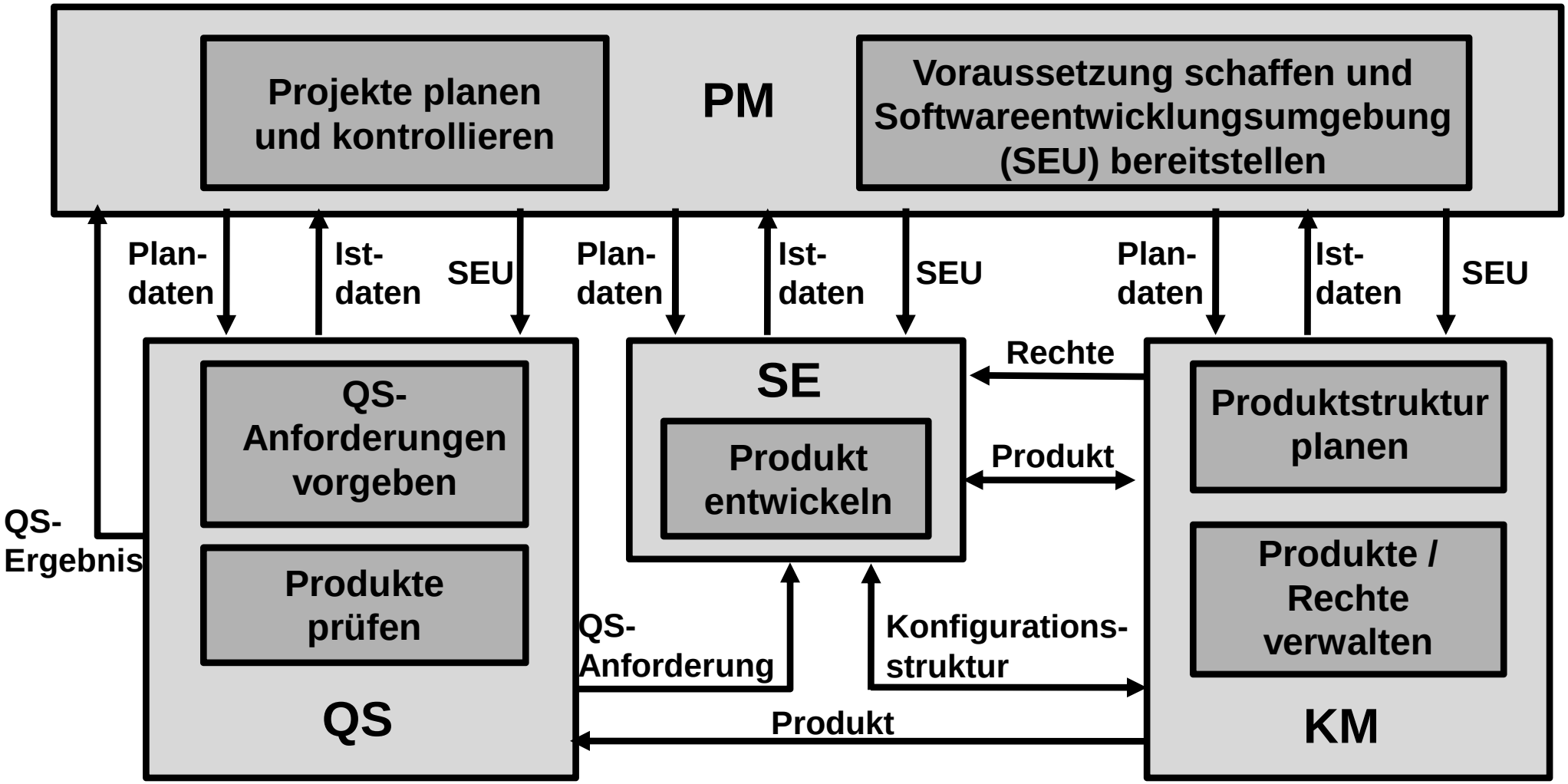
Sprechstunde: mittwochs 14 – 15 und n.V.

Inhalt

4 Konfigurationsmanagement

Inhalt

Schnittstellen des Projektmanagements



SEU: Software-Entwicklungsumgebung
PM: Projektmanagement
QS: Qualitätssicherung
KM: Software-/Konfigurationsmanagement
SE: Software-Entwicklung (beinhaltet auch Wartung und Evolution)

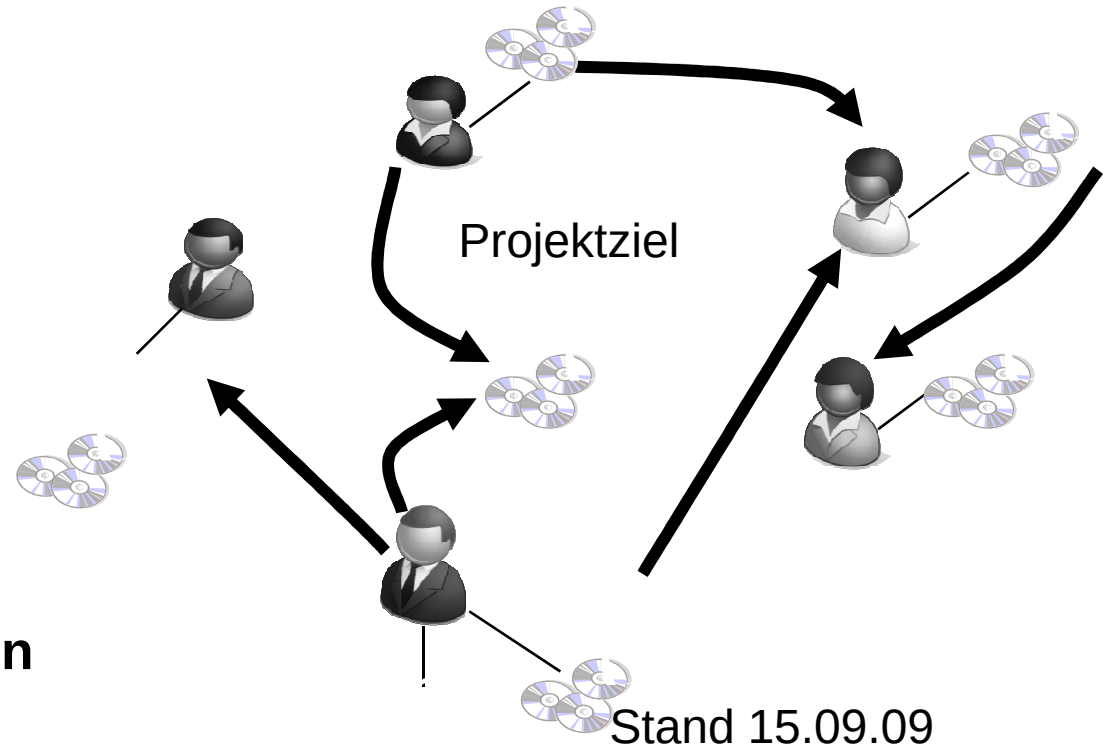
aus [Zuser, SW Engineering, 2004]

