

# Einführung in die Programmierertechnik

## Versionsverwaltung

# Software Configuration Management (SCM)

- Aufgaben:
  - Verwaltung des checkin/checkout von Quellen
    - Festhalten von Zeit, Autor, Zweck und Inhalt von Änderungen
  - Steuerung der Softwareproduktion (builds)
  - ...
- Ziele:
  - Investitionsschutz, sogar falls Hauptentwickler die Firma verlassen
  - Wiederherstellung alter (fehlerhafter) Versionen (Konfigurationen)
    - zur Fehleranalyse
    - zur Bestimmung der funktionalen Unterschiede zwischen zwei Versionen
  - Vereinfachung des Produktionsprozesses
  - Unterstützung von formalen Änderungsprozessen (change management)
  - Unterstützung von gleichzeitiger Arbeit mehrerer Entwickler an einem Projekt
    - eventuell global verteilt

# Revision Control

- **Aspekt von SCM**
  - oft synonym zu SCM verwendet
- **Revision: benannte Änderung eines Dokuments**
  - evtl. change sets: Gruppe von Änderungen, die alle den gleichen Namen bekommen
  - Name der Revision: Nummer, evtl. hierarchisch
    - 1, 2, 3, 4
    - 1.1, 1.2, 1.3, 1.3.1.4, ...

# Bekannte Versionsverwaltungssysteme

- Bazaar (Canonical/GNU-Projekt, Martin Pool u.a.)
- BitKeeper (BitMover, Larry McVoy)
- ClearCase (IBM, ehemals Rational, ehemals Atria, ehemals Apollo)
- CVS (Concurrent Versions System, Brian Berliner, Jeff Polk u.a.)
- git (Linus Torvalds u.a.)
- Mercurial (Matt Mackall)
- Perforce (Perforce Software)
- PVCS (Silver Lake Partners, ehemals Serena, Merant, Intersolv, ...)
- RCS (Revision Control System, Walter F. Tichy)
- SCCS (AT&T Bell Labs)
- Subversion (CollabNet)
- TeamWare (Sun, Larry McVoy)
- Visual SourceSafe (Microsoft)
- Visual Studio Team Foundation Server (Microsoft)

# Begriffe

- **Repository: Speicher für die Revisionen**
  - Zugänglich lokal oder über Netz
  - Dateien oder relationale Datenbanken
  - Revisionen oft als Änderungen (Deltas) gegenüber anderen Revisionen gespeichert
- **Sandbox/working copy: Lokale Kopie einer Revision**
  - eventuell mit lokalen Änderungen
- **Commit (Checkin): Übertragen lokaler Änderungen in Repository**
- **Checkout: Anlegen einer Sandbox**
  - update: Integration von Änderungen im Repository in eine Sandbox
- **Locking: Sperren eines Dokuments zur Bearbeitung**
  - “concurrent versioning”: Gleichzeitige Bearbeitung des Dokuments ist erlaubt
    - Merging: Integration mehrerer Änderungen
- **Change set: Menge zusammengehörender Änderungen**
- **Tags: Benannter Versionsstand des Gesamtsystems**
- **Branch: Seitenzweig der Entwicklung**

# Versionsverwaltung mit Subversion

- [subversion.apache.org](http://subversion.apache.org)
- Entwickelt mit dem Ziel, CVS abzulösen
  - Behebt Mängel von CVS
    - Umbenennen von Dateien
    - Löschen von Verzeichnissen
    - Unterstützung von Change sets
    - Effizientes diff
    - Integration in andere Werkzeuge
- Client-Server-Architektur
  - Mehrere Zugriffsmethoden auf das Repository: lokale Dateien, svnserve-Protokoll (eventuell über SSH), WebDAV (über http oder https)
  - Verschiedene Authentifizierungs- und Autorisierungsmechanismen (Nutzername/Passwort, Zertifikate)

# Subversion-Klienten

- /usr/bin/svn (Kommandozeile)
- RapidSVN (GUI: Windows, Linux)
- TortoiseSVN (Integration in Windows-Explorer)
- Subclipse (Integration in Eclipse)

# /usr/bin/svn

- import: Kopieren eines Verzeichnisbaums in Repository
  - Standardmäßig aktuelles Verzeichnis
- checkout (co): Checkout eines Teils des Repositories
  - Standardmäßig den gleichen relativen Pfad wie Repository
- commit (ci): Checkin
  - Standardmäßig aller geänderten Dateien im aktuellen Verzeichnis
  - Alternativ: Angabe der zu übertragenden Dateien
- diff: Anzeigen der Unterschiede (unified diff)
  - Standardmäßig zwischen Sandbox und Originalversion
- log: Anzeige der Versionsgeschichte
- blame (ann, annotate): Zuordnung zwischen Zeilen und Autoren
- copy (cp): Anlegen von Tags und Branches
  - “zero-copy”: Kopie ist lediglich Verweis auf Originaldokument
- ...

