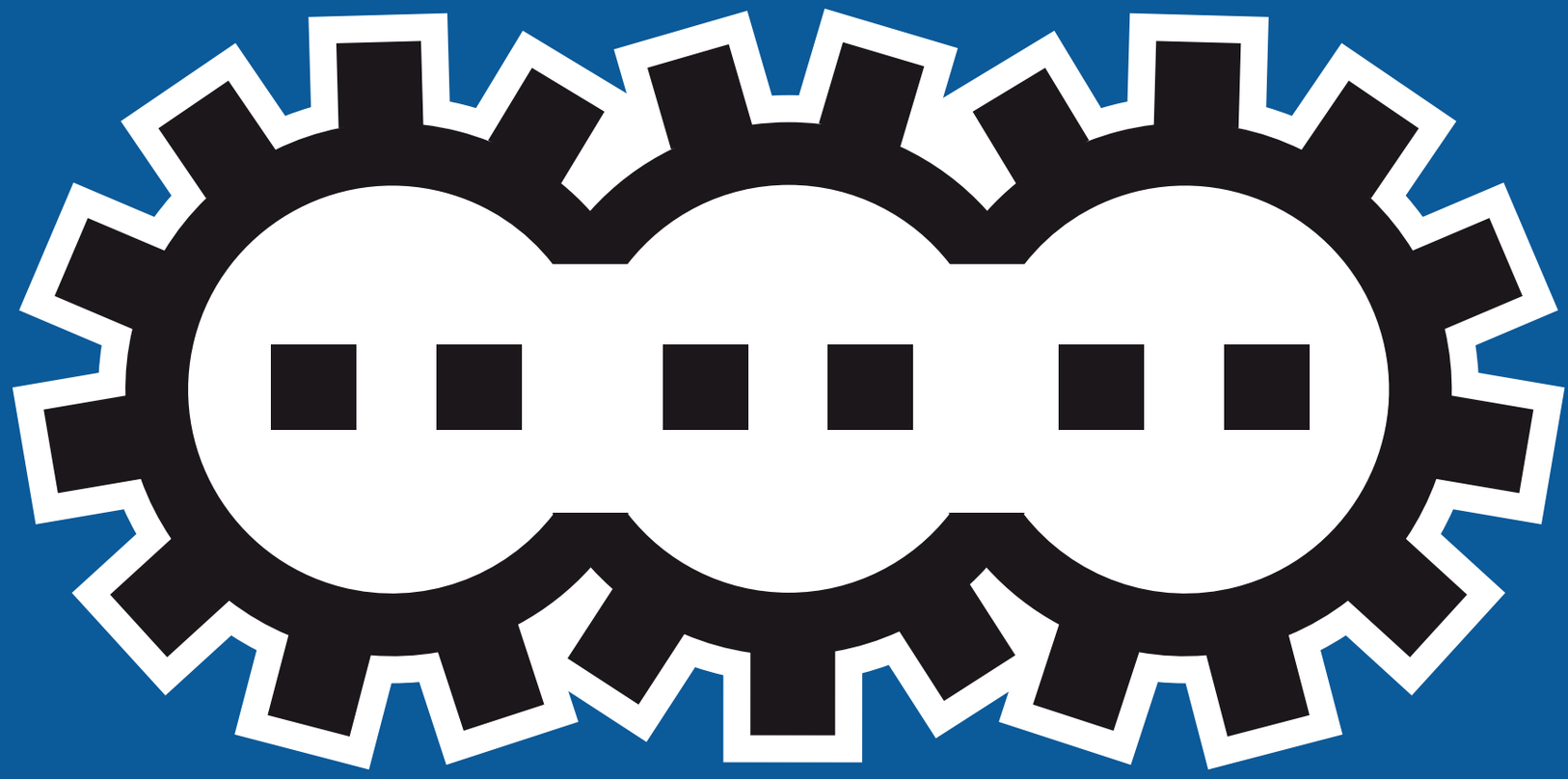
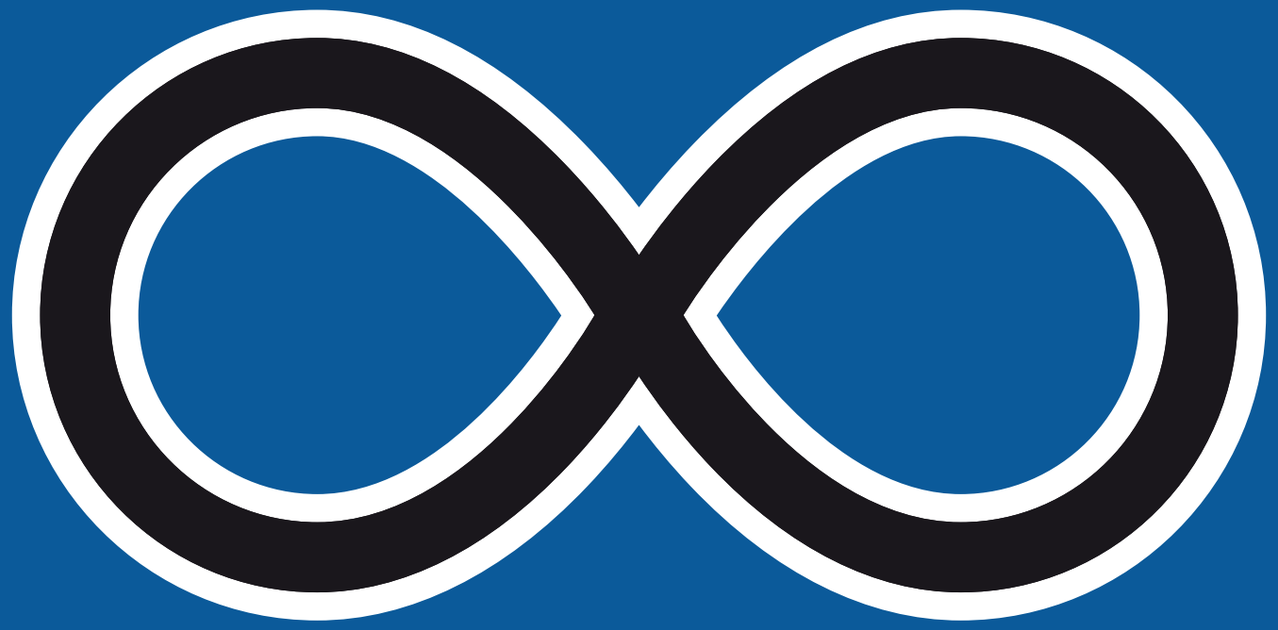


SOFTWAREENTWICKLUNG

IM TEAM MIT OPEN-SOURCE-WERKZEUGEN

01 - zentrales Versionsmanagement

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

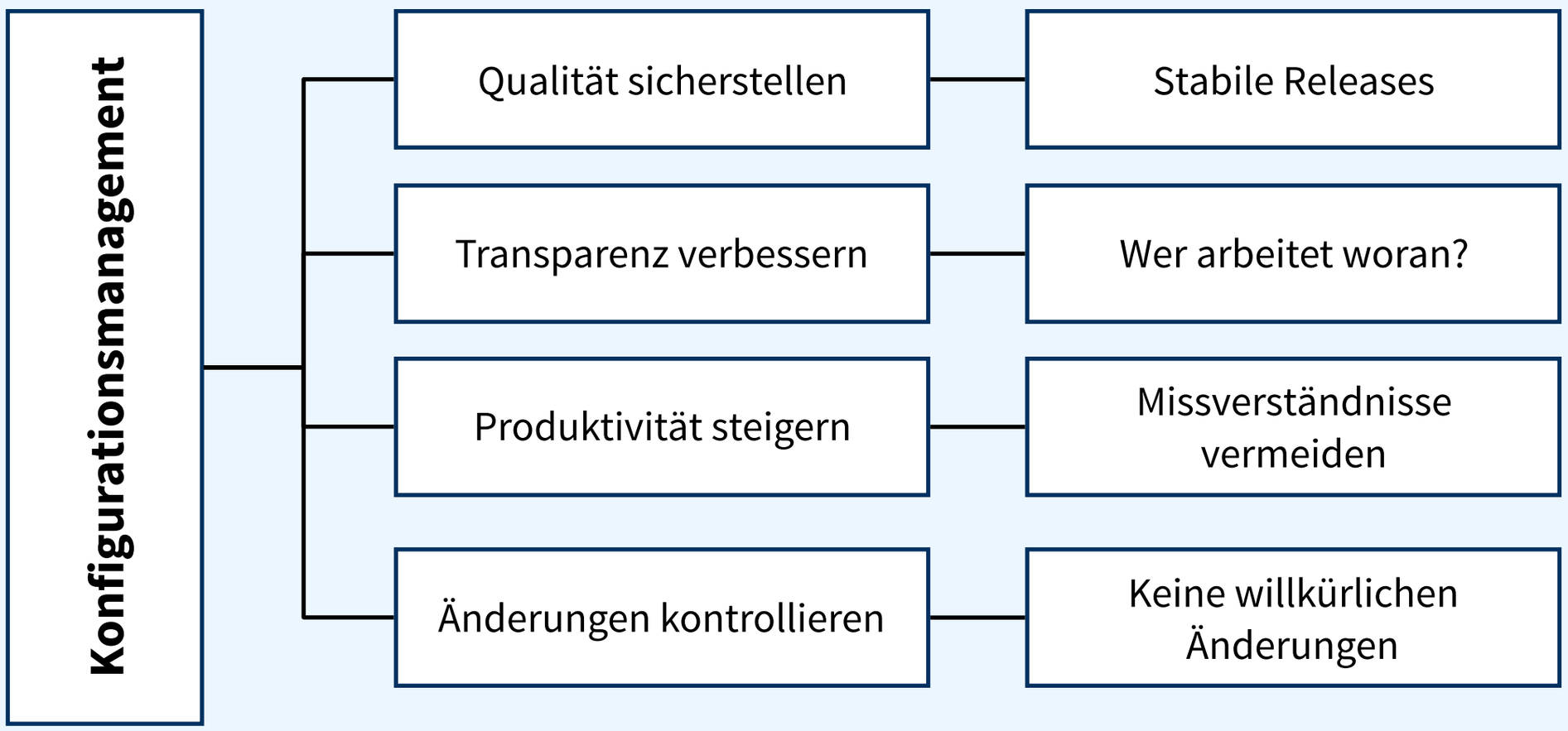


Wiederholung

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

Konfigurationsmanagement

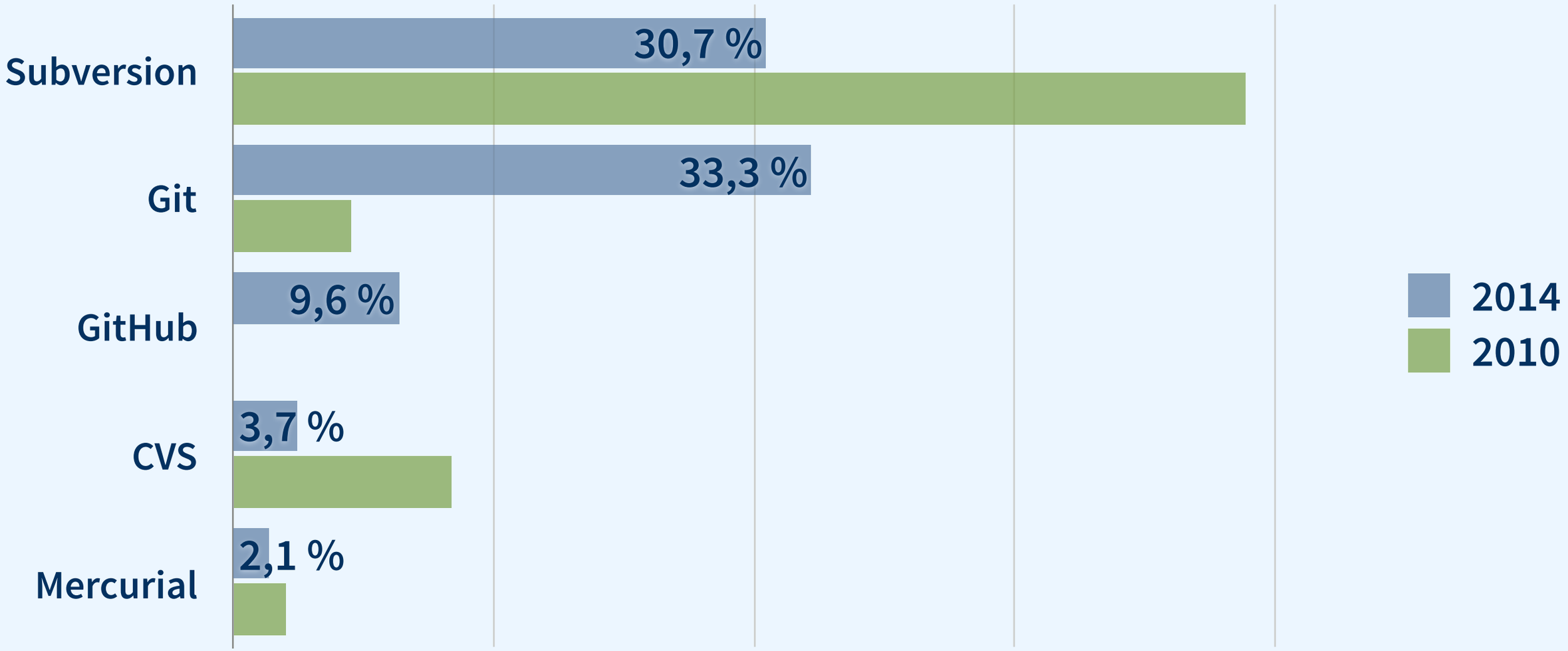
- Statt *Bottom-Up* vielleicht besser der *Top-Down* Ansatz
- Konfigurationsmanagement definiert den Prozess formal
  - IEEE Standard for Software Configuration Management Plans
  - Capability Maturity Model Integration – Configuration Management



