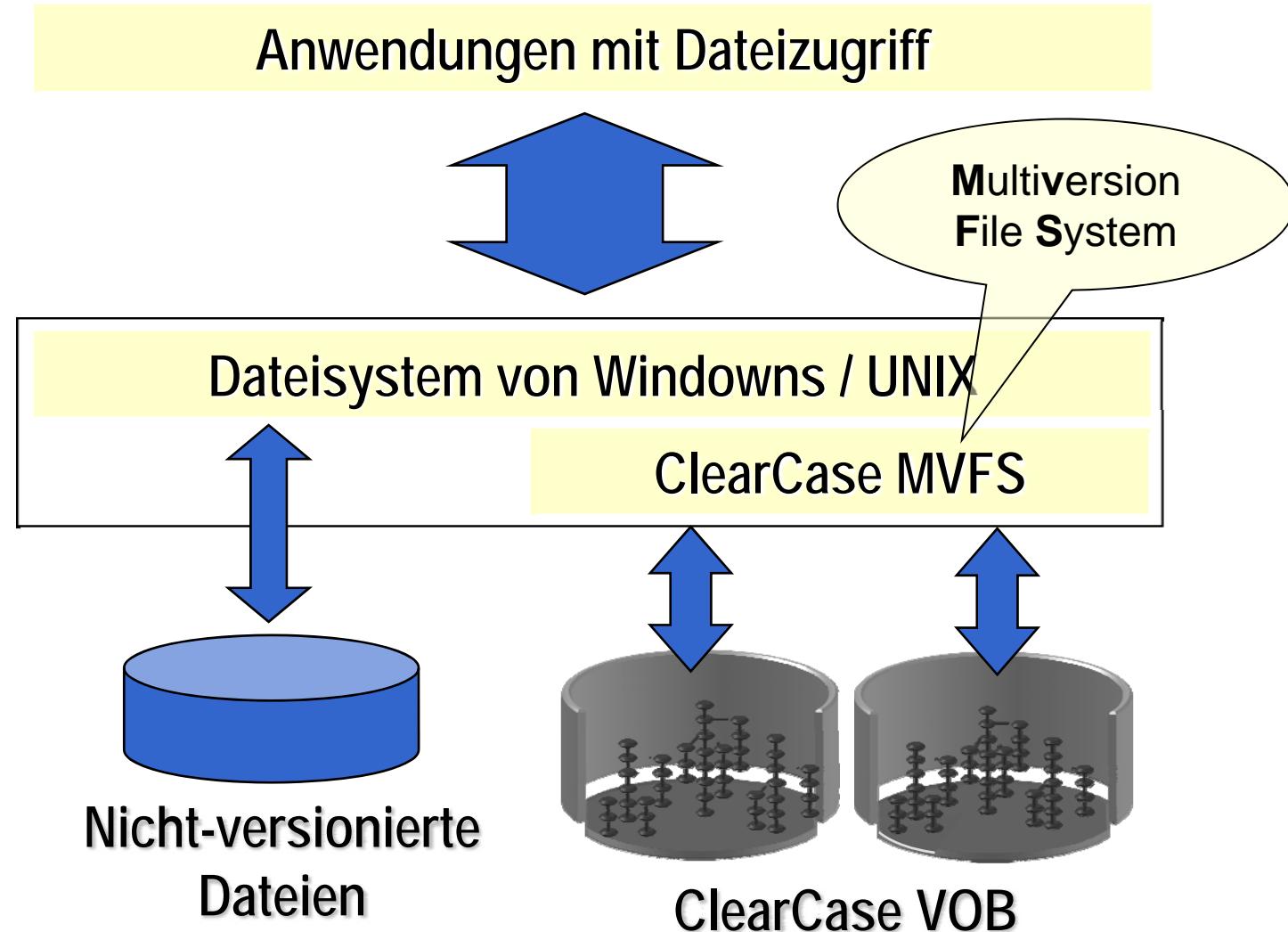


## Exkurs: Clear Case

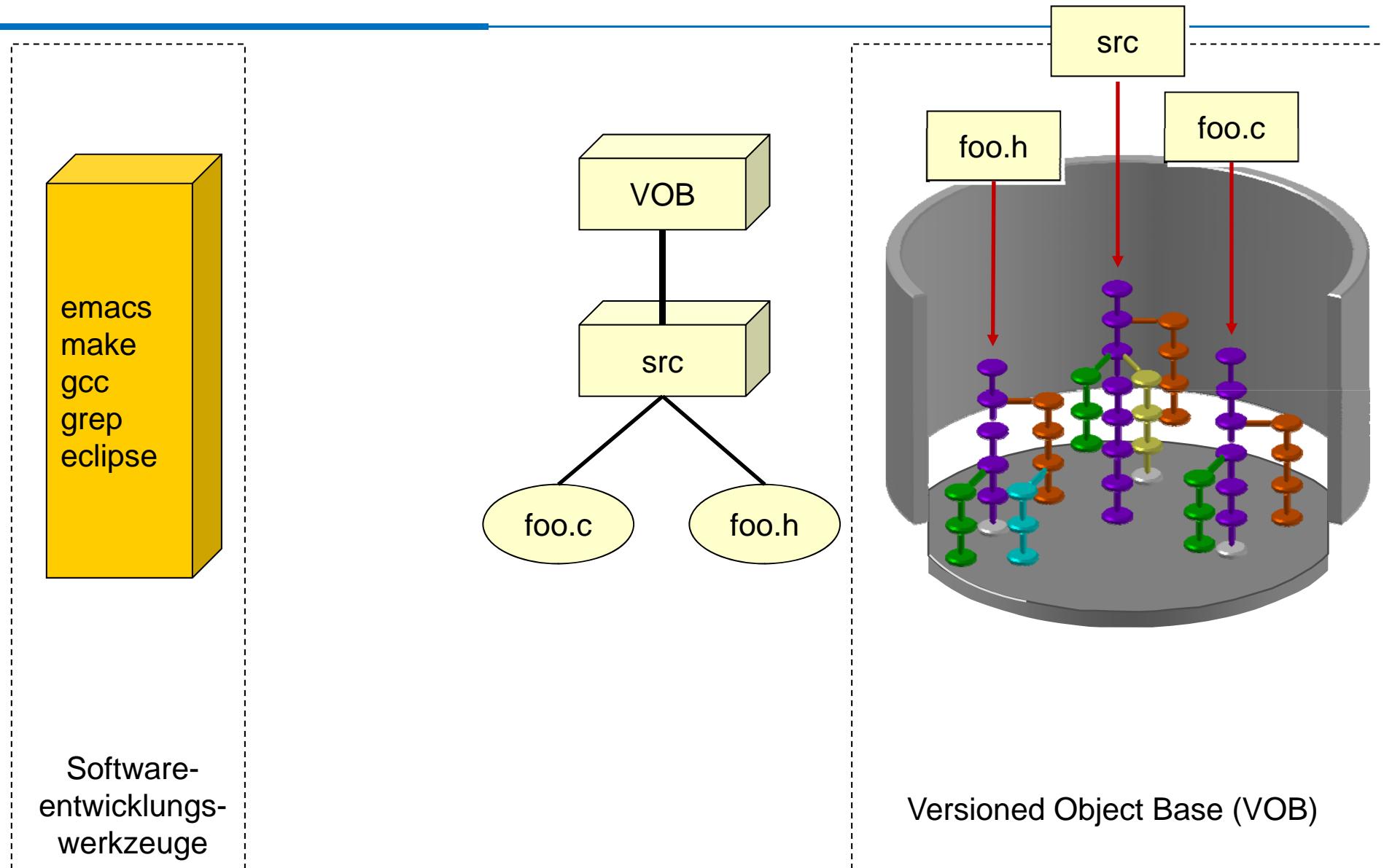
Dieser Foliensatz enthält Details zu ClearCase.

Ab WS 2009 sind die hier enthaltenen Folien nicht mehr Teil des Prüfungsstoffes sondern lediglich Vertiefung für Interessierte.

