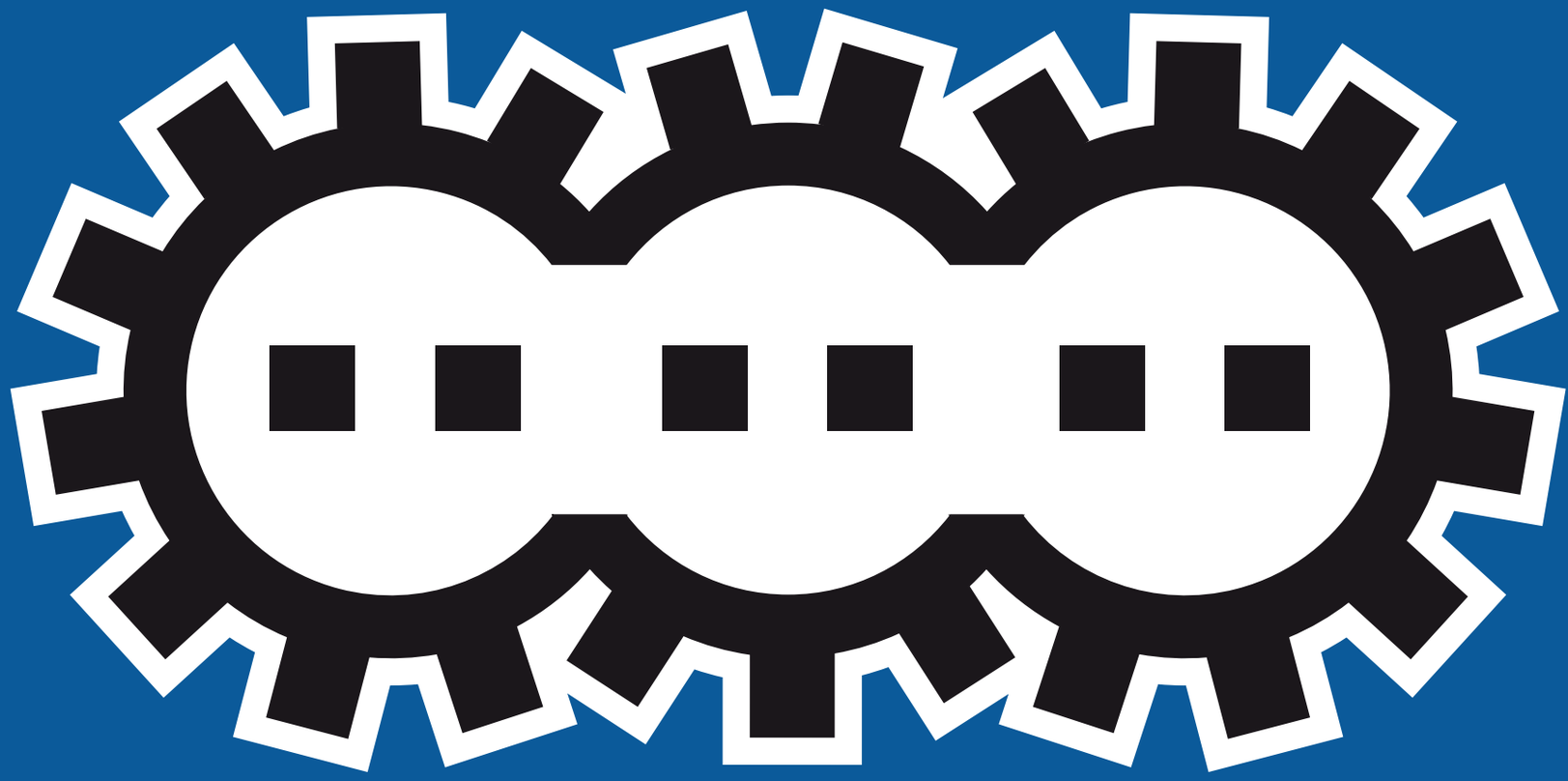
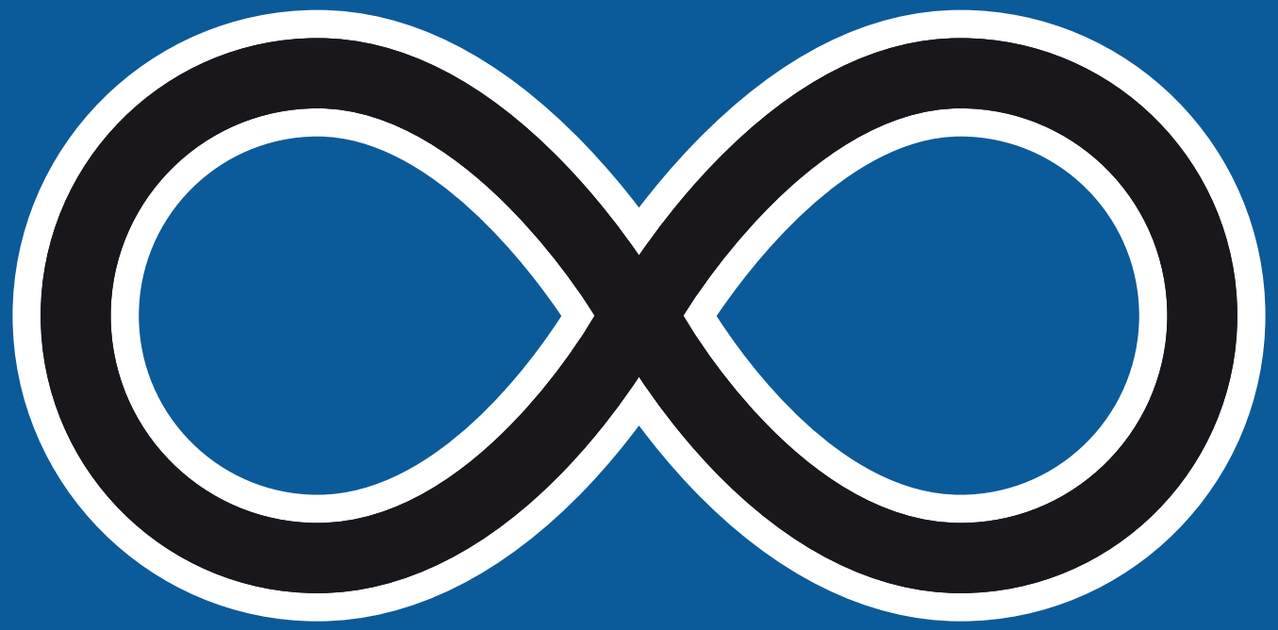