Probleme

Komponenten, Dokumente, (Zwischen-)Ergebnisse

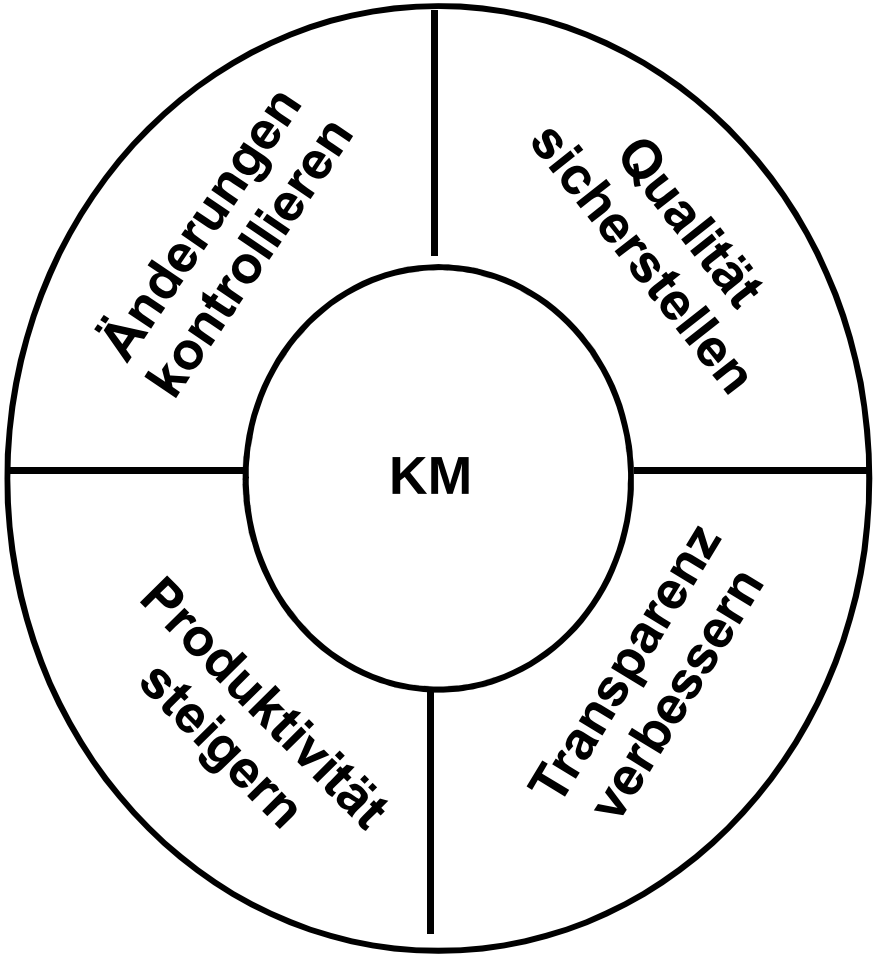
Projektteam-Mitglied (z.B. Entwicklerin)

- **Integration der falschen Komponenten**
- **Bereits entwickelte Komponenten werden übersehen (z.B. Programmteile, Testdaten)**
- **Vorversionen werden unvollständig archiviert**
- **Modifikationen werden versehentlich überschrieben**



⇒ **Fehler durch mangelhafte Übersicht über die Teile und ihre Zusammensetzung**

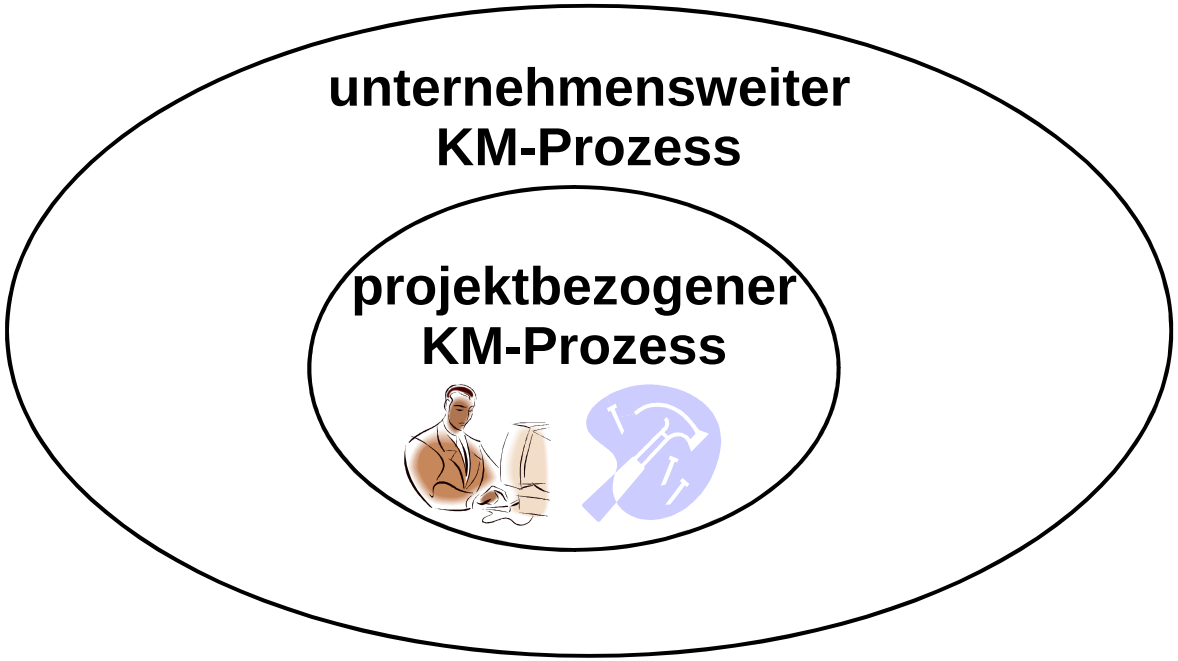
- **Grundidee: Richtlinien, Vorschriften und Werkzeuge zur sicheren Verwaltung und zum kontrollierten Zugriff auf alle Projektergebnisse im Team**



Konfigurationsmanagement (KM)

Software Configuration Management (SCM)

- **Reichweiten von KM-Prozessen**



- **Standards legen Grundzüge des KM fest**

  AFSCM-375-1 (1963, US-Militär, Hardware)
  MIL-STD-483 (1971, US-Militär, Hardware und Software)
  ISO 10007:2003, ISO 12207:1995, IEEE Std 828-1998

4.2 Aufgaben und Verfahren

- **Versionskontrolle (Identifizierung, Archivierung, Schutz/Zugriffsregeln, Bereitstellung, kontrollierte Löschung von Einheiten, Versionen und Varianten)**
- **Konfigurationskontrolle (Sicherstellung der Konsistenz, Dokumentation der Abhängigkeiten, Nachvollziehbarkeit und exakte Reproduzierbarkeit)**
- **Build-Automatisierung (Erzeugung ausührbarer Programme) mit Prüfsummenunterstützung**
- **Änderungskontrolle (Verwaltung und Prüfung von Änderungen)**
- **Koordination der Teamarbeit (zentrale Koordination mittels gemeinsamer Referenzumgebung und Konflikterkennung/-vermeidung)**
- **Ableitung und Bereitstellung von Metriken**

Schritt 1: Identifikation der Konfigurationselemente

- Quelltext
- Schnittstellenverträge
- Anforderungsdokumente (z.B. Use Cases)
- Architektur- und Designdokumente
- Konfigurationsmanagement-Handbuch
- Testspezifikationen und Testdaten
- Build-Skripte
- Meta- und Konfigurationsdaten
- Benutzerdokumentation
- Installationsanleitung, Release-Notes
- Werkzeuge (z.B. Compiler, Entwicklungsumgebungen)
- Bibliotheken
- Generierte Artefakte (zB. HTML-Dokumente, kompilierte Quelltexte)
- Protokolle von Meetings
- Binäre Auslieferungsdateien
- Projektpläne
- Protokolle von Meetings
- Liste offener Punkte, Risikolisten, etc.