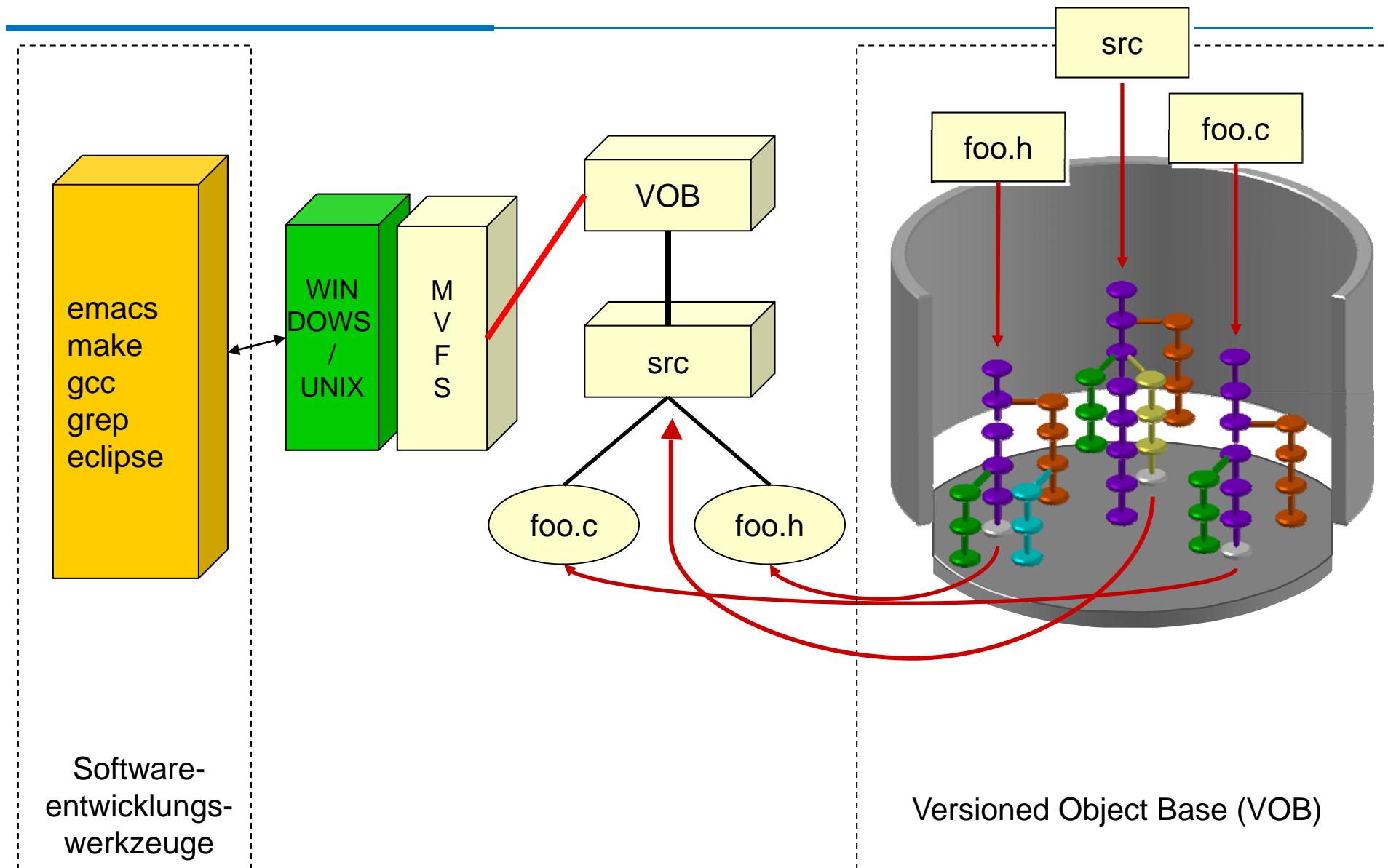
# ClearCase® Architektur



# Virtuelles Dateien System (Virtual File System) bei ClearCase

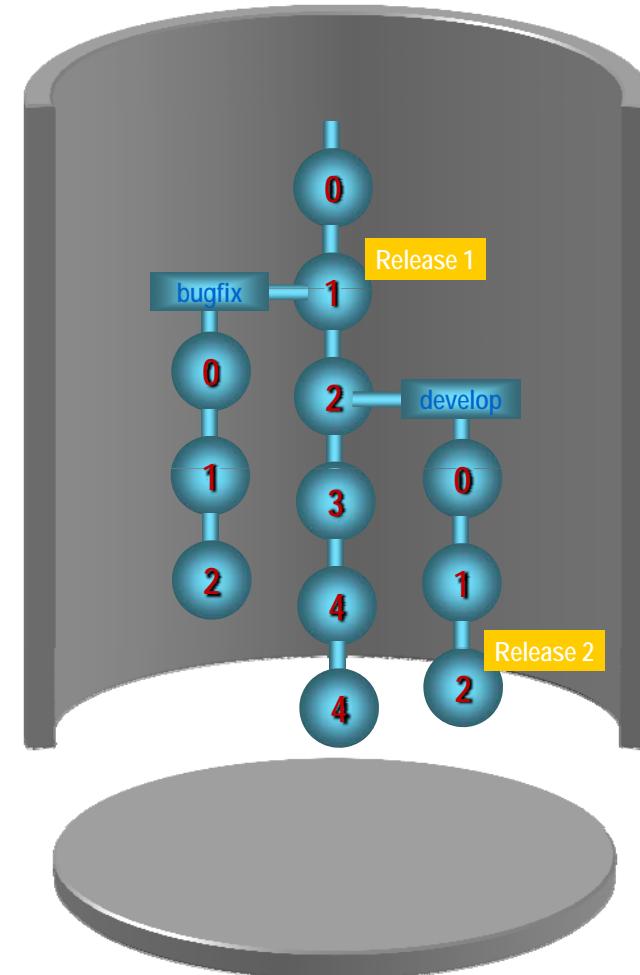


# Virtual File System bei ClearCase



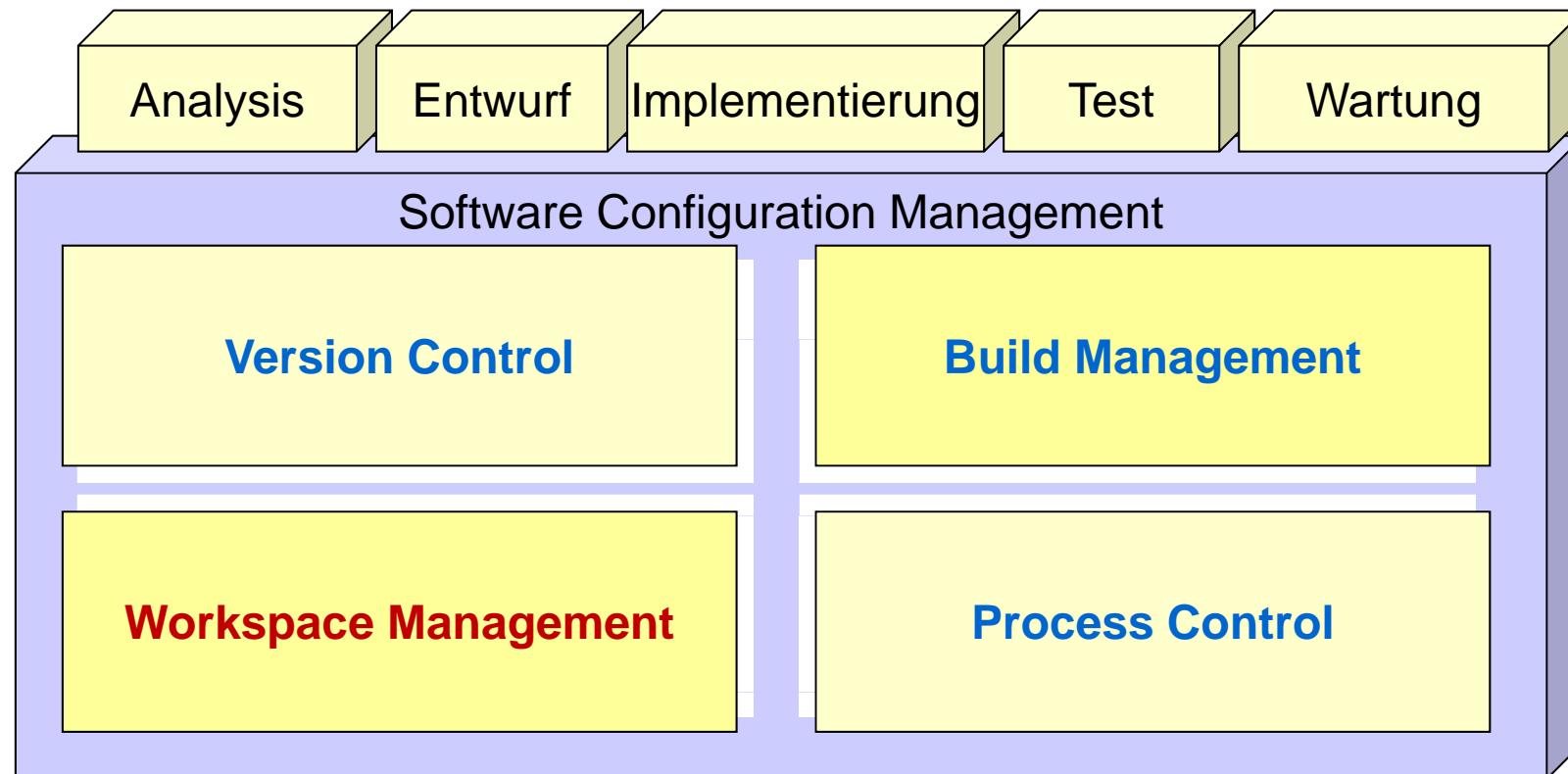
# ClearCase VOB (Versioned Object Base)

- Sicherer Repository
  - ◆ aufbauend auf objektorientierter Datenbank
  - ◆ Zugriff nur mit ClearCase möglich
- Speichert alle versionierten Daten
  - ◆ Source code
  - ◆ Directories
  - ◆ Binaries
  - ◆ Wer hat was, wann, warum getan?
- Gemeinsamer Datenzugriff
- Beliebig viele VOBs im Netzwerk mit beliebig vielen Elementen  
→ Verteile Datenbank



# SCM: Das Fundament des Entwicklungsprozesses

---



# Workspace Management (1)

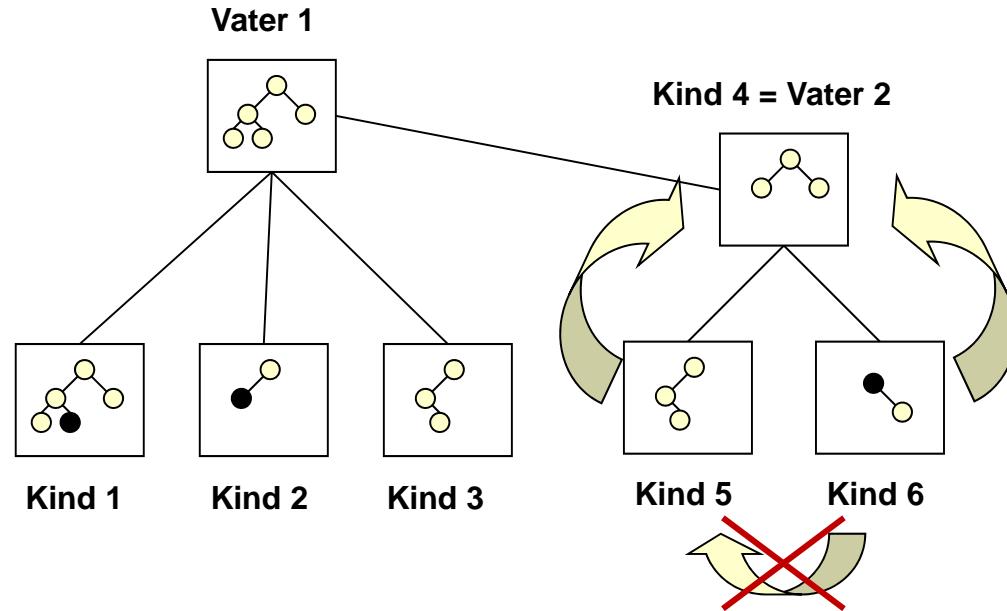
---

- Einrichtung und Pflege einer persönlichen Arbeitsumgebung
  - ◆ d.h. Zusammenstellung der benötigten Objekte
    - ⇒ Sources, Binaries, ...
  - ◆ ... für die entsprechende Aufgabe
    - ⇒ Bugfixing, Neu- / Weiterentwicklung, ...
- Ansätze
  - ◆ **unconstrained**:
    - ⇒ keine wohldefinierte Arbeitsumgebung erforderlich
  - ◆ **hierarchical sub-environments**:
    - ⇒ vorgegebene Arbeitsbereiche können kopiert und lokal geändert werden
  - ◆ **change sets**:
    - ⇒ Grundversion eines Objekts + Menge von Änderungen dieses Objekts
  - ◆ **rule-based environments**:
    - ⇒ benötigte Objekte werden über Regeln definiert

# Workspace Management: Hierarchical Sub-Environments

- Prinzip

- ◆ copy - modify - merge



- Nachteile

- ◆ zur Integration von Änderungen muß gemeinsamer "Vater" vorhanden sein; dieser wird dann unwiderruflich geändert
  - ◆ paarweiser Austausch zwischen "Kindern" nicht direkt möglich
  - ◆ Kombination von Teilespekten einzelner "Kinder" nicht möglich
  - ◆ informative globale Sicht geht durch Isolation der Arbeitsumgebungen verloren

# Workspace Management: Change Sets

---

- Gruppierung von aufeinander aufbauenden Versionen
  - ◆ z.B. Bugfixes
  - ◆ wird oft zur Organisation *einer* Entwicklungslinie benutzt
  - ◆ am Ende des Entwicklungsprozesses werden die kumulativen Veränderungen als neue Versionen herausgebracht (patch bundles)

# Workspace Management: Rule-Based Workspaces

---

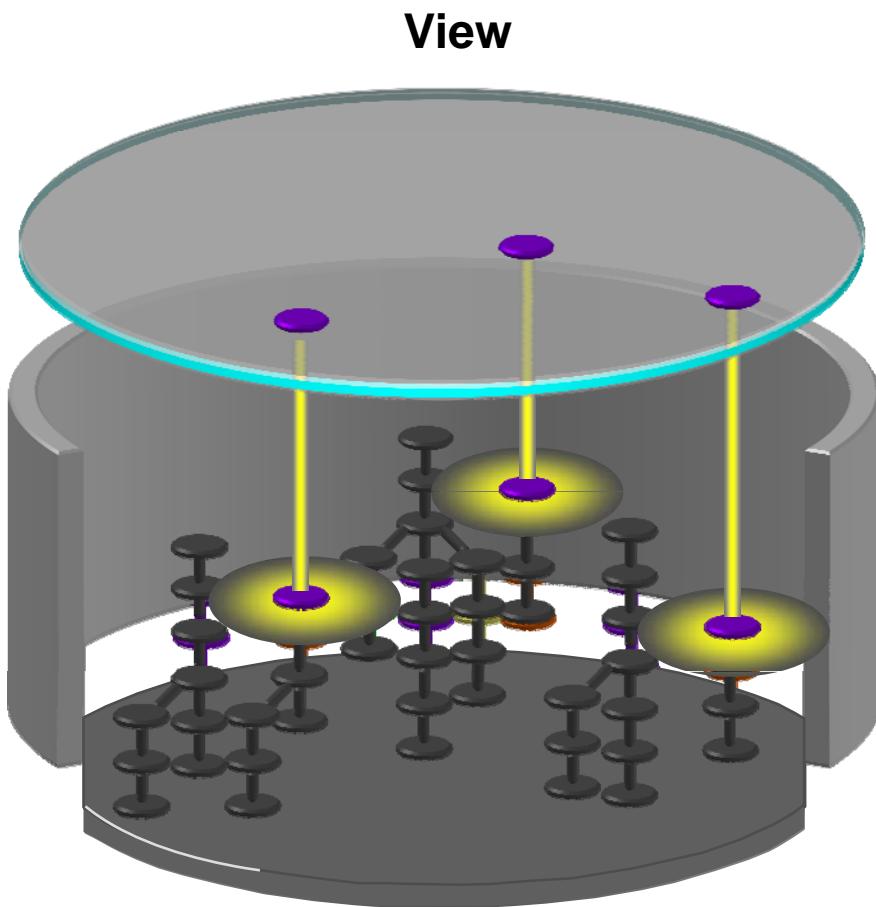
- Prinzip: regeldefinierte Auswahl von Objekten / Versionen
- Statische Regelauswertung
  - ◆ Auswertung der Regeln zum Zeitpunkt des Arbeitsbeginns
  - ◆ ausgewählte Versionen werden in den privaten Arbeitsbereich kopiert
  - ◆ der Benutzer muss am Ende explizit die Synchronisation anstoßen
- Dynamische Regelauswertung
  - ◆ Auswertung der Regeln zum Zeitpunkt des Objektzugriffs
    - ⇒ ClearCase

# Views in ClearCase (1)

---

- Configuration Specification
  - ◆ Sichtdefinition durch benutzerspezifische **Regelmenge**
  - ◆ agieren als Filter
  - ◆ Auswahl einer bestimmten Objektversion verhindert Zugriff auf alle anderen Versionen des selben Objekts
  - ◆ Projektion der ausgewählten Struktur und der spezifizierten Versionen als **normaler Verzeichnisbaum**
  - ◆ nur für den privaten Gebrauch gedacht

# ClearCase Views



- Transparentes Arbeiten mit verschiedenen Releases des gleichen Projekts
  - ◆ “Zeig mir Version 2.20”
  - ◆ Eine Art Filter
- Arbeiten in Echtzeit
- Sofortiger Zugriff auf den gesamten Datenbestand
- Downloads finden nur statt, wenn es erforderlich ist (ein Zugriff)
  - ◆ kein komplettes Kopieren!