# Entwicklungsprozess mit Subversion

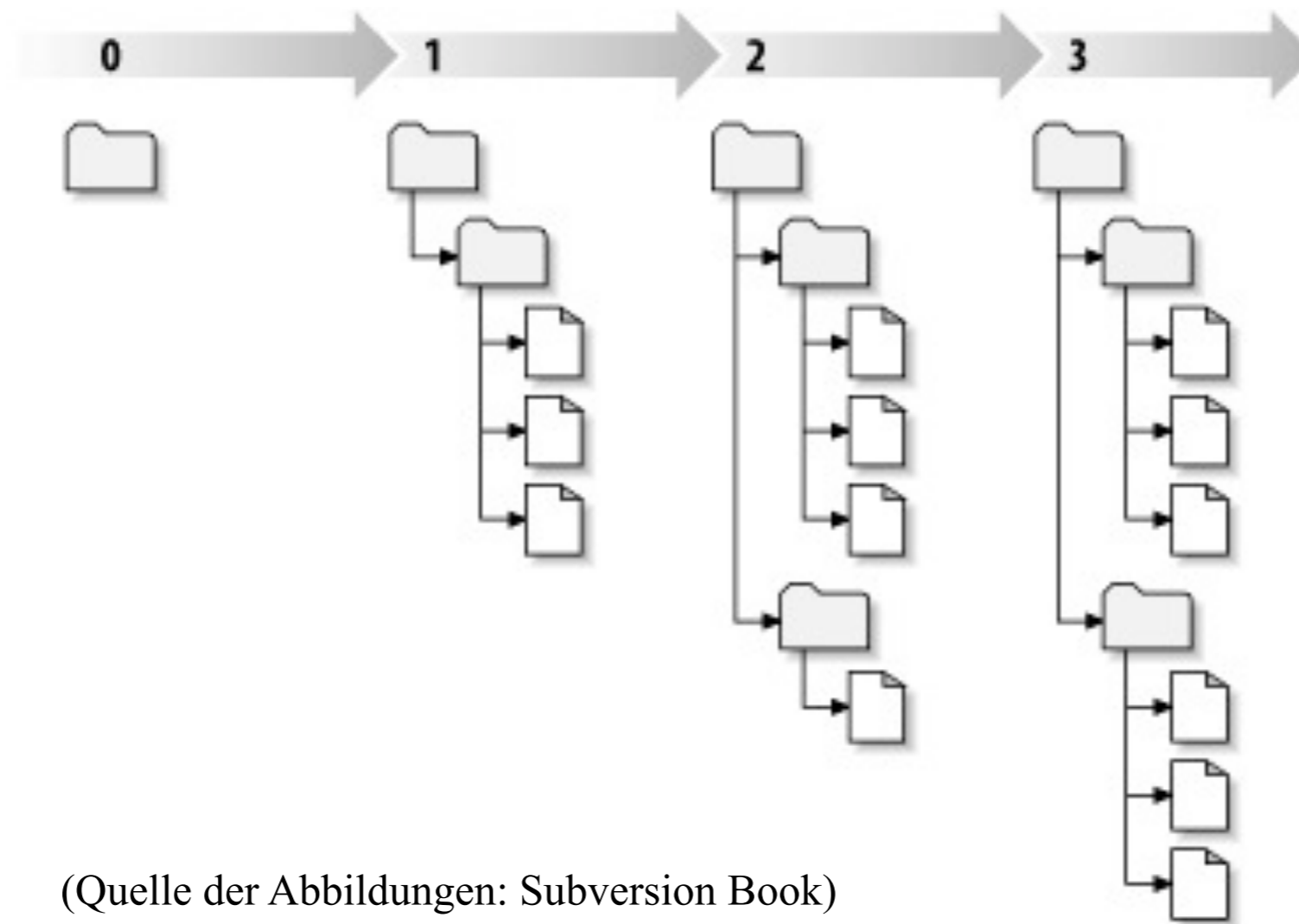
1. Importieren existierender Quellen in Repository (optional)
2. Checkout einer Arbeitskopie
3. Bearbeitung der Quellen (inhaltlich nicht zusammengehörende Änderungen möglichst zeitlich trennen, oder in verschiedenen Arbeitskopien)
  - evtl. Auswahl der Dateien in “svn commit”
4. Testen der Änderung
5. Checkin der Änderung, Fortsetzen mit 3.

