Versionsverwaltung

Seminar Softwareentwicklung in der Wissenschaft

Robert Wiesner

Gliederung

- Motivation
- Allgemeines
- Varianten der Versionsverwaltung
- Versionierungssysteme
- Git als Versionierungssystem-Beispiel
- Zusammenfassung
- Diskussion

Ziele der Versionsverwaltung

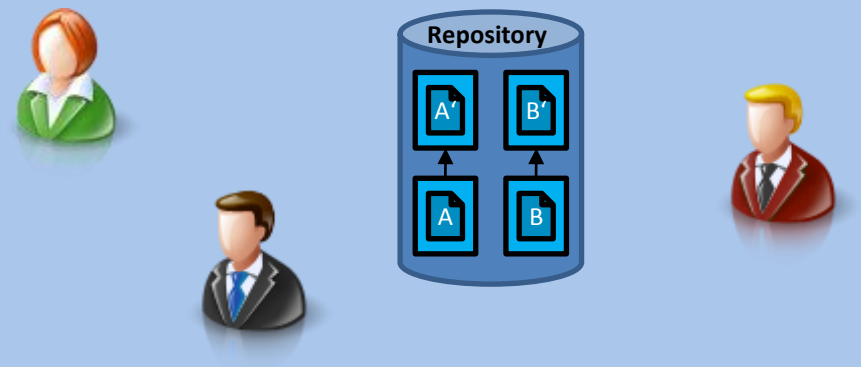
- Inkrementelle Entwicklung von *Versionen*
- Verfügbarkeit der *Historie*
- Parallele Entwicklung von *Branches* (Zweigen)
- *Integration* von Versionen vieler Entwickler
- Nachvollziehbarkeit von Änderungen
- *Koordinierte* Entwicklung
- Effiziente *Fehlerbehandlung* und -suche

Zentrale Begriffe[LOE09]

- Version Control System (VCS)
- Revision Control System (RCS)
- Source Code Manager (SCM)
- Permutationen von
  - „revision“,
  - „version“,
  - „code“,
  - „content“,
  - „control“,
  - „management“,
  - „system“

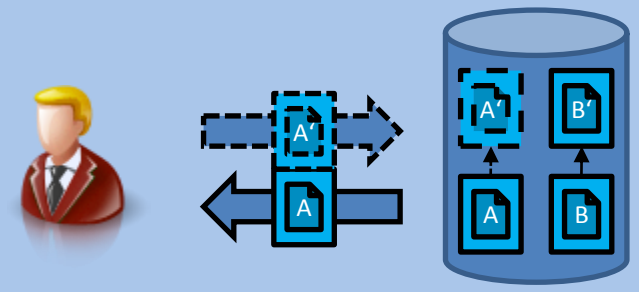
Zentrale Begriffe

- Das *Repository* als Datenbank
- Verwaltet *Versionen* von Daten
- Zeitlich geordnete Versionen als *Historie*
- *Arbeitskopie:* Datenversion in Bearbeitung



Arbeitsweise

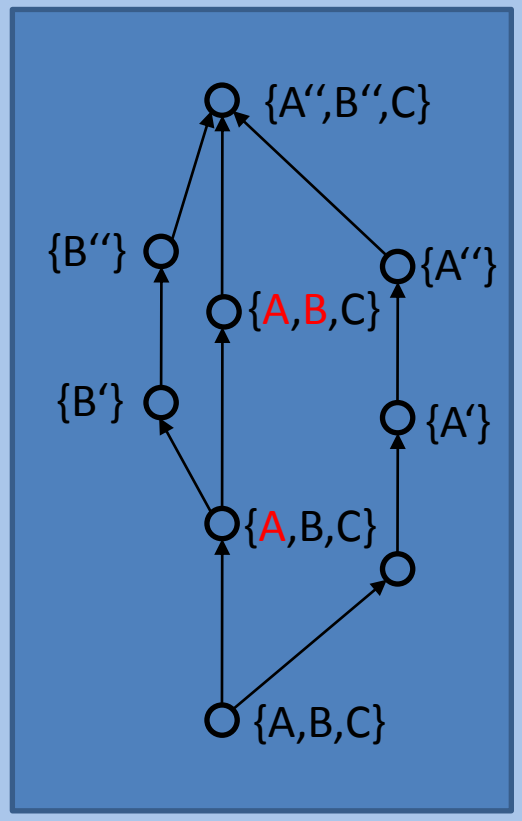
- *Aktualisiere/Initialisiere/Klone* ein Repository
- Ändere Daten in der Arbeitskopie
- Aktualisiere das Repository (*commit*)

