Versionskontrolle: Motivation

BBZW, Modul 426

Patrick Bucher

22.02.2024

Wozu?

Wozu brauchen wir eine Versionskontrolle?

Beispiel: Projektarbeit

Sie arbeiten während längerer Zeit an einem Projekt.

- Aus Versehen löschen Sie in einem Dokument eine ganze Seite.
- Wenn Sie das sofort bemerken, können Sie die Änderung rückgängig machen.
- Wenn Sie aber weiterarbeiten, immer wieder speichern und das Dokument schliessen, geht die gelöschte Seite verloren.

Sie machen tägliche Backups, verlieren im schlimmsten Fall die Arbeit eines Tages.

“Lösung” 1: Handgestrickte Versionskontrolle

Vorher (ohne Versionskontrolle):

```
Projektarbeit.docx
```

Nachher (mit “Versionskontrolle”):

```
AKTUELLSTE-Projektarbeit.docx
NEU-AKTUELLSTE-Projektarbeit.docx
Projektarbeit.docx
Projektarbeit-backup-Montag.docx
Projektarbeit-alt.docx
Projektarbeit-V3.docx
Projektarbeit-V3-besser.docx
Projektarbeit-neues-Design.docx
Projektarbeit-nach-Zwischenkorrketur.docx
```

Probleme der handgestrickten Versionskontrolle

- Schlechte Übersicht:
  - Welches Artefakt repräsentiert den aktuellen Stand?
  - Welche alten Artefakte können gelöscht werden?
- Keine konsequente Versionierung:
  - Wann und warum wird eine Datei “gesichert”?
- Schwere Rückverfolgung:
  - Auf welchem Stand basiert ein Artefakt?
- Keine Historisierung:
  - Welche Änderungen wurden in welcher Reihenfolge vorgenommen?
  - Warum wurde eine bestimmte Änderung vorgenommen?

“Lösung” 2: Cloud Storage, Dokumentverwaltung

Dropbox, Sharepoint, OneDrive usw. lösen das Problem teilweise:

- Übersicht: Verzeichnis repräsentiert einen zeitlichen Zustand
- Rückkehr zu früherem Zustand möglich
- Änderungshistorie vorhanden (wer, wann)

Ungelöste Probleme (je nach Lösung):

- Versionsübergänge implizit
  - Version = Speichervorgang
- Rückverfolgung schwer
  - Umbenennung = neues Artefakt
- Historisierung rudimentär
  - nur lineare Zeitachse
  - Änderungskommentare optional

Dokumentverwaltung und Codeverwaltung

Dokumente

- Binärdateien (z.B. `.docx`, `.pdf`): schwer vergleichbar
  - Zusammenführung von Änderungen oft manuell
- Änderungen betreffen oft nur eine Datei

Quellcode

- Textdateien (z.B. `.go`, `.py`): zeilenweise vergleichbar
  - Zusammenführung von Änderungen weitgehend automatisch
- Änderungen betreffen häufig viele Dateien

Die Verwaltung von Programmcode erfordert mächtigere Werkzeuge.

Dokumente *können* Quellcode (z.B. Markdown, LaTeX) sein.

Zentrale Versionskontrolle (I)

- Alle Dateien werden auf einem zentralen Server gespeichert.
- Clients laden nur den aktuellen Stand (*Snapshot*) herunter.
- Vorteile:
  - ein aktueller Stand (global)
  - man sieht, was andere machen
  - zentrale Rechteverwaltung, zentrales Backup
- Nachteile:
  - *Single Point of Failure*
  - funktioniert nur teilweise offline
  - Konflikte werden erst beim Hochladen bemerkt
- Beispiele: CVS, Subversion, Perforce

Zentrale Versionskontrolle (II)

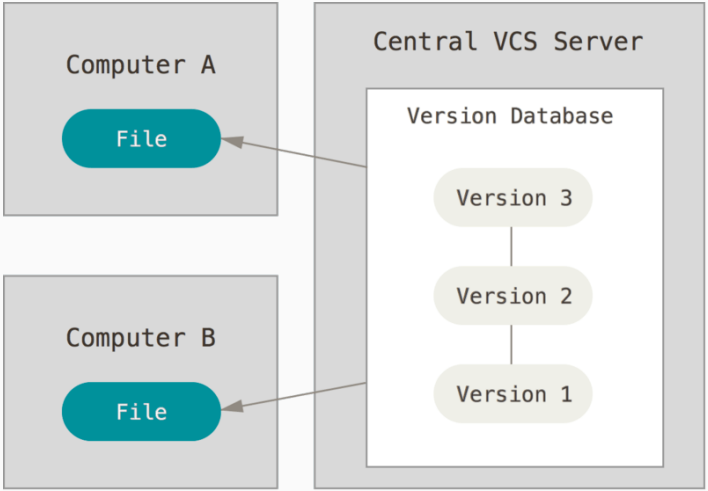

Abbildung 1: Zentrale Versionskontrolle
(`https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F`)

Verteilte Versionskontrolle (I)

- Alle Daten werden lokal verwaltet.
- Optional können die Daten auf beliebig vielen Servern abgelegt werden.
  - Normalfall bei Projekten mit mehreren Mitarbeitenden
- Vorteile:
  - arbeiten offline uneingeschränkt möglich
  - gemeinsamer Austausch über Server trotzdem möglich
  - erhöhte Sicherheit gegen Ausfälle und Datenverlust
- Nachteile:
  - Konflikte können an mehreren Orten auftauchen
  - mehr Konzepte müssen gelernt werden
  - es gibt keine Ausreden mehr (“Ich kann nicht arbeiten, der Server ist down!”) ;-)
- Beispiele: Git, Mercurial, Bazaar, Darcs

Verteilte Versionskontrolle (II)

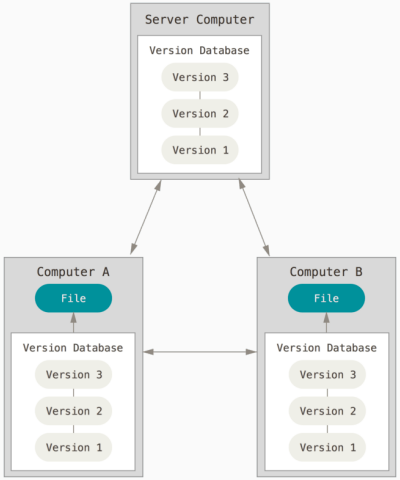

Abbildung 2: Verteilte Versionskontrolle
(`https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F`)

Vorschau: Git

Im Rahmen des Moduls 426 werden wir die dezentrale Versionsverwaltung **Git** verwenden.

Git hat sich als quasi-Standard etabliert; wer etwas anderes als Git verwenden will, braucht gute Gründe.

Prominente Gegenbeispiele:

- OpenBSD (CVS, jedoch mit Git-Mirrors)
- FreeBSD (erst neulich von Subversion auf Git umgestiegen)

Glossar

- **(D)VCS**: (Distributed) Version Control System
  - (verteiltes) Versionskontrollsystem
- **SCM**: Source Code Management
  - Quellcodeverwaltung
- **CVS**: Concurrent Versions System
  - nicht zu verwechseln mit CSV (Comma-Separated Values)

Quellen und Links

- Pro Git (en)
- Pro Git (de)
- Git Reference (en)