Quelle: Gunther Popp

Versionsverwaltung

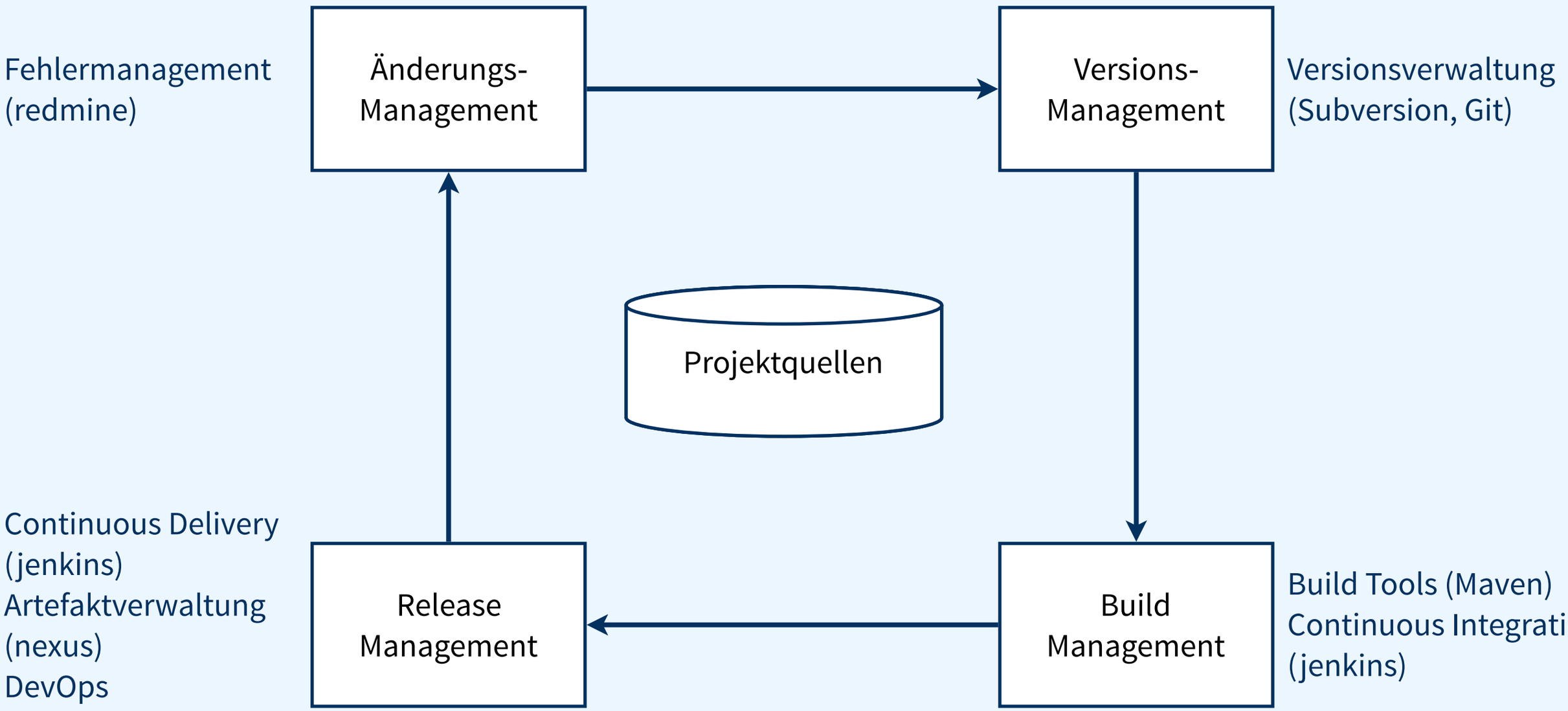
Welches Versionsverwaltungssystem (VCS) setzen Sie ein?



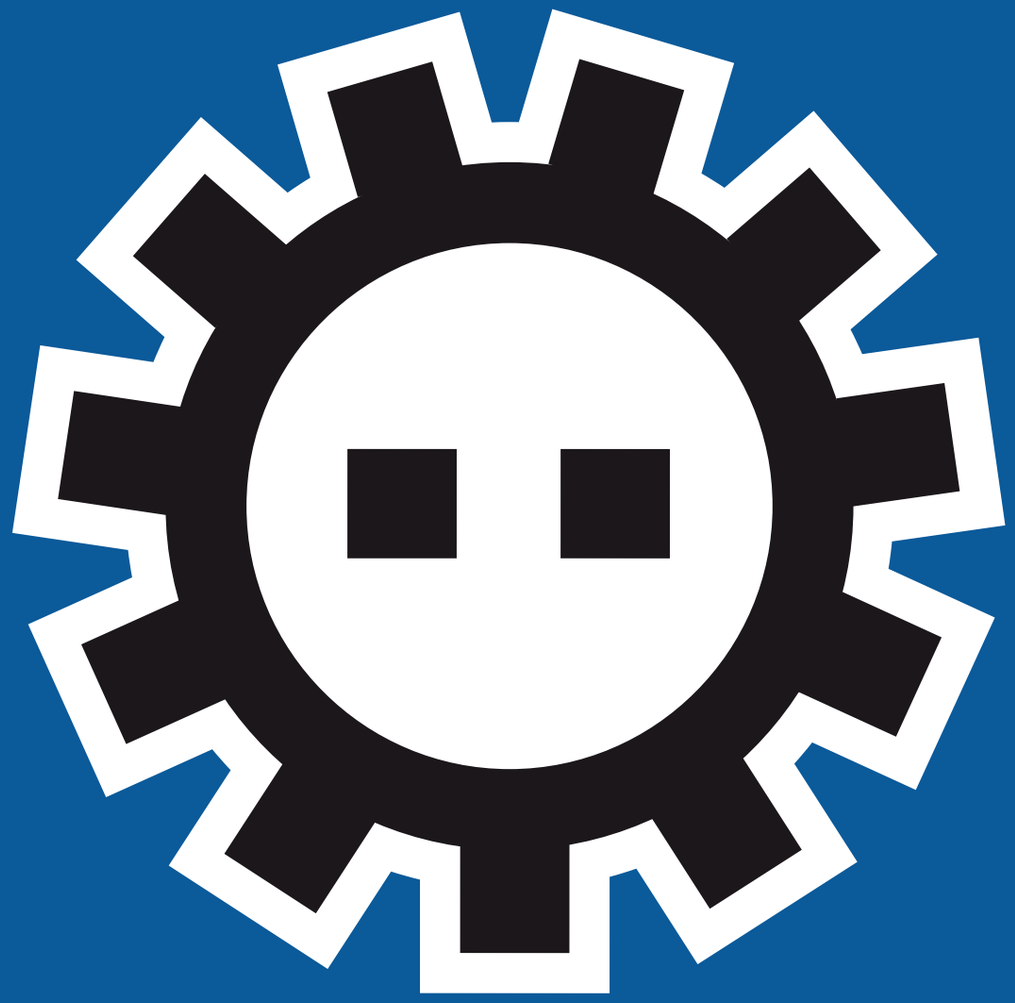
Git hat den gleichen Marktanteil wie Subversion!

Konfigurationsmanagement

(pragmatisch, mit Open-Source-Werkzeugen)



Quelle: Gunther Popp, Konfigurationsmanagement



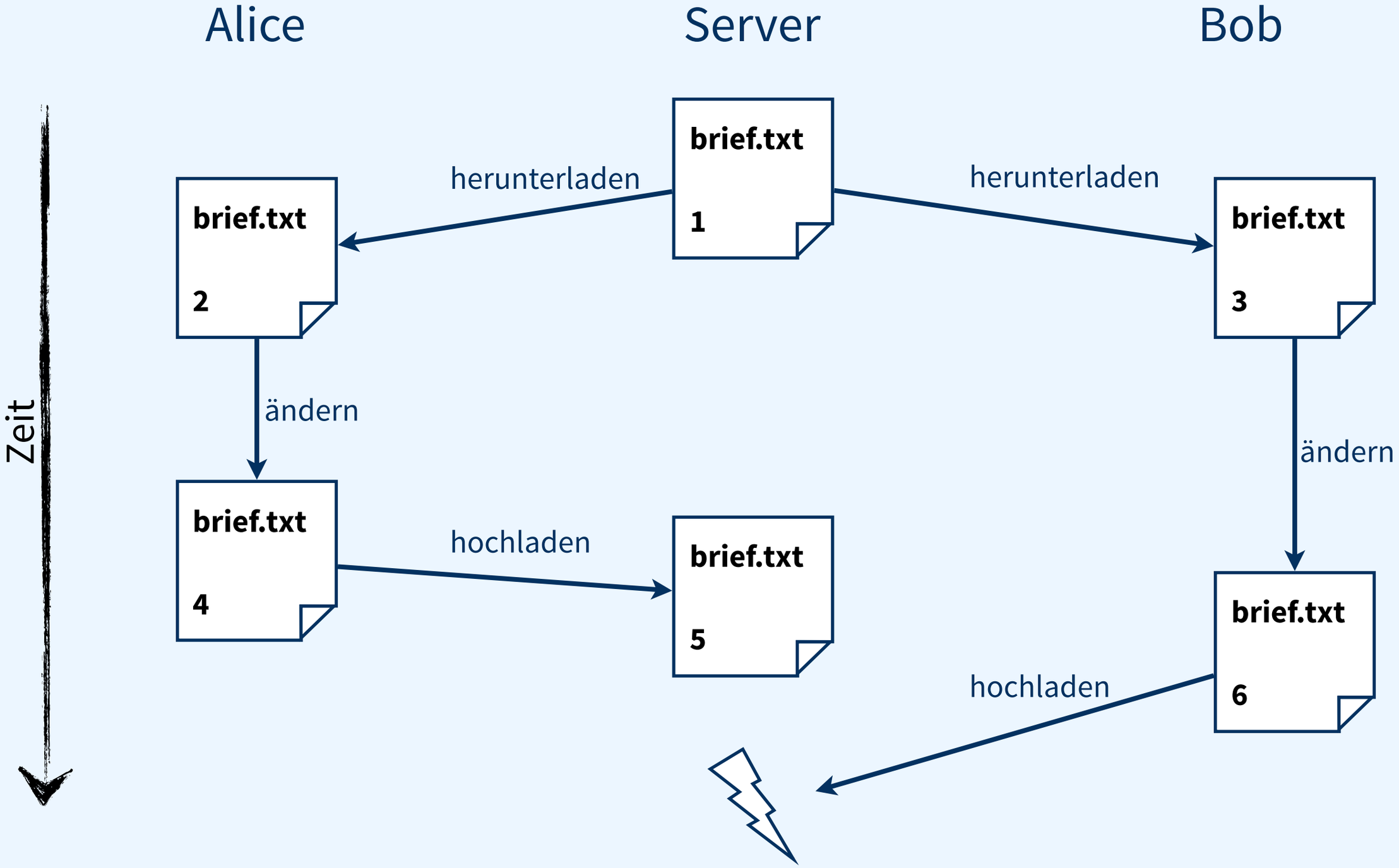
Motivation

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

Ausgangslage

- Softwareprojekte bestehen aus vielen (Quell-)Dokumenten:
  - Programmcode
  - HTML-Dokumente
  - UML-Diagramme
  - Dokumentation
- Projekte werden in Teamarbeit entwickelt:
  - verschiedene Personen müssen auf die gleichen Dokumente zugreifen können
  - verschiedene Personen ändern gleichzeitig das gleiche Dokument

Problem



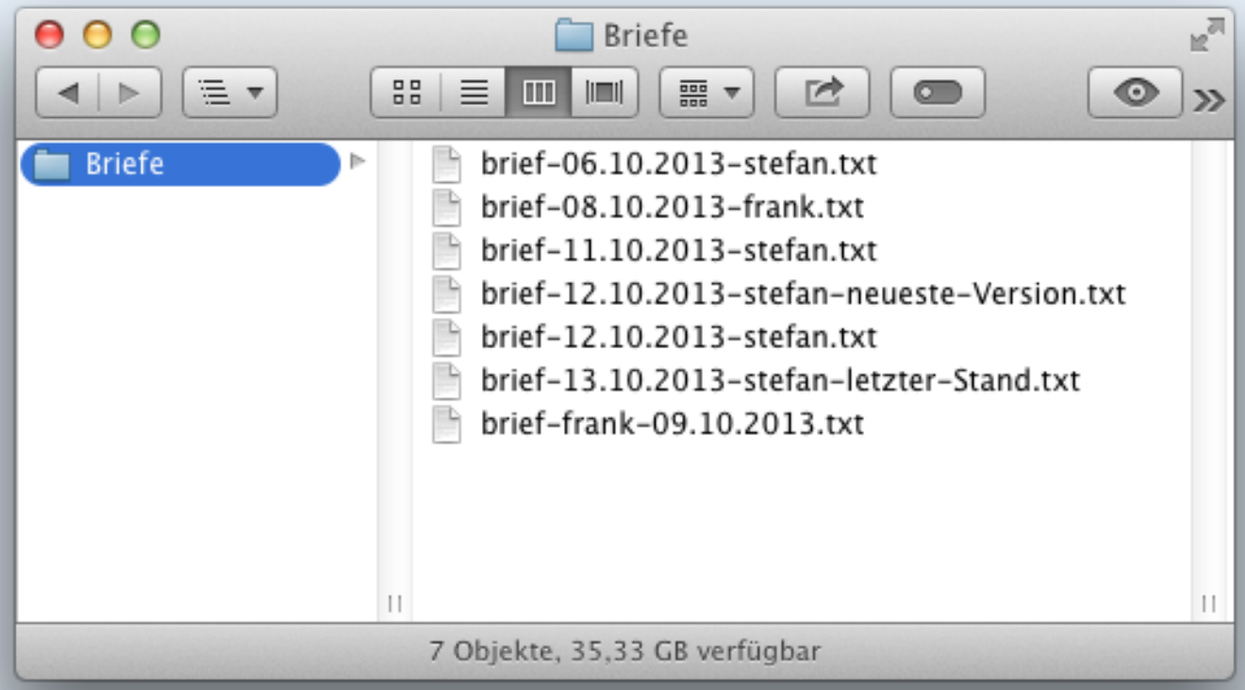
Es existieren drei Varianten von »brief.txt«, welche?