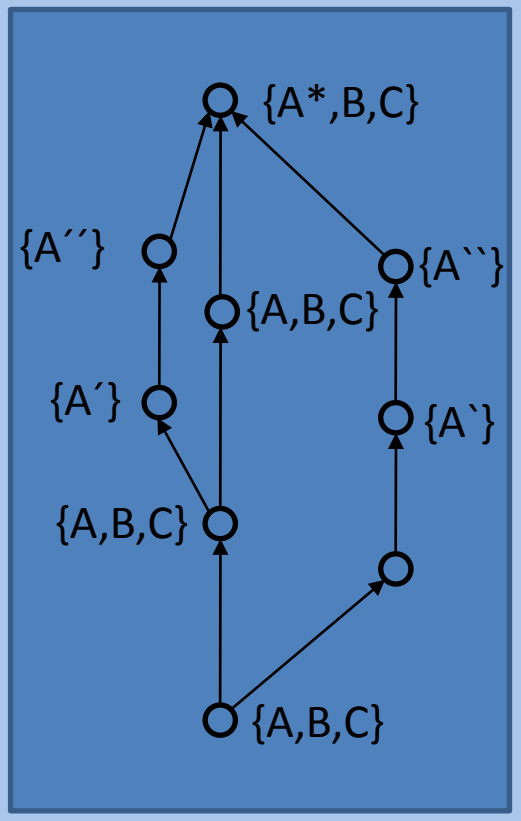


Arbeitsweise

- Lock – Modify – Write
  - Verteilung über logische Einheiten
  - Sperrobjekte nicht nebenläufig bearbeitbar
  - Kein *Zusammenführen* (*Merging*) notwendig
- Copy – Modify – Merge
  - Mehrere Arbeitskopien einer Datenversion
  - Ermöglicht *Verzweigung*
  - Erfordert gegebenenfalls Zusammführen

Lock – Modify – Write / Copy – Modify – Merge





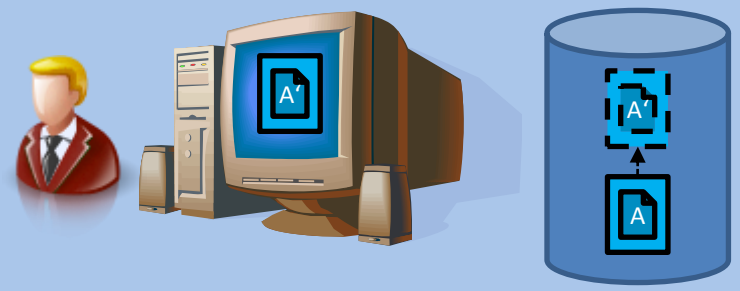
Varianten der Versionsverwaltung

- Lokale Versionsverwaltung
  - Einzelner Rechner
- Zentrale Versionsverwaltung
  - Vernetzte Rechner
- Verteilte Versionsverwaltung
  - Vernetzte Rechner