SOFTWAREENTWICKLUNG

IM TEAM MIT OPEN-SOURCE-WERKZEUGEN

02 – fortgeschrittenes Versionsmanagement

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

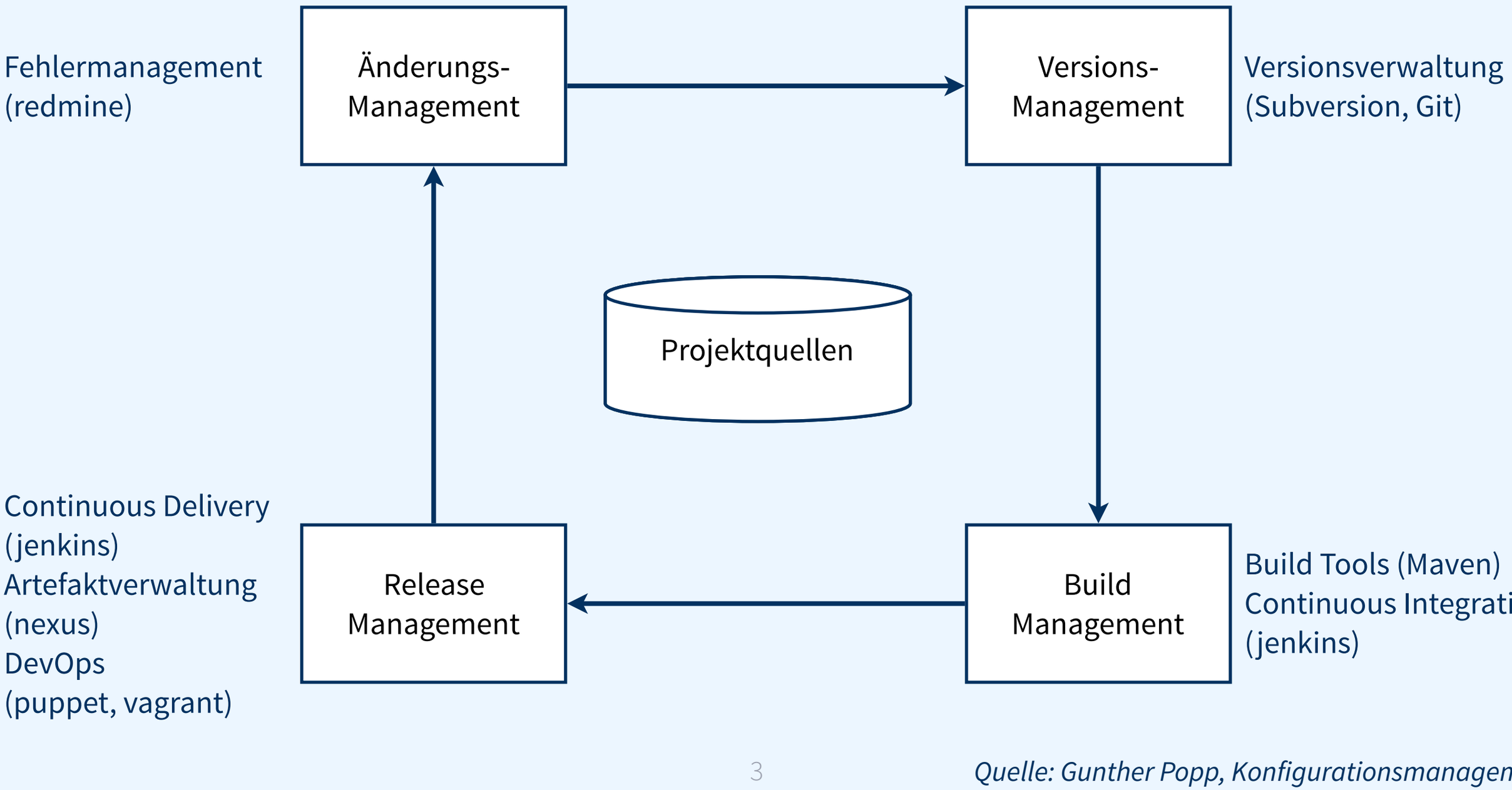


Wiederholung

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

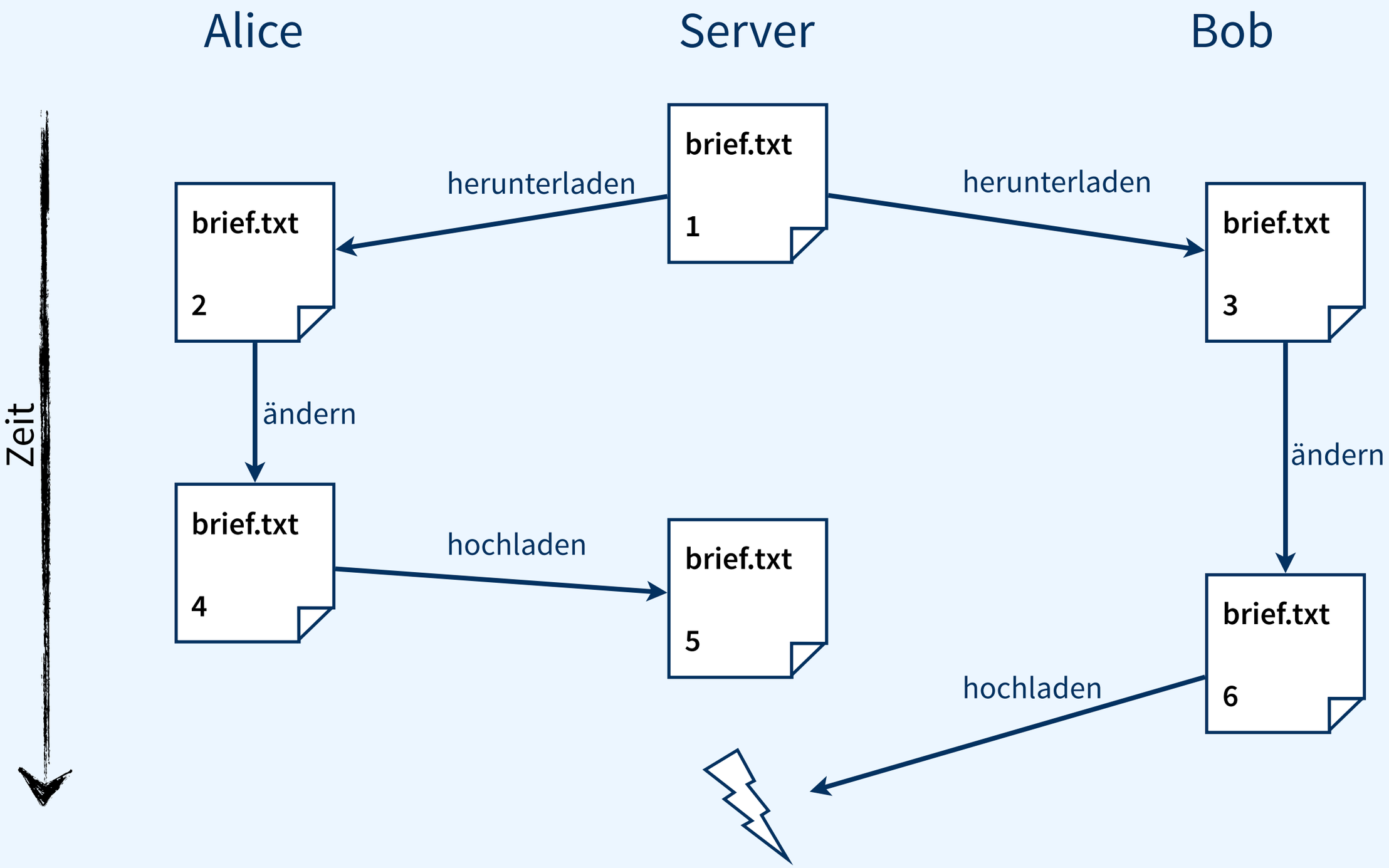
Konfigurationsmanagement

(pragmatisch, mit Open-Source-Werkzeugen)



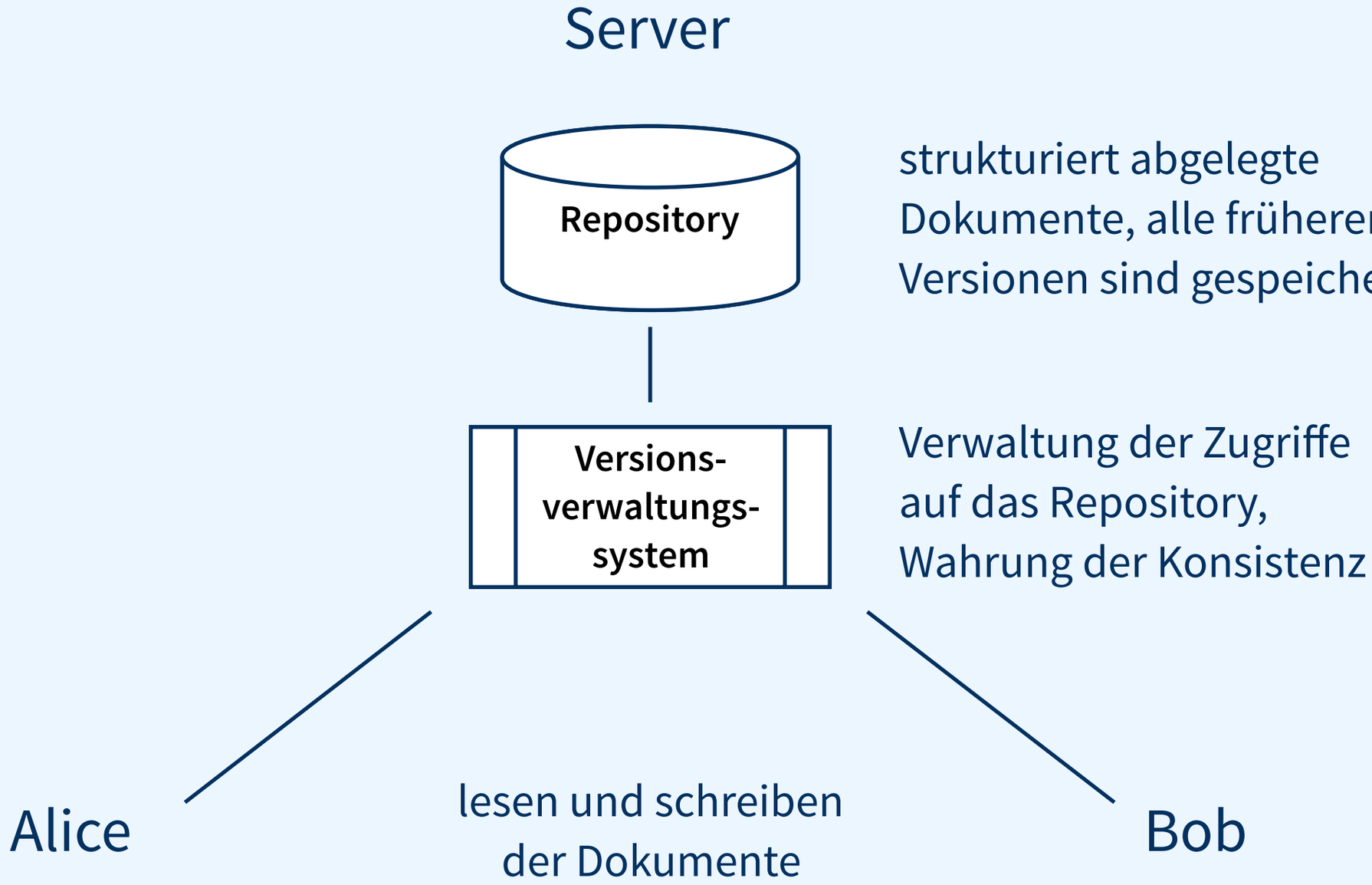
Quelle: Gunther Popp, Konfigurationsmanagement