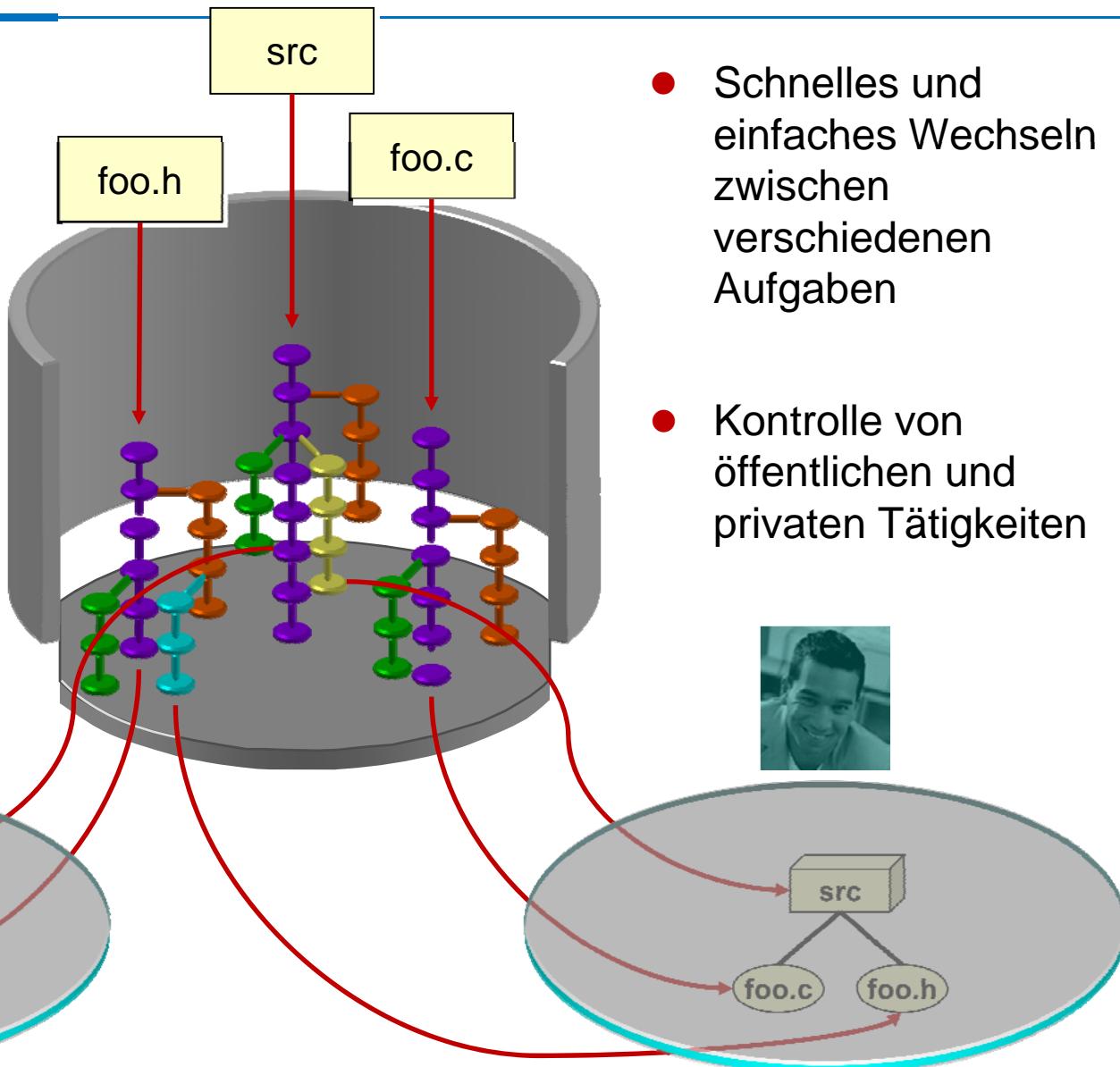
# ClearCase Views

---

- Alle Objekte im VOB sind schreibgeschützt
- **Check-Out**
  - ◆ Ein Entwickler muß ein check-out-Kommando absetzen, um ein Objekt verändern zu können
  - ◆ Dieses wird dann in den privaten Speicherbereich kopiert, ist aber noch unter dem ursprünglichen Namen und Pfad ansprechbar
- **Check-In**
  - ◆ Das check-in-Kommando erzeugt eine neue Objektversion im VOB und löscht die private Kopie
  - ◆ Locking: Nur derjenige Entwickler, der das check-out-Kommando abgesetzt hat, darf die neue Objektversion auch wieder einchecken
- Variante: „unreserved check-out“:
  - ◆ „first check-in wins“; alle anderen müssen mergen

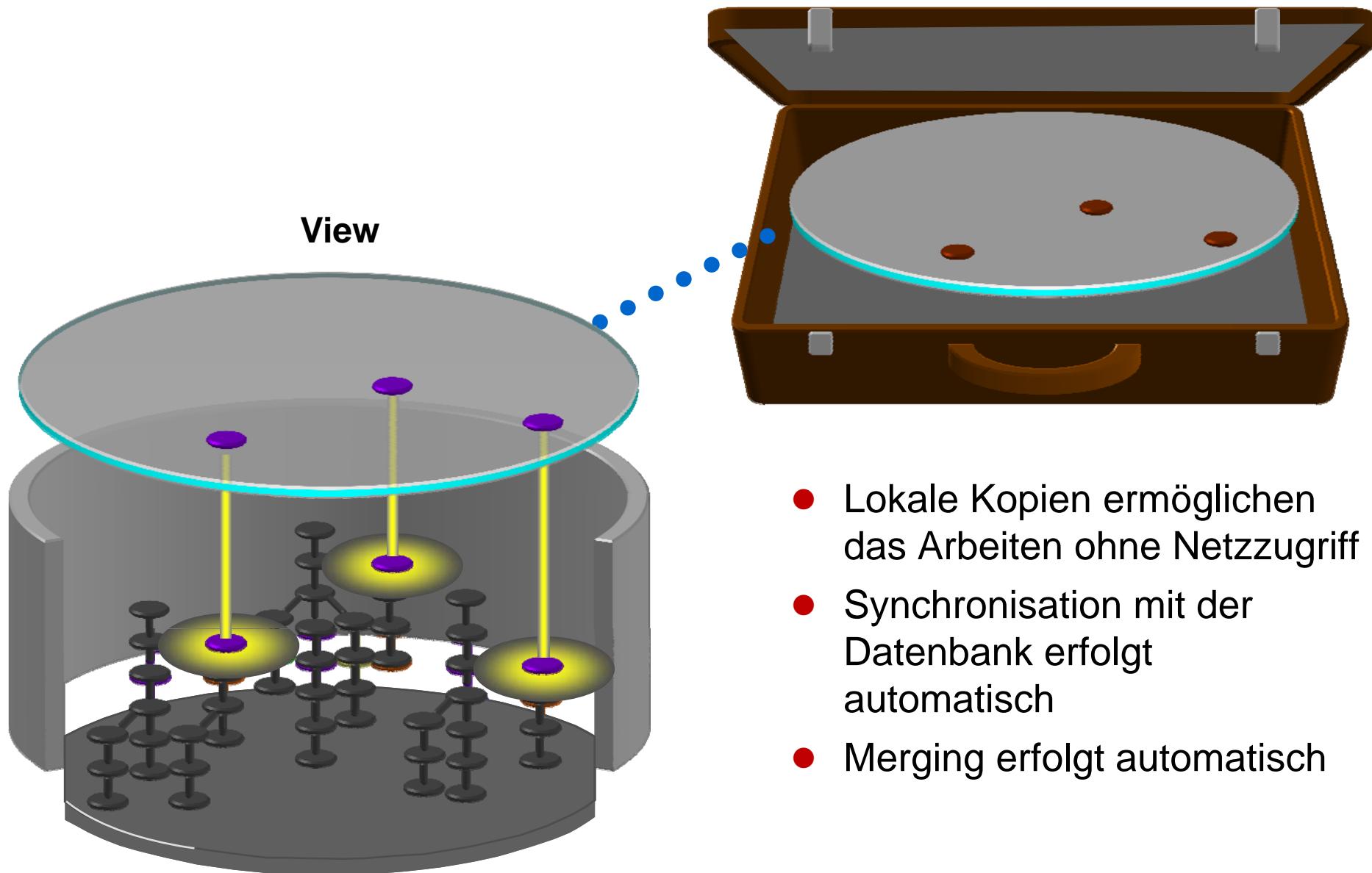
# ClearCase Views

- Einfacher Weg, viele Aufgaben zu managen
- Erlaubt dynamisches Teilen der Arbeit



# ClearCase Views: Private Storage

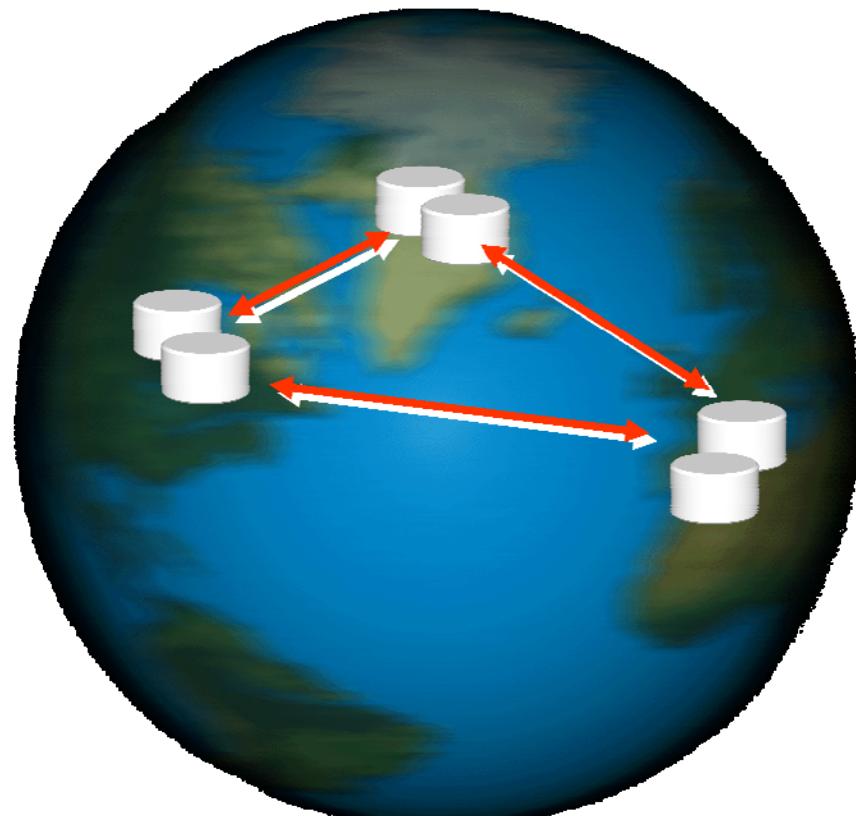
---



- Lokale Kopien ermöglichen das Arbeiten ohne Netzzugriff
- Synchronisation mit der Datenbank erfolgt automatisch
- Merging erfolgt automatisch