Lokale Versionsverwaltung

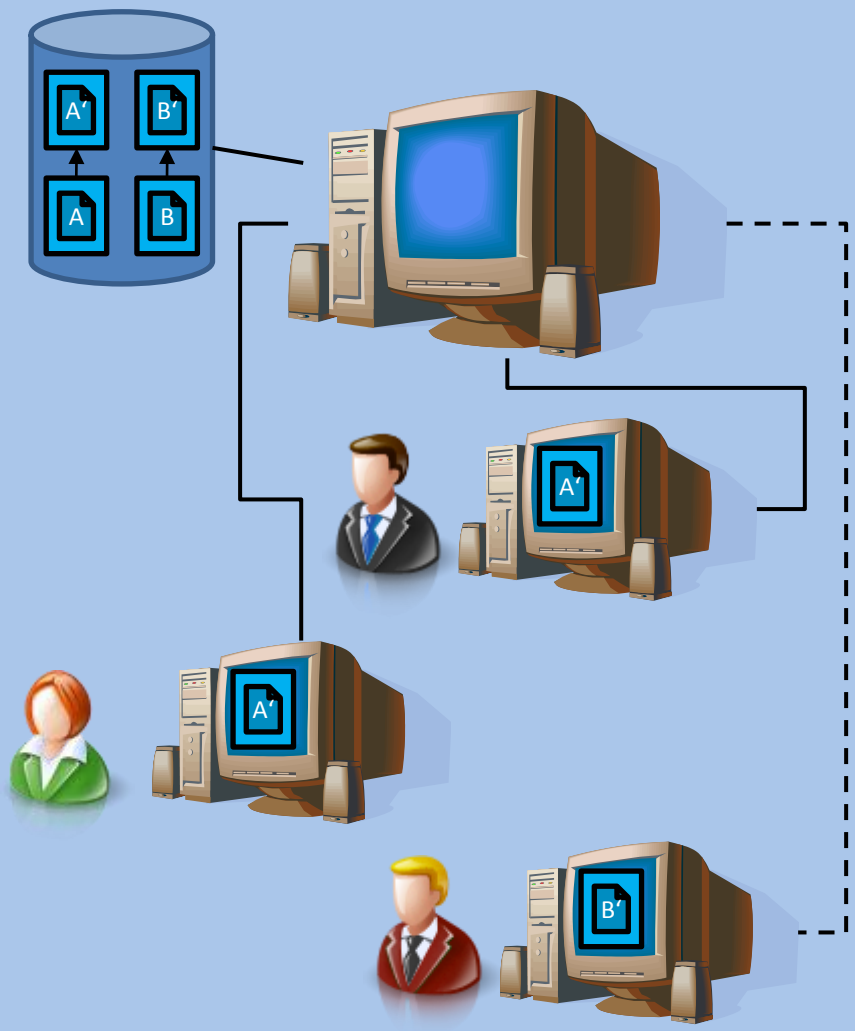
Lokale Versionsverwaltung:

- Dateiorientiert
- Serielle Entwicklung
- In Büroanwendungen
- Für SIW eher uninteressant



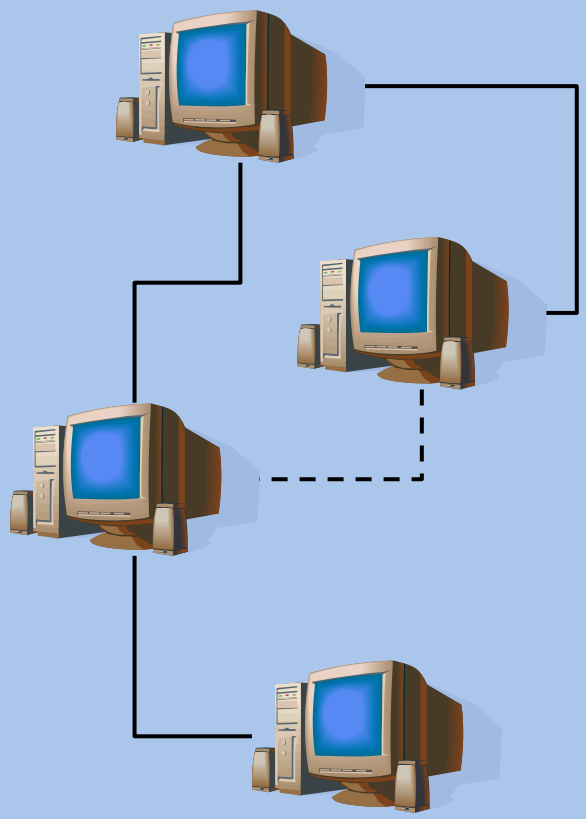
Zentrale Versionsverwaltung

- Client-Server-Architektur
- Ein Repository
- Streng hierarchisch
- Leicht Koordinierbar
- Parallele Entwicklung
- Beliebig viele Entwickler
- Beliebig viele Kopien