Typische Fragen

- Wo ist die aktuelle zentrale Version?
- Wo ist die Version 2.3?
- Wo ist die Version vom 12. Oktober 2013?
- Was hat sich seit letzter Woche geändert?
- Wer hat diese Änderung gemacht?
- Warum wurde die Änderung gemacht?

triviale Lösung

- Austausch der Dateien per
  - E-Email
  - Netzwerkfestplatte
  - USB-Stick
  - Cloud



- Definition einer Namenskonvention
  - z.B.: brief-12.10.2013-stefan.txt
- Kann es eine Namenskonvention für »*Wer hat wann was warum gemacht?*« geben?

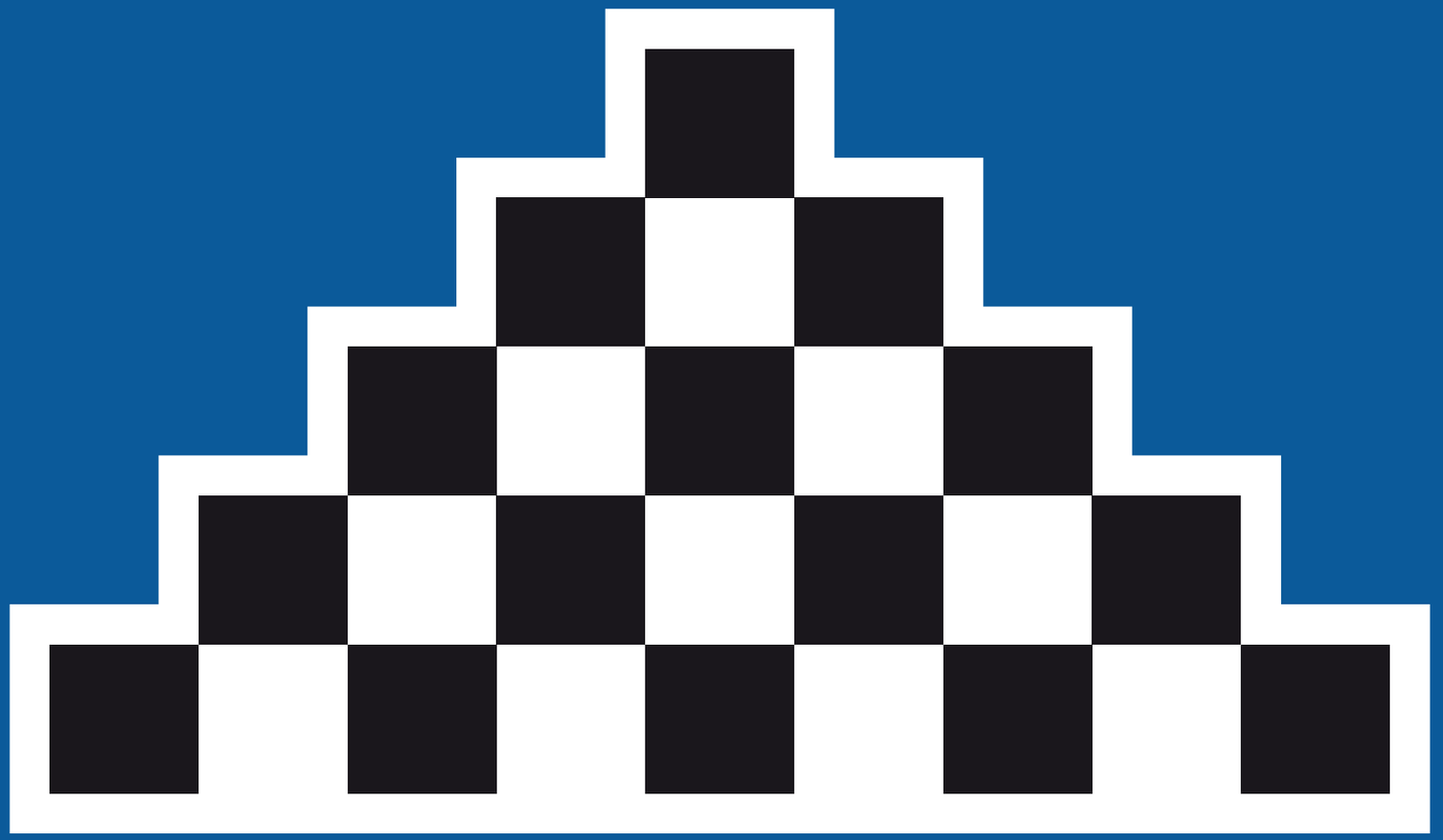
neue Probleme

- Konventionen werden eventuell nicht eingehalten
- Koordination ist aufwändig
- Verschiedene Zweige eines Dokuments werden manuell verwaltet
- Unterschiede zwischen Versionen nur schwer zu ermitteln

➡ Konventionen müssen technisch erzwungen werden!

Ziele

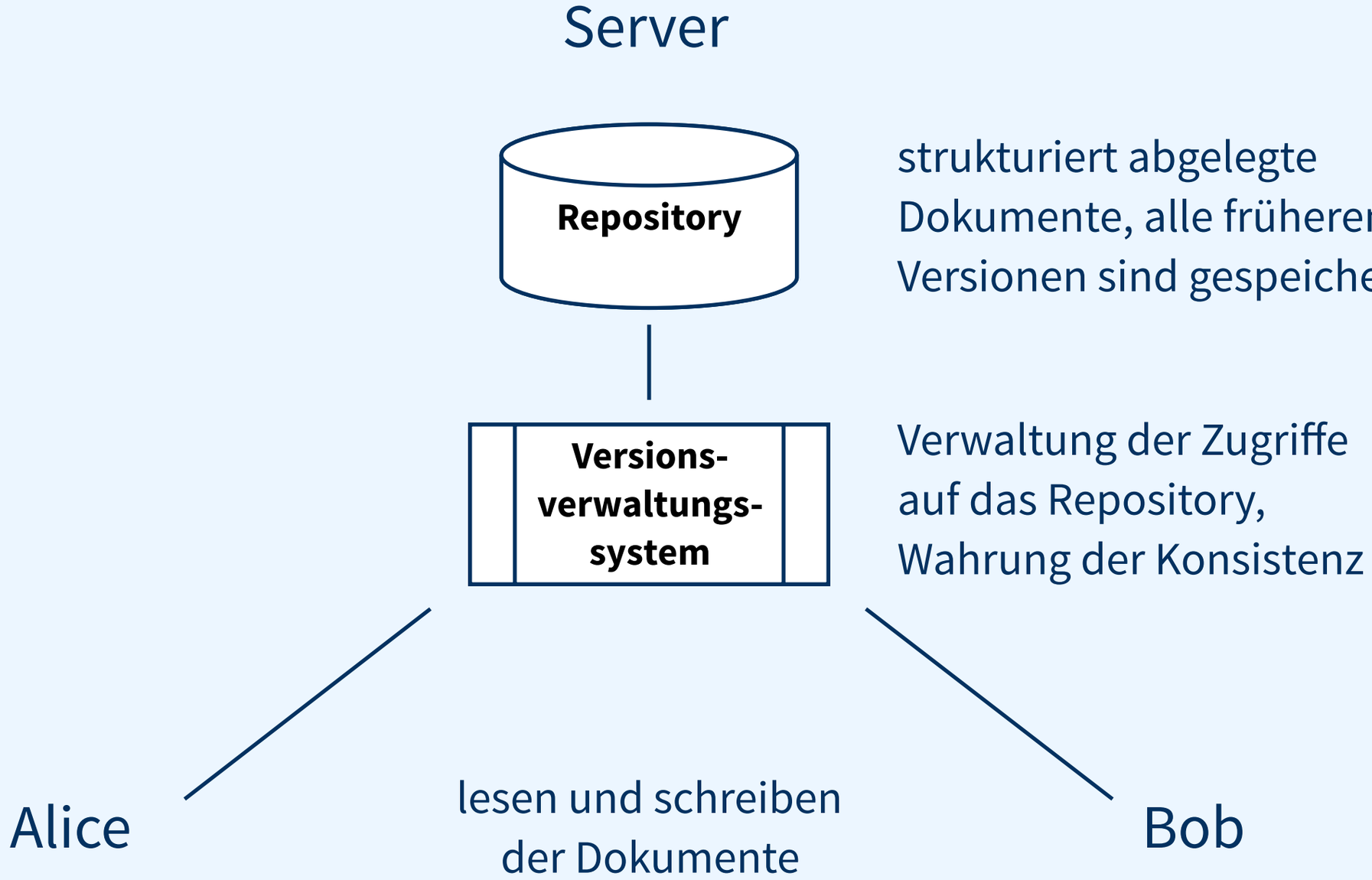
- Zugriff von allen Beteiligten auf die Entwicklungsartefakte
  - zentrale Verwaltung der (Quell-)Dokumente
- Konflikte bei der Bearbeitung vermeiden
  - mischen von verschiedenen Änderungen
  - alte Versionen sind wiederherstellbar
- Änderungen nachvollziehbar machen
  - »Wer« hat »wann« »was« geändert? (automatisch)
  - »Warum« wurde die Änderung vollzogen? (manuell)



Grundlagen

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

Versionsmanagement-Werkzeug



Konsistenz

- Ähnlich wie Datenbanken erlaubt das Versionsmanagement den schreibenden Zugriff mehrerer Nutzer
- Ähnlich wie bei Datenbanken führt die parallele Änderung des gleichen Datensatzes zu Inkonsistenzen