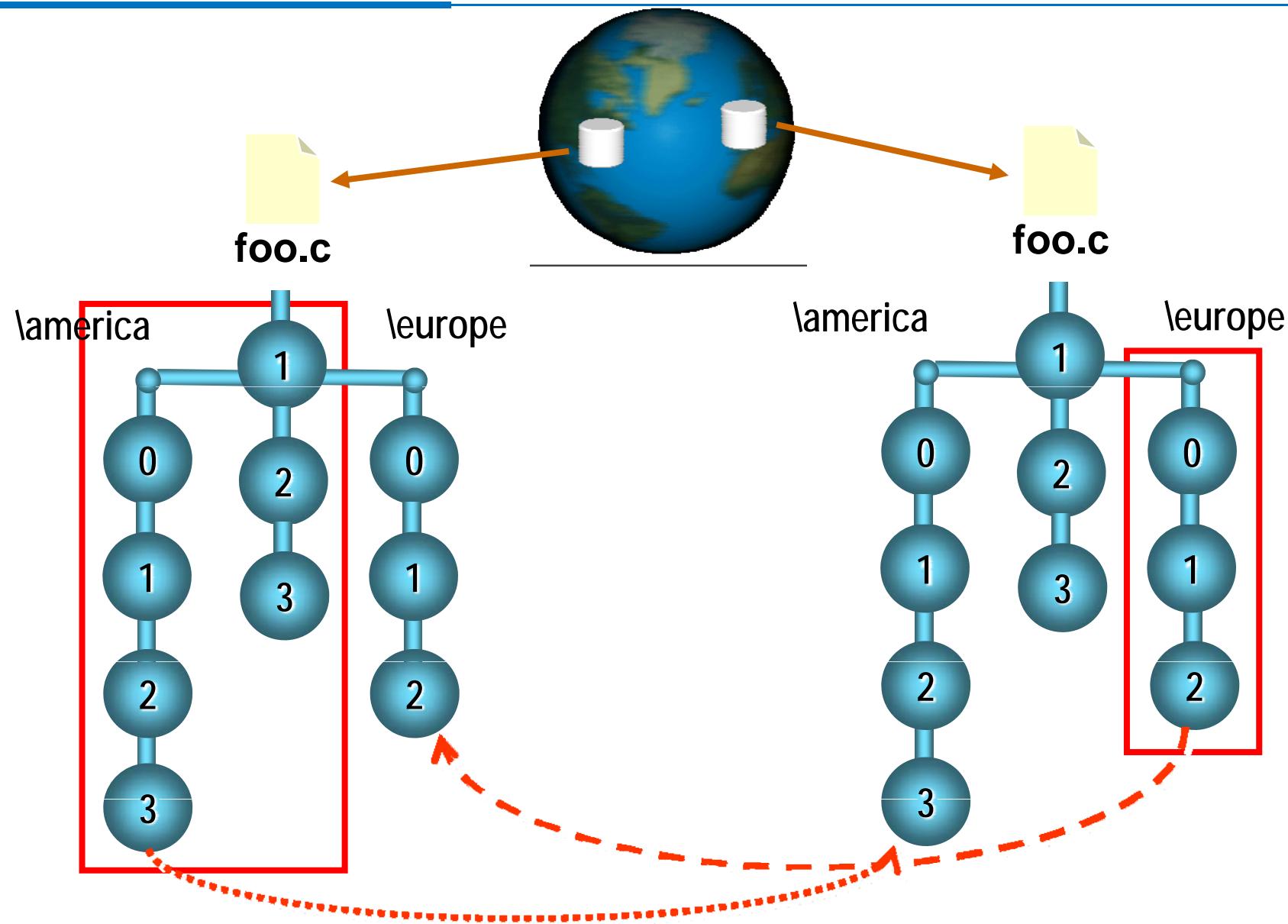
# ClearCase: Verteile Entwicklung

---



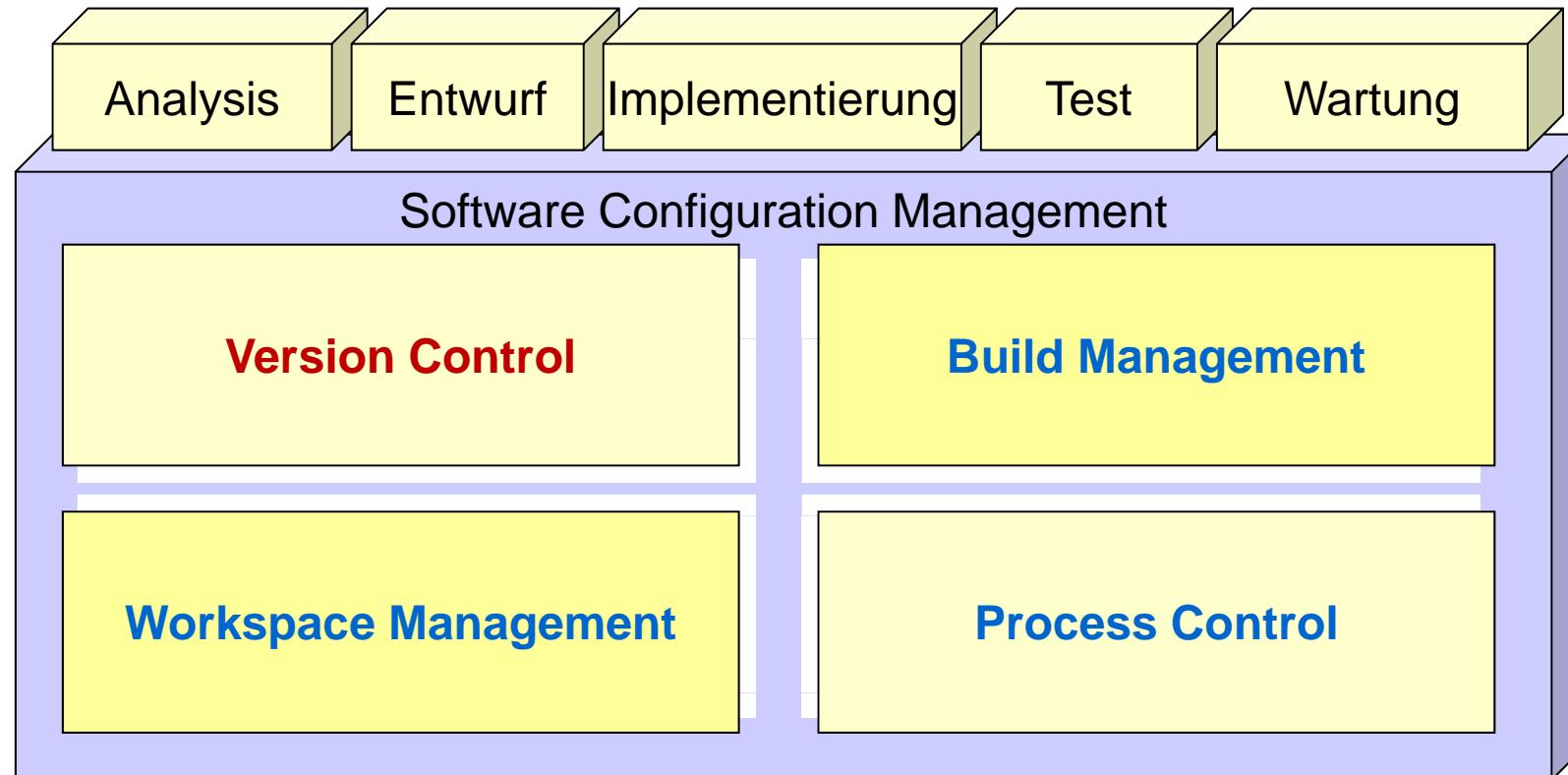
- Paralleles Entwickeln mit geographisch verteilten Projekt-Teams
- Automatische Updates und Kopien der VOBs
  - ◆ mit oder ohne Netzzugriff
- Nur ein Team hat "Mastership" pro branch

# ClearCase® Workspace Management: Multi Site Parallel Development



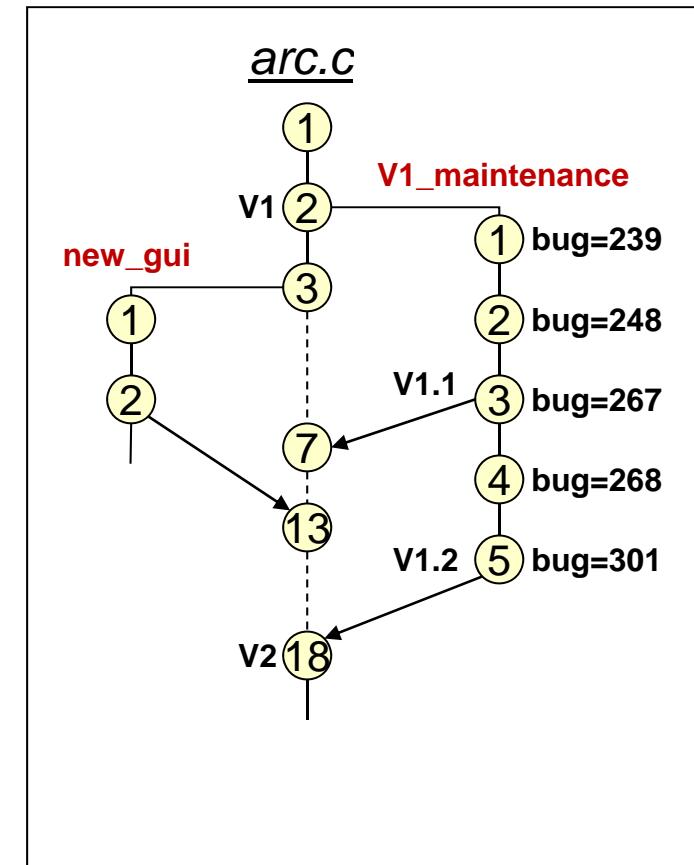
# SCM: Das Fundament des Entwicklungsprozesses

---



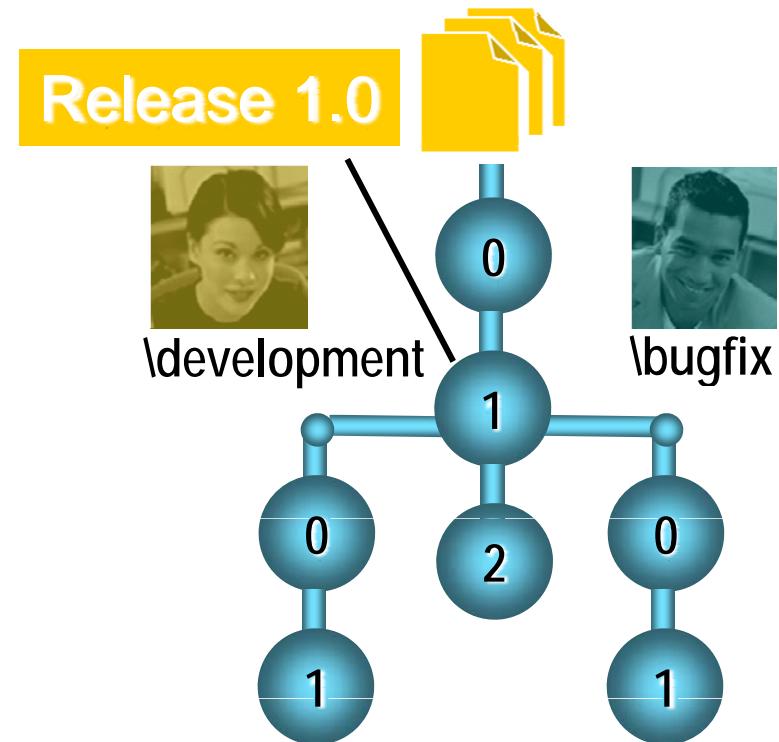
# Versionskontrolle: Branching (1)

- „Branching Graph Model“
  - ◆ Darstellung der wichtigsten Entwicklungslinien und Change Sets
  - ◆ **symbolische Namen** zum besseren Verständnis und einfacheren Referenzieren
  - ◆ jeder Zweig (Branch) hat zusätzliche administrative Annotationen
    - ⇒ symbolischer Name
    - ⇒ Verweis auf Abspaltungspunkt
    - ⇒ Kommentare
    - ⇒ ...