Problem

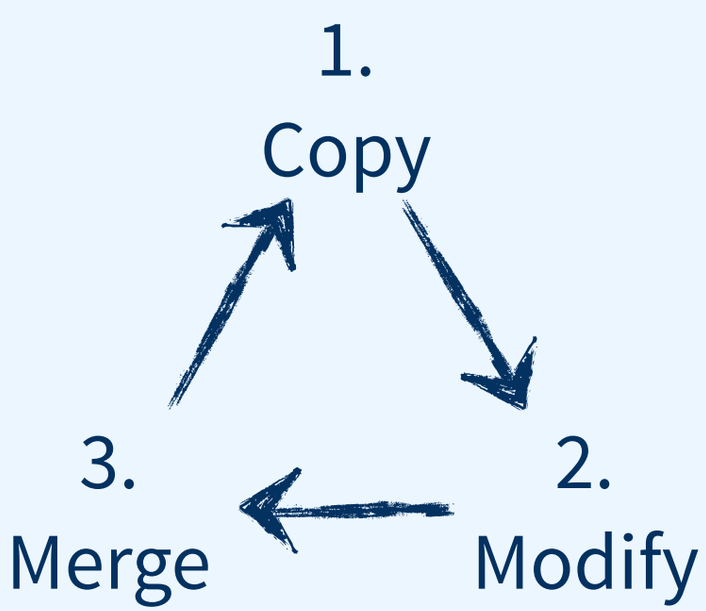


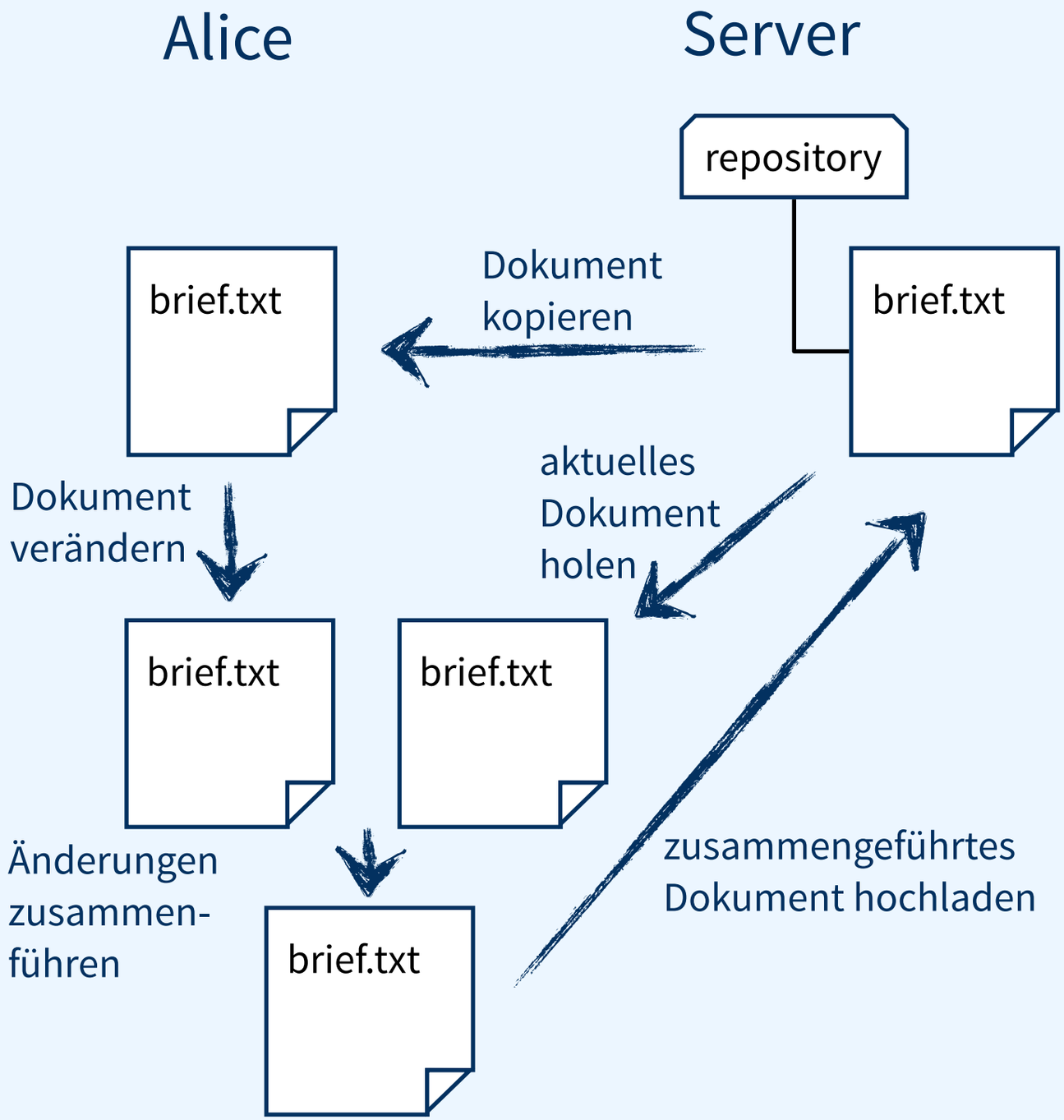
Es existieren drei Varianten von »brief.txt«, welche?

Versionsmanagement-Werkzeug

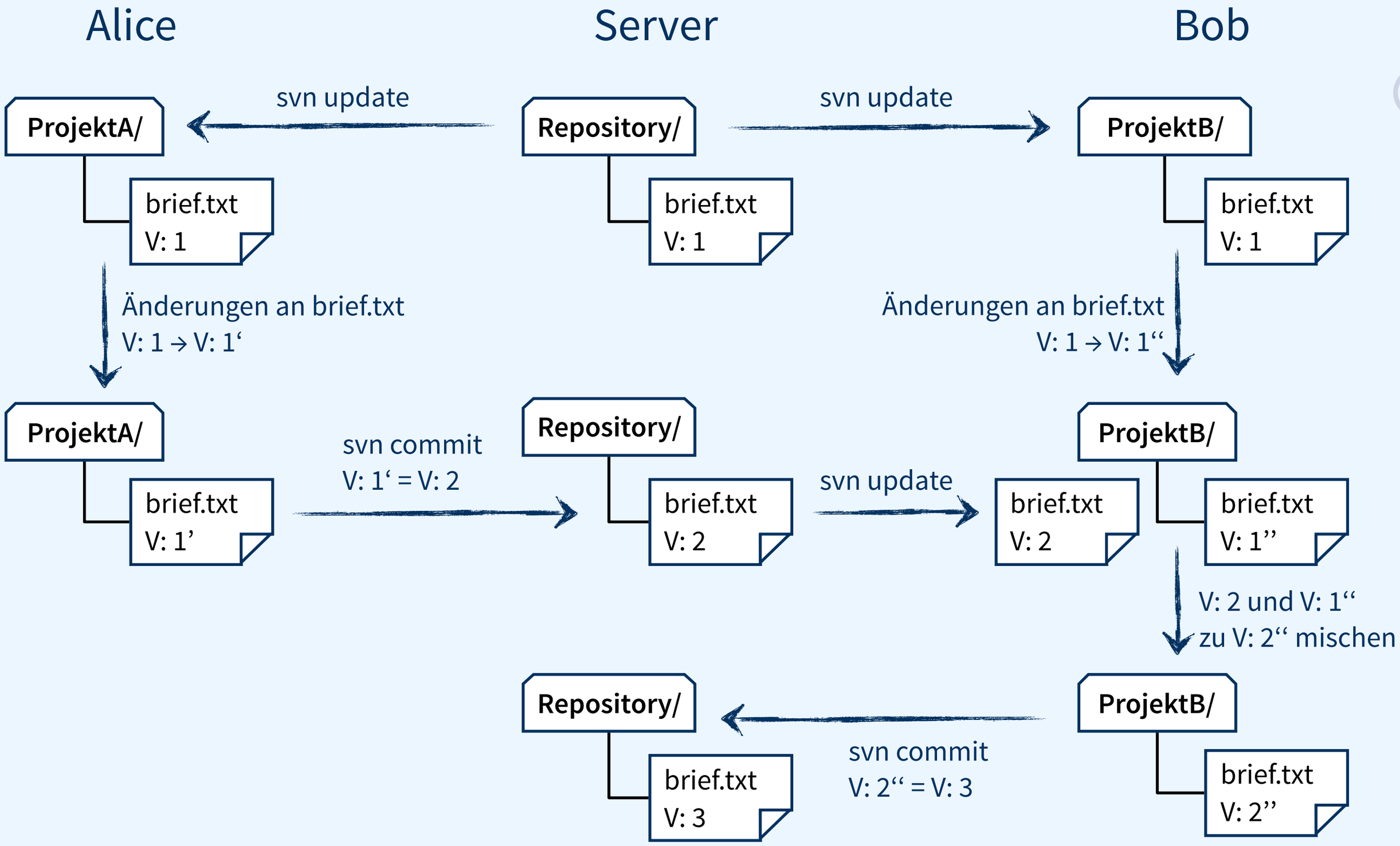


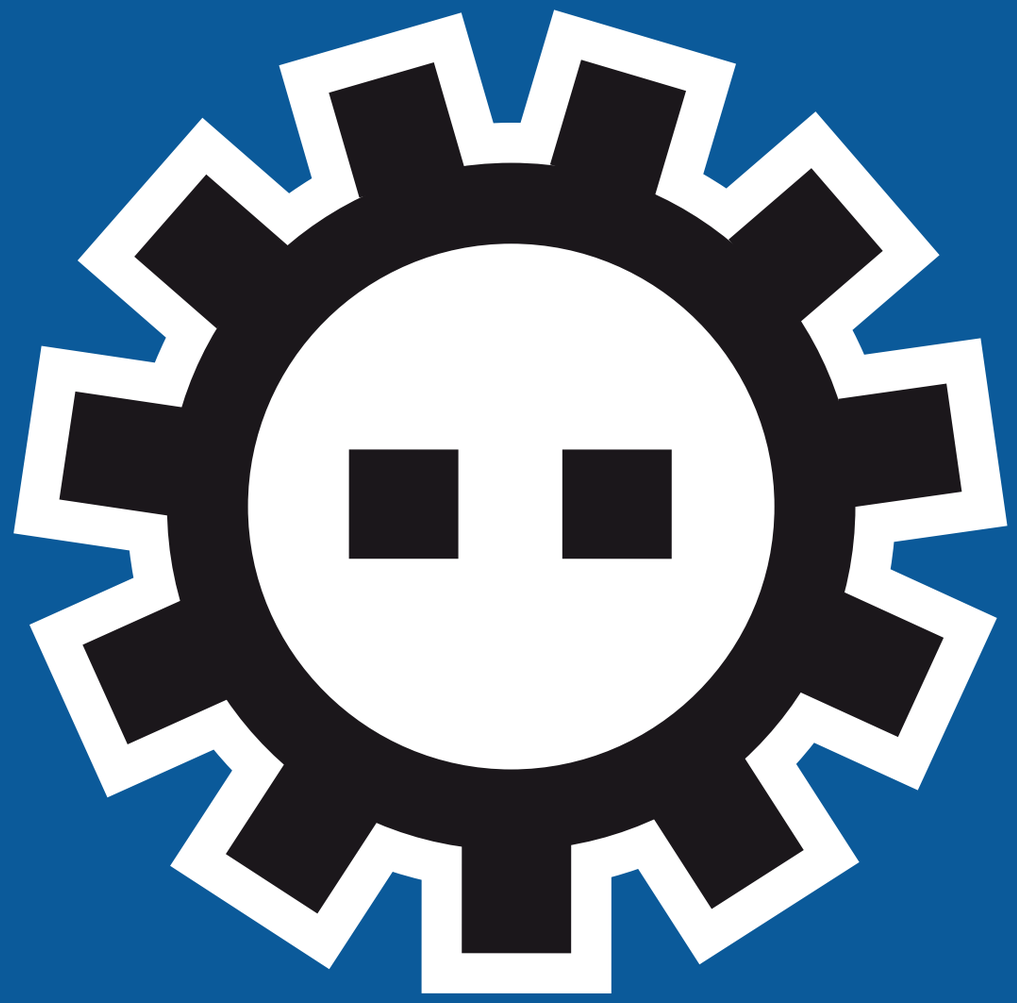
optimistisches Verfahren





Parallele Änderungen





Motivation

Basisfunktionalität

- Bis jetzt können wir mit Subversion die Basisfunktionalität eines Versionsverwaltungswerkzeugs nutzen:
  - Ein Projekt aus dem Repository auschecken
    - » erstellt Arbeitskopie
  - Änderungen hoch- und runterladen
  - Änderungen bei Bedarf mischen und Konflikte beseitigen
- Dies reicht für 90% der Fälle
- Diese Woche werden wir fortgeschrittene Techniken kennenlernen