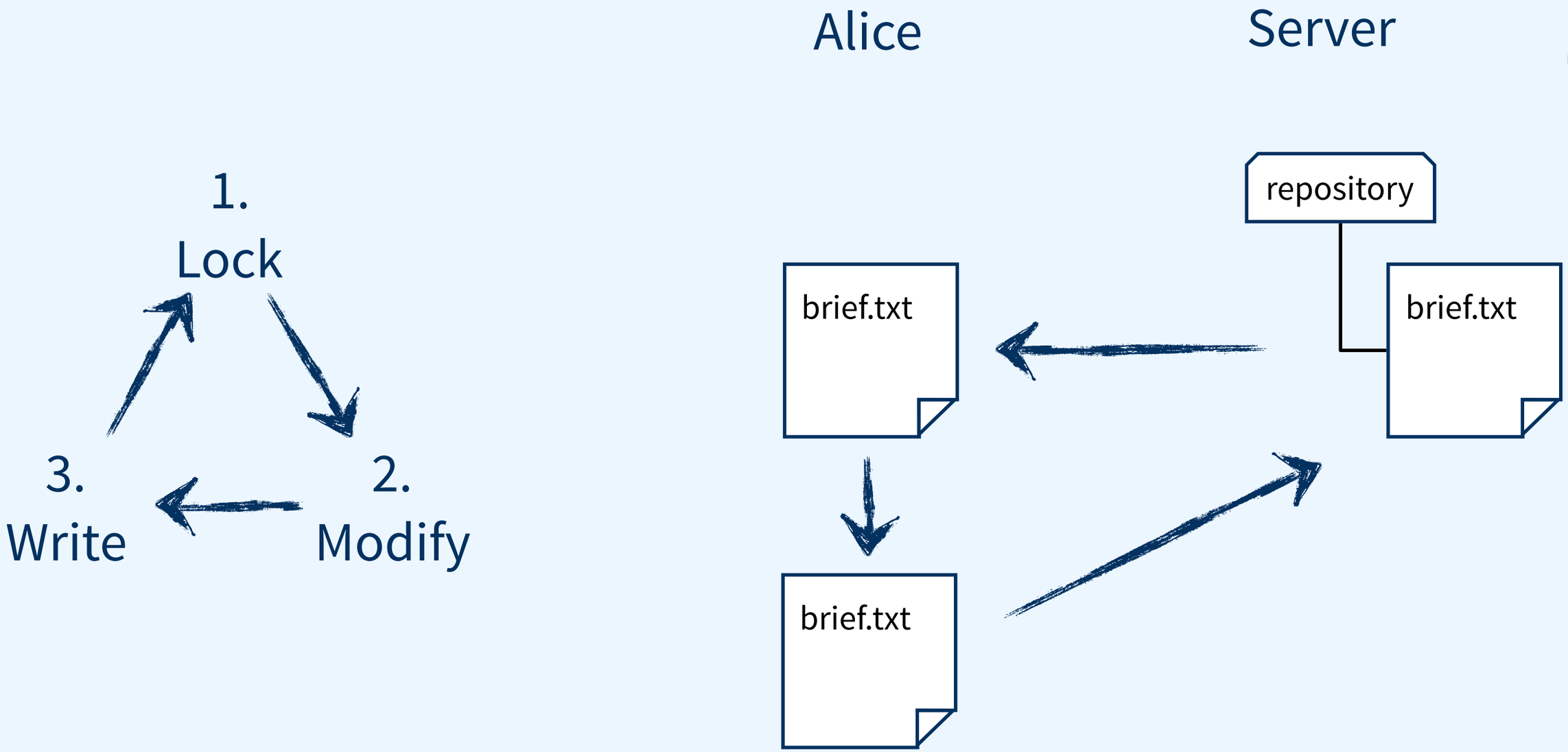
optimistisches Verfahren

- **System erlaubt gleichzeitiges Bearbeiten eines Dokuments**
- **Konflikte müssen aufgelöst werden**
- **Nur konfliktfreie Dokumente landen im Repository**
- **Konfliktauflösung: automatisch/manuell**

pessimistisches Verfahren

- **System verbietet gleichzeitiges Bearbeiten eines Dokuments**
- **Dokumente müssen erst gesperrt werden**
- **Sind nur vom Nutzer mit Sperre bearbeitbar**
- **Entsperren darf nicht vergessen werden!**

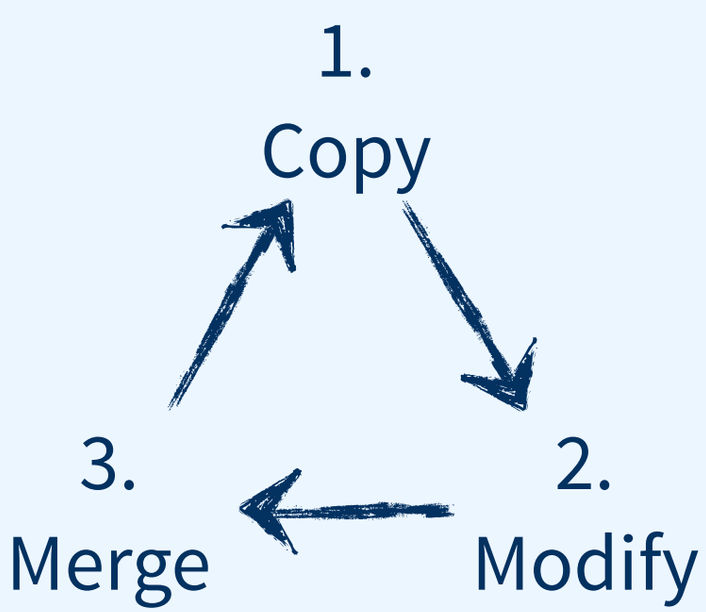
pessimistisches Verfahren

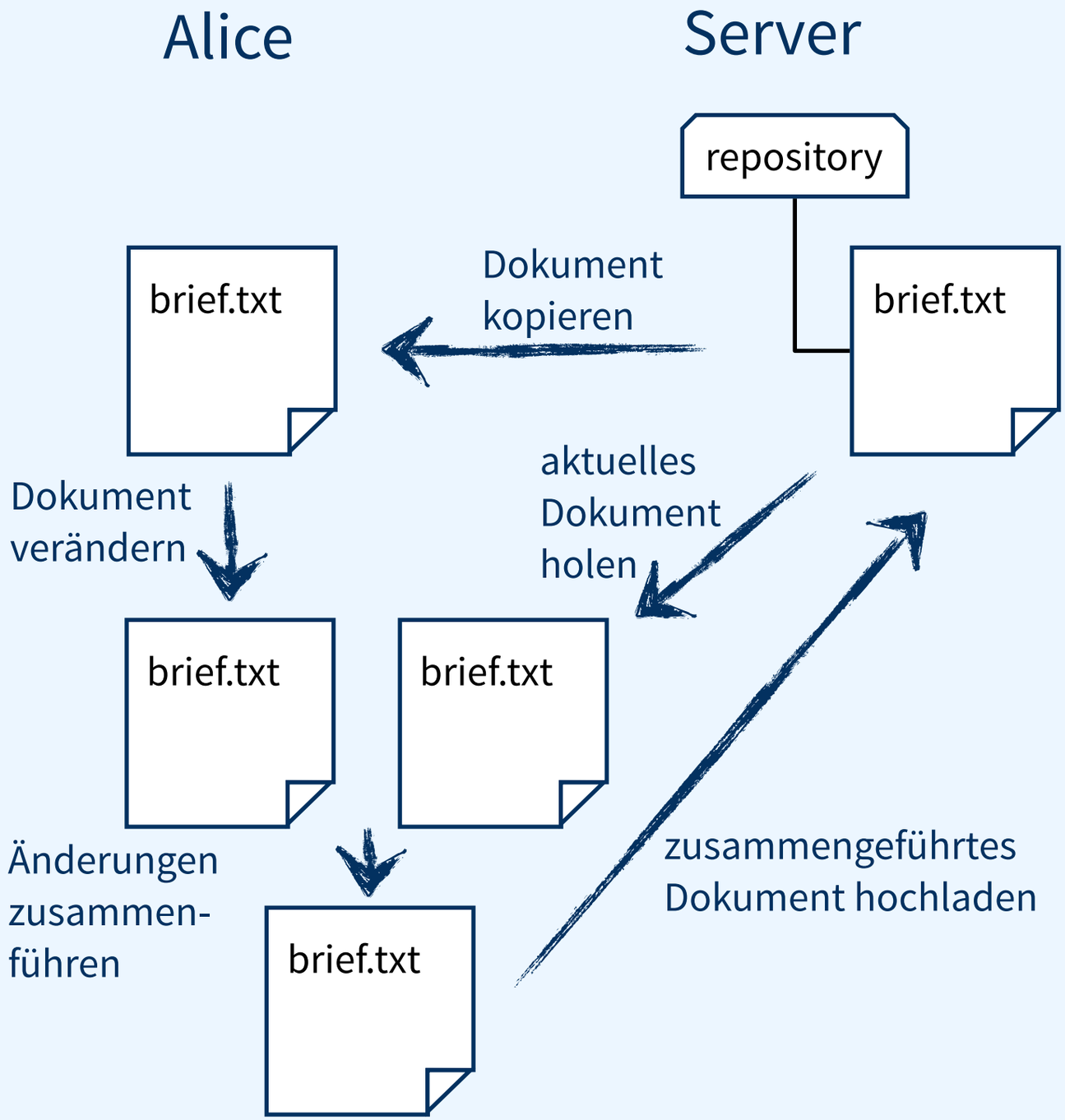


pessimistisches Verfahren

- Vorteil: Niemals Konflikte
- Sperrdauer aber sehr lang
  - mehrere Stunden → programmieren
  - mehrere Wochen → Programmierer im Urlaub
- Sperren sind häufig unnötig, siehe nächstes Verfahren
- Pessimistisches Verfahren wird nur selten verwendet
  - in dieser Veranstaltung gar nicht mehr

optimistisches Verfahren





optimistisches Verfahren

- Vorteil: Dokumente sind immer änderbar
- Nachteil: Konflikte können auftreten
- Zusammenführen von Dokumenten muss definiert sein
  - bei Binärdokumenten schwierig
    - » eine Änderung muss verworfen werden
  - bei Textdokumenten einfach
    - » unabhängige Teile des Dokuments können *automatisch* zusammengeführt werden
    - » Änderungen an der gleichen Textstelle müssen *manuell* zusammengeführt werden

Werkzeuge

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

Historie

SCCS

- 1972
- Source Code Control System
- Verwaltet nur einzelne Dateien

RCS

- 1980
- Revision Control System
- hohe Ähnlichkeit mit SCCS

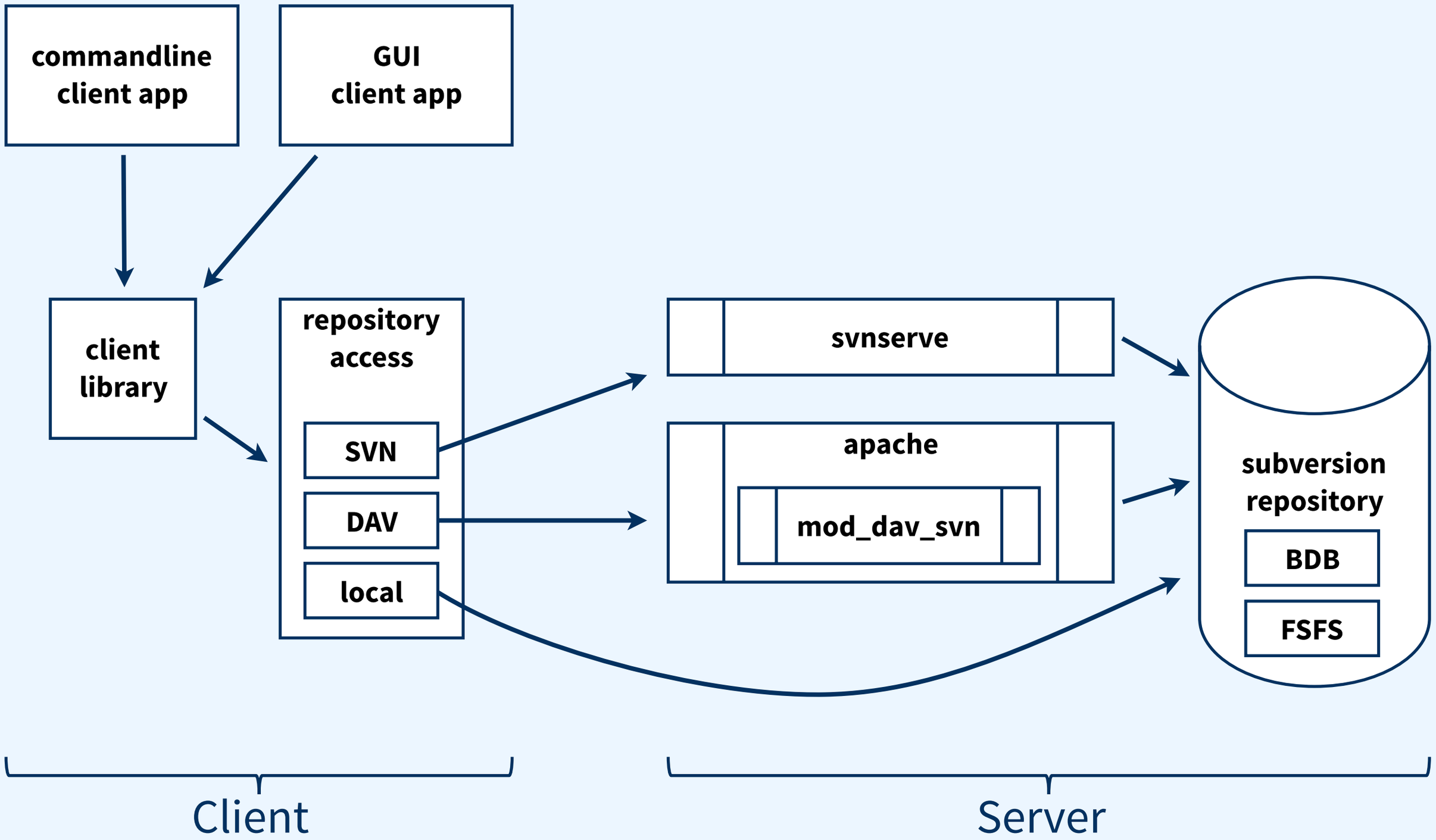
CVS

- 1989
- Concurrent Versions System
- kann Projekte verwalten

SVN

- 2000
- Subversion
- bessere Unterstützung für Verzeichnisse

Architektur



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

Kommandozeilen Client

- Ein Apache-Projekt
- Kann für alle gängigen Betriebssysteme heruntergeladen werden:
  - http://subversion.apache.org/packages.html
- Enthält Client- und Server-Komponenten
  - allerdings nur Kommandozeilenprogramme
- Alle lokalen Operationen von Subversion werden mit einem Kommando ausgeführt: »svn« gefolgt von der Operation
  - z.B.: »svn help«

```
$ svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client, version 1.7.5.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules
```

Repository

- Speicherbereich, in dem Subversion Dokumente und deren Metadaten ablegt
- Kann entweder lokal oder auf einem entfernten Server abgelegt sein
- Zugriff nur mittels Subversion-Clients
  - niemals direkt das Dateisystem manipulieren
- Erstellung eines lokalen Repository mittels »`svnadmin`«:

```
$ svnadmin create Repository
$ ls -la Repository
total 16
drwxr-xr-x   8 stefan  staff  272 22 Nov 15:44 .
drwxr-xr-x  12 stefan  staff  408 22 Nov 15:44 ..
-rw-r--r--   1 stefan  staff  229 22 Nov 15:44 README.txt
drwxr-xr-x   5 stefan  staff  170 22 Nov 15:44 conf
drwxr-sr-x  16 stefan  staff  544 25 Nov 10:02 db
-r--r--r--   1 stefan  staff    2 22 Nov 15:44 format
drwxr-xr-x  11 stefan  staff  374 22 Nov 15:44 hooks
drwxr-xr-x   4 stefan  staff  136 22 Nov 15:44 locks
```

Arbeitskopie

- Privater Arbeitsbereich (Verzeichnis) eines Nutzers
- Subversion verbindet Arbeitsbereich mit einem Bereich im Repository
- Nur aus einem Arbeitsbereich können Änderungen zum Repository geschickt werden
- Subversion legt ein verstecktes Verzeichnis an (.svn)
  - internes Verzeichnis, nicht manuell verändern!