Schritt 2: Erstellung eines KM-Plans

Einleitung
(Ziele, Begriffe, Beschreibung der Konfigurationselemente)

Management
(Verantwortlichkeiten für die KM-Aktivitäten)

Aktivitäten
(KM-Aufgaben und Verfahren)

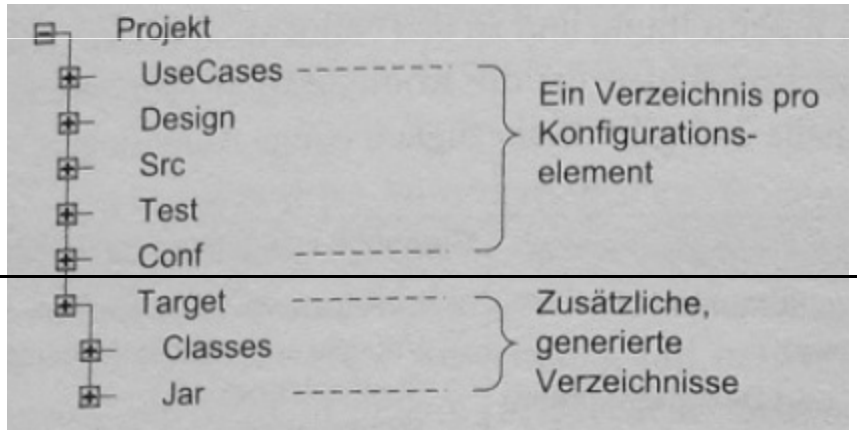
Zeitplanung
(zeitliche Abfolge der KM-Aktivitäten)

Ressourcen
(Werkzeuge, Personal)

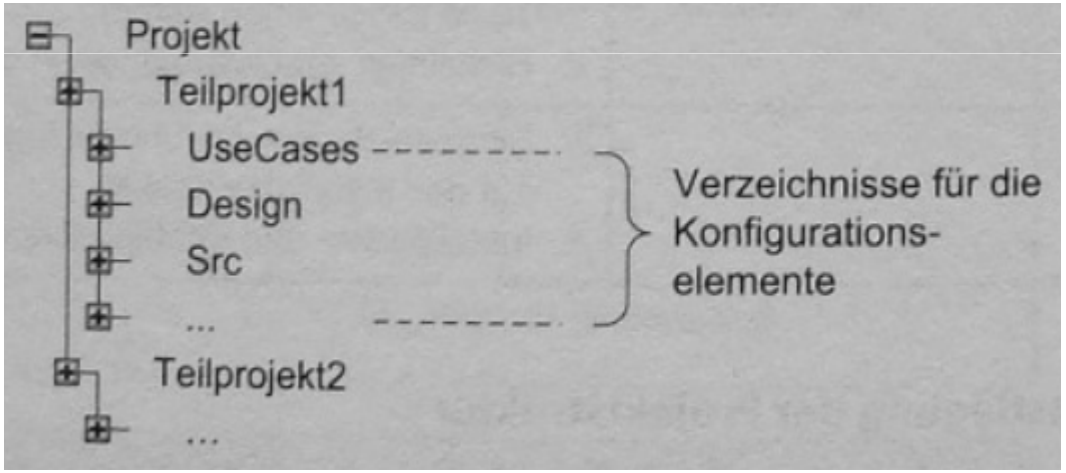
Pflege
(Änderungsplanung des Plans)

Schritt 3: Festlegung einer Projektstruktur

- **Hierarchie der Verzeichnisse im Projekt** ⇒ **Projektstruktur**
  ⇒ **Systemstruktur/Architektur (Conway's Law 1968)**

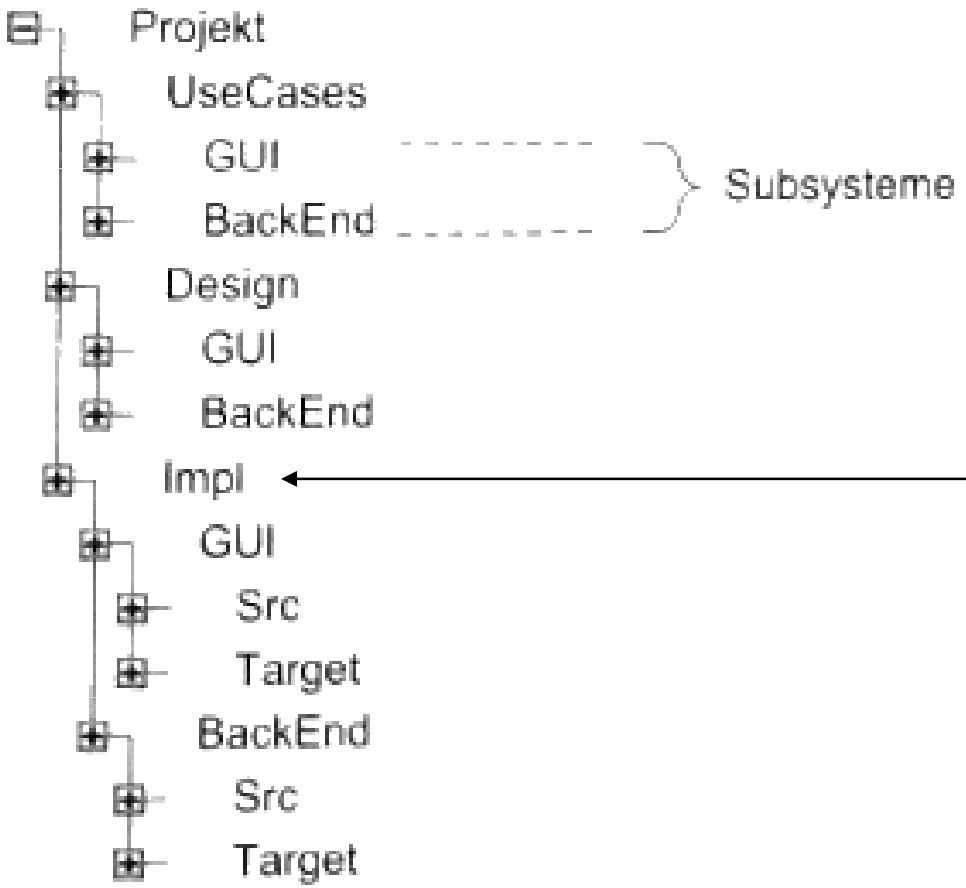


Keine Konfigurationselemente (temporärer Charakter)