# Entwicklungsprozess: Mehrere Entwickler

- Entwickler 1: Importieren der Quellen, Checkout einer Arbeitskopie, Beginn der Bearbeitung
- Entwickler 2: Checkout der Arbeitskopie, Beginn der Bearbeitung
  - Annahme: Entwickler 2 ist zuerst fertig
- Entwickler 2: Checkin (commit)
- Entwickler 1: Versuch des Checkin, Fehler: nicht im Besitz der aktuellen Quellen
- Entwickler 1: Update der Arbeitskopie auf den aktuellen Stand
  - eventuell: Behebung von Konflikten
- Entwickler 1: Erneuter Checkinversuch
- Entwickler 2: Update

# Versionen in Subversion

- revision number: natürliche Zahl
  - bezeichnet eine Version des gesamten Dateibaums im Repository



(Quelle der Abbildungen: Subversion Book)

# *Revision numbers und change sets*

- Jede Commit-Operation erzeugt eine neue *revision number*
- Commit-Operation betrifft u.U. mehrere Dateien
  - Anlegen, Löschen, umbenennen von Dateien und Verzeichnissen
  - Ändern von Dateien
- Gesamtheit der Änderungen pro Commit: Changeset
- Änderungen sollten logisch zusammengehören:
  - eine gemeinsame “commit message”
  - Einfaches Erfassen aller Änderungen in einem Changeset (svn info)
  - Einfaches Erfassen der Änderungen seit einer Version (svn ann, svn diff)
  - Einfaches Zurücknehmen einer Änderung (svn merge)

# Arbeitskopien

- Enthält Ausschnitt des Verzeichnisbaums
- Initial: alle Dateien in Arbeitskopie stammen aus gleicher *revision*
  - später: manche Dateien stammen aus neueren Versionen (etwa nach *commit*)
- Zustand einer Datei in der Arbeitskopie
  - unverändert und aktuell: keine Aktion erforderlich
  - lokal verändert und aktuell: letztlich “svn commit”
    - svn update hat keine Auswirkung
  - unverändert und veraltet: svn update
  - lokal verändert und veraltet: svn update, eventuell Konflikte beheben (svn resolved)
    - “svn commit” hier nicht erlaubt

# Arbeitskopien (2)

- Subversion speichert Zusatzdaten in Verzeichnis `.svn`:
  - Pfadname zum Repository
  - Versionsnummern der Dateien
  - Originaler Text der Dateien
    - “svn diff” erfordert keine Kommunikation mit dem Repository

# Basiskommandos

- Aktualisierung der Arbeitskopie: `svn update`
- Durchführung von Änderungen:
  - `svn add`
  - `svn delete`
  - `svn cp`
  - `svn move`
  - `svn mkdir`
- Betrachtung der Änderungen:
  - `svn status`
  - `svn diff`
  - `svn revert`
- Integration fremder Änderungen:
  - `svn update`
  - `svn resolved`
- Rückschreiben der Änderungen: `svn commit`

# Repository URLs

- Zugriff über HTTP, WebDAV:
  - `http(s)://server/path`
- Zugriff über lokales Dateisystem
  - `file:///path`
- Zugriff über Subversion-Protokoll
  - `svn://server/path`
  - `svn+ssh://server/path`

# svn update

- Buchstabencodes für durchgeführte Änderungen:
  - U: Datei aktualisiert
  - A: Datei oder Verzeichnis hinzugefügt
  - D: Datei oder Verzeichnis gelöscht
  - R: Datei oder Verzeichnis ersetzt (alte Version gelöscht, neue hinzugefügt)
  - G: Änderungen in lokale Version integriert (merGed)
  - C: Änderungen konnten nicht integriert werden (Conflict)
- Ähnliche Codes bei svn status:
  - M: lokale Datei verändert
  - ?: Subversion weiß nichts über Datei (und Datei steht auch nicht in der Liste ignoriertes Dateien - svn:ignore)
  - !: Datei steht unter Versionsverwaltung, ist aber nicht vorhanden
  - A, D, C, R wie bei update
  - ...