Verteilte Versionsverwaltung

- Ein Repository pro Entwickler
- Beliebig viele Entwickler
- Beliebig viele Kopien
- Kein Lock – Modify – Write
- Parallele Entwicklung
- Komplexe Koordination
- Zentrale Repositories möglich

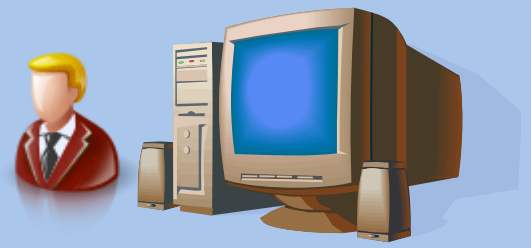


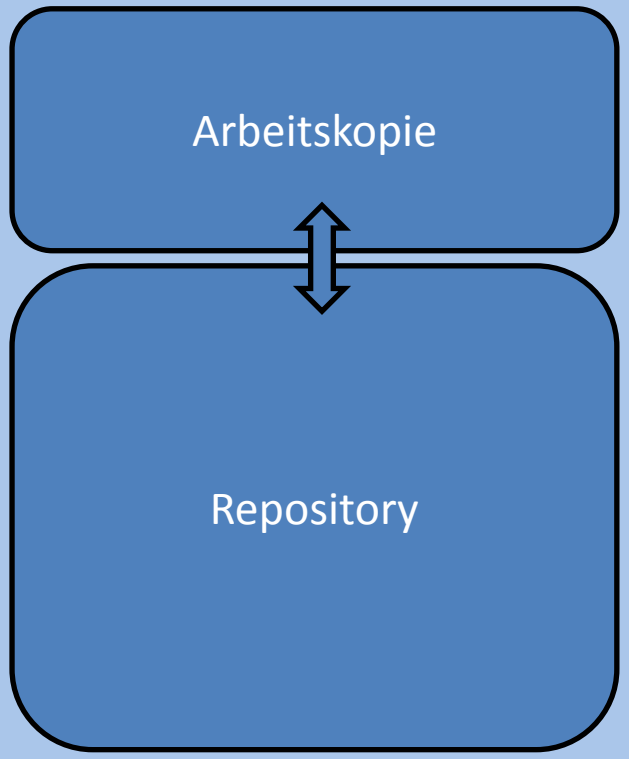
Versionierungssysteme[BAE05]

Lokale Versionierungssyteme	Zentrale Versionierungssyteme	Verteilte Versionierungssysteme
SCCS (1972)	Concurrent Version System (CVS , 1986)	MS Visual SourceSafe (komerziell, 1994)
SourceSafe (=> MS SourceSafe)	Subversion (SVN, 2004)	BitKeeper (kommerziell)
		Monotone
		Mercurial (hg)
		Git (2005)

Git als Versionierungssystem-Beispiel

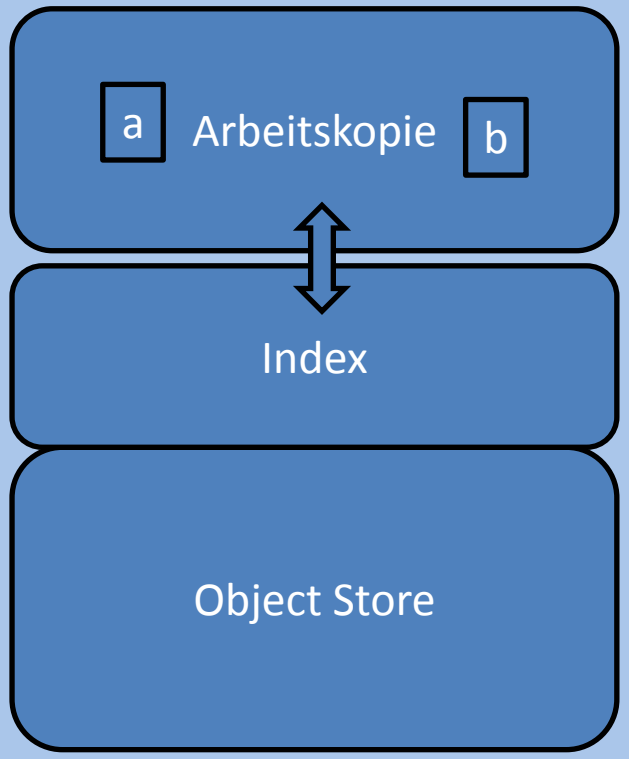
- Eigenschaften von Git
  - Verteiltes Versionierungssystem
  - Copy – Modify – Merge
  - Zentrale Elemente:
    - Arbeitskopie
    - Repository