- **Einfluss der Projektorganisation: Gliederung in Teilprojekte**

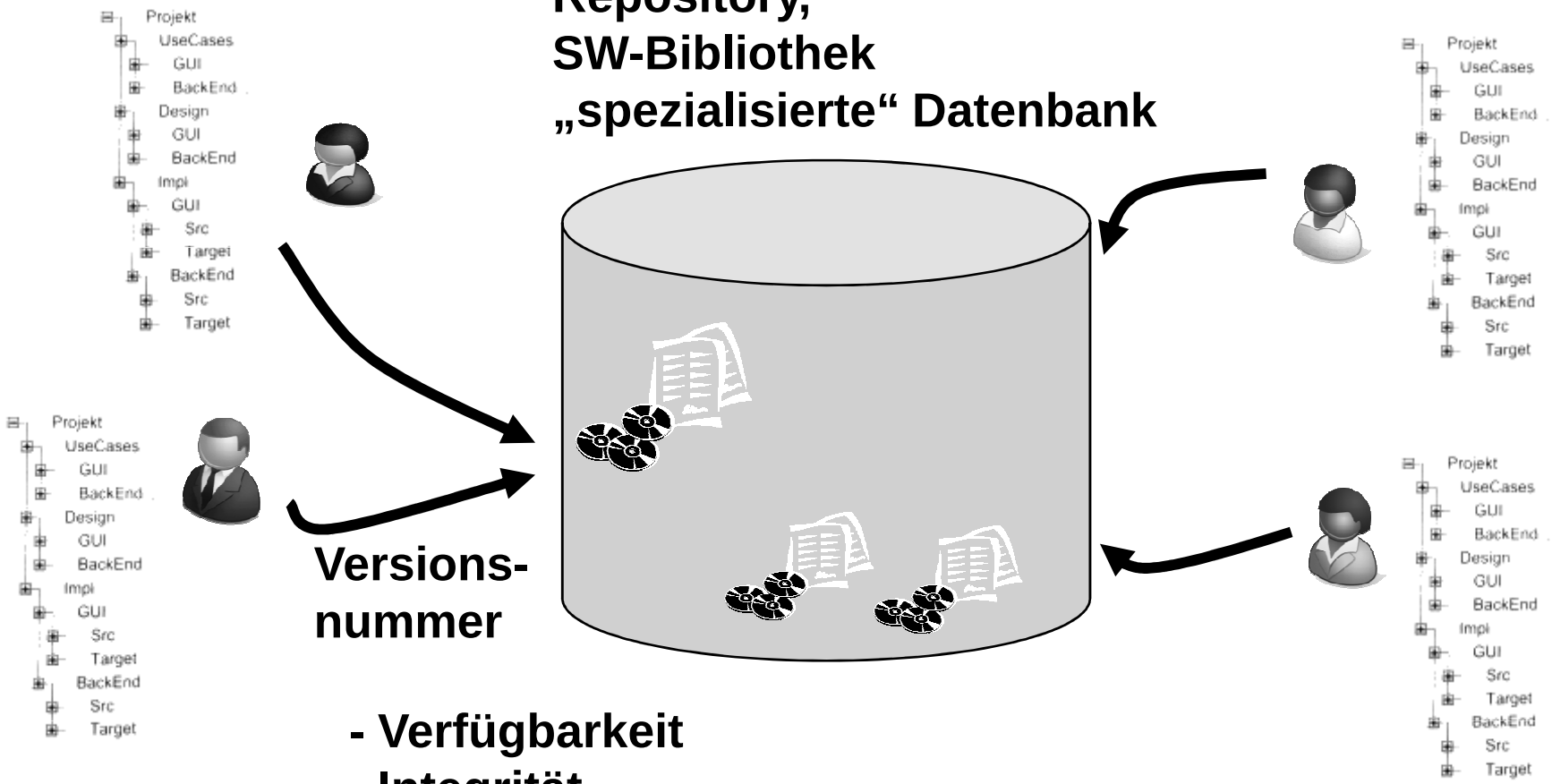
⇒ **Unabhängigkeit**
⇒ **Architektur ist dem organisatorischen Rahmen unterworfen**



- **Einfluss der SW Architektur: Detaillierung der Projektstruktur auf Basis der Softwarearchitektur**

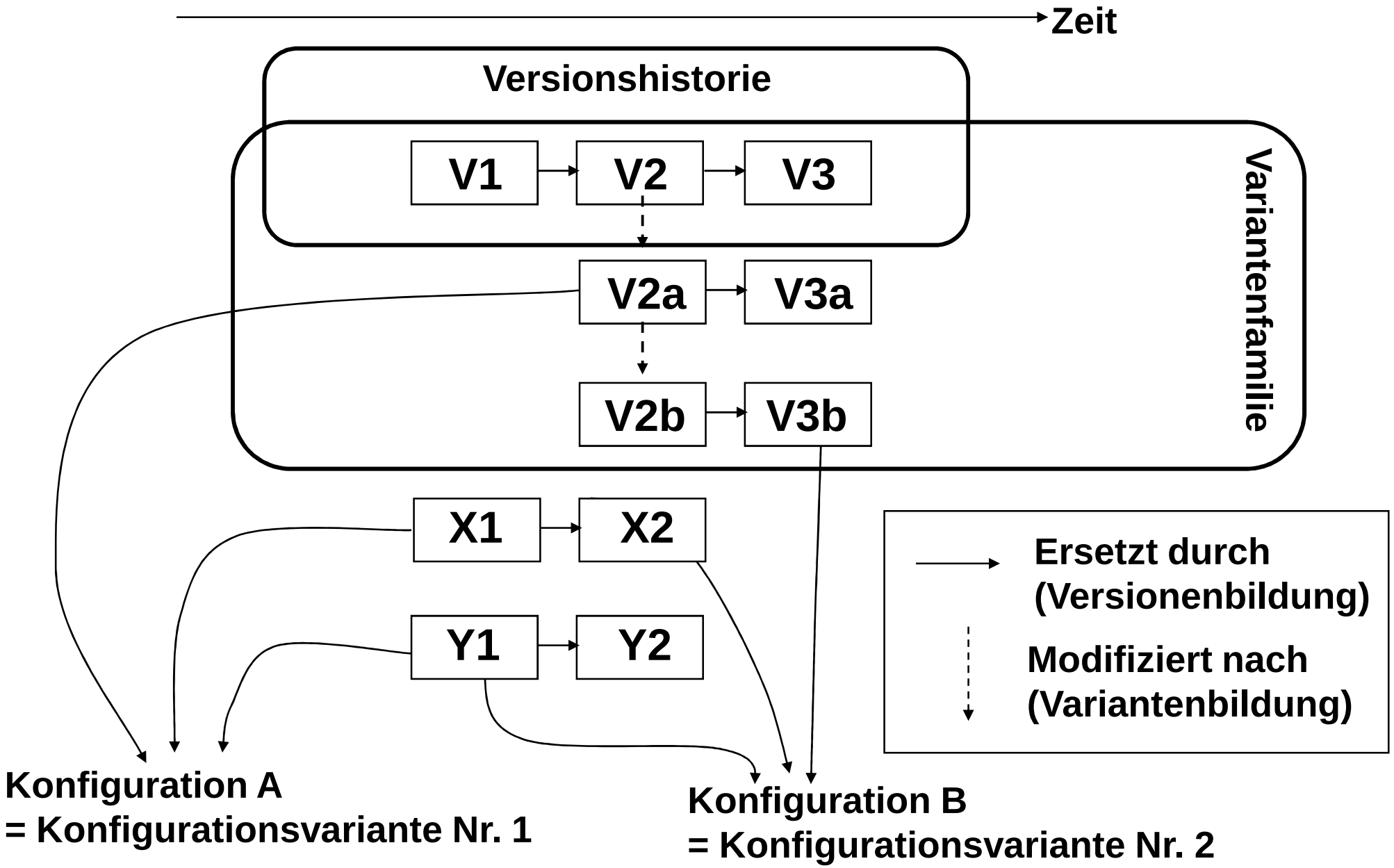
ggf. Hilfsordner zur Gruppierung aller Konfigurationselemente, die für die Implementierung relevant sind