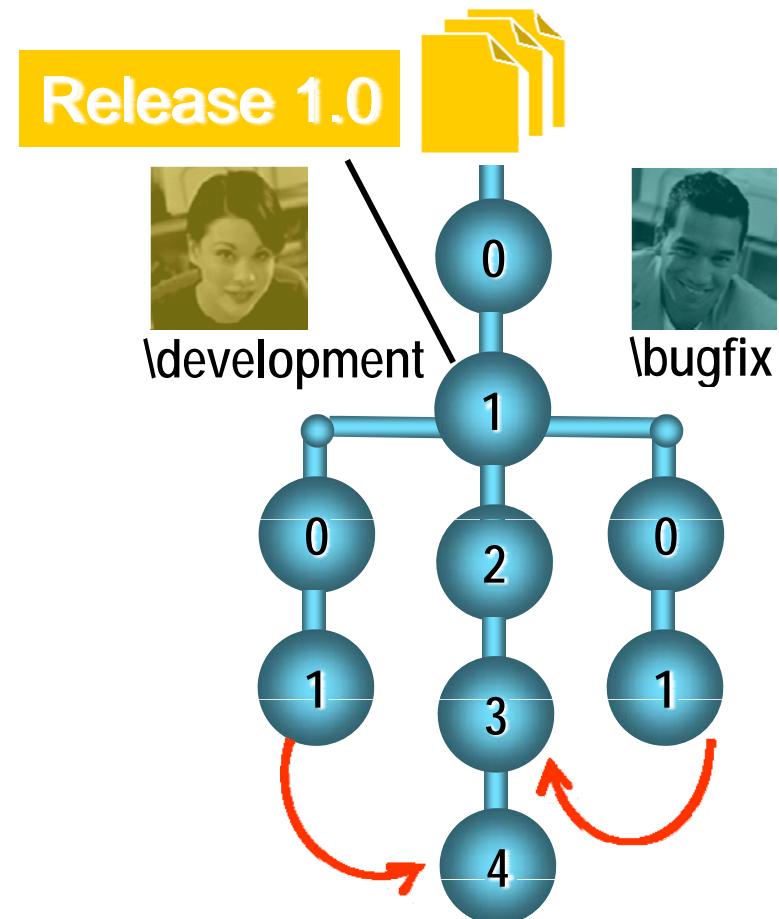
# Versionskontrolle: Branching (2)

- Uneingeschränktes branching wird unterstützt
- Automatisches branching, wann immer es nötig ist
- Jedes Mitglied hat einen isolierten Arbeitsbereich
- Verschiedene Arbeitsbereiche für verschiedene Aktivitäten
- Erlaubt parallele und durchgängige Entwicklung



# Versionskontrolle: Merging (1)

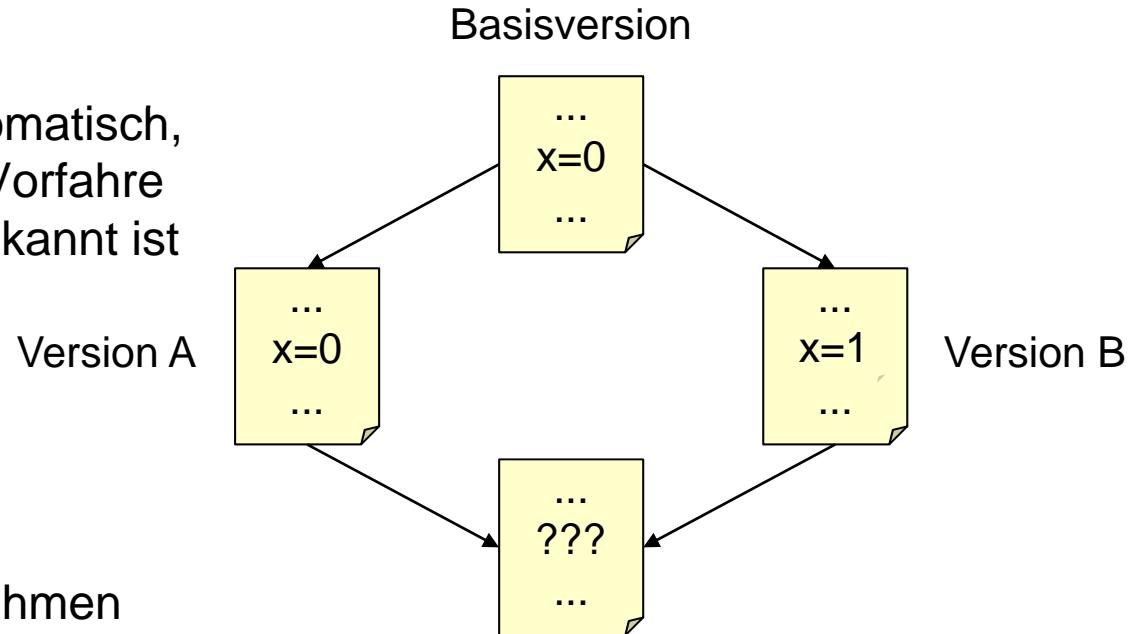
- Isolierte Arbeitsbereiche werden integriert
- Fördert und schafft Transparenz
- Ein *Merge Manager* unterstützt automatisches Merging und hilft beim Auflösen eventueller Konflikte
- Merging ist in alle Richtungen möglich



# Versionskontrolle: Merging (2)

- Problematisierung

- ◆ merging erfolgt automatisch, falls gemeinsamer Vorfahre der zwei Dateien bekannt ist



- Idee

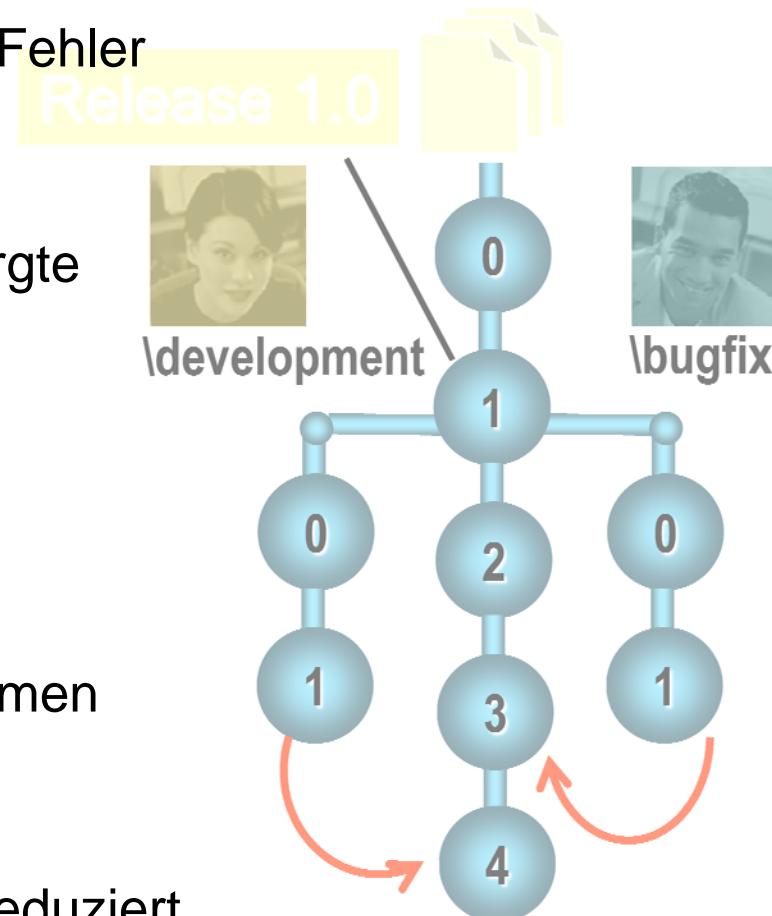
- ◆ Änderungen übernehmen

- Experimentelle Resultate

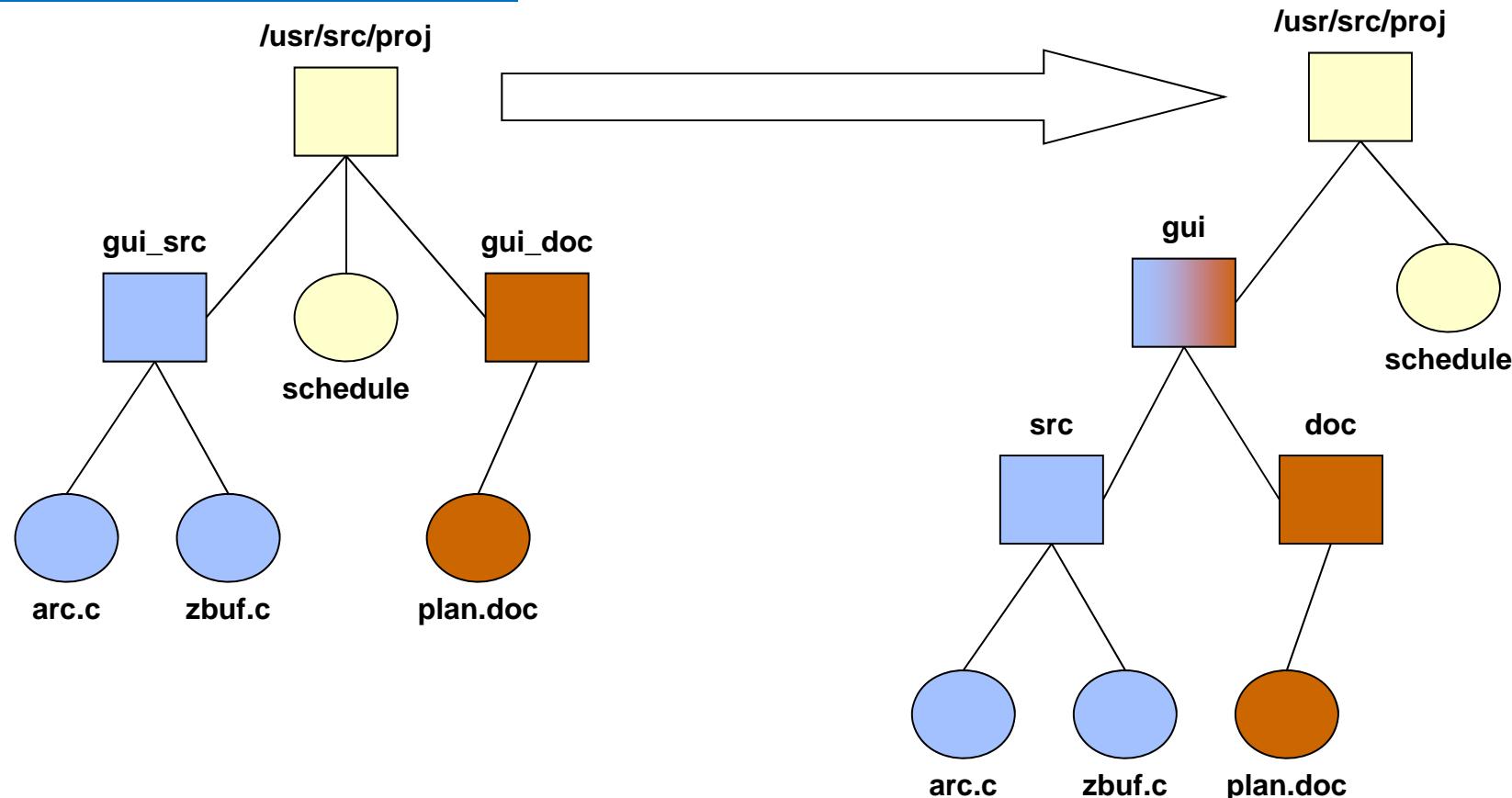
- ◆ In ca. 90% der Fälle konnte das Zusammenführen ohne Rückfragen durchgeführt werden.
  - ◆ Ca. 1% der automatischen Zusammenführungen war fehlerhaft, was meist beim Übersetzen entdeckt wurde (Syntaxfehler)

# ClearCase® Versionskontrolle: Branching & Merging

- Isoliert risikante Entwicklungen und Tests
- Stellt sicher, dass einmal behobene Fehler für immer verschwinden
- Merge-Utility findet noch nicht gemergte Dateien
- Änderungen an Releases werden transparent
- Releases auf verschiedenen Plattformen werden ermöglicht
- Integrationsaufwand wird drastisch reduziert