Neue Begriffe

Diffs anzeigen

Branches

Head

Logs anzeigen

Import/ Export

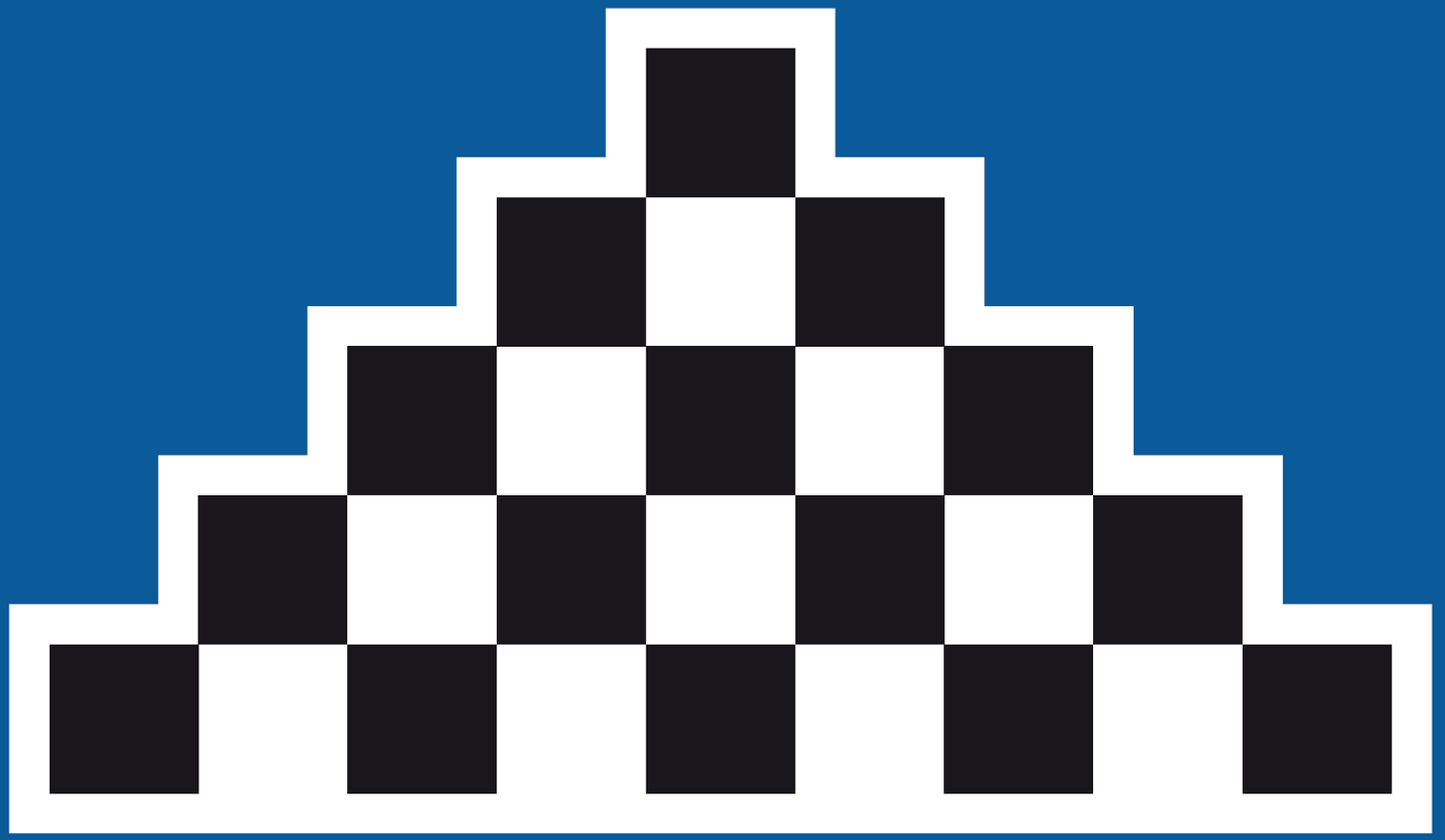
Trunk

Tags

Blame

Merge

Release



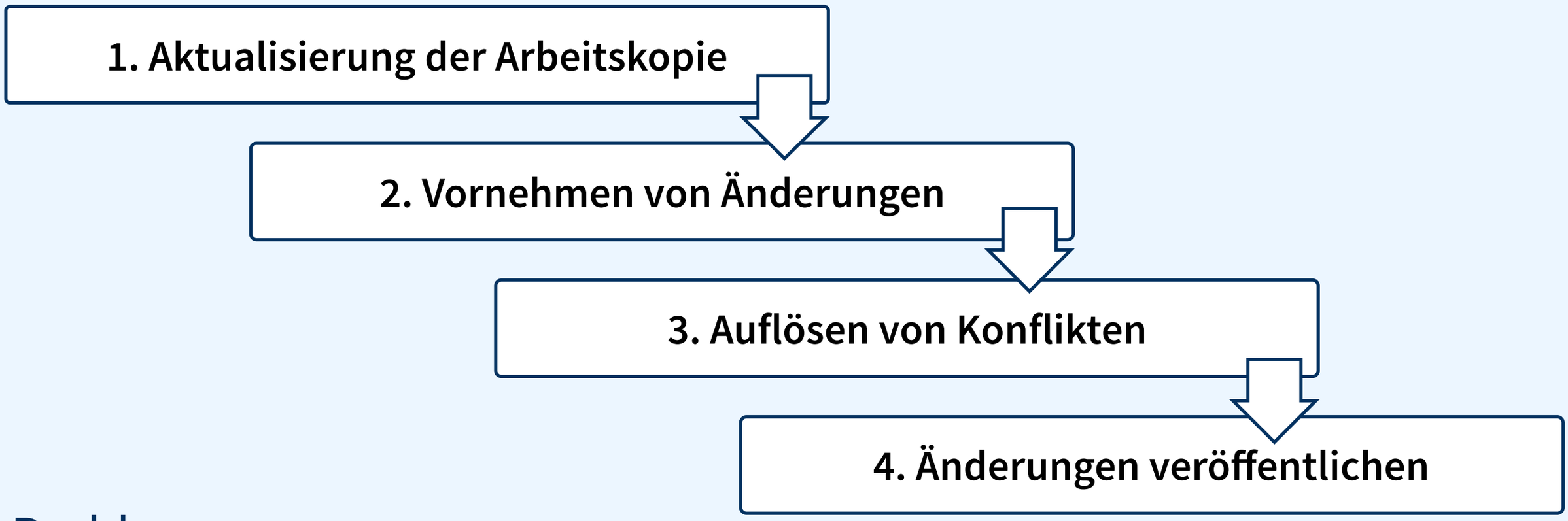
Grundlagen

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Entwicklungszweige in Subversion

Branches

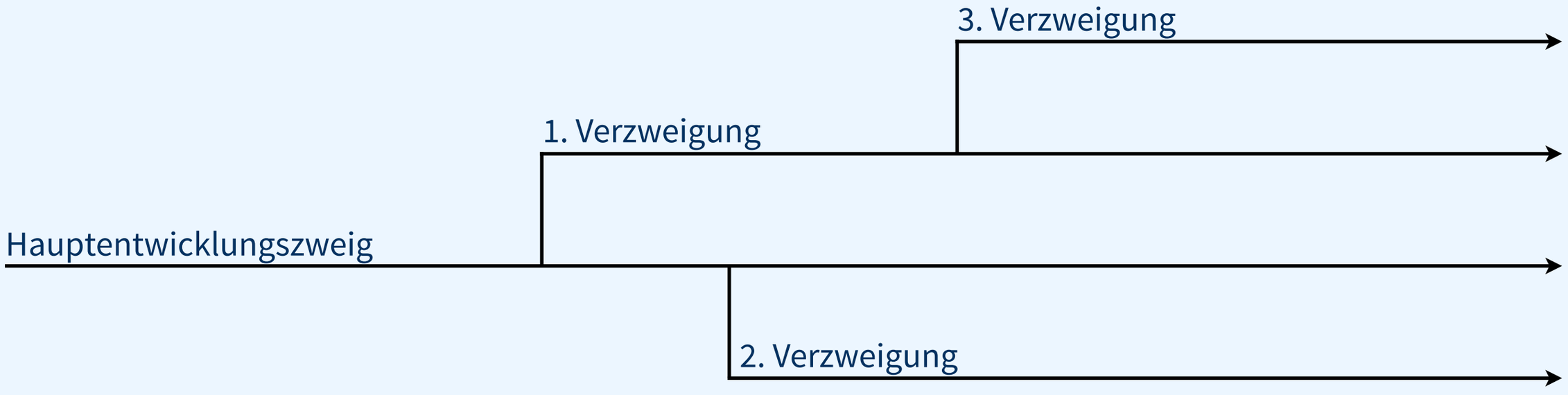
Einfacher Arbeitsablauf



Probleme:

- Was passiert bei großen Änderungen?
  - wir verbleiben lange in Punkt 2 (z.B. mehrere Wochen)
  - alle Änderungen sind nur lokal und unversioniert
  - was passiert bei Datenverlusten in der Arbeitskopie?