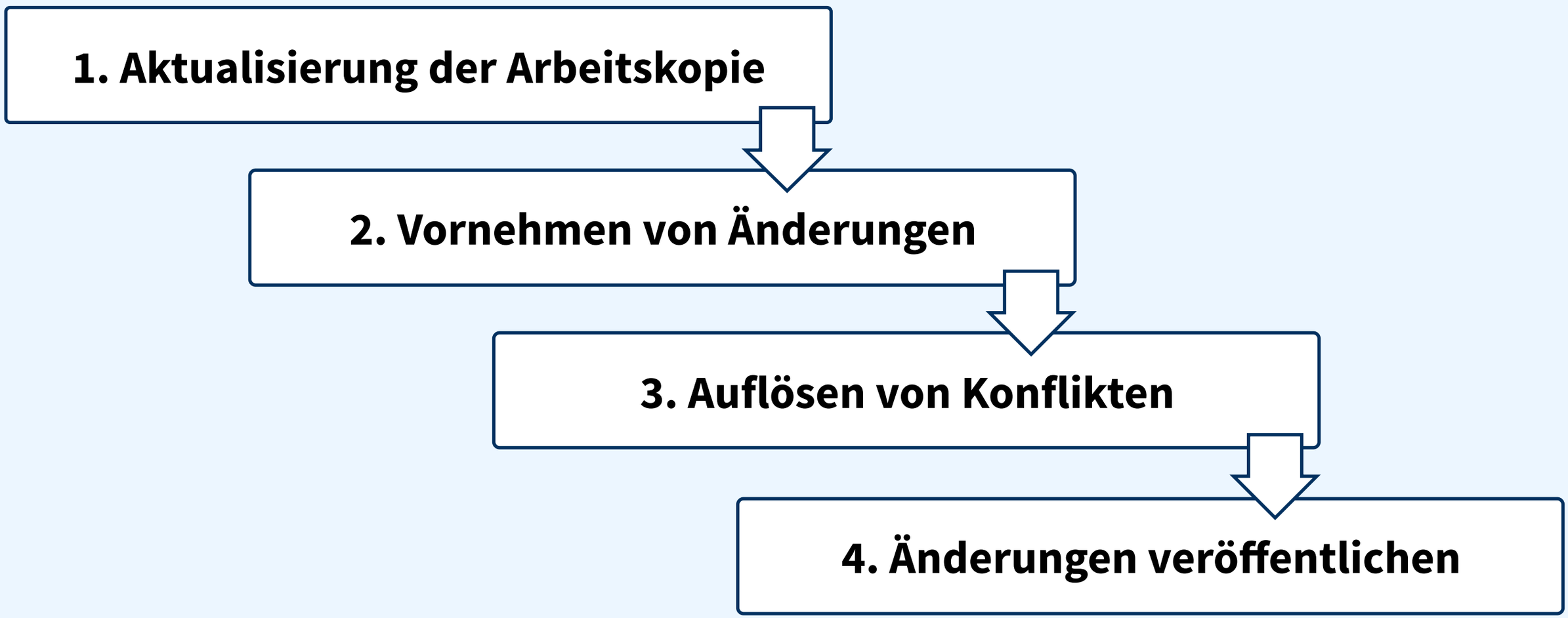
Check-out

- Um ein lokales Verzeichnis zu einer Arbeitskopie zu machen, sollten Sie ein Verzeichnis aus dem Repository »aus-checken«
- Subversion klont die Inhalte aus dem Repository (im Augenblick leer) und erstellt Kontrollstrukturen im Arbeitsverzeichnis:

```
$ mkdir ProjektA
$ cd ProjektA
/ProjektA $ svn checkout file:///Users/stefan/Temp/Repository/ .
Checked out revision 0.
/ProjektA $ ls -la
total 0
drwxr-xr-x   3 stefan  staff  102 22 Nov 11:16 .
drwxr-xr-x  11 stefan  staff  374 22 Nov 11:14 ..
drwxr-xr-x   7 stefan  staff  238 22 Nov 11:16 .svn
```

Arbeitszyklus

- Das einrichten des Repository und der Arbeitskopie ist eine einmalige Tätigkeit
- Der weitere Arbeitszyklus hat folgende Struktur:



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

Arbeitskopie aktualisieren

- Um Änderungen anderer Entwickler aus dem Repository in die Arbeitskopie zu übernehmen, muss diese aktualisiert werden
- Subversion bietet dafür das »`svn update`« Kommando
- Wir gehen später detaillierter auf diese Kommando ein!
  - leeres Repository → nichts zu aktualisieren

```
/ProjektA $ svn update
Updating '.':
At revision 0.
```

Vornehmen von Änderungen

- In der Arbeitskopie können Sie beliebig arbeiten:
- Existierende Dateien bearbeiten
- Neue Dateien erstellen
  - Subversion erkennt zwar neue erstellte Dateien, verwaltet diese Dateien aber nicht automatisch
  - mittels »`svn add Datei`« zur Verwaltung hinzufügen

```
/ProjektA $ emacs brief.txt
/ProjektA $ svn status
?       brief.txt
/ProjektA $ svn add brief.txt
A       brief.txt
/ProjektA $ svn status
A       brief.txt
```

Änderungen veröffentlichen

- Die Änderungen der Arbeitskopie werden an das Repository übertragen
- Die früheren Revisionen der Dateien bleiben im Repository gespeichert
- Das Kommando »`svn commit`« startet den Vorgang
- Commit funktioniert nur, wenn die Datei im Repository nicht durch einen anderen Entwickler verändert wurde (Konflikt)
- Commit erwartet eine kurze Beschreibung der Änderungen (Log Message)

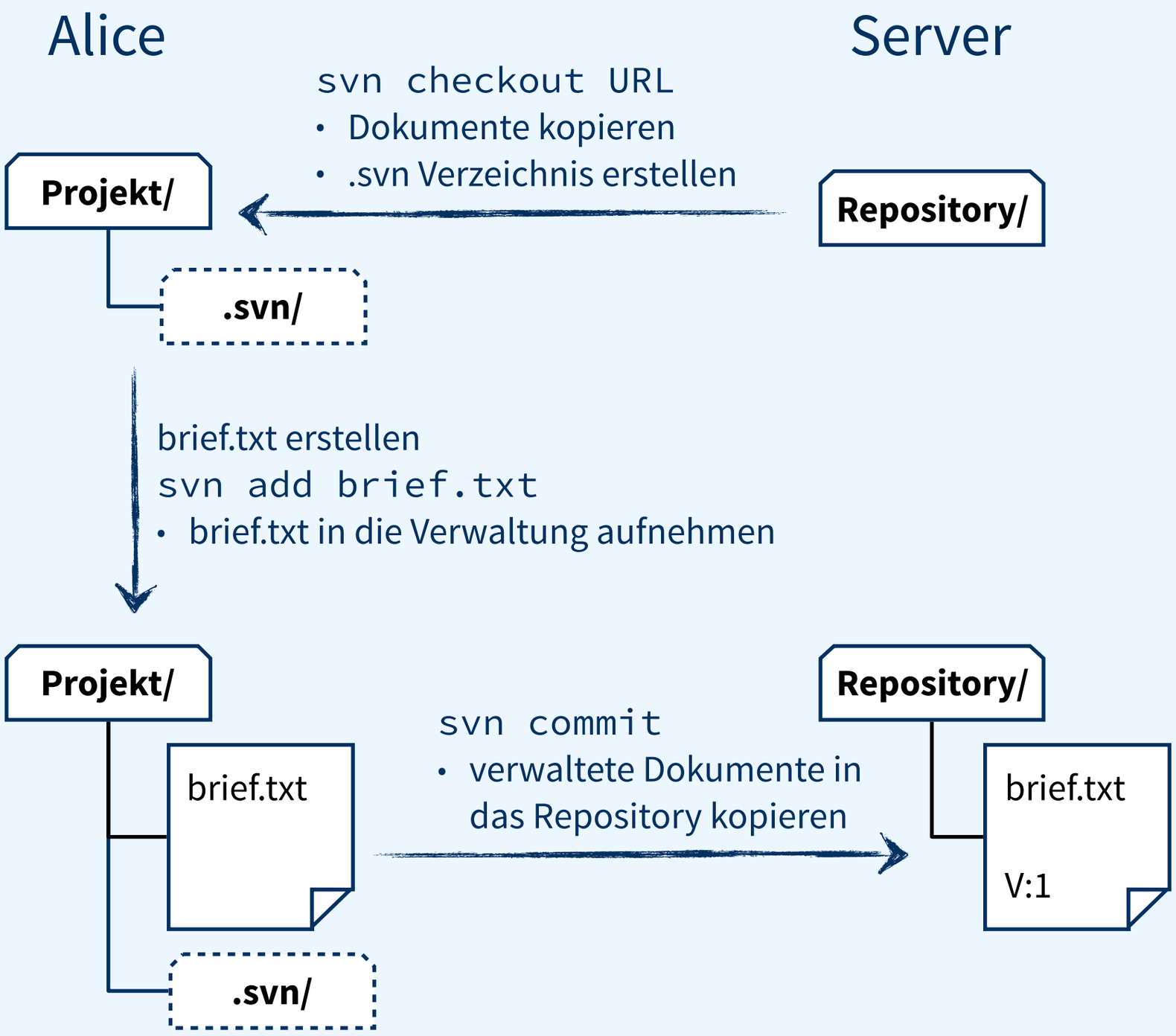
```
/ProjektA $ svn commit -m "brief hinzugefügt"
Adding         brief.txt
Transmitting file data .
Committed revision 1.
```

LOG-Message

- Eine Änderung sollte eine aussagekräftige Beschreibung bekommen (Log-Message)
- Sollte menschenlesbar den Grund der Änderungen beschreiben
- Kann die Kommunikation im Team erleichtern
- Können mit »`svn log [datei]`« abgerufen werden:

```
/ProjektA $ svn log brief.txt
------------------------------------------------------------------------
r1 | stefan | 2013-11-25 09:58:29 +0100 (Mo, 25 Nov 2013) | 1 line

brief hinzugefügt
------------------------------------------------------------------------
```

Arbeitszyklus

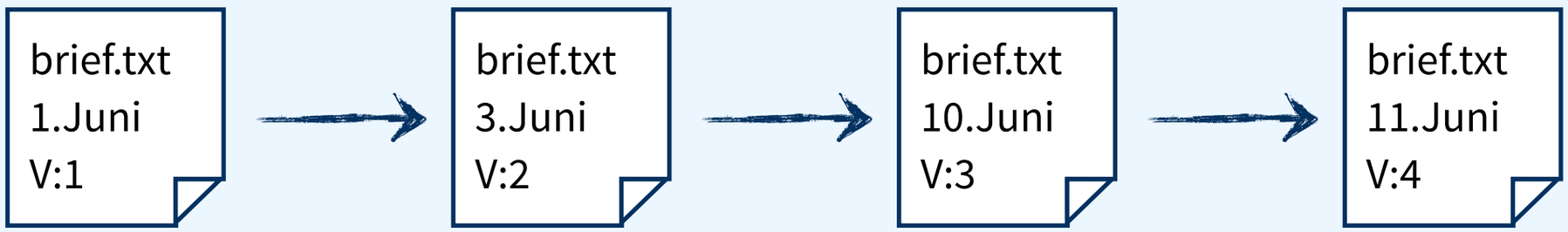


Welcher Schritt fehlt im Diagramm?