# Konfliktbehandlung

- Mehrere Änderungen an “der gleichen Stelle”
  - Heuristik, wann zwei Änderungen die gleiche Stelle betreffen
- Subversion legt Dateien für alle beteiligten Versionen:
  - filename.mine: lokale Änderungen
  - filename.rALT: Version, auf der lokale Änderungen basieren
  - filename.rNEU: aktuelle Version
  - filename: “Mischversuch”
- Subversion versucht, Änderungen zu mischen
  - *conflict marker* zeigen überlappende Änderungen an
  - i.d.R. Lösung des Konflikts durch Editieren der Markierungen
- Nach Behebung des Konflikts: `svn resolved`
  - löscht alle Hilfsdateien

# Beispiel: *conflict marker*

```
$ cat sandwich.txt
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<< .mine
Salami
Mortadella
Prosciutto
=====
Sauerkraut
Grilled Chicken
>>>>>> .r2
Creole Mustard
Bottom piece of bread
```

# Änderungsdateien

- diff(1)
- svn diff: Automatisch “unified diff”
  - ---: alte Version
  - +++: neue Version
  - - <zeile>: gelöschte Zeilen
  - + <zeile>: hinzugekommene Zeilen
- svn diff <dateiname>: Inspektion lokaler Änderungen
  - svn diff -rV1:V2 <dateiname>: Inspektion der Änderungen zwischen V1 (ausschließlich) und V2 (einschließlich)
- i.d.R.: Inspektion der Änderungen, dann commit
- read-only Repository: Senden der Änderungen an Schreibberechtigten
  - Dort: Integration mittels patch(1)

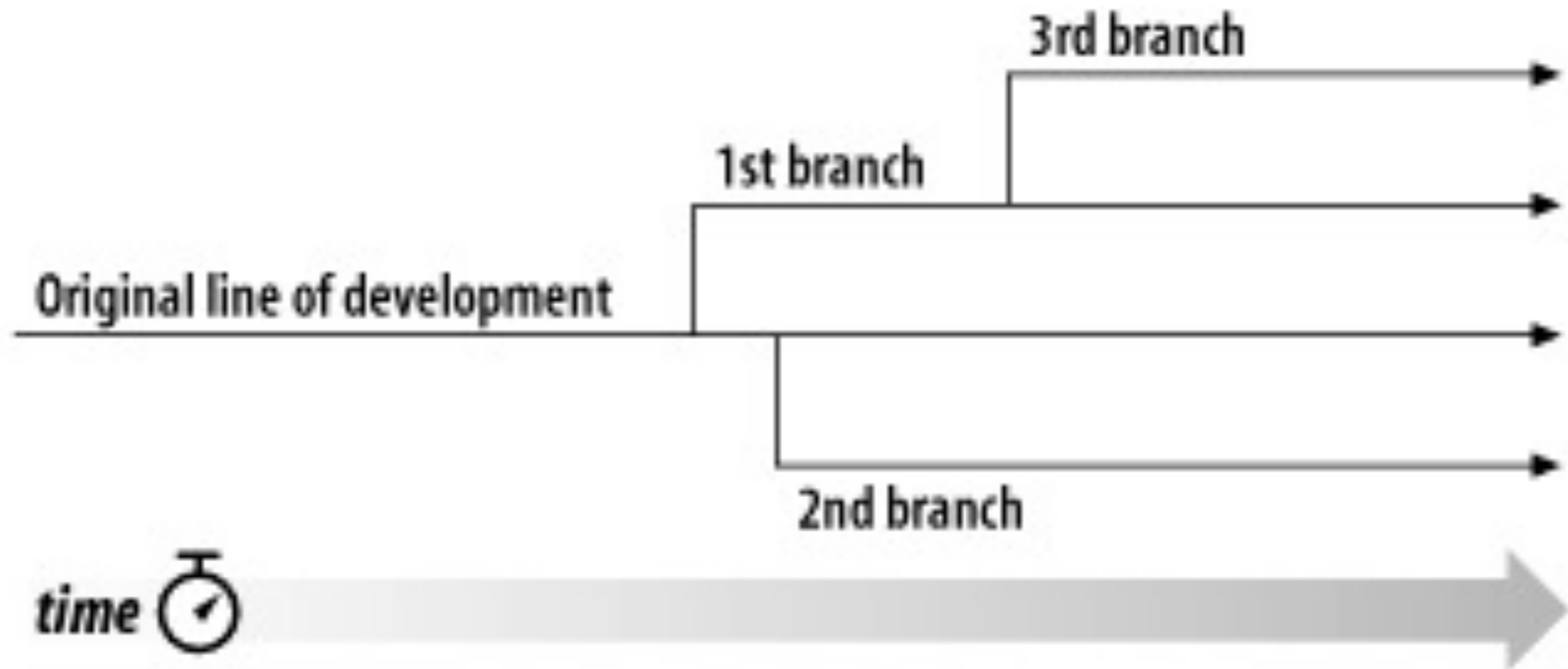
# Beispiel: Änderungsdateien

```
$ svn diff
Index: rules.txt
===== [ . . . ]
--- rules.txt (revision 3)
+++ rules.txt (working copy)
@@ -1,4 +1,5 @@
  Be kind to others
  Freedom = Responsibility
  Everything in moderation
- Chew with your mouth open
+ Chew with your mouth closed
+ Listen when others are speaking
$
```