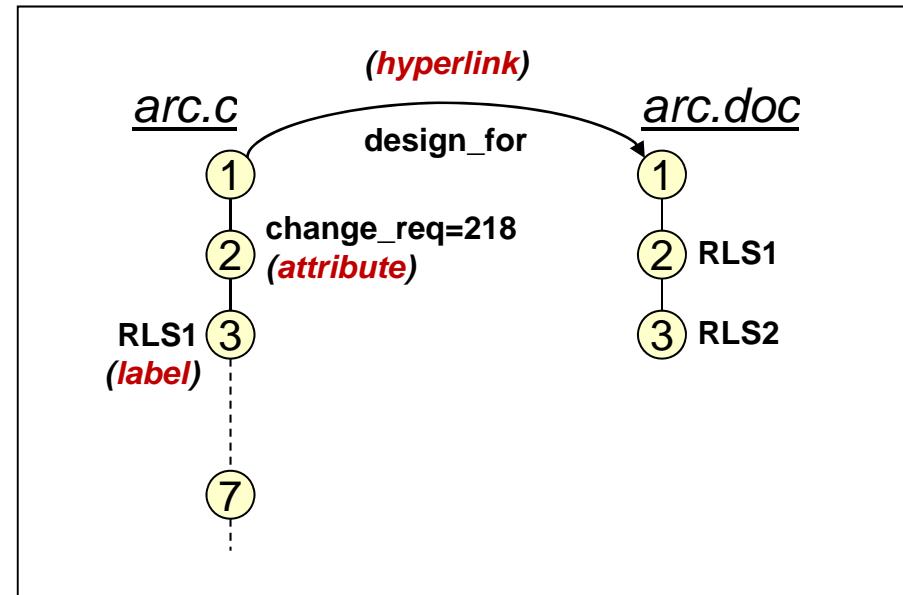
# Versionierung von Ordner



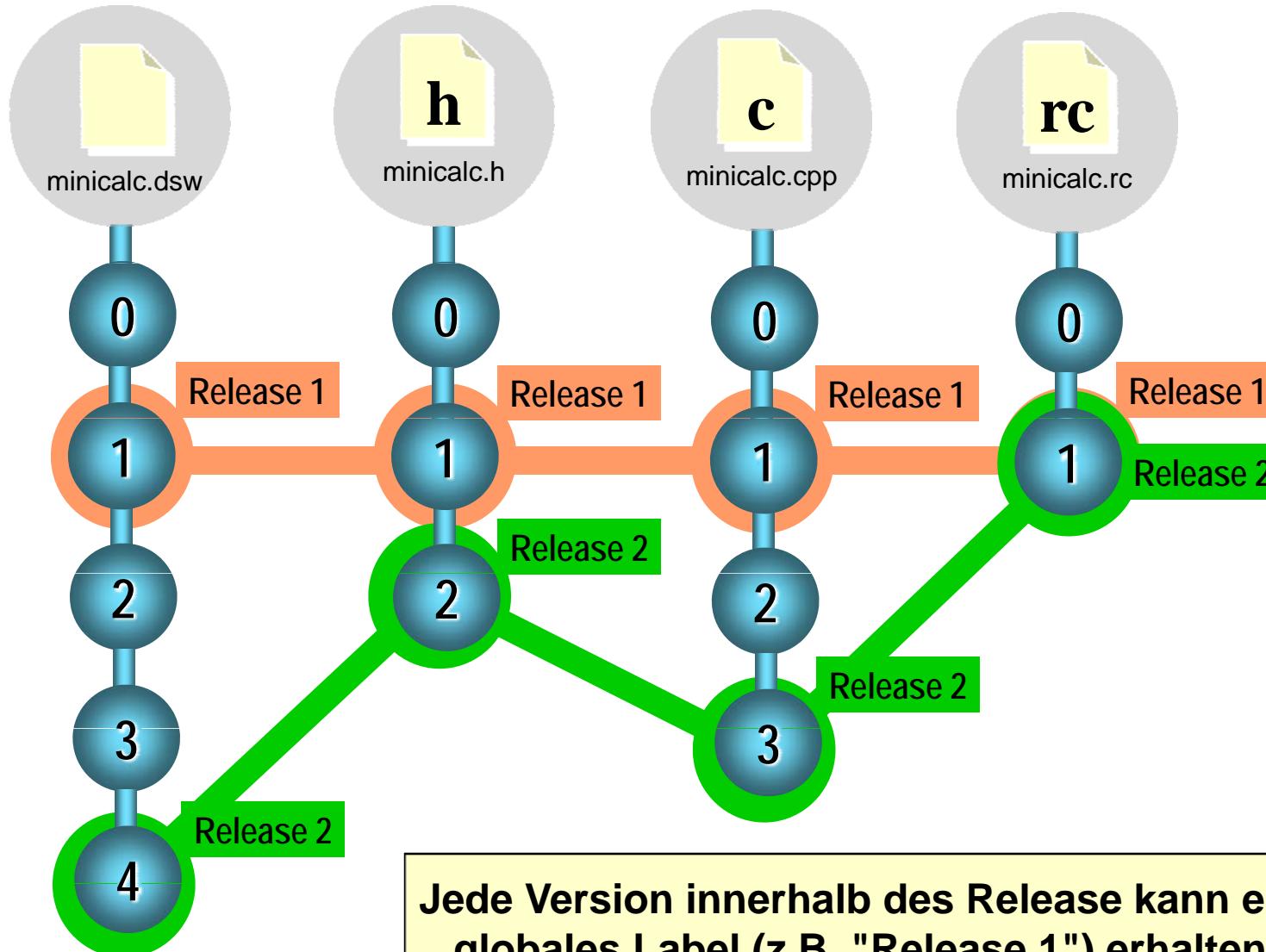
- Obige Änderung entspricht einer neuen Version des Ordners `/usr/src/proj/`

# Versionskontrolle (3): Erweiterungsmöglichkeiten

- Labels
  - ◆ symbolische Namen
- Attribute
  - ◆ zusätzliche Anmerkungen
- Hyperlinks
  - ◆ Beziehungen
- Trigger
  - ◆ ECA-Regeln
- Change Sets
- Annotationen (auf Code-Ebene)
- Namespaces
- ...



# ClearCase Baselining



# ClearCase® Übersicht

---

**Version Control**

**Build Management**

**Workspace Management**

**Process Control**

# Build Management: Mit herkömmlichen Werkzeugen (make)

---

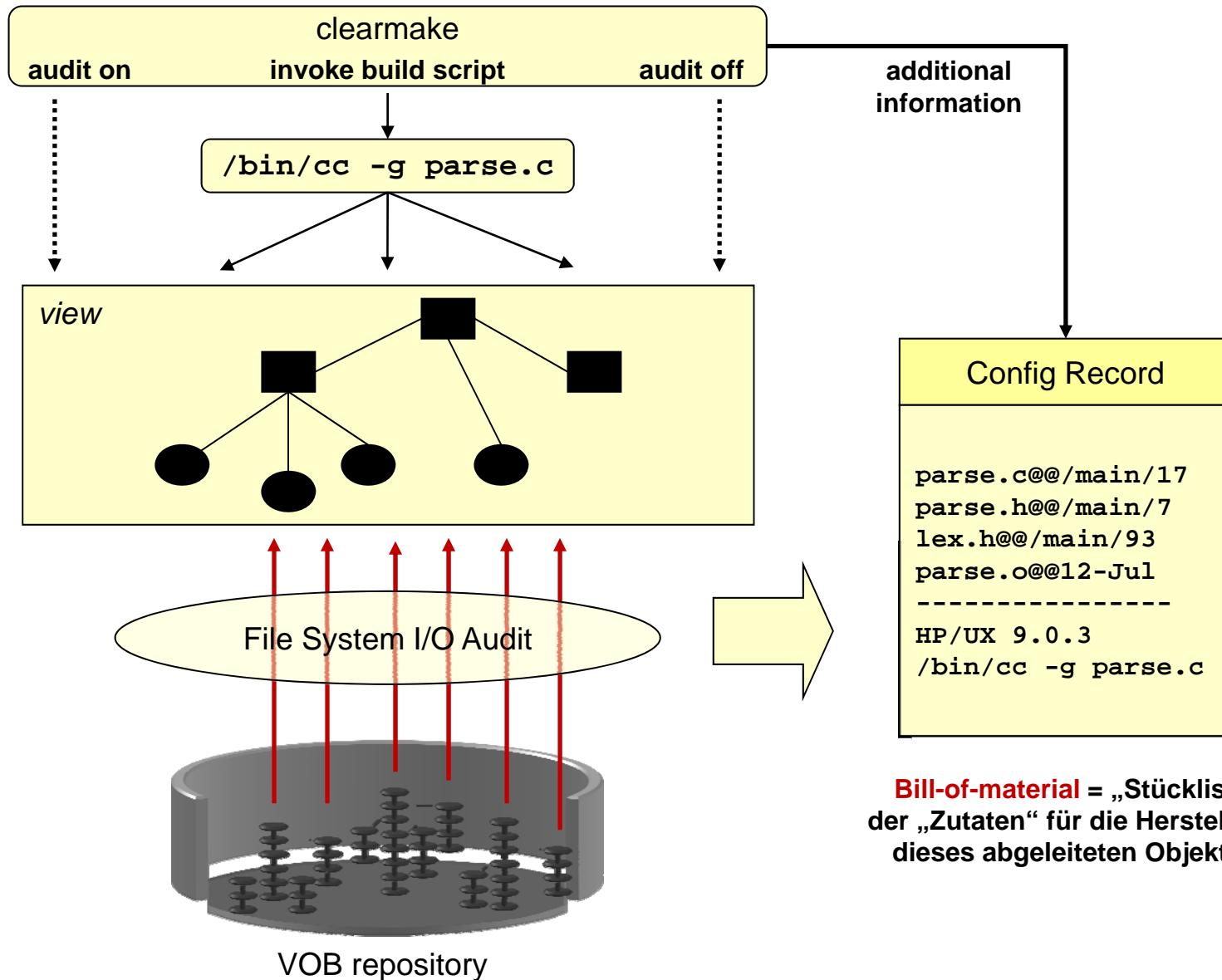
- Keine automatisierte Abhängigkeitsanalyse
  - ◆ Abhängigkeiten werden vom Benutzer manuell beschrieben (makefile)
  - ◆ Das ist aber in einer sich dynamisch ändernden Umgebung sehr aufwendig und fehleranfällig
- Kein „Bill of materials“
  - ◆ unklar ob zwei abgeleitete Produkte gleichen Namens (foo.o) wirklich gleich sind, da nicht klar ist, ob sie auch aus den gleichen „Zutaten“ und nach der gleichen „Rezeptur“ erstellt wurden
- Kein „Binary sharing“
  - ◆ Das "normale" Make weiß nichts darüber, dass ein Kollege evtl. ein aufwendig zu erstellendes abgeleitetes Objekt schon erzeugt hat

# Build Management: Mit ClearCase

---

- Verwaltung von Informationen zum **reproduzierbaren Erstellen von abgeleiteten Objekten**
- Technische Grundlage
  - ◆ Durch das VFS ist es möglich, die I/O-Zugriffe vom Compiler oder anderen Werkzeugen auf alle beteiligten Objekte erfassen und protokollieren zu lassen.
  - ◆ **clearmake** erledigt diese Aufgabe
- Protokollierung der benötigte Daten (**Auditing**) → Bestandsliste (**bill-of-materials**)
  - ◆ Korrekte Versionen aller Quell-Dateien und anderer beteiligter Objekte
  - ◆ Namen und Versionen der verwendeten Werkzeuge
  - ◆ benutzte Parametereinstellungen
- Resultierende nützliche Eigenschaften
  - ◆ Abgeleitete Objekte können von verschiedenen Benutzern genutzt werden (**binary sharing**)
  - ◆ Nur die notwendigsten Operationen zum Erstellen eines abgeleiteten Objekts müssen durchgeführt werden (**minimal rebuilding**)