1. **Aktualisierung der Arbeitskopie**
2. **Vornehmen von Änderungen**
3. **Auflösen von Konflikten**
4. **Änderungen veröffentlichen**

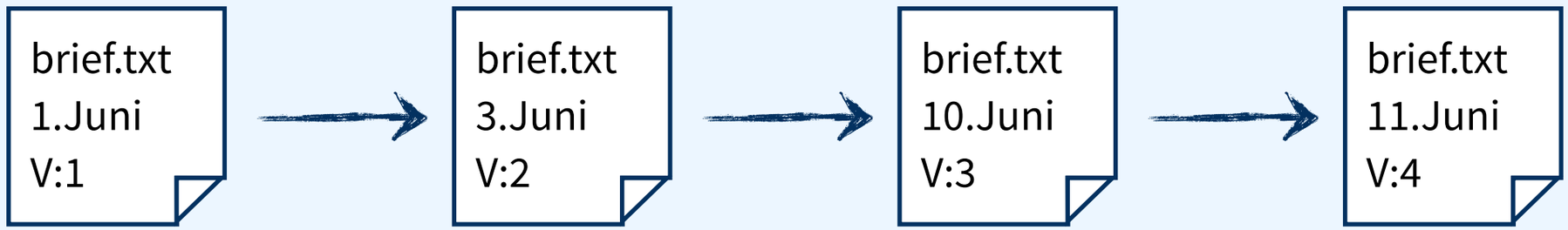
Revisionen

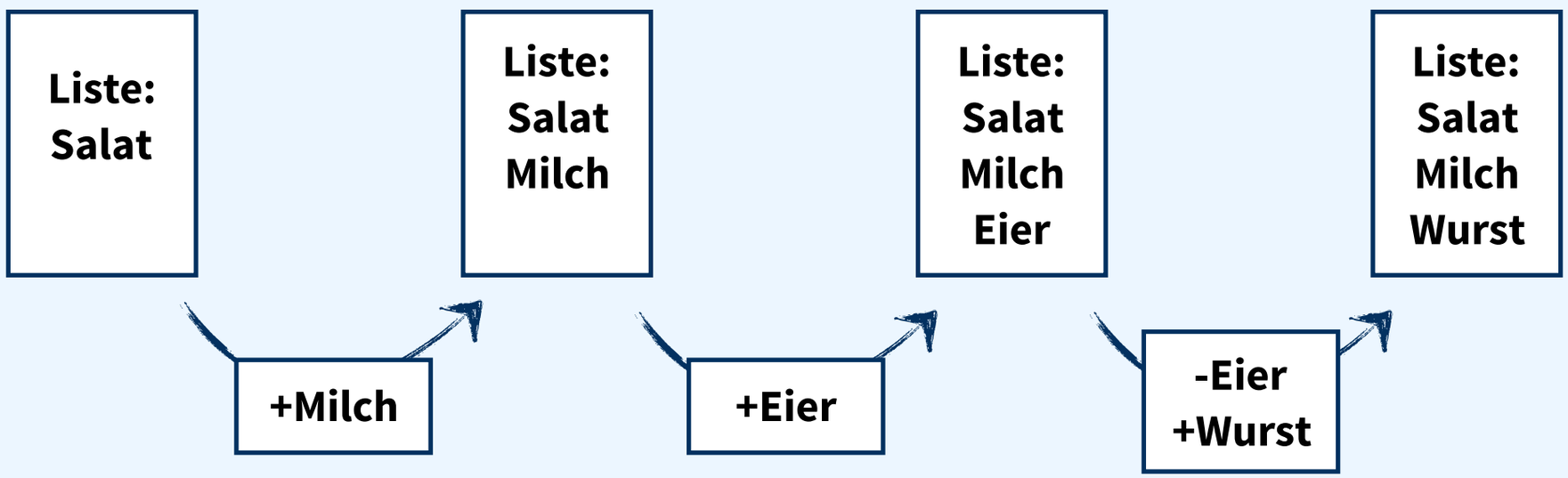
- Revisionen sind Änderungen, die im Versionsmanagementsystem gespeichert werden
- Eine Revision enthält:
  - Delta-Änderung des Dokuments
  - Autor, Datum, Revisionsnummer
  - Log-Nachricht
- Die Revisionen einer Datei werden als Historie bezeichnet
- Revisionen sind nicht löschbar!



Delta Beispiel

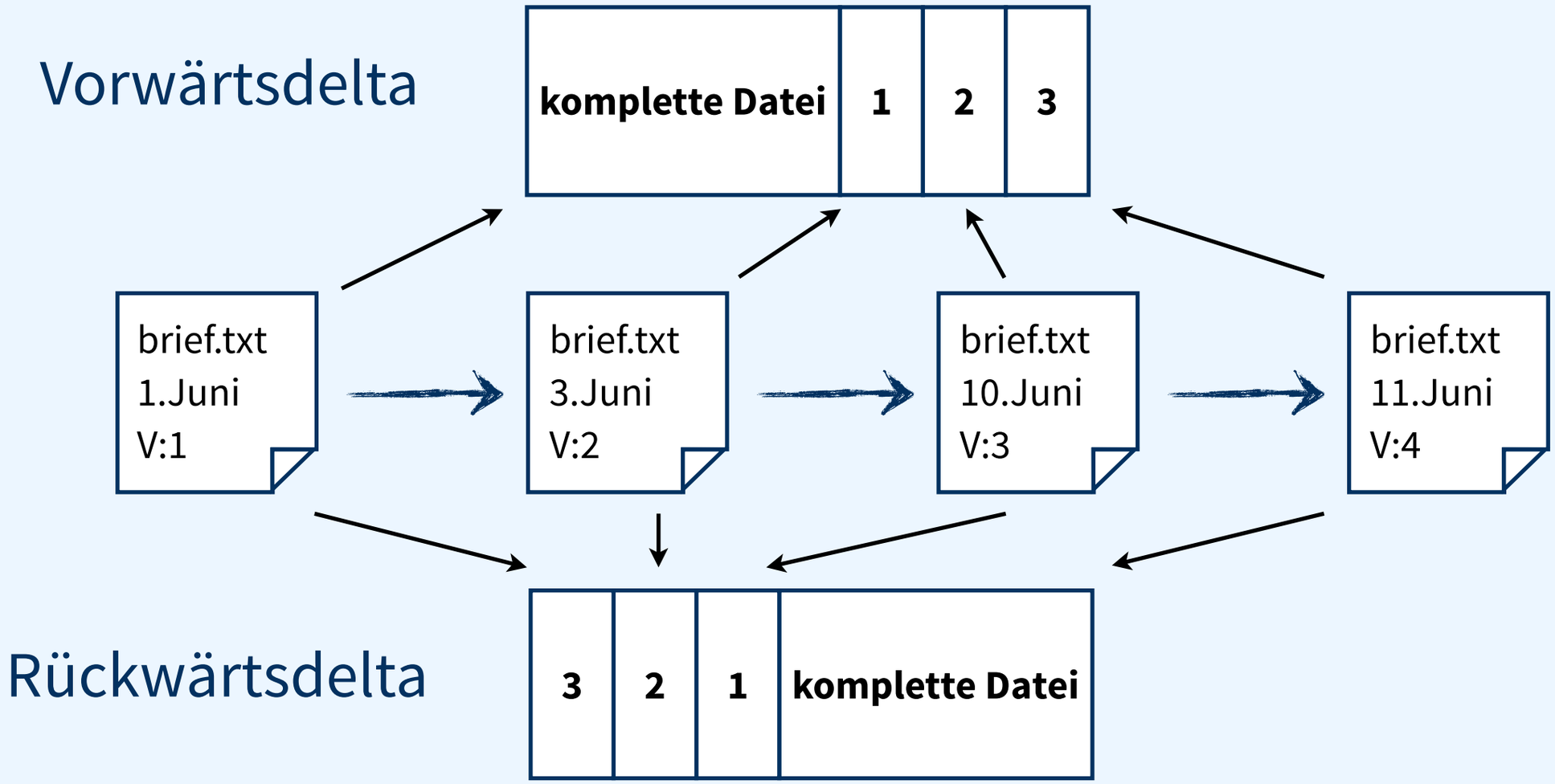
brief.txt enthält eine Einkaufsliste, die vier mal aktualisiert wird:





Delta-Änderungen

- Änderungen zwischen zwei Revisionen werden platzsparend als Deltas abgelegt
- Das Repository verwaltet Deltas intern, zwei Verfahren sind möglich:



Revisionsnummern

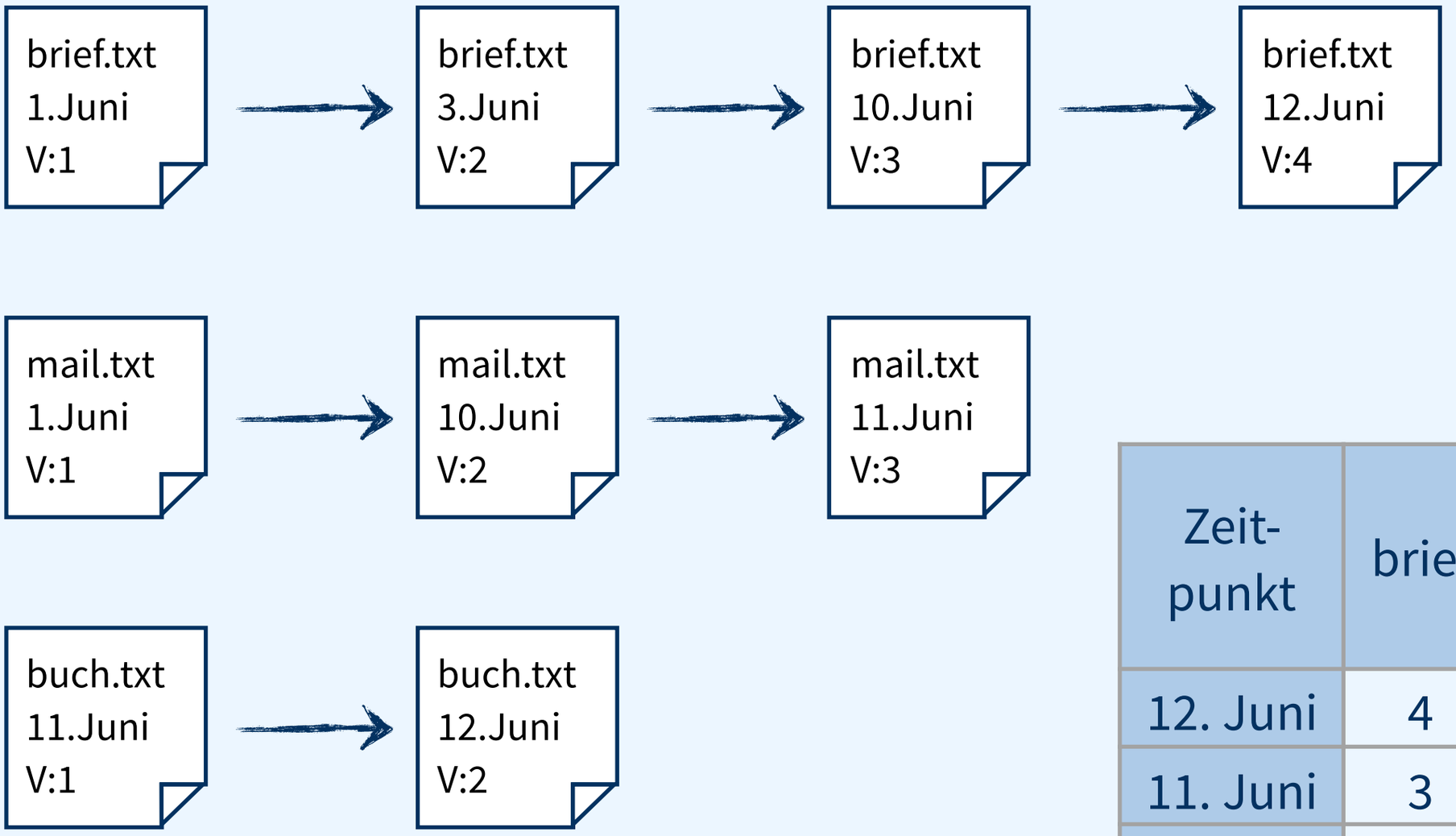
Dokument-Revisionen

- Jedes Dokument erhält eigene Revisionsnummer
- Bei jeder Änderung wird diese individuell hochgezählt
- Projekt-Revision ist Menge verschiedener Dokumentrevisionen

Commit-Revisionen

- Jedes Commit erhält eigene Revisionsnummer
- Jede geänderte Datei bekommt Revisions-nummer des Commits
- Projekt-Revision ist ein konkrete Commit-Revisionen