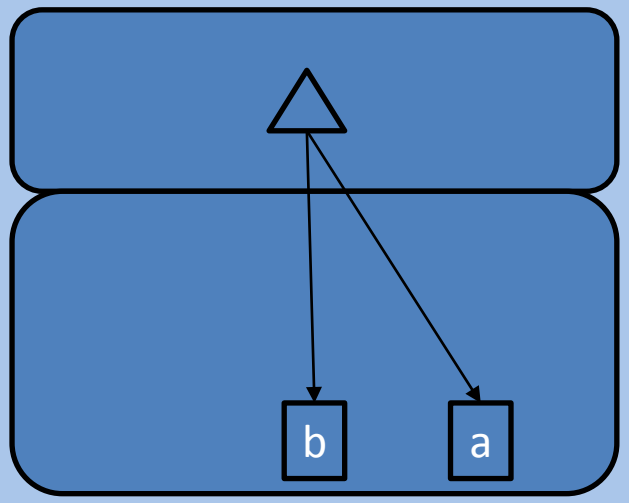
Zentrale Elemente

- Arbeitskopie
  - Eine Datenkopie aus dem Repository
  - Datensatz in Bearbeitung
- Repository
  - Index
  - Object Store



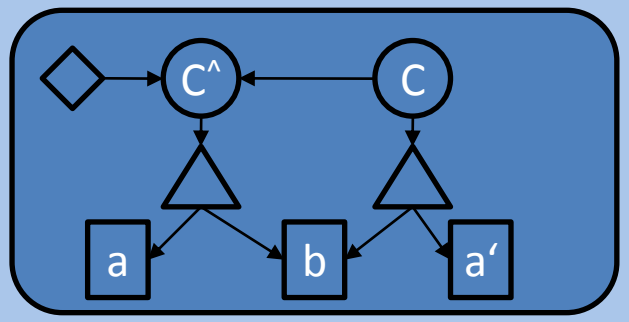
Repository

- Index
  - *Staging Area:*
    - Änderungen der Arbeitskopie im Repository speichern
  - Schnittstelle zwischen Arbeitskopie und Repository
  - Wesentliche Operationen
    - Hinzufügen (*git add <file>)*
    - Löschen (*git rm <file>)*
    - *Vergleichen (git diff <--cached>)*