- **Technische Einlussfaktoren (z.B. Verzeichnisse, die das KM-Werkzeug vorschreibt)**

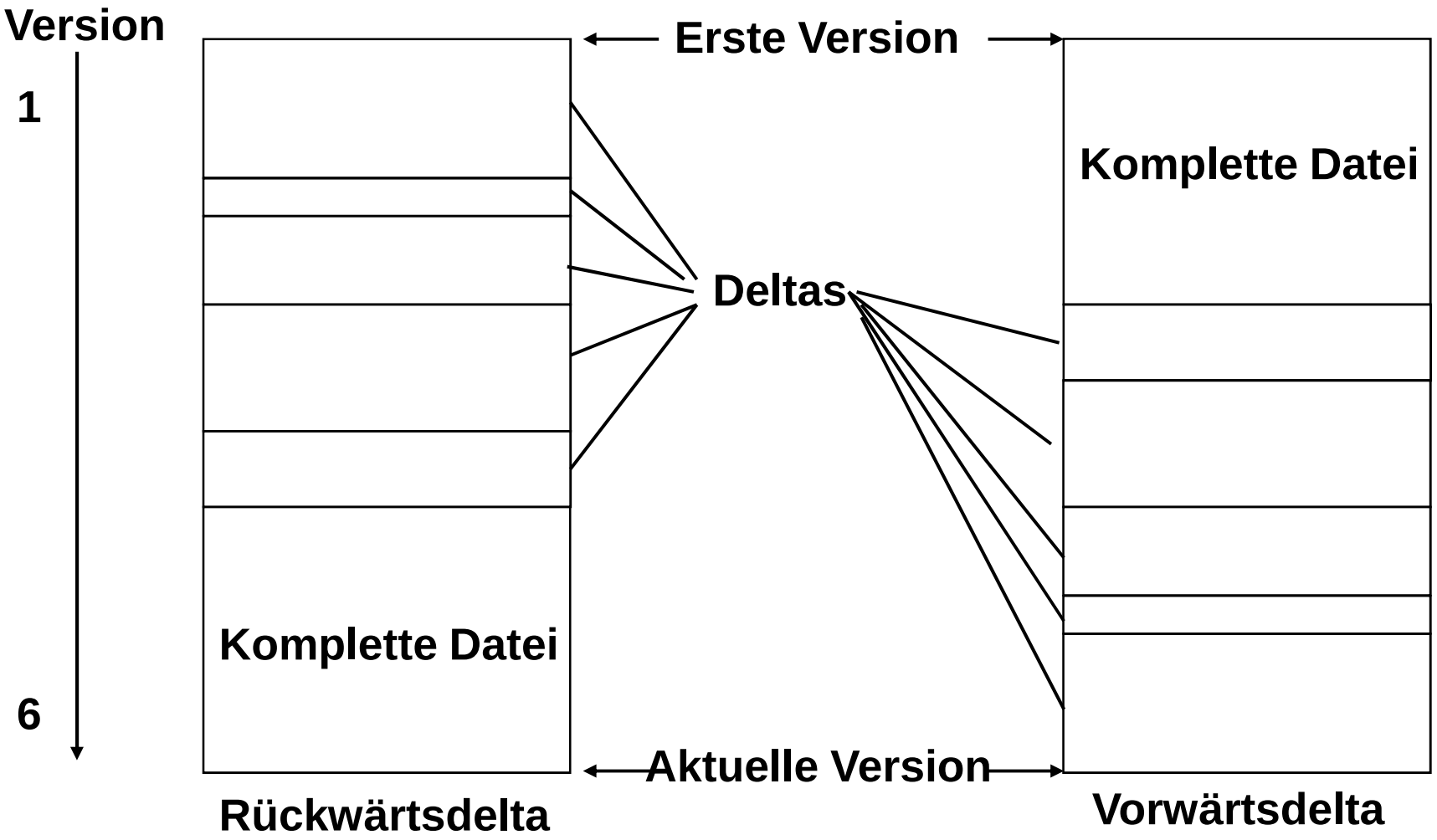


- Verfügbarkeit
- Integrität
- Zugriffkontrolle
- Änderungsverfolgung und Wiederherstellung (Versionshistorie)
- Effizienz bei großen Datenmengen

- Abbildung ins Repository (ohne temporäre Ordner)


Zeit
Versionshistorie
V1
V2
V3
V2a
V3a
V2b
V3b
Variantenfamilie
X1
X2
Y1
Y2
Ersetzt durch
(Versionenbildung)
Modifiziert nach
(Variantenbildung)
Konfiguration A
= Konfigurationsvariante Nr. 1
Konfiguration B
= Konfigurationsvariante Nr. 2

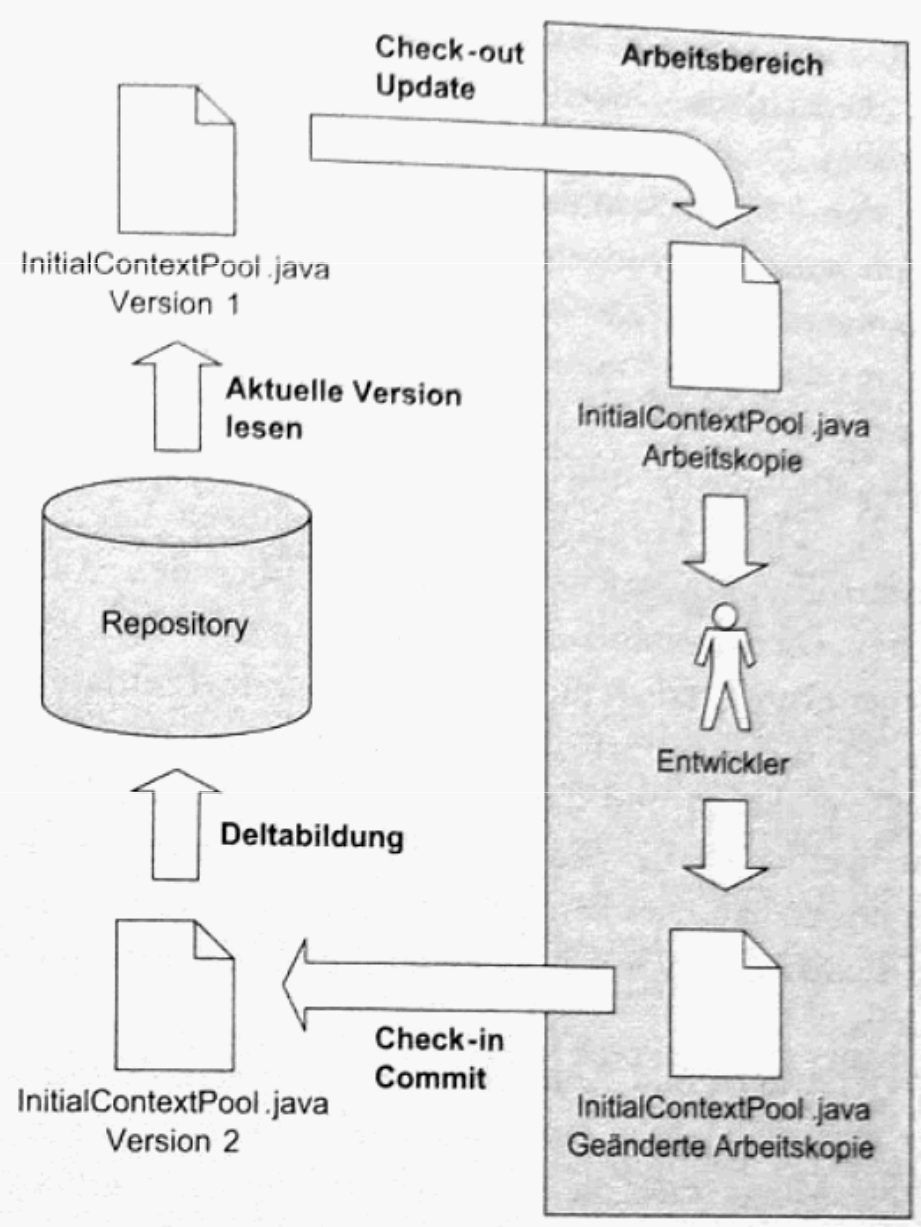
- **Deltamechanismus ⇒ Reduktion des Speicherbedarfs**