Entwicklungszweige



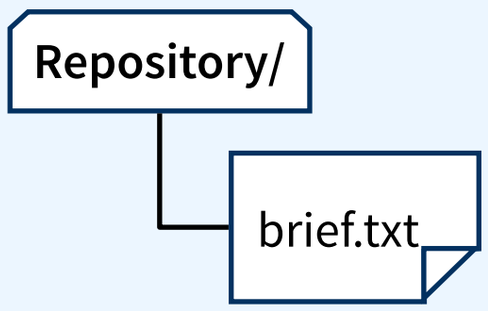
- Bei jeder Verzweigung werden alle Dokumente des Ursprungszweigs in den neuen Zweig kopiert
  - eine Arbeitskopie bezieht sich (meist) nur auf einen Zweig
  - Änderungen in den Zweigen sind unabhängig
- Zweige können bei Bedarf wieder vereinigt werden

Quelle: http://svnbook.red-bean.com

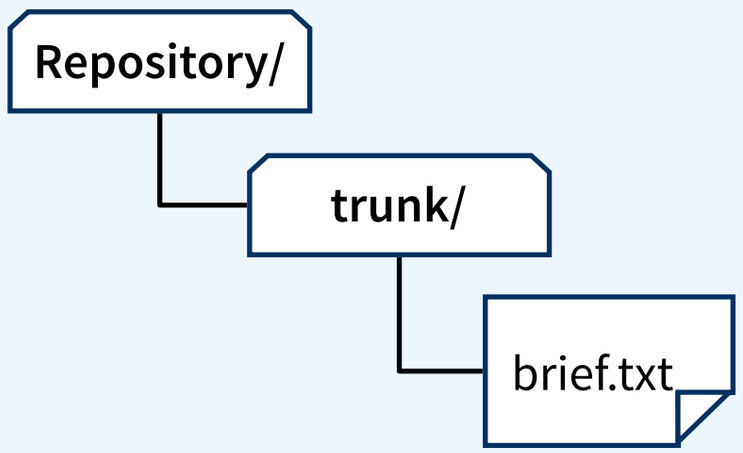
Trunk

- Der Hauptentwicklungszweig wird »trunk« (Stamm) genannt
- Subversion gibt keine Struktur des Repositories vor
  - Konvention ist, dass ein Ordner namens »trunk« im Hauptverzeichnis des Repositories angelegt wird
  - in diesem Ordner befinden sich die Dokumente des Hauptentwicklungszweigs

Alte Struktur
ohne Verzweigungen



Neue Struktur
mit einem Hauptzweig »trunk«



Trunk erzeugen

- Trunk-Verzeichnis im Repository erzeugen
- Dateien in das Trunk-Verzeichnis verschieben
- Arbeitskopien neu auschecken

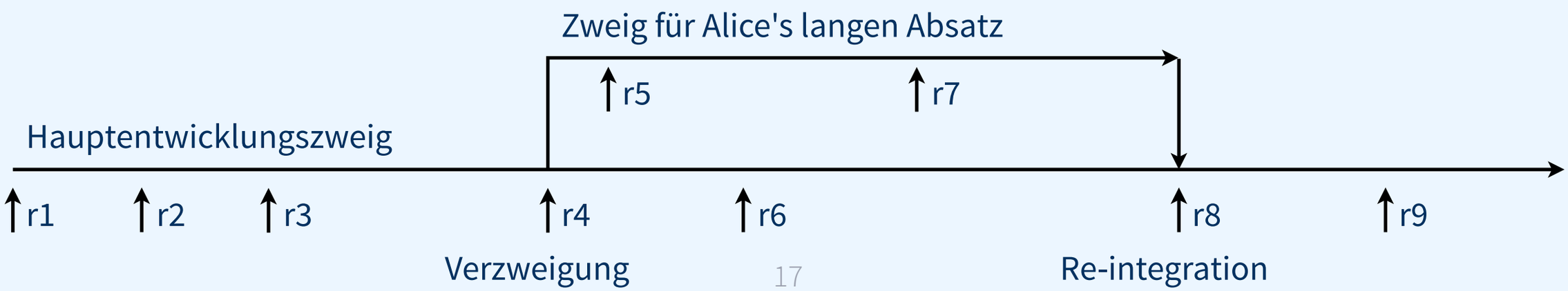
```
/$ svn mkdir file:///Users/stefan/Temp/Repository/trunk -m "trunk erzeugt"

Committed revision 4.
/$ svn move file:///Users/stefan/Temp/Repository/brief.txt file:///Users/stefan/Temp/
Repository/trunk/brief.txt -m "brief kopiert"

Committed revision 5.
/$ rm -rf ProjektA ProjektB
/$ svn checkout file:///Users/stefan/Temp/Repository/trunk ProjektB
A    ProjektB/brief.txt
Checked out revision 5.
```

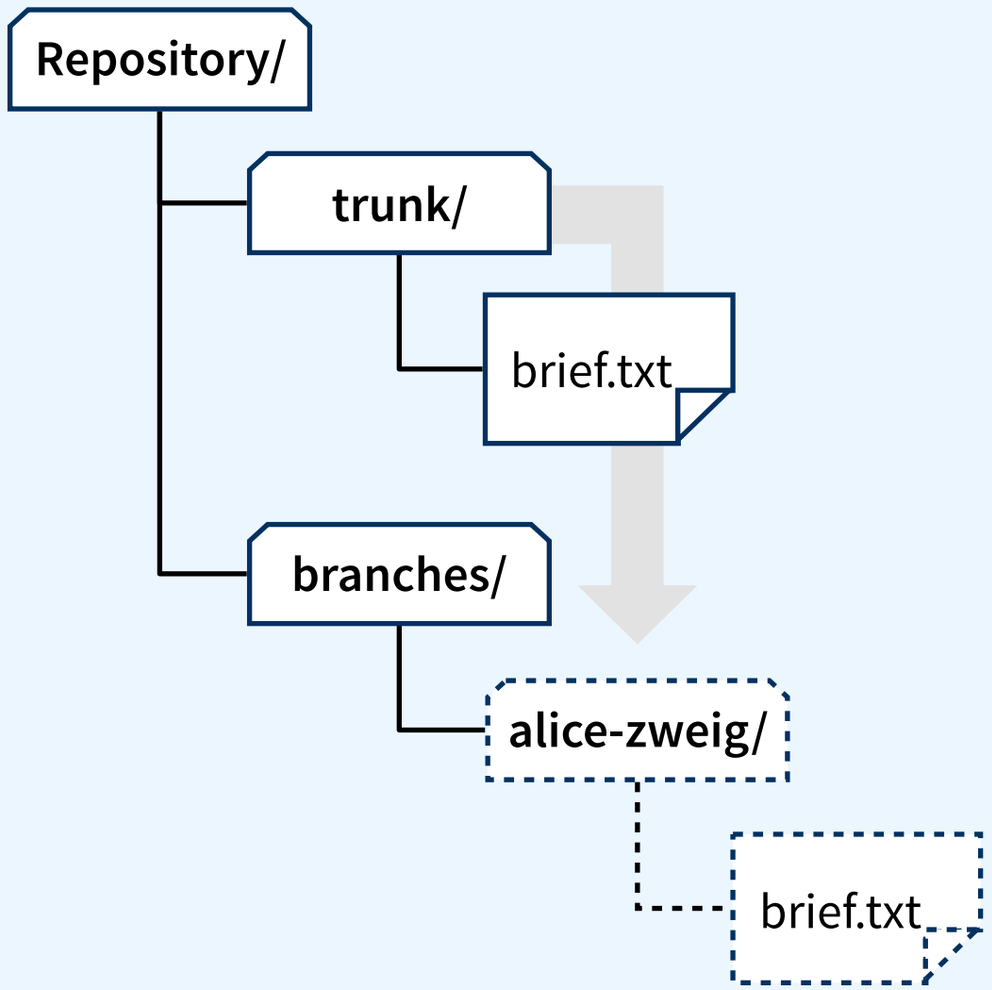
Herausforderung

- Was ist, wenn …
  - … Alice einen langen Absatz im Brief einfügen möchte
  - … Bob nur kleinere Fehler im Brief korrigieren möchte
- Alice erzeugt einen neuen Zweig, um ihre Änderungen regelmäßig zu veröffentlichen
- Bob arbeitet weiterhin auf dem Hauptzweig
- Alice kann ihre Änderungen später in den Hauptzweig re-integrieren:



Branches

- Verzweigungen werden laut Konvention im »branches« Ordner verwaltet
- »branches« Ordner muss manuell erstellt werden
- Verzweigungen sind Verzeichniskopien
- Da es mehrere parallele Verzweigungen geben kann, muss jede Verzweigung einen eigenen Namen erhalten:



Billige Kopien

- Änderungen werden nur als Deltas gespeichert!
- Kopien großer Projektverzeichnisse sind billig
  - keine Änderung an den Dokumenten
  - nur geringer Platzbedarf für Metadaten
  - konstante Zeit → O(1)
- Aus Sicht der Clients sind es vollwertige Kopien
  - Dokumente in den Zweigen können unabhängig geändert werden

➡ Erstellen Sie Zweige so oft Sie wollen!

Schlüsselkonzept Zweige

- Subversion besitzt kein internes Konzept für Verzweigungen:
  - ein Verzeichnis ist ein Zweig, weil wir ihm die Bedeutung geben
  - ein Verzeichnis in Subversion kennt immer seine Historie, auch wenn es kopiert wurde
- Zweige sind immer Verzeichniskopien:
  - müssen sich also im Verzeichnisbaum des Repository befinden
  - laut Konvention im Wurzelverzeichnis »branches«

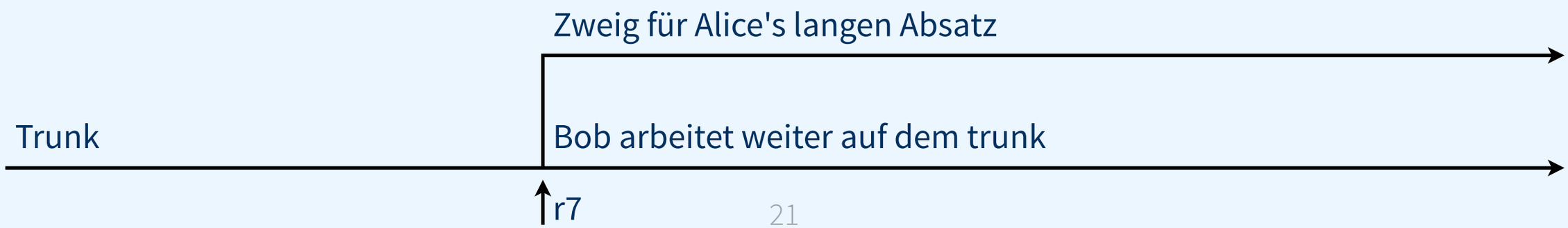
Verzweigung Beispiel

- Branches-Verzeichnis im Repository erzeugen
- Trunk-Verzeichnis in ein neues Branches-Unterverzeichnis kopieren
- Arbeitskopie aus dem Branches-Unterverzeichnis auschecken

```
/$ svn mkdir file:///Users/stefan/Temp/Repository/branches -m "branches erzeugt"

Committed revision 6.
/$ svn copy file:///Users/stefan/Temp/Repository/trunk file:///Users/stefan/Temp/
Repository/branches/alice-zweig -m "alice zweig kopiert"

Committed revision 7.
/ $ svn checkout file:///Users/stefan/Temp/Repository/branches/alice-zweig ProjektA
A    ProjektA/brief.txt
Checked out revision 7.
```