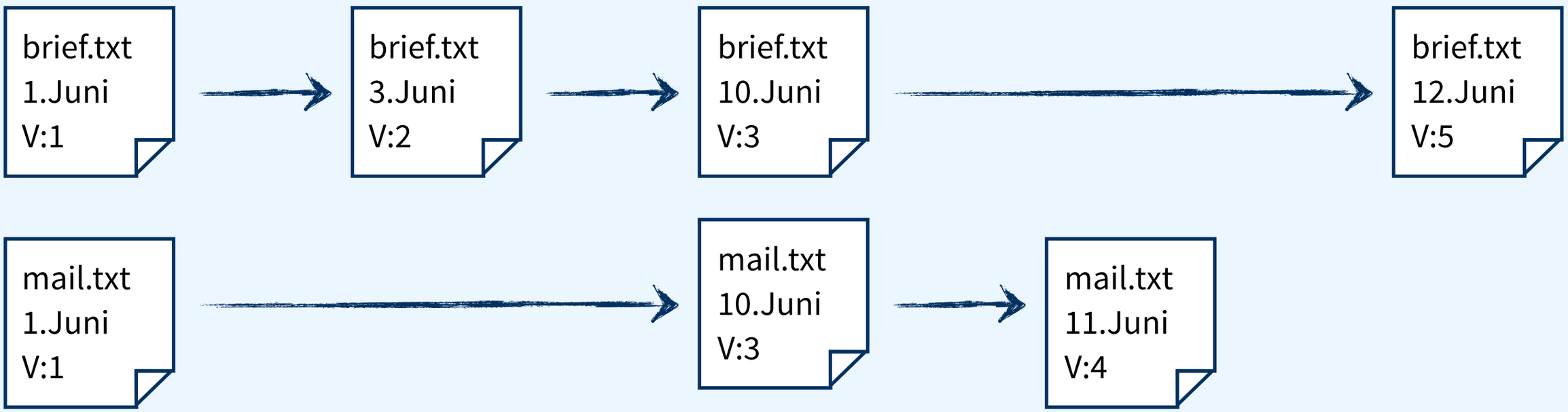
Dokument-Revisionen



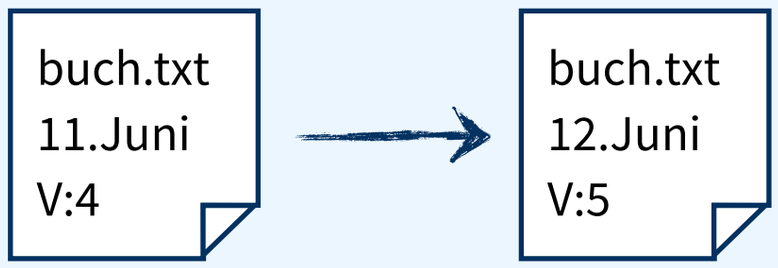
Zeit-punkt	brief	mail	buch
12. Juni	4	3	2
11. Juni	3	3	1
10. Juni	3	2	
3. Juni	2	1	
1. Juni	1	1	

Welchen Revisionen sind an welchen Zeitpunkten aktuell?

Commit-Revisionen



- Welche Dokumente (in welchen Revisionen) werden zurückgeliefert, wenn Commit-Revision 5 angefragt wird?

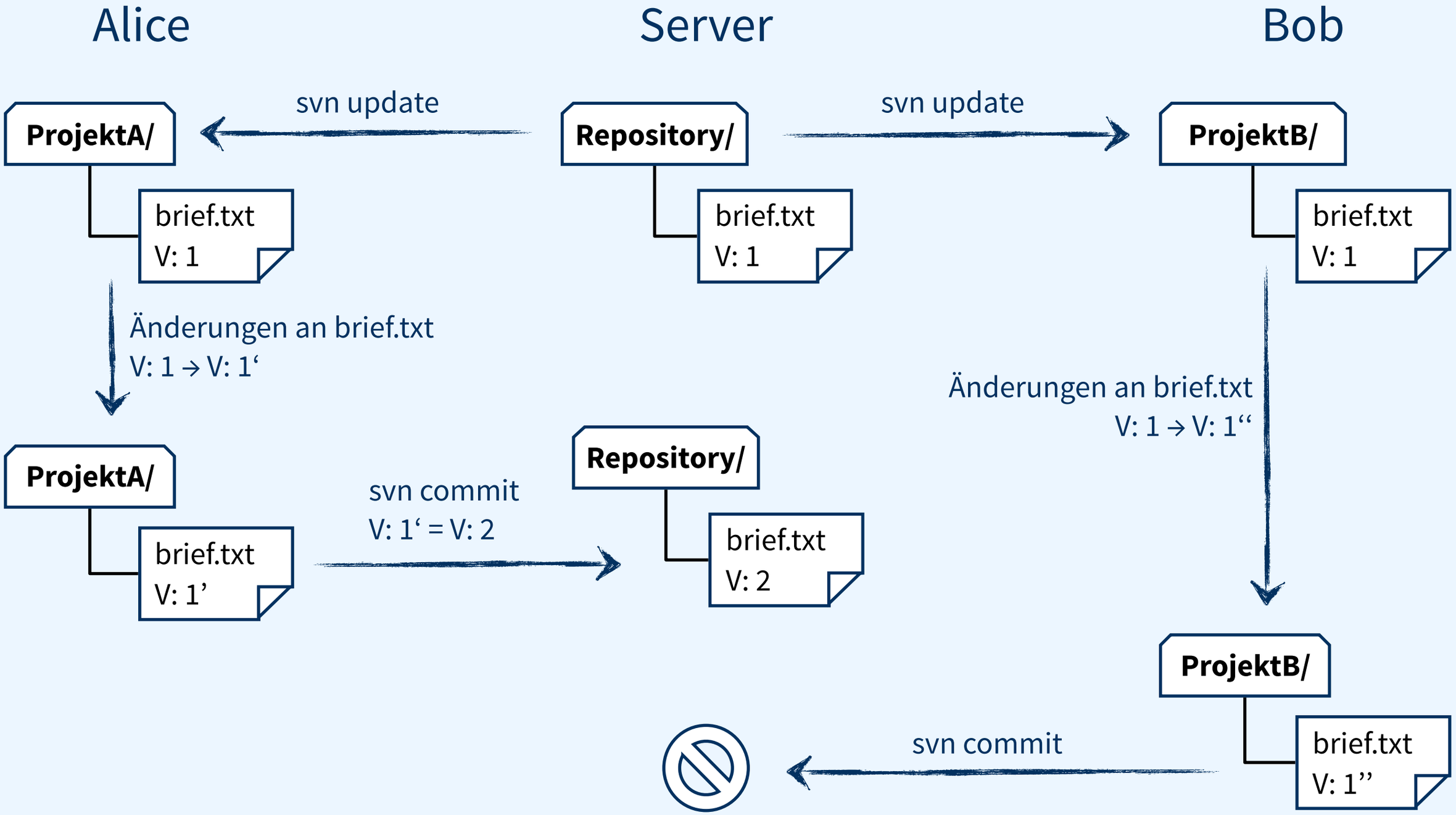


- **Nachteil:** Revisionsnummer ändern sich, obwohl Dokument (evtl.) unverändert
- **Vorteil:** Ein Zeitpunkt kann durch eine eindeutige Revisionsnummer identifiziert werden

Parallele Änderungen

- Bis jetzt hat nur ein Nutzer auf das Repository zugegriffen
  - trivialer Fall
  - Konflikte sind nicht möglich
- Das Repository soll aber von mehreren Nutzer verwendet werden
  - potenziell parallele Änderung der gleichen Datei
  - Konflikte können entstehen und müssen gelöst werden

Parallele Änderungen



Parallele Änderungen

Alice:

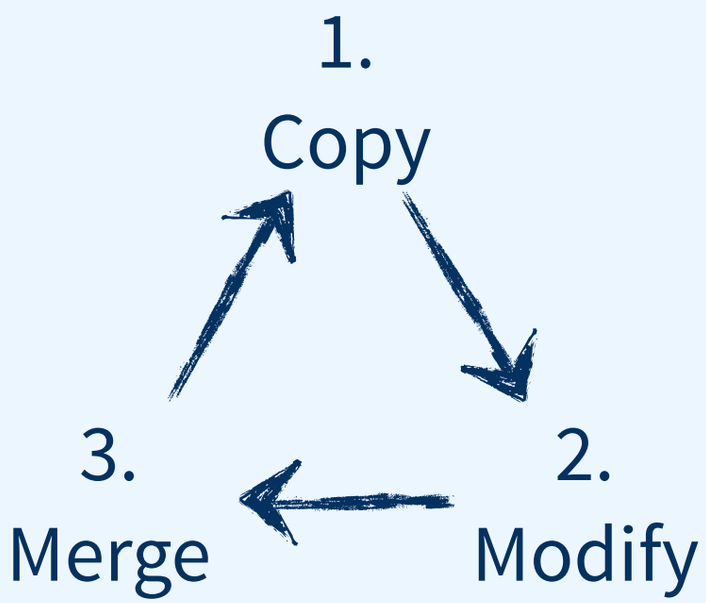
```
/ProjektA $ svn update
Updating '.':
At revision 1.



/ProjektA $ emacs brief.txt
/ProjektA $ svn status
M       brief.txt
/ProjektA $ svn commit -m "Änderungen alice"
Adding         brief.txt
Transmitting file data .
Committed revision 2.
```

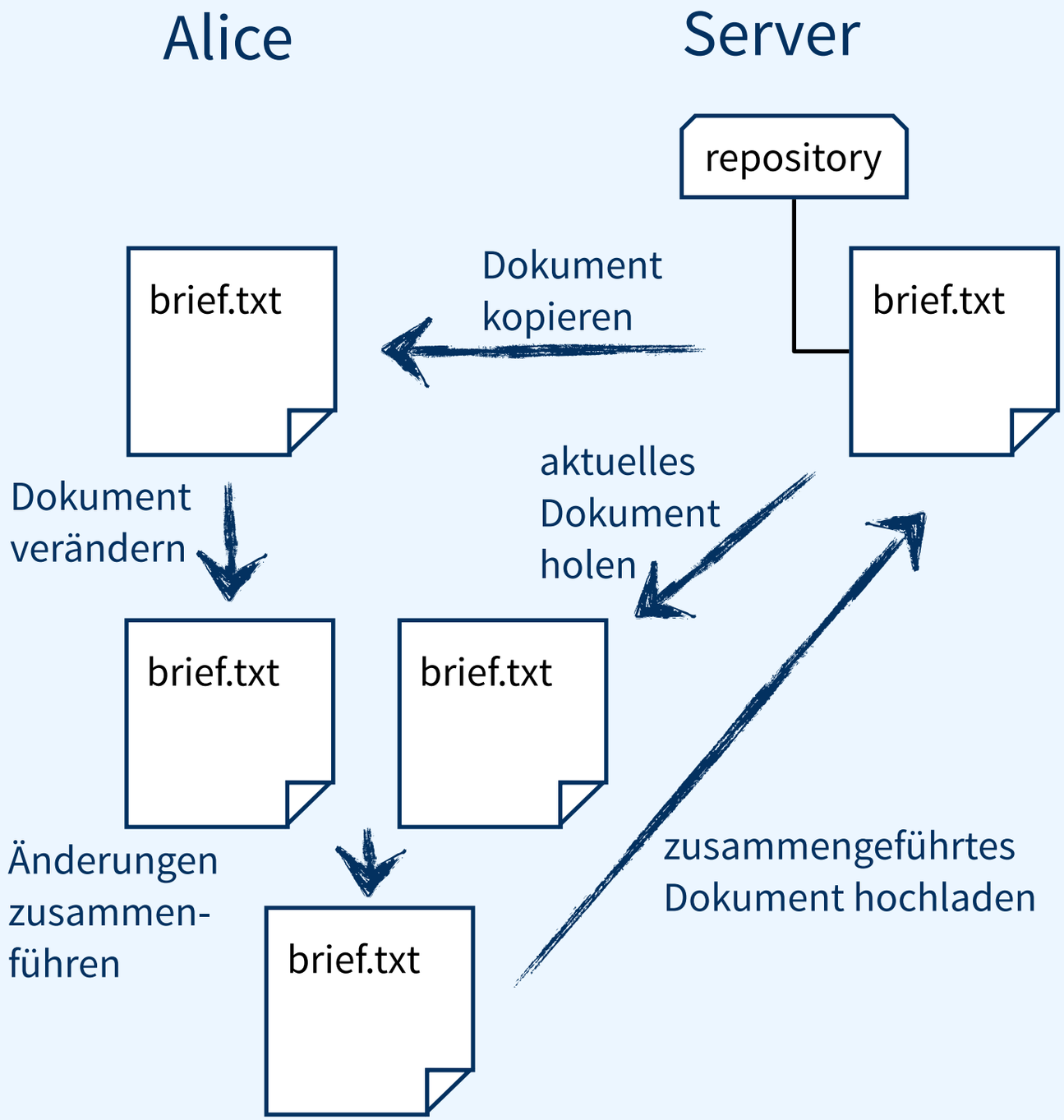
Bob:

```
/ $ mkdir ProjektB
/ $ cd ProjektB
/ProjektB $ svn checkout file:///Users/
stefan/Temp/Repository/ .
Checked out revision 1.
/ProjektB $ emacs brief.txt
/ProjektB $ svn status
M       brief.txt




/ProjektB $ svn commit -m "Änderungen bob"
Sending        brief.txt
svn: E155011: Commit failed (details
follow):
svn: E155011: File '/Users/stefan/Temp/
ProjektB/brief.txt' is out of date
```

Wie könnte dieses Problem gelöst werden?