Konfigurationselement: Textdatei, Binärdatei, …

- **Check-in / Check-out**

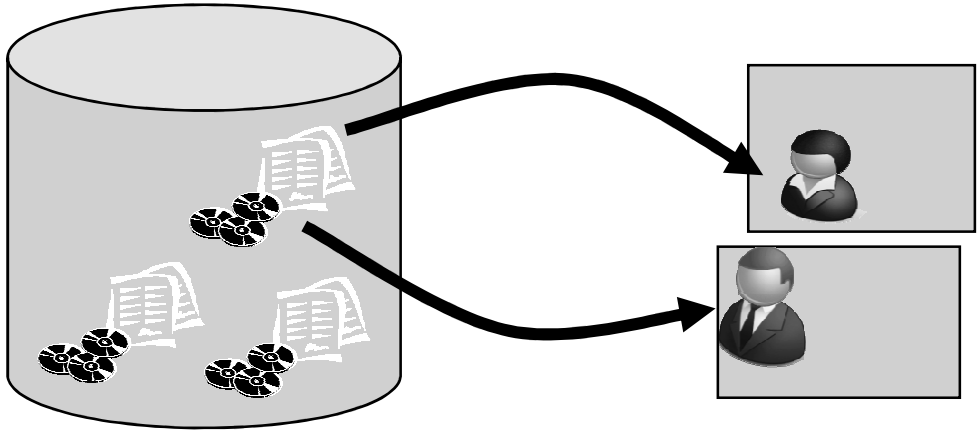
⇒ Bildung von Versionen



4.6 Verwaltung der Konfigurationselemente

- Check-in / Check-out

⇒ Kontrolle paralleler Änderungen



Check-out Lock

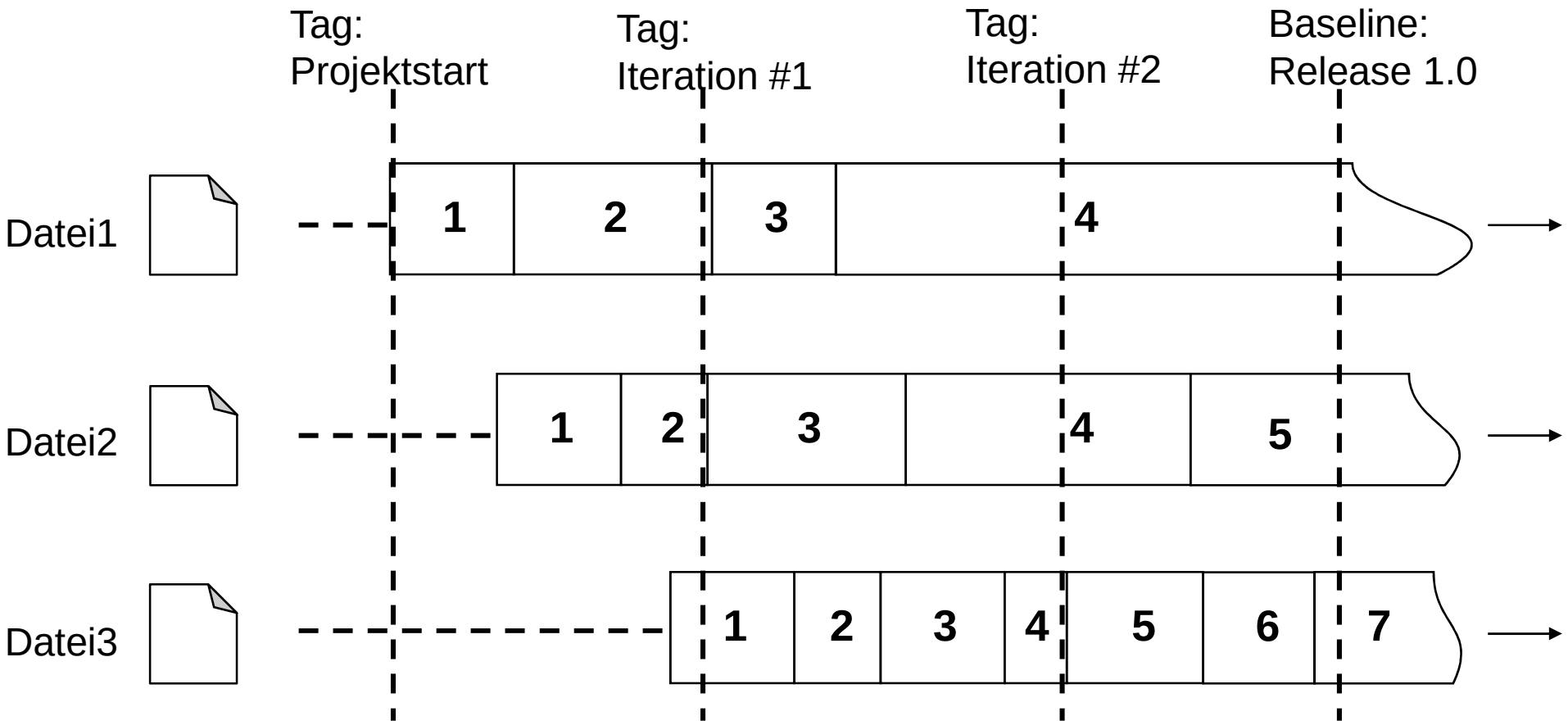
Unlock Check-in

0) Vermeidung durch Planung und Abstimmung

1) Lock-Modify-Unlock:
- Kontrolle zum Check-out / Lock-Zeitpunkt
- Sperrmechanismus: Keine parallele Veränderung

2) Copy-Modify-Merge
- Kontrolle zum Check-in / Unlock-Zeitpunkt
- Konfliktauflösungsmechanismus: (werkzeugunterstütztes) Merge