Re-Integration

- Zielzweig als Arbeitskopie auschecken
- svn merge mit dem Quellzweig aufrufen
- Änderungen einchecken und branch löschen

```
/$ cd ProjektA
/ProjektA $ emacs brief.txt
/ProjektA $ svn commit -m "Änderungen im Zweig"
Sending        brief.txt
Transmitting file data .
Committed revision 8.
/ProjektA $ cd .. && rm -rf ProjektA
/$ svn checkout file:///Users/stefan/Temp/Repository/trunk ProjektA
A    ProjektA/brief.txt
Checked out revision 8.
/$ cd ProjektA
/ProjektA $ svn merge --reintegrate file:///Users/stefan/Temp/Repository/branches/alice-
zweig
--- Merging differences between repository URLs into '.':
U    brief.txt
/ProjektA $ svn commit -m "Alice Zweig re-integriert"
Sending        .
Sending        brief.txt
Transmitting file data .
Committed revision 9.
```

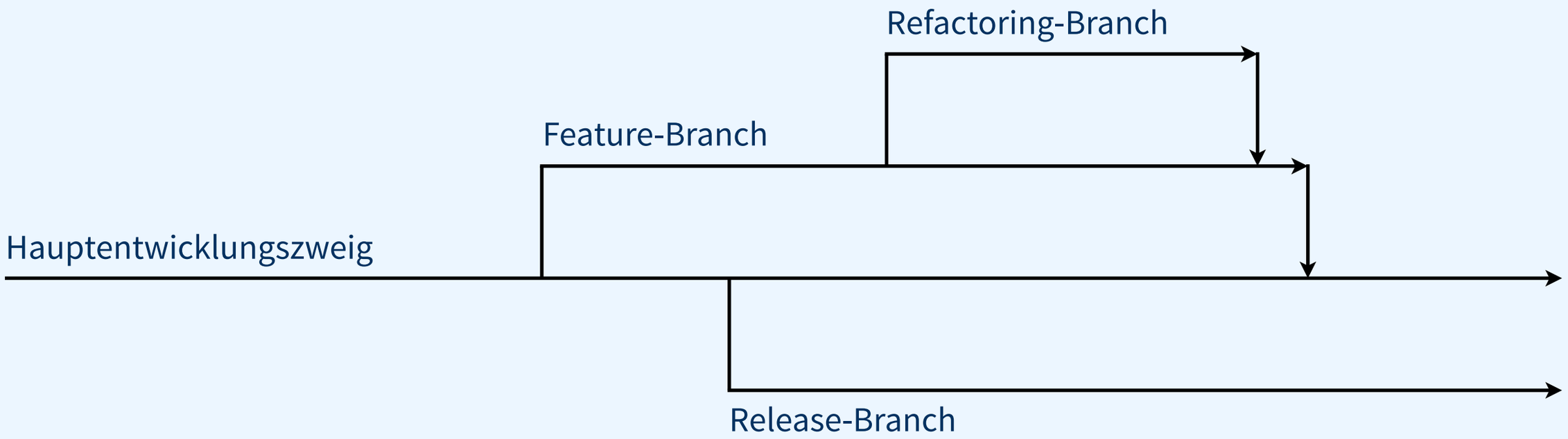
Branch Typisierung

Verschiedene Arten von Verzweigungen:

- **Feature-Branches**
  - isolierte Erstellung einer neuen Funktion
  - wird in den »trunk« re-integriert und gelöscht
- **Refactoring-Branches**
  - aufräumen des Source-Codes, z.B. Einführung von Pattern
  - wird (evtl.) in den »trunk« re-integriert und gelöscht
- **Release-Branches**
  - Branch für eine veröffentlichte Softwareversion
  - langlebig, wird nicht gelöscht, evtl. weiterentwickelt

temporäre / permanente Zweige

- Temporäre Zweige werden in den Quell-Zweig re-integriert
  - und danach gelöscht
- Permanente Zweige bleiben parallel zum »trunk« erhalten
  - eventuell mit Austausch von Änderungen



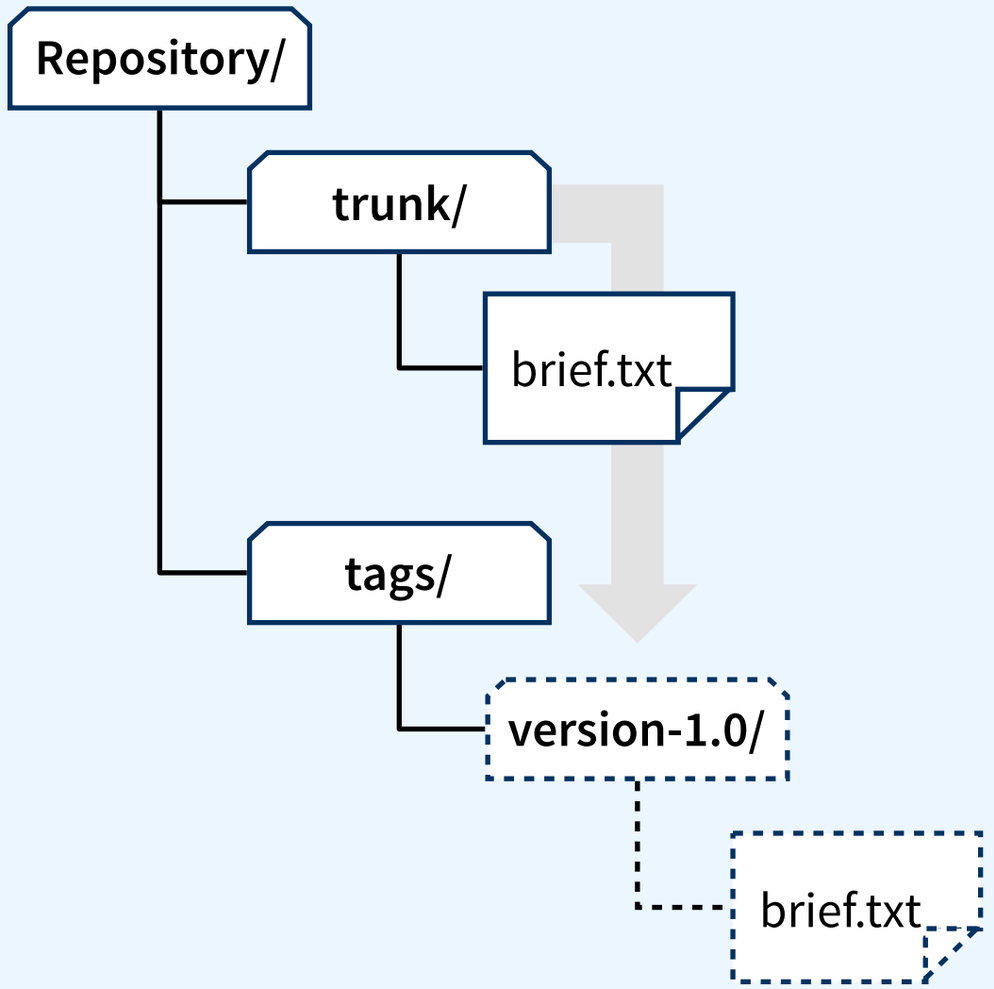
Tags

- Subversion verwaltet
  - keine Software-Versionen
  - sondern Software-*Re*visionen
- Jede (kleine) Änderung erhöht die globale Revisionsnummer
- Revisionsnummern sollten nicht nach außen gegeben werden
- Stattdessen sollte zu definierten Zeitpunkten…
  - … die Software als konkrete Version veröffentlicht werden
  - … der Versionsname vom Marketing bestimmt werden
    - » Oracle 10g, Windows XP, Eclipse Helios
  - … im Repository eine Verzeichniskopie erstellt werden …

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Tags

- Tags werden laut Konvention im »tags« Ordner verwaltet
- »tags« Ordner muss manuell erstellt werden
- Tags sind Verzeichniskopien
- Da es mehrere parallele Tags geben kann, muss jeder Tag einen eigenen Namen erhalten:
- Tags unterscheiden sich nur semantisch von branches, technisch ist es dasselbe!
- Tags sollten nach der Erstellung nicht mehr geändert werden!



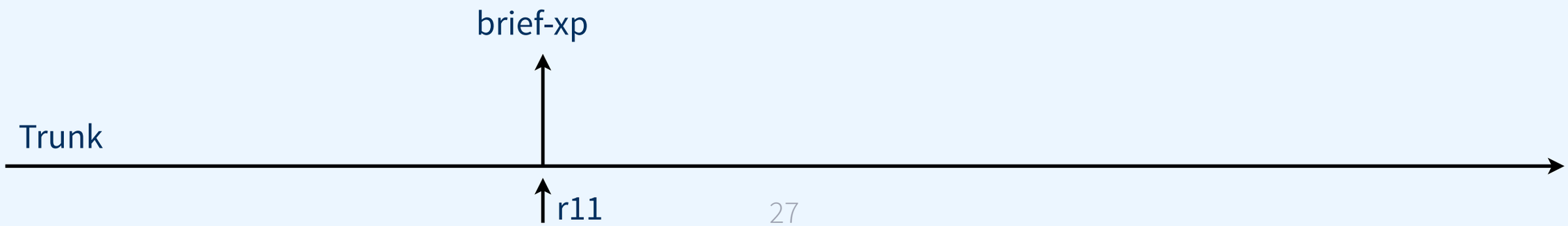
Tag Beispiel

- Tag-Verzeichnis im Repository erzeugen
- Trunk oder Branch in ein neues Tag-Unterverzeichnis kopieren
- Tags sollten nicht als Arbeitskopie ausgecheckt werden

```
/$ svn mkdir file:///Users/stefan/Temp/Repository/tags -m "tags erzeugt"

Committed revision 10.
/$ svn copy file:///Users/stefan/Temp/Repository/trunk file:///Users/stefan/Temp/
Repository/tags/brief-xp -m "brief.txt als XP getaggt"

Committed revision 11.
```



verbliebene Konzepte und Operationen von Subversion

Verschiedenes

Head

- Als Head bezeichnet man die aktuellste Revision des Repository
  - »svn update« aktualisiert z.B. per Default auf Head
  - »svn checkout« holt die aktuellste Revision
- Bei vielen Subversion-Operationen kann aber auch mit dem Parameter »-rREV« eine explizite Revisionsnummer angegeben werden, z.B.:

```
/$ svn checkout -r8 file:///Users/stefan/Temp/Repository/trunk ProjektA
A    ProjektA/brief.txt
Checked out revision 8.
/$ cd ProjektA
/ProjektA $ svn update
Updating '.':
U    brief.txt
 U   .
Updated to revision 11.
```