Rückblick: optim. Verfahren





Merge

- Bevor Sie Änderungen einspielen müssen Sie:
  - aktuelle Version aus dem Repository holen
  - lokale und entfernte Änderungen mischen (merge)
- Danach klappt auch der »commit«
- Triviale Änderungen werden automatisch gemischt:

Bob:

```
/ProjektB $ svn commit -m "Änderungen bob"
Sending        brief.txt
svn: E155011: Commit failed (details follow):
svn: E155011: File '/Users/stefan/Temp/ProjektB/brief.txt' is out of date
/ProjektB $ svn update
Updating '.':
G    brief.txt
Updated to revision 2.
/ProjektB $ svn commit -m "Änderungen bob"
Sending        brief.txt
Transmitting file data .
Committed revision 3.
```

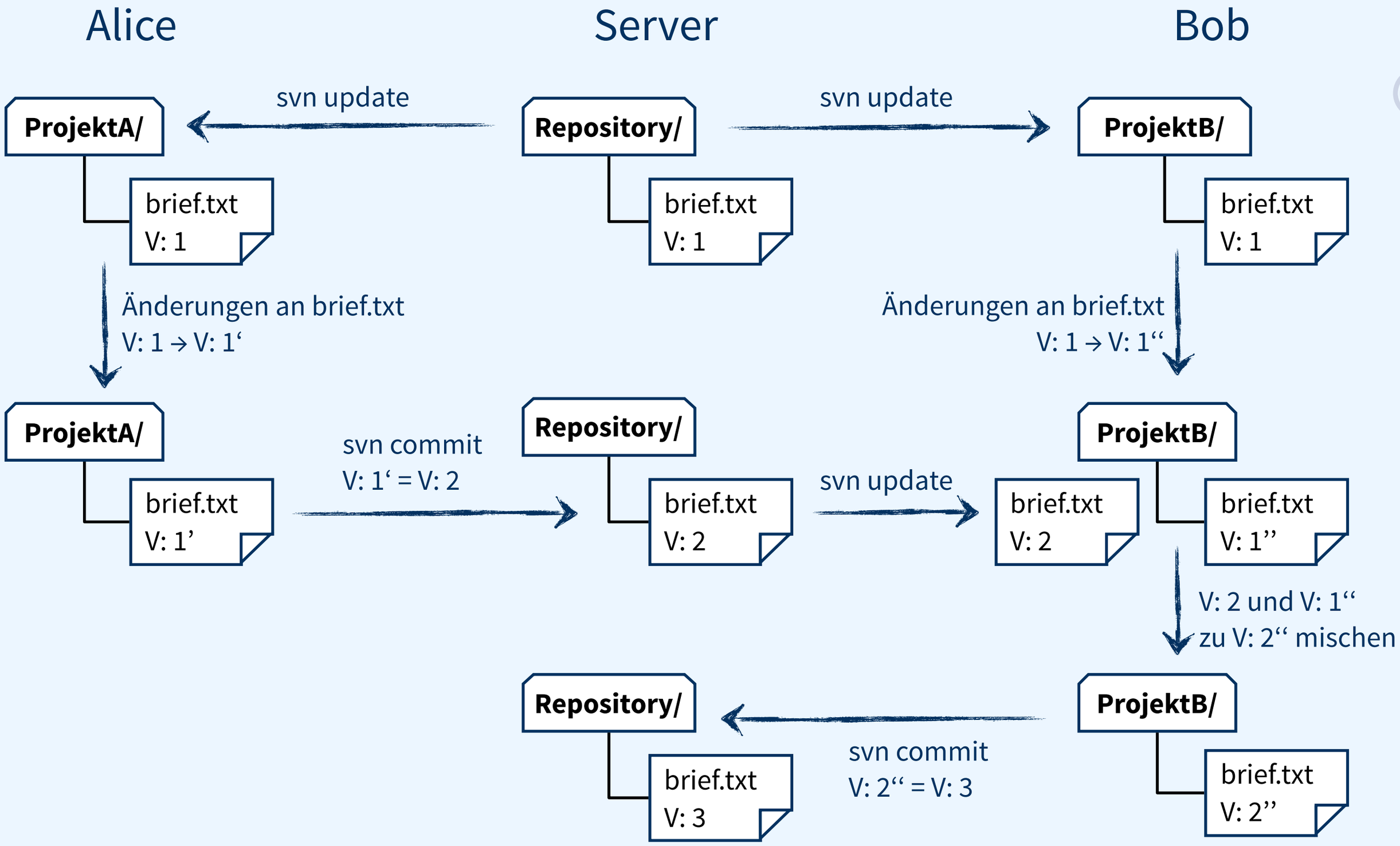
manuelles Merge

- Änderungen an den gleichen Stellen können nicht automatisch gemischt werden
  - mit einem Texteditor manuell mischen (unten)
  - besser ein GUI-Programm dafür verwenden (später)

Bob:

```
/ProjektB $ svn commit -m "Änderungen bob"
Sending        brief.txt
svn: E155011: Commit failed (details follow):
svn: E155011: File '/Users/stefan/Temp/ProjektB/brief.txt' is out of date
/ProjektB $ svn update
Updating '.':
Conflict discovered in '/Users/stefan/Temp/ProjektB/brief.txt'.
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: r
G    brief.txt
Updated to revision 2.
/ProjektB $ svn commit -m "Änderungen bob"
Sending        brief.txt
Transmitting file data .
Committed revision 3.
```

Parallele Änderungen



Zusammenfassung

Begriffe

Begriff	Erklärung
Repository	zentraler Ablageort, gesamte Historie aller Dokumente
Arbeitskopie	lokaler Arbeitsbereich, jeder Nutzer besitzt einen eigenen
Revision	Erstellung einer neuen Version des Dokuments im Repository
Delta	Unterschiede eines Dokument zwischen den Revisionen
Log	Vom Nutzer einzugebende Beschreibung der Änderungen
Konflikt	Gleichzeitige Änderung der Arbeitskopie und des Repository

Operationen

Operation	Erklärung
Create	Erstellung eines Repository; einmalige Aktion, durchgeführt von einem Administrator
Check-Out	Herunterladen von Dokumenten aus dem Repository und Erstellung einer lokalen Arbeitskopie
Update	Aktualisieren der lokalen Arbeitskopie mit neuen Dokumenten (anderer Nutzer) aus dem Repository
Commit	Hochladen der Änderungen aus der lokalen Arbeitskopie in das Repository
Merge	Mischen der durch ein »update« heruntergeladenen Änderungen aus dem Repository mit der Arbeitskopie

Status Meldungen

Statuscodes in den Ausgaben von Subversion:

```
/ProjektA $ svn status
?       email.txt
M       brief.txt
```

Status	Bedeutung
'U'	Dokument wurde aus dem Repository aktualisiert
' '	keine Änderung
'A'	Dokument zum Hinzufügen vorgesehen
'D'	Dokument zum Löschen vorgesehen
'M'	Dokument wurde lokal verändert
'C'	Dokument befindet sich im Konflikt mit dem Repository
'?'	Dokument unterliegt nicht der Versionskontrolle

Best Practices

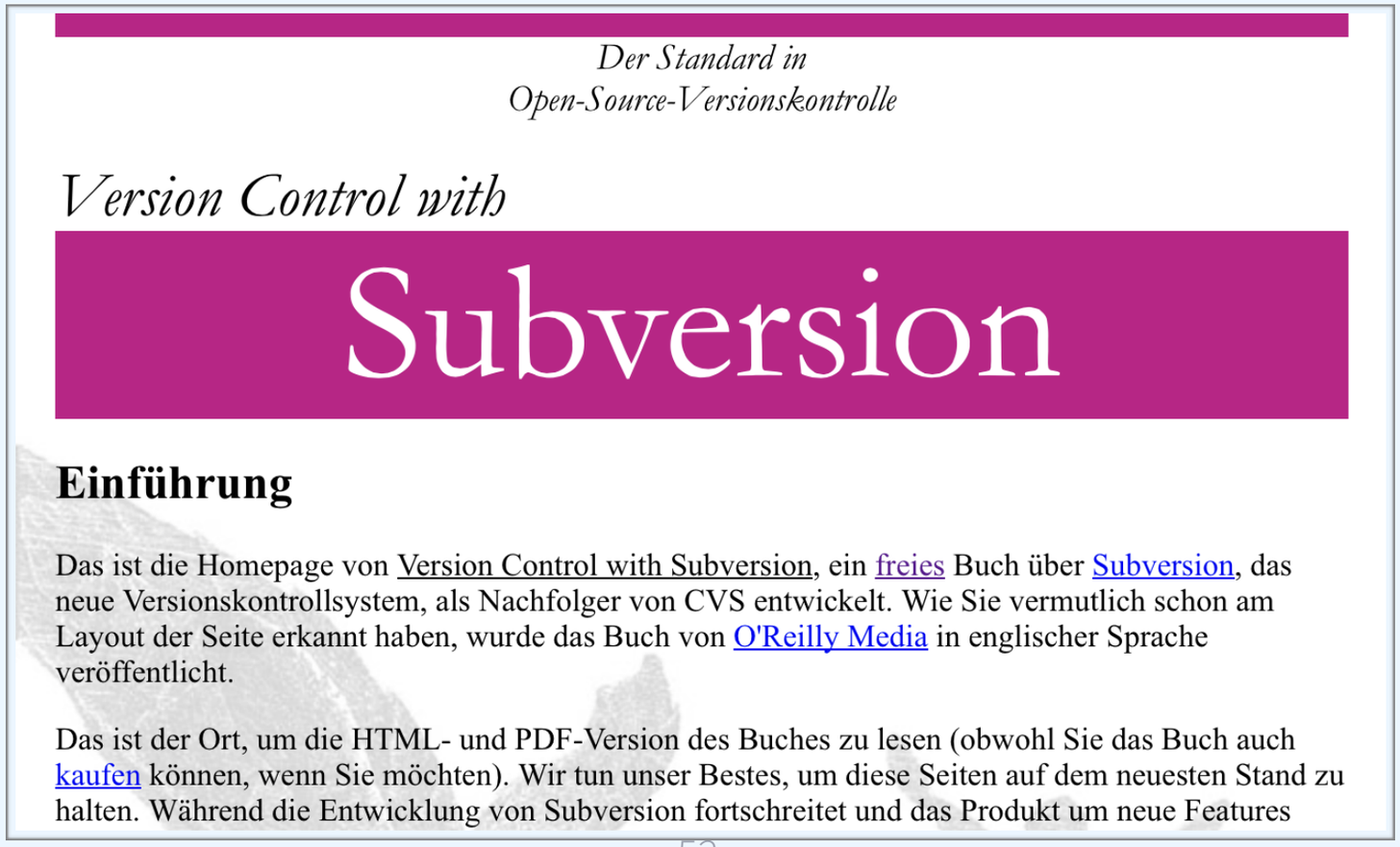
- Nur Quelldokumente einchecken
  - nur `*.java`, keine `*.class`
  - mittels `svn:ignore` können lokale Dokumente dauerhaft vom einchecken ausgeschlossen werden
- Keinen fehlerhaften Code einchecken
  - Projekte werden auch von anderen gebaut
    - » Kompilierfehler bremsen alle Entwickler aus
  - für größere Experimente Branches verwenden (nächste Woche)

Best Practices

- Aussagekräftige Commit-Kommentare
  - schlecht: „calculateSum() geändert“
  - besser: „uninitilialisierten Zugriff behoben“
  - am Besten: „Bug #123 behoben, verursacht durch uninitilialisierten Zugriff in Methode calculateSum()“
- Keine sehr langen Zeilen einchecken
  - z.B. in HTML oder LaTeX
  - da zeilenbasiertes Mischen
- Binär-Dateien nur wenn nötig einchecken
  - »Merge« nicht möglich, Konflikte nicht auflösbar
  - Revisionen verschwenden Speicherplatz

Literatur

- SVN Buch »Versionskontrolle mit Subversion«
  - in gedruckter Form von O'Reilly Media
  - in elektronischer Form unter der Creative-Commons-Lizenz: http://svnbook.red-bean.com



Der Standard in
Open-Source-Versionskontrolle

Version Control with

Subversion

Einführung

Das ist die Homepage von Version Control with Subversion, ein freies Buch über Subversion, das neue Versionskontrollsystem, als Nachfolger von CVS entwickelt. Wie Sie vermutlich schon am Layout der Seite erkannt haben, wurde das Buch von O'Reilly Media in englischer Sprache veröffentlicht.

Das ist der Ort, um die HTML- und PDF-Version des Buches zu lesen (obwohl Sie das Buch auch kaufen können, wenn Sie möchten). Wir tun unser Bestes, um diese Seiten auf dem neuesten Stand zu halten. Während die Entwicklung von Subversion fortschreitet und das Produkt um neue Features

Subversion Cheat Sheet

- Kurzzusammenfassung der wichtigsten Kommandos:
  - http://cheatography.com/davechild/cheat-sheets/subversion

Cheatography

Subversion Cheat Sheet

by Dave Child (DaveChild) via cheatography.com/1/cs/3/

Subversion Resources

Homepage	http://subversion.apache.org/
SVN Book	http://svnbook.red-bean.com/

Subversion Components

svn	Command line program
svnversion	Revision of working copy
svnlook	Inspect repository
svnadmin	Repository administration
svndumpfilter	Filter repository stream

Subversion Checkout Working Copy

$ svn checkout "/path/to/repository"

Checkout working copy into current folder

$ svn checkout "/path/to/repository" "/path/to/folder"

Checkout working copy into target folder

Subversion Update Working Copy

$ svn update "/path"

Update path

$ svn update -r9 "/path"

Update path to revision 9

Danke

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