# Build Management: Lösung mit ClearCase



# ClearCase® Build und Release Management

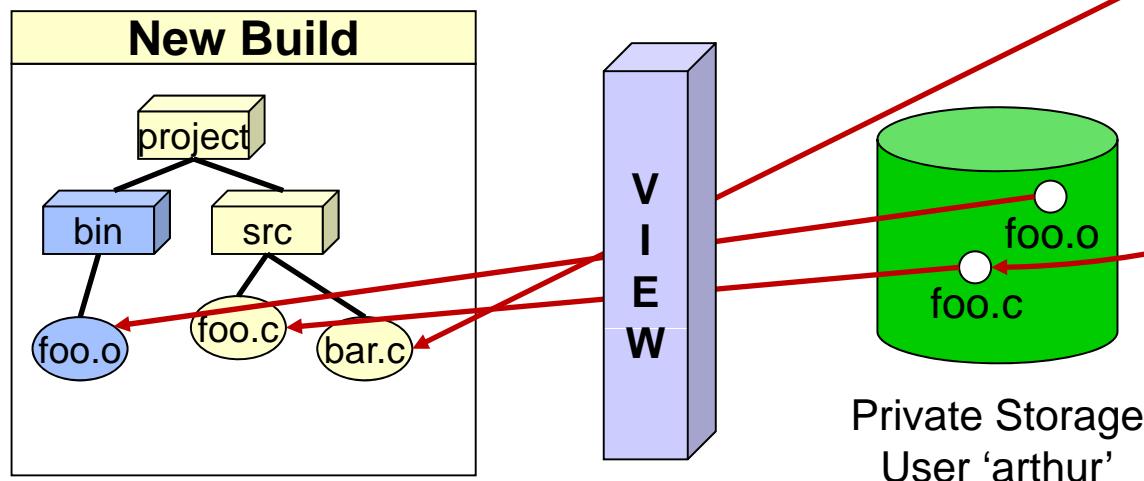
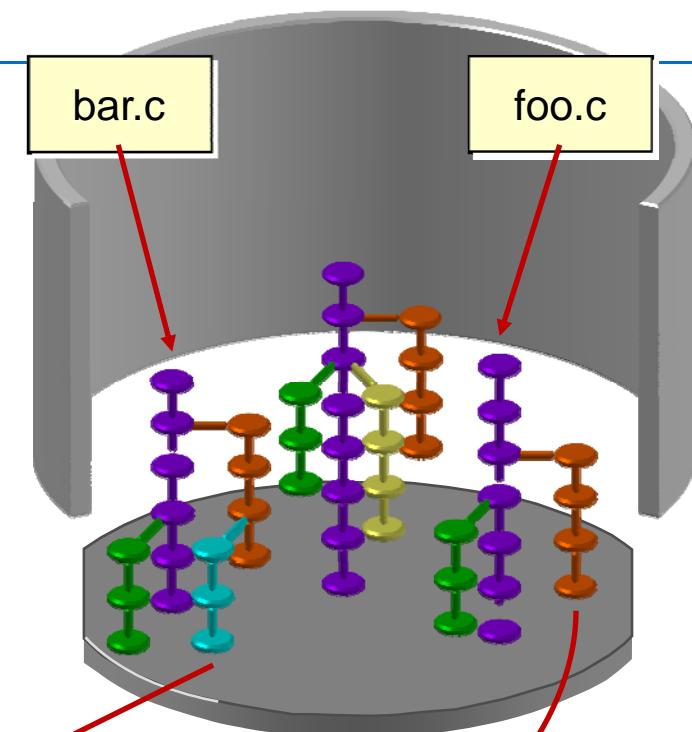
---

- Build Maintenance
  - ◆ Automatisches Erkennen von Abhängigkeiten
  - ◆ Stellt sicher, dass ein Build die richtigen Versionen einzelner Elemente beinhaltet
  - ◆ Konfigurationsprofile bestimmen exakt jede Version der benutzten Elemente

# ClearCase® Build und Release Management

- Build Auditing (Protokollierung)
  - ◆ Reproduzierbarkeit von builds durch Anfertigen von "Stücklisten"

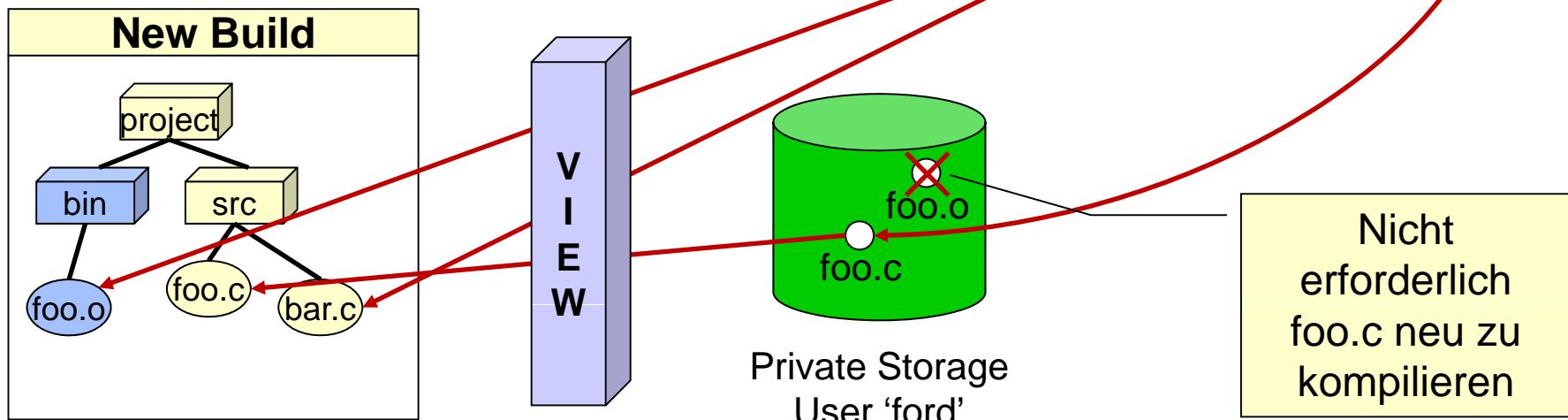
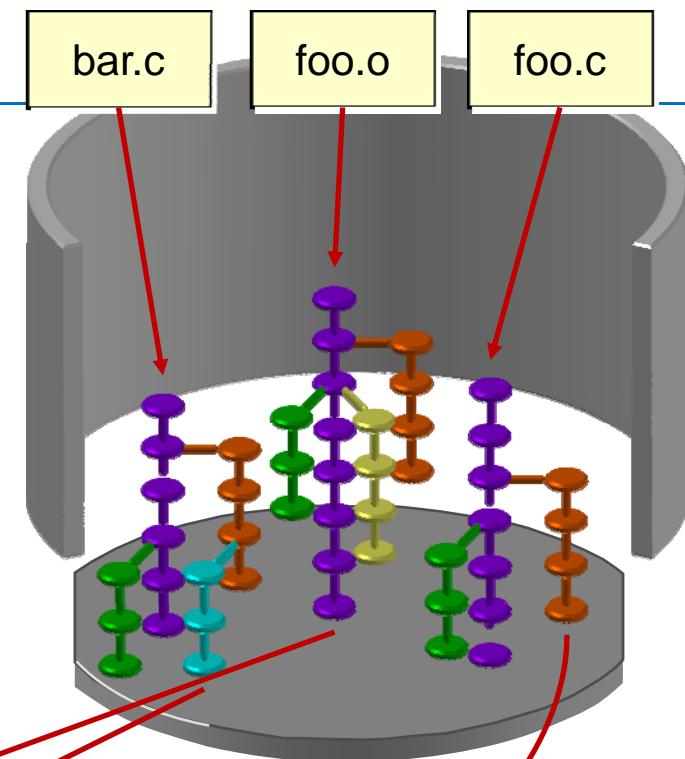
```
Target foo.o built on host 'falcon' by arthur
Reference Time 4-June-2001, 1:30:42
View was falcon:/home/falcon/arthur/myview
/usr/vob/src@@/main/12
/usr/vob/src/foo.c@@/main/bugfix/4
/usr/vob/src/foo.h@@/main/17
/usr/vob/src/foo.h@@main/9
Build Script:
cc -g -o foo foo.c
```



# ClearCase® Build und Release Management

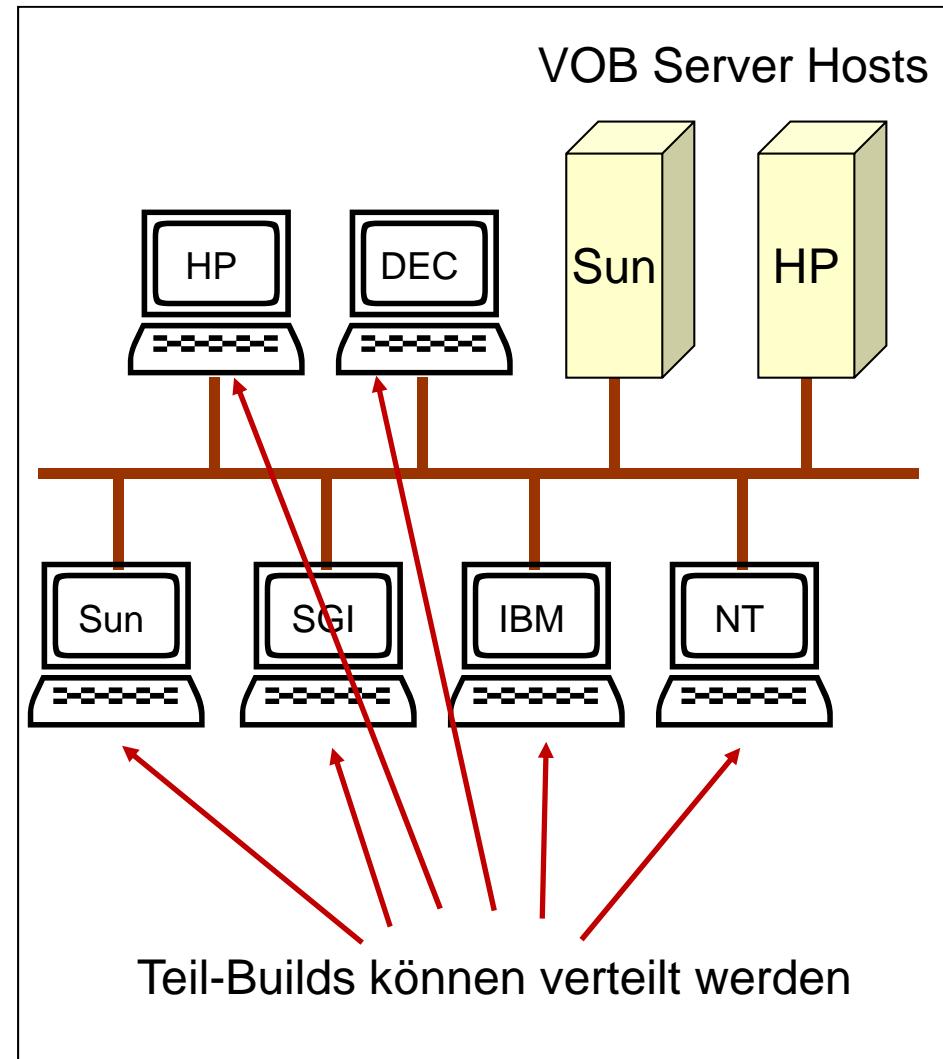
- Build Avoidance (Vermeidung)
  - ◆ Entwickler können "abgeleitete Objekte" teilen

```
Target foo.o built on host 'falcon' by ford
Reference Time 5-June-2001, 9:14:42
View was falcon:/home/falcon/ford/myview
/usr/vob/src@@/main/12
/usr/vob/src/foo.c@@/main/bugfix/4
/usr/vob/src/foo.h@@/main/17
/usr/vob/src/foo.h@@main/9
Build Script:
cc -g -o foo foo.c
```



# ClearCase® Build And Release Management

- Build Performance
  - ◆ Ermöglicht parallele und verteilte builds über das Netzwerk (UNIX)



# ClearCase® Build und Release Management: Zusammenfassung

---

- Build Maintenance
  - ◆ Automatisches Erkennen von Abhängigkeiten
- Build Auditing (Protokollierung)
  - ◆ Reproduzierbarkeit von builds durch Anfertigen von "Stücklisten"
- Build Avoidance (Vermeidung)
  - ◆ Ermöglicht das Teilen von "abgeleiteten Objekten" zwischen Entwicklern
- Build Performance
  - ◆ Ermöglicht parallele und verteilte builds