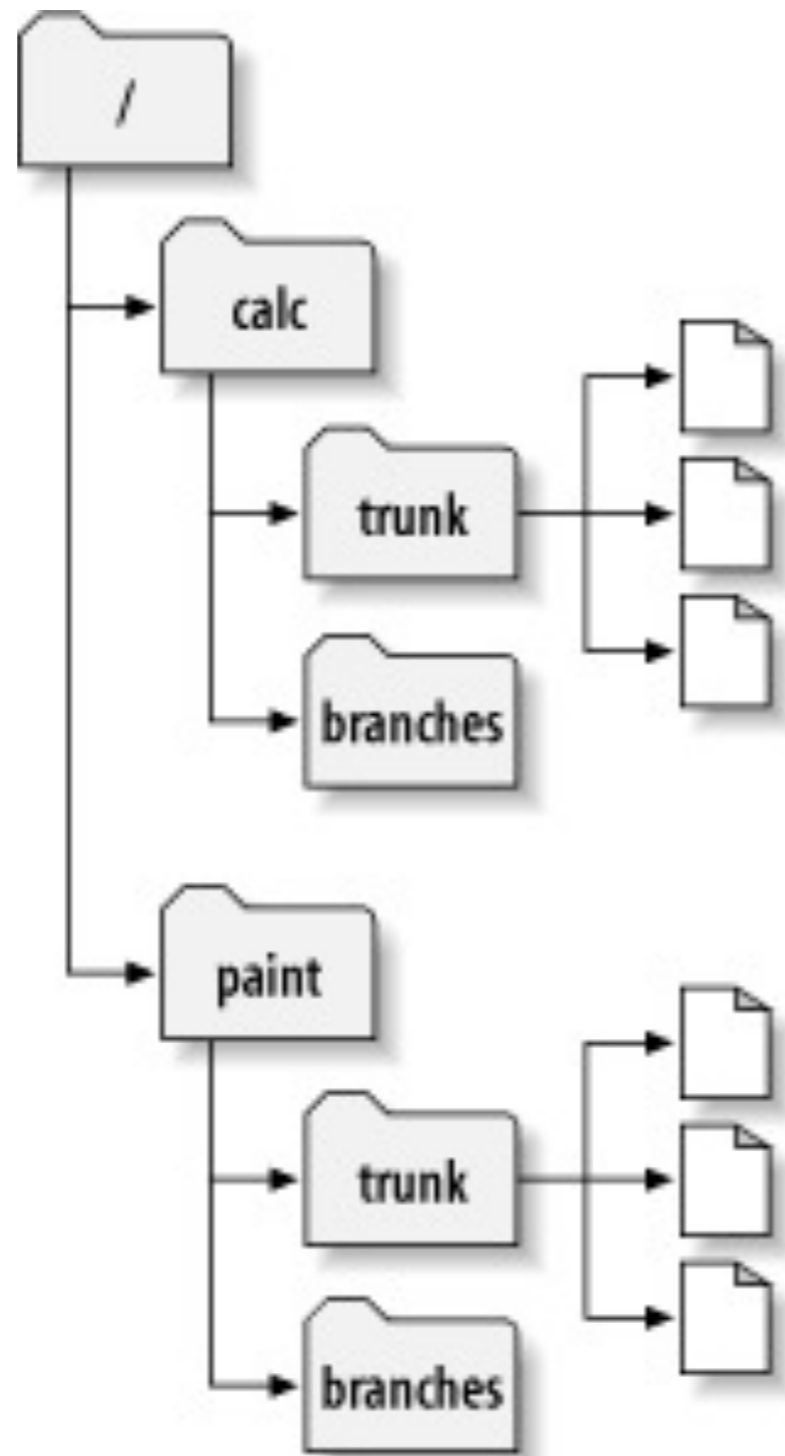
# Tags und Branches

- **Problem: Revision numbers sind nicht frei wählbar**
  - benannte Versionen (z.B. “Release 2.4.17”) wünschenswert
- **Lösung: Tags (symbolische Versionsnamen)**
  - Subversion-Strategie: Erzeugen eines Namens durch Anlegen einer Kopie des gesamten Dateibaums
    - Kopien sind “billig”
- **Problem: Seitenzweige der Entwicklung müssen auch gespeichert werden**
  - Lösung 1: verschiedene Arbeitskopien für unvollendete Projekte
    - aber: mehrere Entwickler?
  - Lösung 2: Branches
    - Subversion-Strategie: Wieder Kopie des gesamten Baums