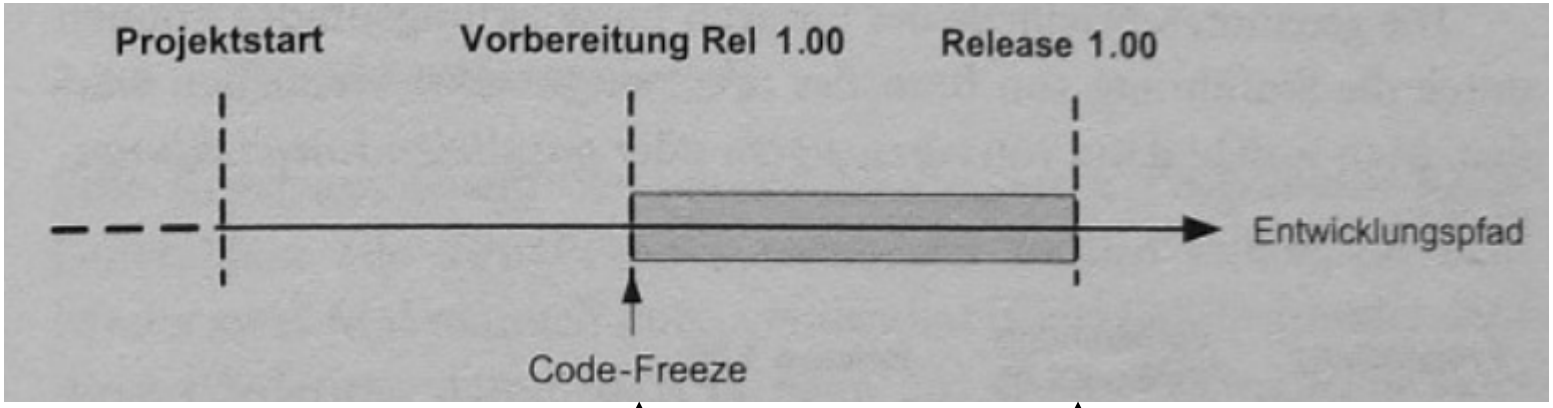
Werkzeuge können teils beides pro Konfigurationselement (z.B. Subversion)



- **Tag: Markierung und Bezeichner für die gültigen Versionen aller Konfigurationselemente im Repository zu einem bestimmten Zeitpunkt**
- **Baseline: stabile Konfiguration, Bezugspunkt für die weitere Entwicklung**
- **Release: Baseline mit dem Zweck der Auslieferung an den Kunden (lauffähig)**

- **Erstellung von Baselines und Releases: Branch Patterns**

1) Linearer Entwicklungspfad mit Code-Freeze



Abschluss von Änderungen
Aktualisierung des Repositories
Setzen eines Tags

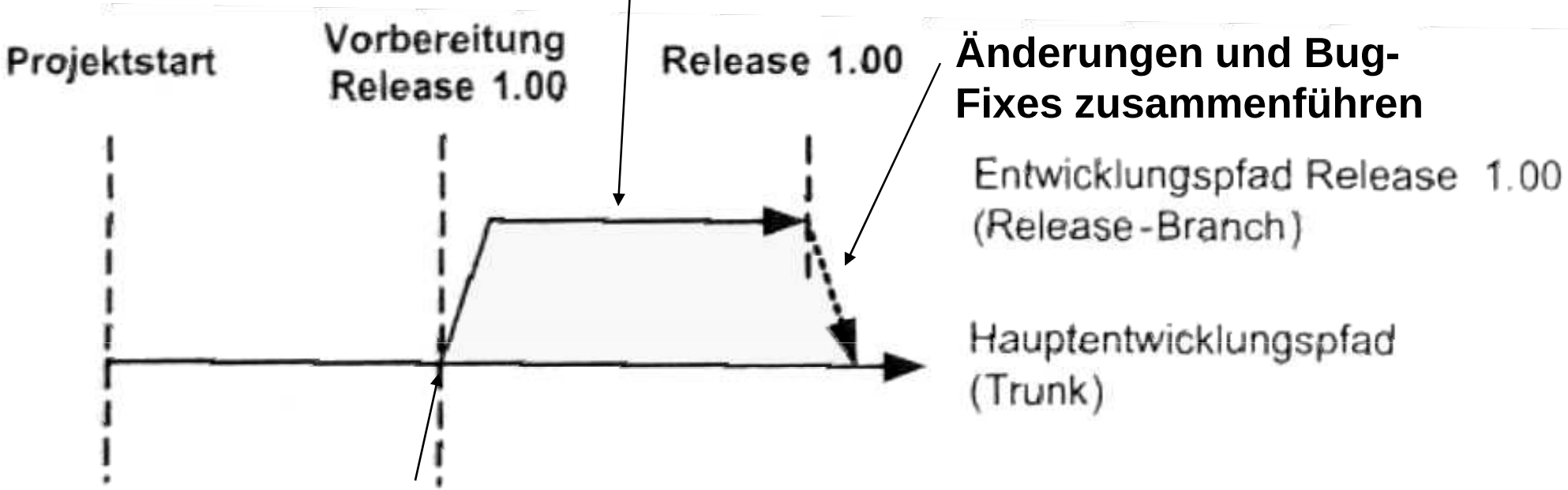
Tag setzen
Code-Freeze aufheben

Streng kontrollierte Änderungen:
Fehlerkorrekturen ja
Funktionserweiterungen nein

- **Erstellung von Baselines und Releases: Branch Patterns**

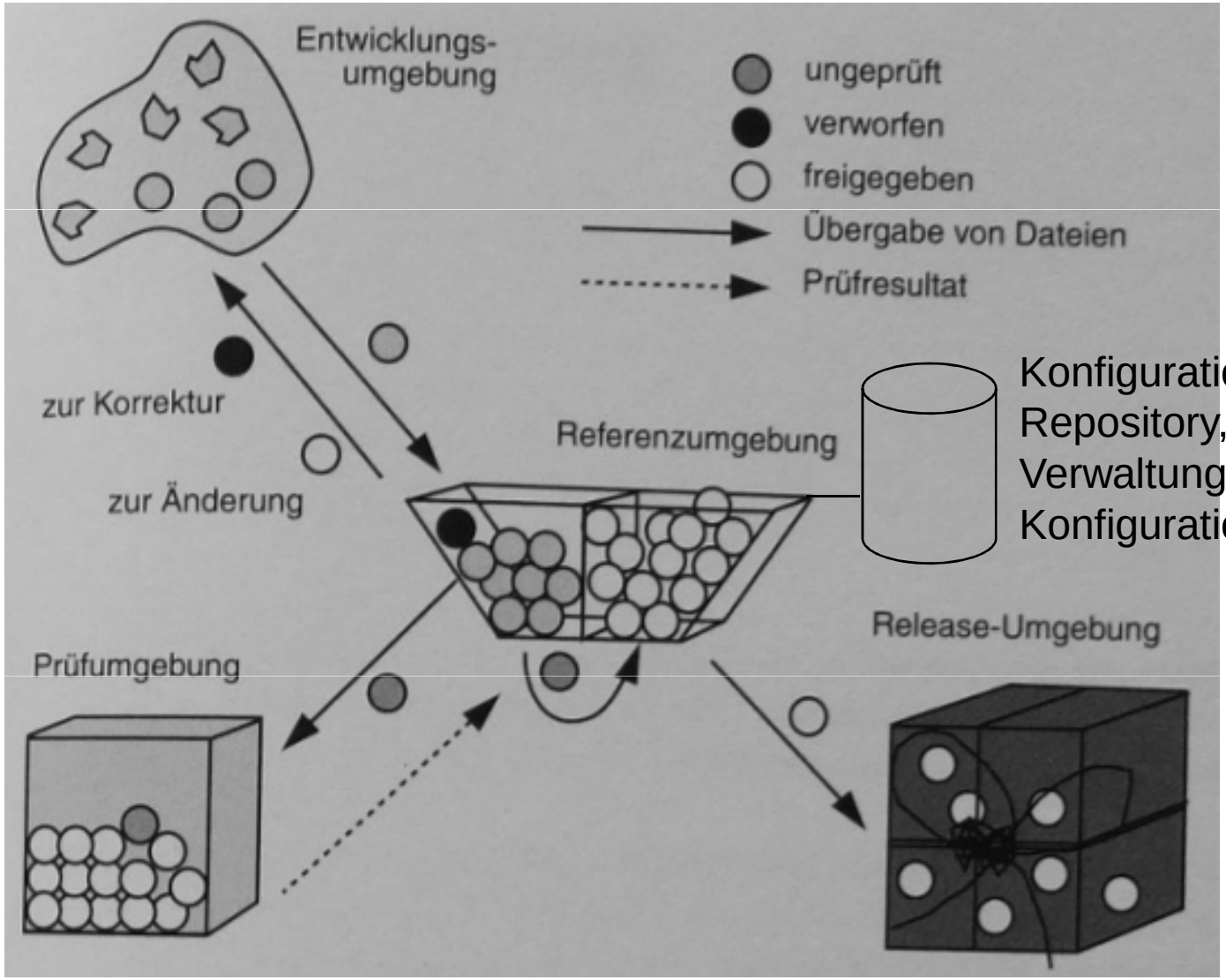
2) Verzweigte / Parallele Entwicklungspfade (mit Branches)