# ClearCase® Übersicht

---

**Version Control**

**Build Management**

**Workspace Management**

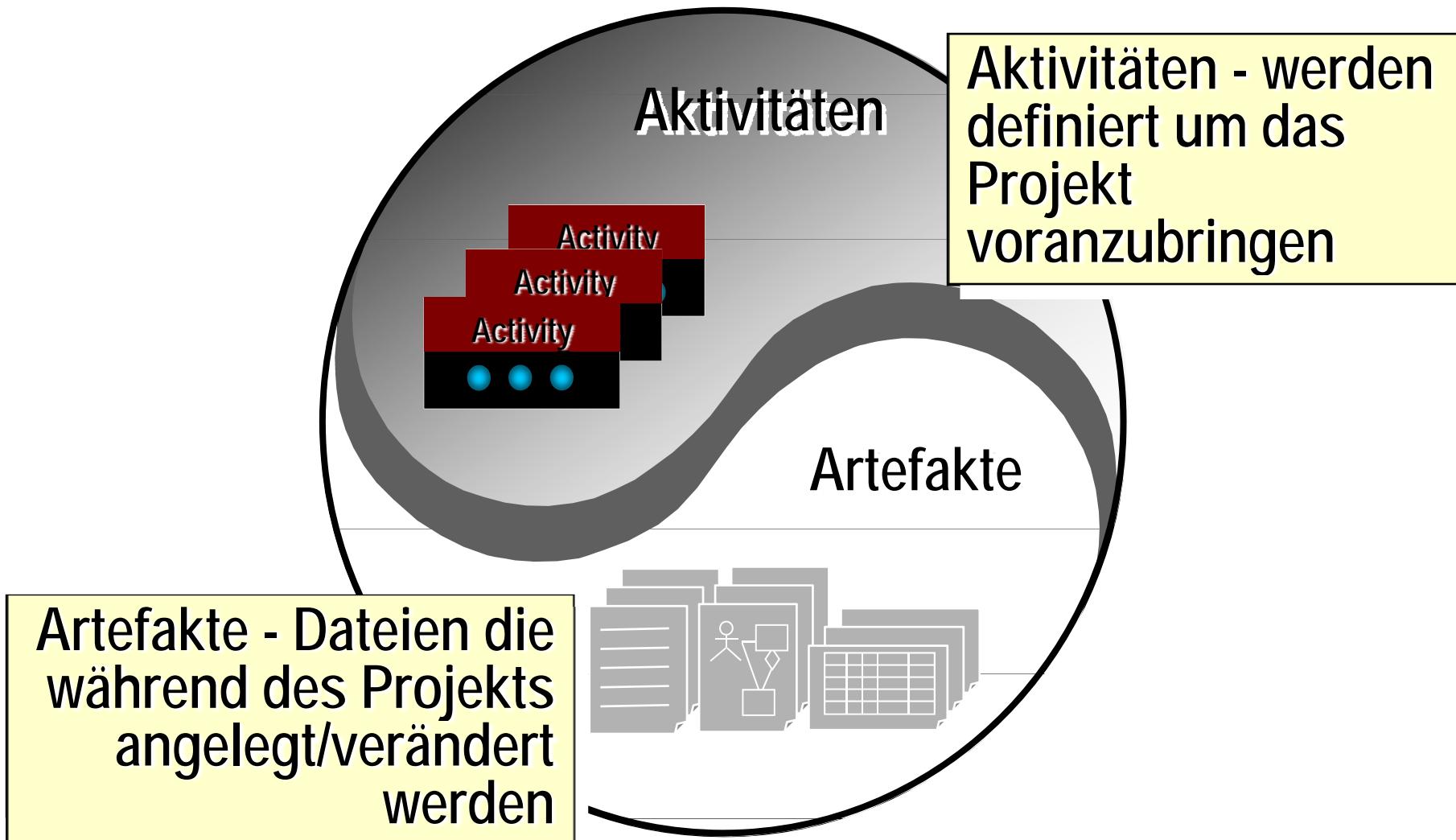
**Process Control**

# Process Control

---

- Entwicklungsprozeß umfaßt Analyse, Design, Implementierung und Wartung des Software-Produkts
- SCM ist nur ein Teil in diesem Prozeß neben
  - ◆ der Eingrenzung von Problemen,
  - ◆ der Definition von Maßnahmen,
  - ◆ dem Erstellen von Zeitplänen,
  - ◆ dem (automatischen) Testen,
  - ◆ dem Messen von Software-Eigenschaften
  - ◆ und anderen Aufgaben.
- Geeignete Werkzeuge zur Automatisierung von Entwicklungs-Prozessen sind sog. "**Workflow Manager**“.
- Deren grundlegendes Konzept ist **Action-Response**, d.h. die Bearbeitung von Teilaufgaben kann andere Teilaufgaben beeinflussen oder anstoßen.

# Die Lösung: Unified Change Management



# Die Lösung: Unified Change Management



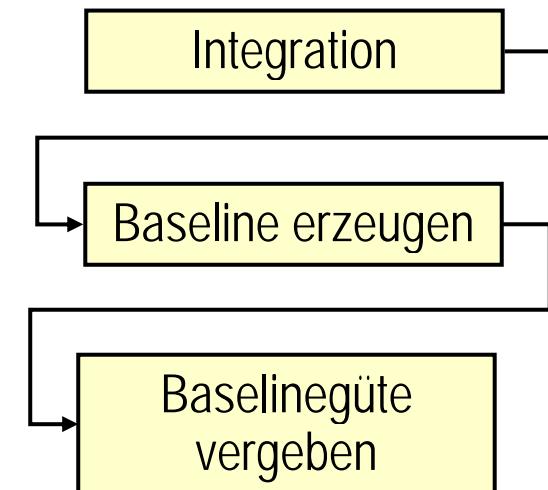
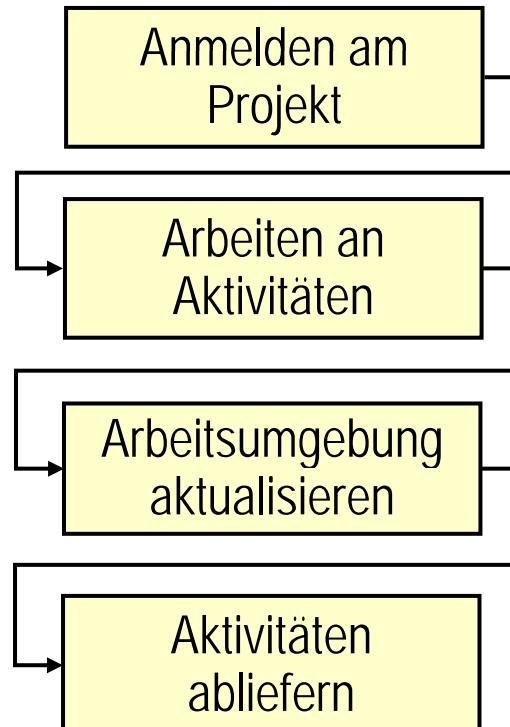
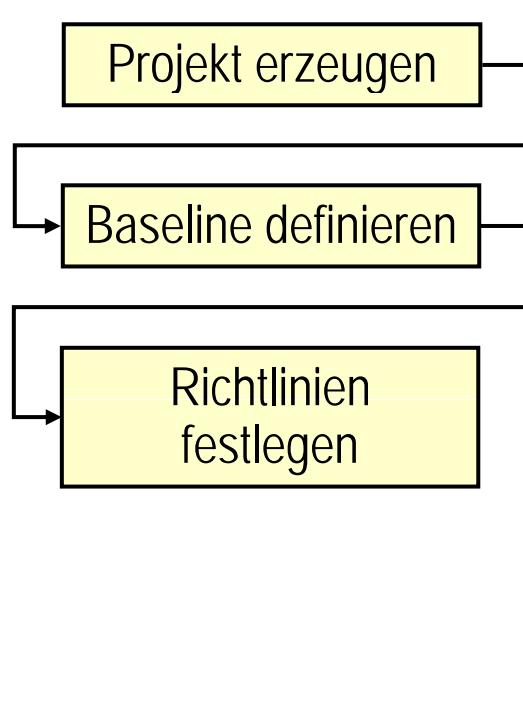
Projekt Manager



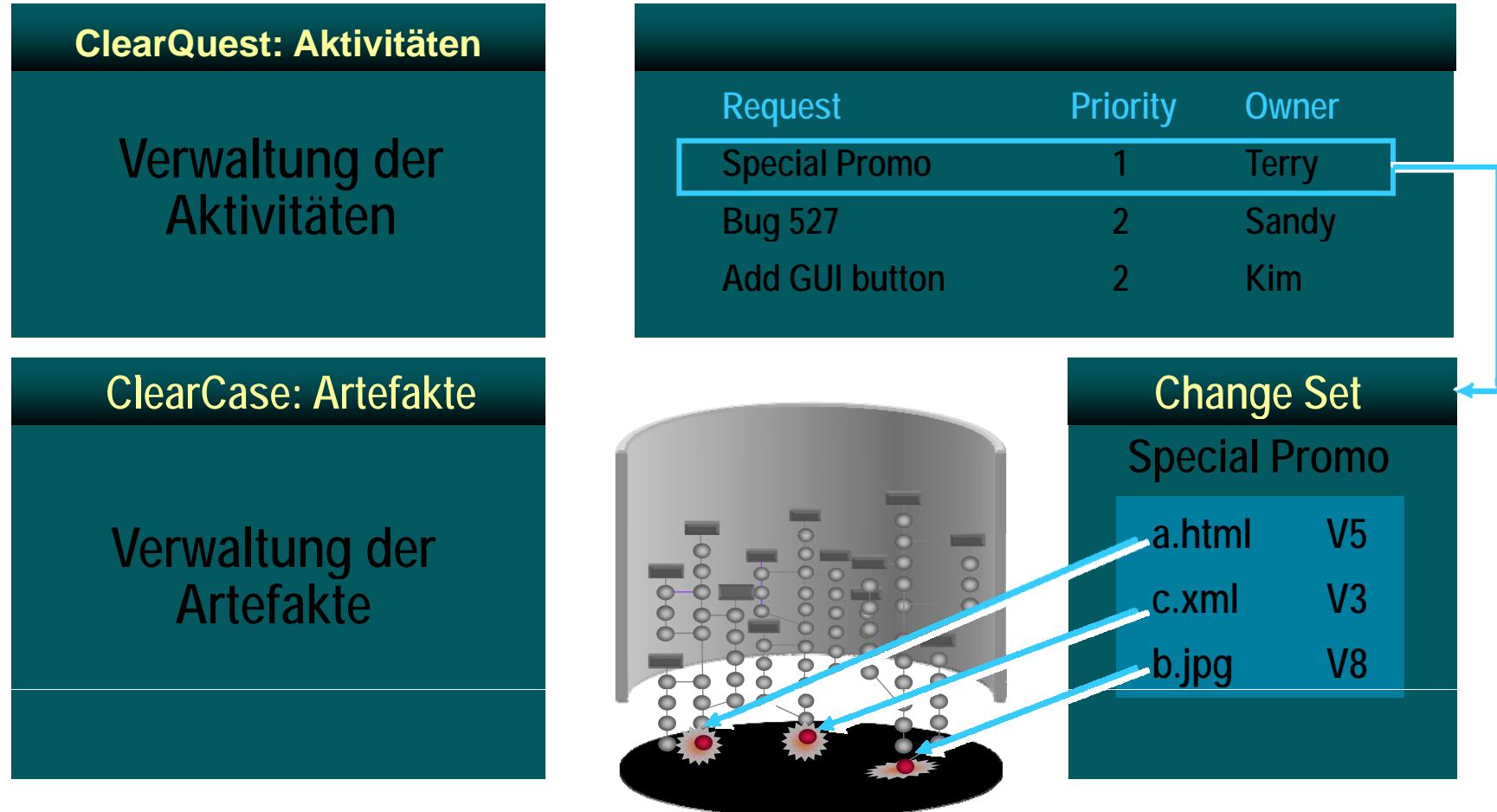
Entwickler



Integrator



# UCM: Verbindung zwischen Aktivitäten und Artefakten



# ClearCase® Zusammenfassung

---

## Version Control

- Full Branching & Labeling
- All File System Elements
- Directories
- Graphical Merge & Compare
- Conversion Utilities

## Build Management

- makefile compatible
- Automatic Dependency Mgmt
- Build Auditing - Config Records
- Parallel, Distributed Building
- Binary Sharing

## Workspace Management

- Rule-based Configurations
- Dynamic Evaluation
- Transparent Access
- Scalable
- Fully Distributed

## Process Control

- Attributes
- Hyperlinks
- Pre-Event, Post-Event Triggers
- Locks & Permissions
- Completely Customizable

# ClearCase Summary

---

## ClearCase Pros

- Industrial strength
- Excellent merge tools
- Proven reliability
- Scales up well
- Good GUI on Windows

## ClearCase Cons

- Very expensive
- Heavyweight server and client
- Very steep learning curve
- High administration burden
- Server communication over a high latency link is SLOW
- All merges are server based
  - ◆ You can't merge or diff without connectivity to the servers
  - ◆ Merges over high latency links are SLOW
- You need to do many things the ClearCase way
- Weak GUI on Unix platforms

# Zusammenfassung: SCM De Luxe

---

- Virtuelles Dateisystem
  - ◆ → Transparenz der Datenhaltung
- Regelbasierte Konfiguration des Arbeitsbereiches
  - ◆ → Flexibilität
- Dynamische Regelauswertung
  - ◆ → Aktualität
- Automatisches Merging
  - ◆ → Geringer Integrationsaufwand
- Build Management
  - ◆ → Reproduzierbare „build“-Ergebnisse, Effizienz durch „minimal rebuilding“
- ECA-Regeln
  - ◆ → Workflow-Management
- Verteilung, automatische Spielelung, verteiltes Building
  - ◆ → Unterstützung für internationale Unternehmen