# Branches

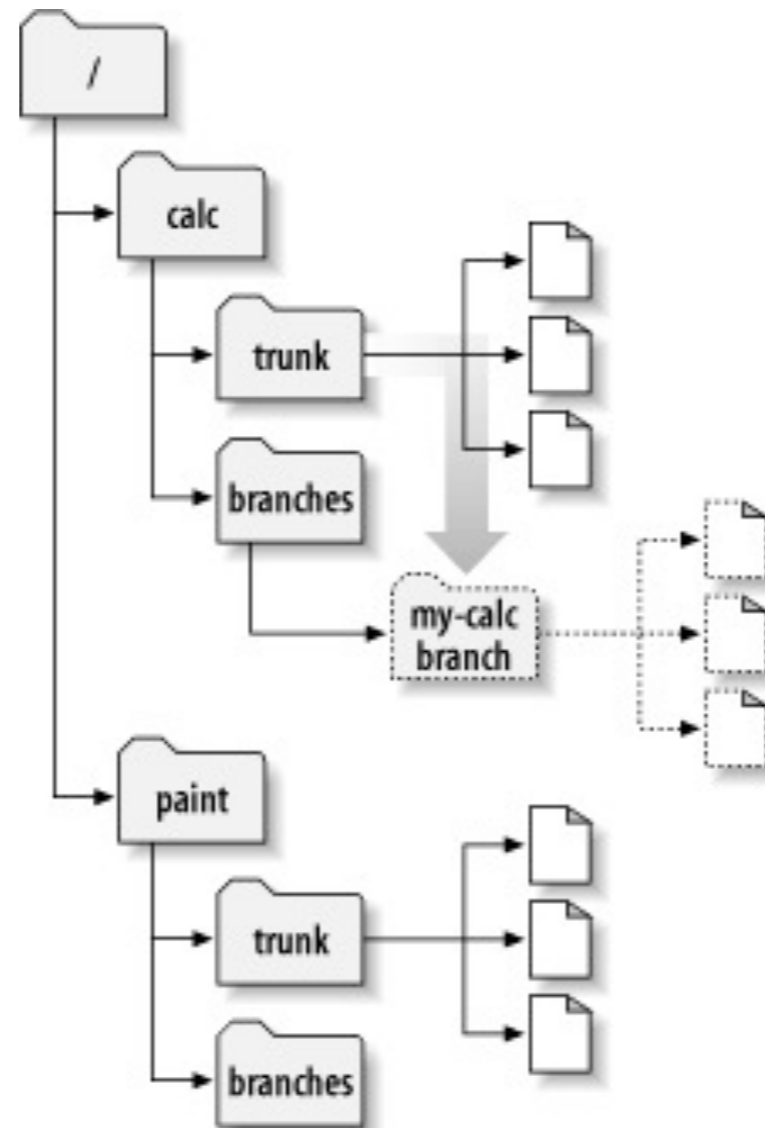


# Verzeichnisstruktur für Branches



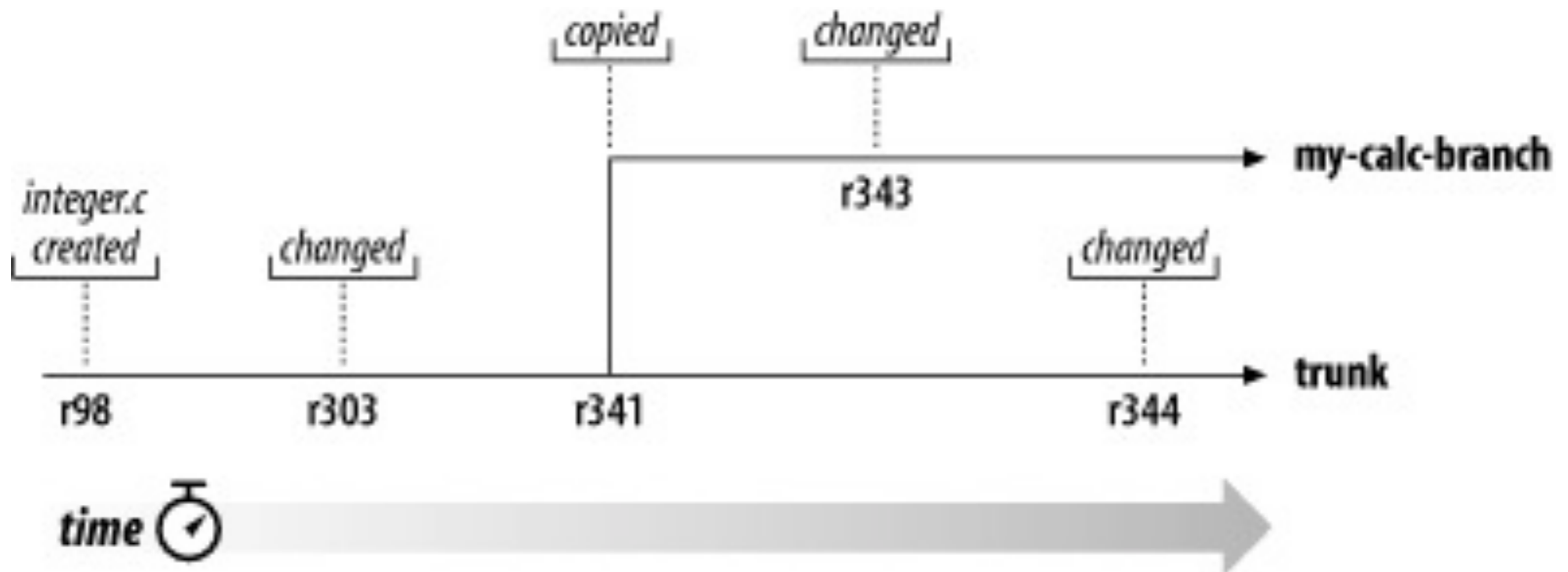
# Anlegen eines Branches

```
$ svn copy http://svn.example.com/repos/calc/trunk \  
          http://svn.example.com/repos/calc/branches/my-calc-branch \  
-m "Creating a private branch of /calc/trunk."
```



# Arbeit mit dem Branch

- Strategie 1: `svn co <pfad zum Branch>`
- Strategie 2: `svn switch`



# Integration von Branches

- `svn merge`
  - Siehe Subversion Book

# Tags

- Unterverzeichnis “tags” neben “trunk” und “branches”
- Anlegen eines Tags:

```
$ svn copy http://svn.example.com/repos/calc/trunk \  
           http://svn.example.com/repos/calc/tags/release-1.0 \  
-m "Tagging the 1.0 release of the 'calc' project."
```

```
Committed revision 351.
```

# Weitere Themen

- Properties
- Repository-Administration
- ...