Branch zur Release-Erstellung (Ablauf wie beim Code-Freeze)

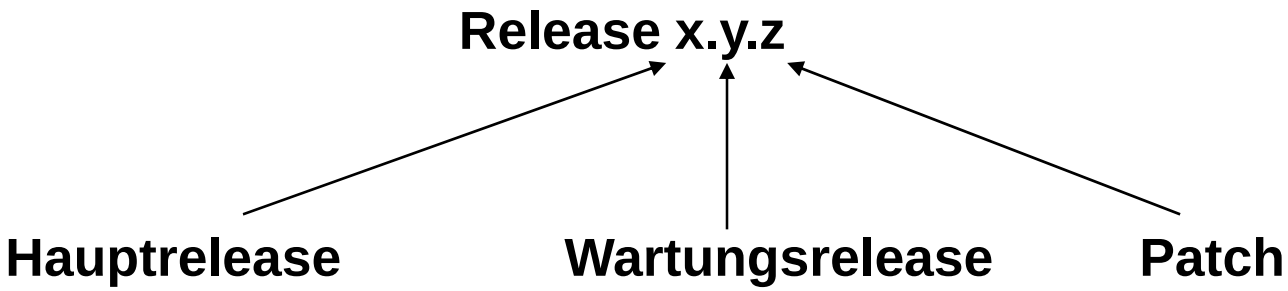


gemeinsame Versionen

3) Echte Parallelentwicklung: kein Zusammenführen (Pflege und Weiterentwicklung mehrerer Releases)



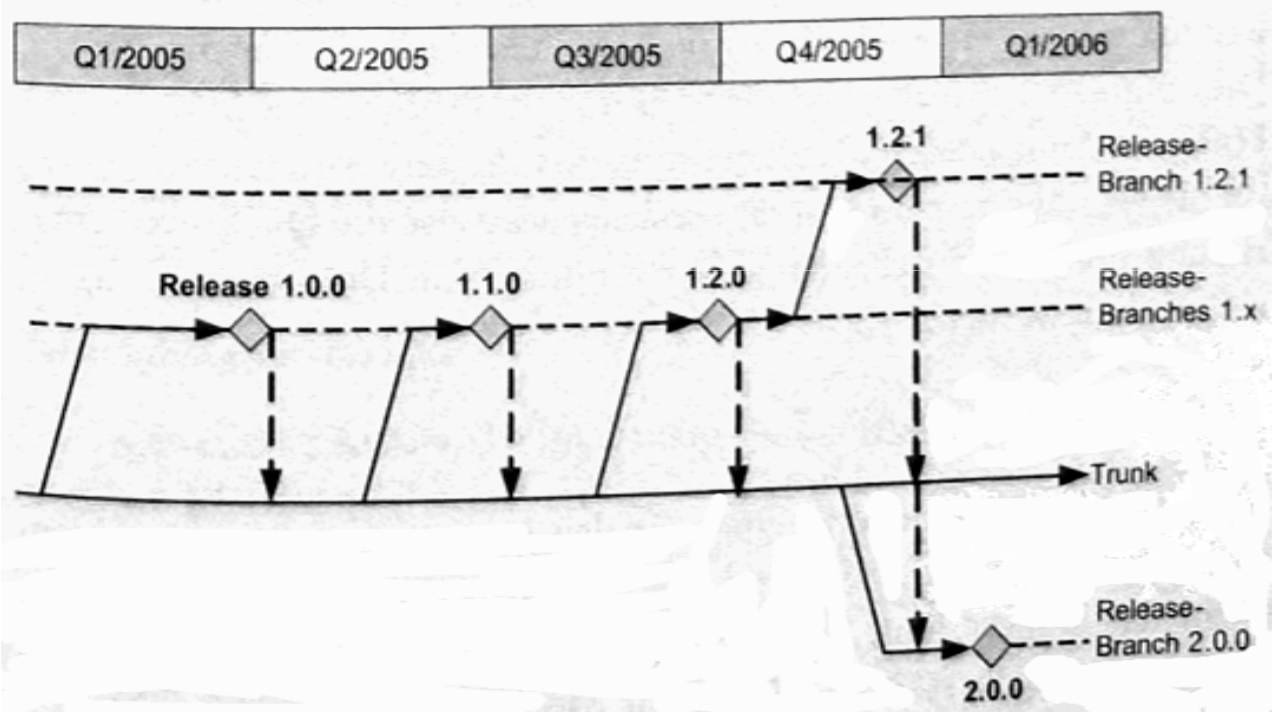
Konfigurationsdatenbank, Repository, SW-Bibliothek: Verwaltung der Konfigurationselemente


Release x.y.z
Hauptrelease
Wartungsrelease
Patch

Rel.-Nr.	Inhalt	Vorlauf	Auslieferung
1.0.0	Initiale Auslieferung	10 Wochen	01.04.2005
1.1.0	CR002, CR010, CR011 Fixes für Defects: 102, 110, 150	2 Wochen	01.07.2005
1.2.0	CR004, CR020 Fixes für Defects: 170, 175, 180	2 Wochen	01.10.2005
1.2.1	Fix für Defect: 191	Keiner	04.12.2005
2.0.0	Neues Modul Auswertungen	8 Wochen	01.02.2006

Release-Beschreibung (Kern des Release-Plans)

← **„Jetzt“**



Grafische Übersicht des Release-Plans

4 Konfigurationsmanagement

Zusammenfassung und Ausblick

1 Software-Krise und Software Engineering
2 Grundlagen des Software Engineering
3 Projektmanagement
4 Konfigurationsmanagement
5 Software-Modelle
6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle
7 Qualität
8 … Fortgeschrittene Techniken

- 4.1 Motivation und Begriffe
- 4.2 Aufgaben und Verfahren
- 4.3 Konfigurationselemente
- 4.4 KM Plan
- 4.5 Projektstruktur
- 4.6 Verwaltung der Konfigurationselemente
- 4.7 Release-Management
- 4.8 Werkzeug Subversion
- 4.9 Build-Prozess

→ Softwareentwicklung in Projekten benötigt Management der im Projekt anfallenden Einheiten mit guten Werkzeugen und automatisiertes Management des Zusammenbaus der im Projekt anfallenden Einheiten (Build-Prozess)