Logs anzeigen

- »svn log *DATEI*« zeigt alle Veränderungsoperationen, die eine Datei (oder ein Verzeichnis) betreffen
- Mit Parameter »–g« auch Logs aus den Zweigen

```
/ProjektA $ svn log -g brief.txt
------------------------------------------------------------------------
r9 | stefan | 2013-12-03 15:09:13 +0100 (Di, 03 Dez 2013) | 1 line
Alice Zweig re-integriert
------------------------------------------------------------------------
r8 | stefan | 2013-12-03 14:58:54 +0100 (Di, 03 Dez 2013) | 1 line
Merged via: r9
Änderungen im Zweig
------------------------------------------------------------------------
r7 | stefan | 2013-12-03 14:46:25 +0100 (Di, 03 Dez 2013) | 1 line
Merged via: r9
alice zweig erzeugt
------------------------------------------------------------------------
r5 | stefan | 2013-12-03 14:28:00 +0100 (Di, 03 Dez 2013) | 1 line
brief kopiert
```

Blame

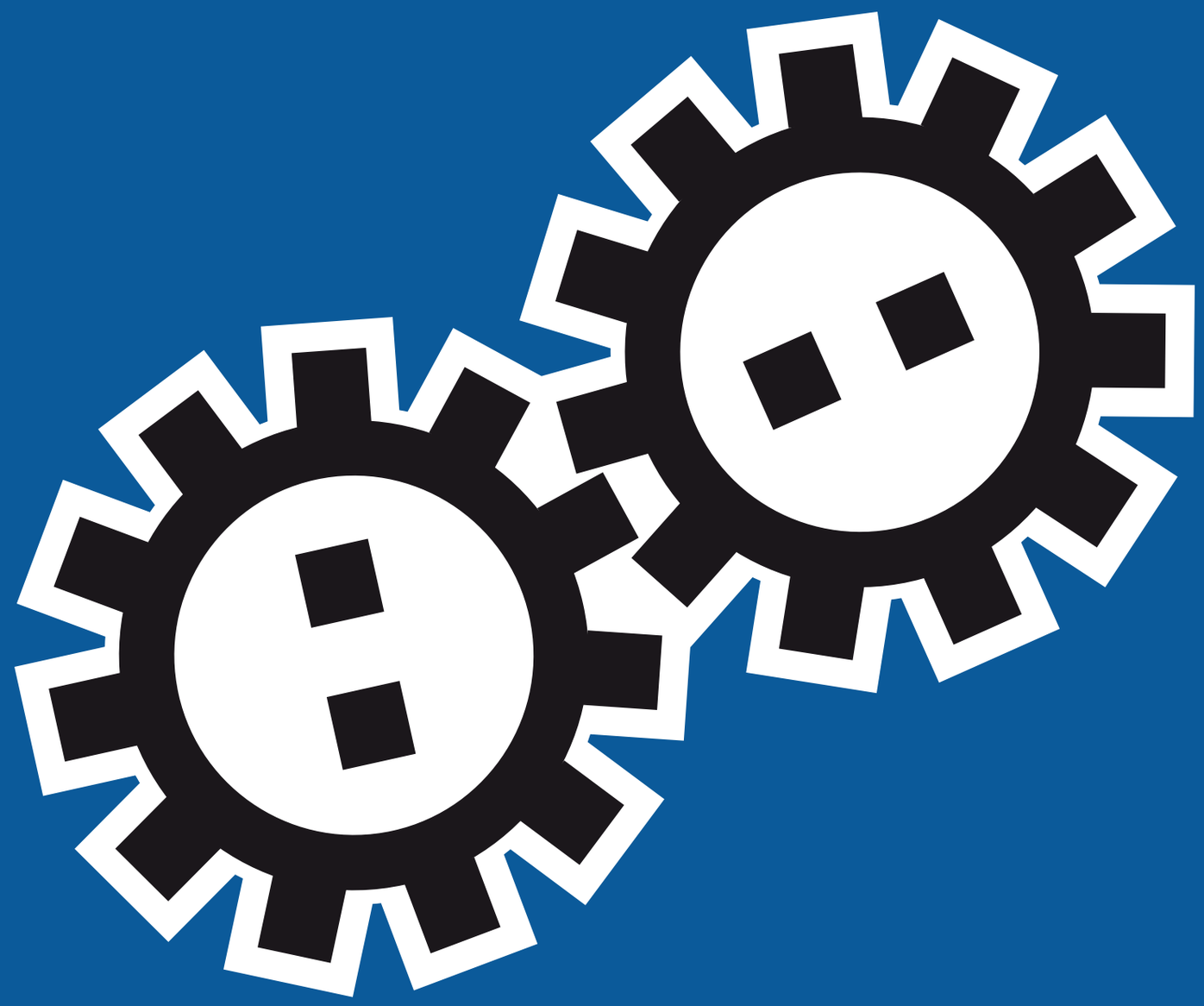
- Mittels »svn blame *Datei*« wird für jede Zeile die Revisionsnummer (und Autor) der Änderung eingeblendet

```
/$ svn blame brief.txt
     2      stefan Einkaufsliste:
     1      stefan Apfel
     1      stefan Birne
     1      stefan Banane
     1      stefan Zwiebeln
     3      stefan Brot
     9      stefan
     9      stefan Wir sollten dringend frische Milch kaufen!
```

Diffs anzeigen

- Mit »svn diff –rREV DATEI« können Sie sich die Änderungen einer Datei zu einer älteren Revision anzeigen lassen
- Im Format einer Patch-Datei, GUI-Tools sind komfortabler

```
/$ svn diff -r8 brief.txt
Index: brief.txt
===================================================================
--- brief.txt    (revision 8)
+++ brief.txt    (working copy)
@@ -4,3 +4,5 @@
 Banane
 Zwiebeln
 Brot
+
+Wir sollten dringend frische Milch kaufen!
```



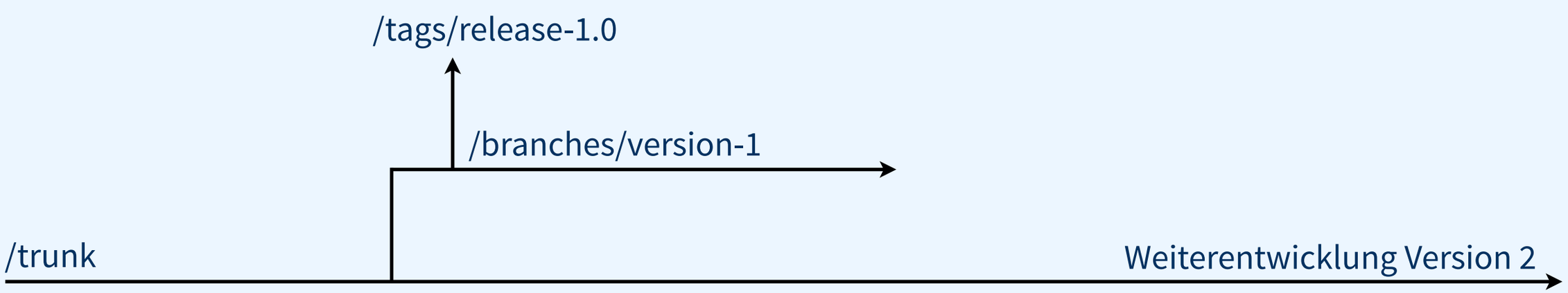
Techniken

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

Release

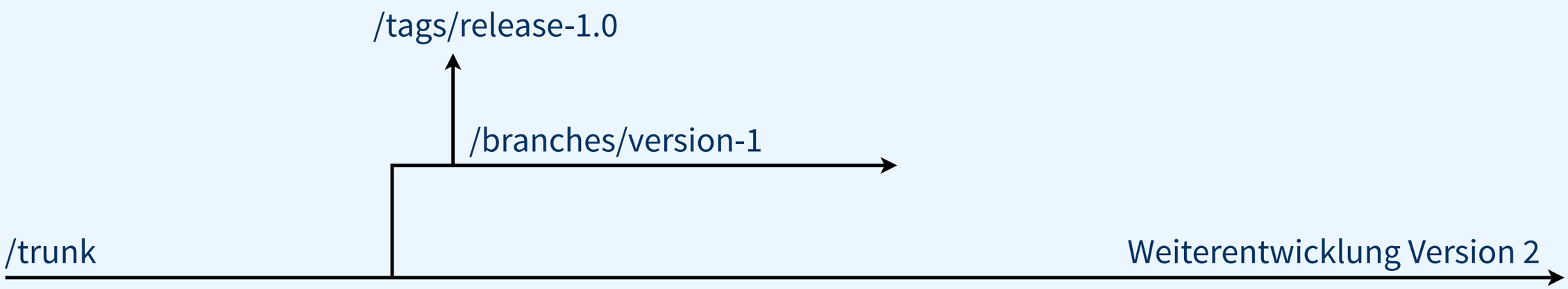
Als Release bezeichnet man die Operationen, die im Vorfeld einer Veröffentlichung ausgeführt werden sollten:

Versions-Branch aus Trunk erzeugen → **Versions-Tag aus Versions-Branch erzeugen** → **Trunk für Weiterentwicklung nutzen**



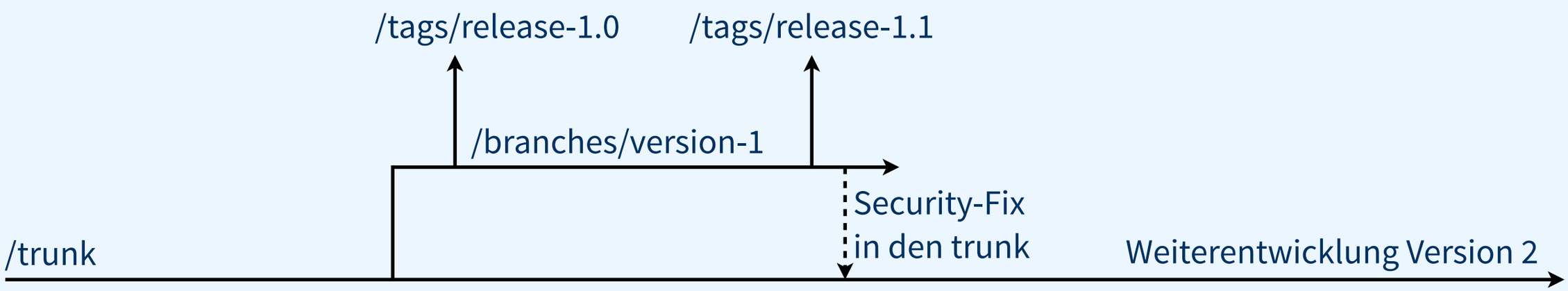
Problem

- Version 1.0 Ihrer Software ist ausgeliefert
- Sie arbeiten mit Hochdruck an Version 2
- In Version 1.0 wird eine Sicherheitslücke gefunden
  - diese muss schnellstens geschlossen werden
- Wie gehen Sie vor?



Vorgehensweise

- Arbeitskopie von /branches/version-1 auschecken
- Fehler beheben
- Tag Version 1.1 erstellen
- Korrigierten Code ebenfalls in den trunk überführen
  - »Cherry-Picking«
  - würde hier zu weit führen



Werkzeuge

Prof. Dr. Stefan Betermieux | Fakultät Informatik | Hochschule Furtwangen

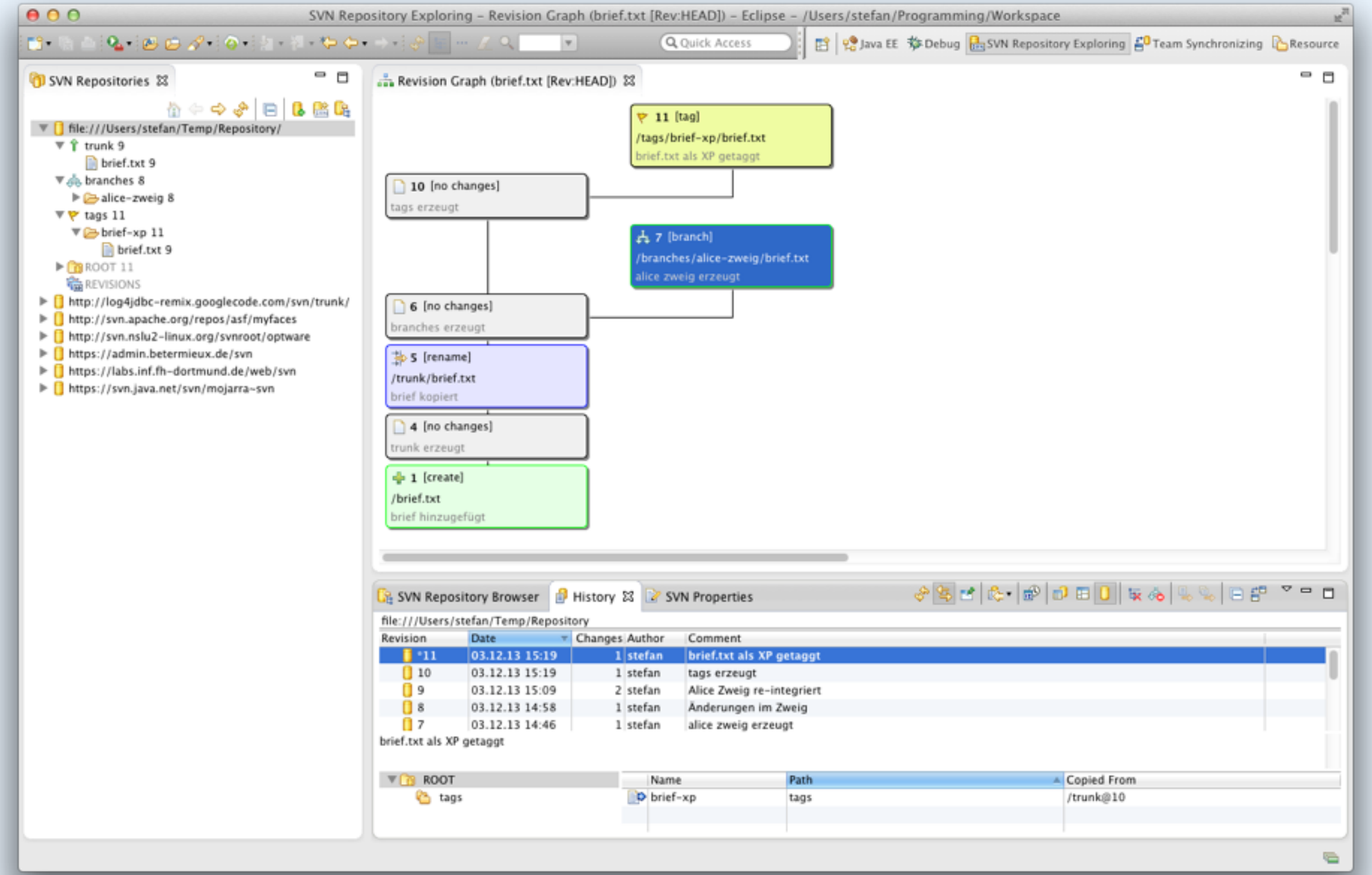
TortoiseSVN

- Open-Source Windows Client
  - http://tortoisesvn.net/downloads.html
- Integriert sich in den Windows File Explorer
  - Anzeige des Versionierungsstatus mittels kleiner Icons
  - Aktionen im Kontextmenü von Dateien
- Versionsgraph / Commit-Statistiken
- Eigene Merge-/Blame-Editoren

Eclipse Client

- Eclipse Plugin »Subversive«
- Im Marketplace (Menü Help → Eclipse Marketplace…)
  - »Subversive – SVN Team Provider« installieren
- Connector installieren (Menü Einstellungen → Team → SVN)
  - Reiter »SVN Connector« auswählen
  - Falls kein Connector in der Liste erscheint:
    - » Knopf »get SVN Connector« anklicken
    - » neuesten SVNKit-Connector installieren
- Subversion Client ist fertig eingerichtet
  - Tutorial unter: http://www.cs.wustl.edu/~cytron/cse132/HelpDocs/Subversive/

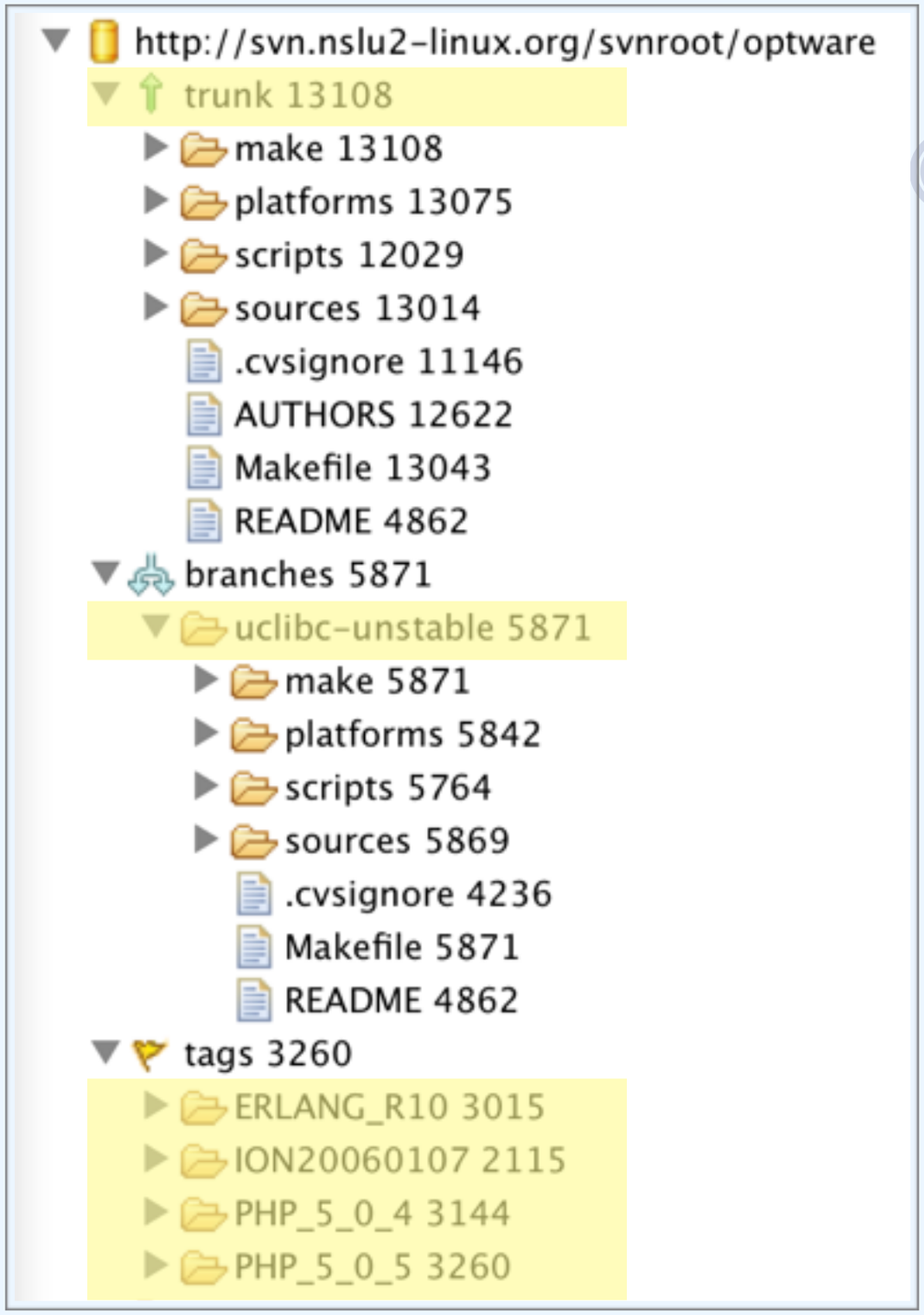
Demo Eclipse



Zusammenfassung

Verzeichnisse

- trunk
  - Hauptentwicklungszweig
  - ein einzelner Zweig
- branches
  - Nebenentwicklungszweige
  - mehrere parallele Zweige
- tags
  - Markierungen bestimmter Zeitpunkte
  - z.B. zur Versionsnummerierung
- Für SVN normale Verzeichnisse, keine unterschiedlichen Konzepte für Tags und Branches



Was passiert?

lokale Datei	svn update	svn commit
unverändert und aktuell	nichts	nichts
lokal geändert und aktuell	nichts	Änderung hochladen
unverändert und nicht aktuell	Änderung herunterladen	nichts
lokal geändert und nicht aktuell	Änderung herunterladen und mischen	schlägt fehl

Danke