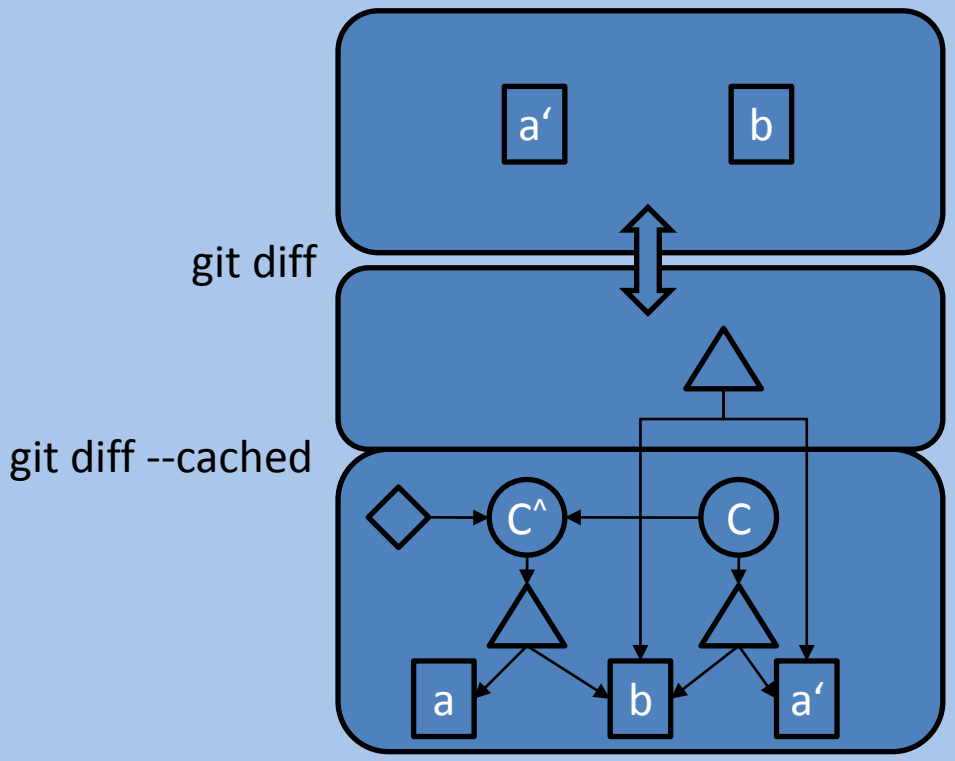
Repository

- Object Store
  - Speichert Daten mit *git commit*
  - Speichert Inhalte, keine Dateien (!= SVN)
  - SHA1-Hash als Bezeichner
  - Typen:
    - Blobs
    - Trees
    - Commits
    - Tags



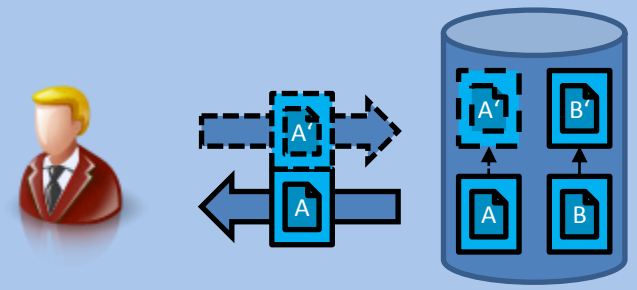
Zwischenfazit

- Verteiltes Versionierungssystem
- Arbeitskopie
- Repository
  - Index
  - Object Store



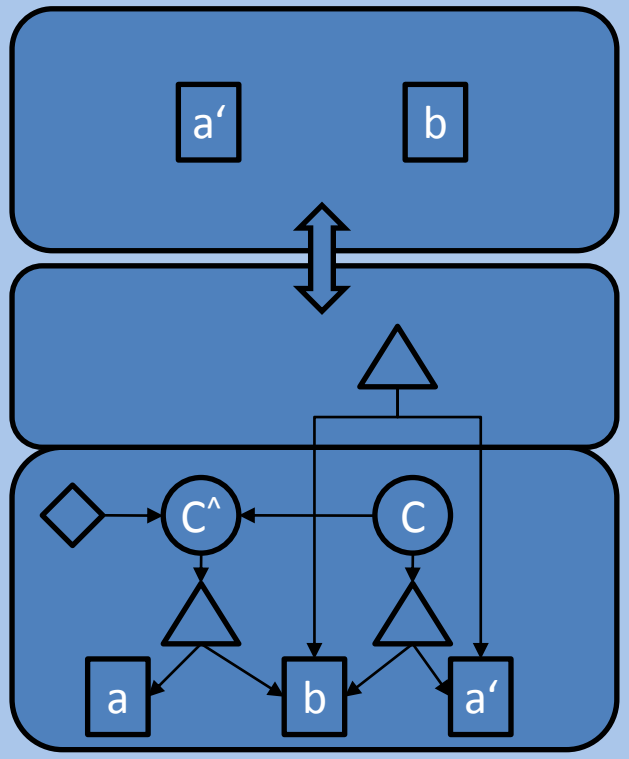
Arbeitsweise - Wiederholung

- *Aktualisiere/Initialisiere/Klone* ein Repository
- Ändere Daten in der Arbeitskopie
- Aktualisiere das Repository (*commit*)
- Hilfe: git help <cmd>



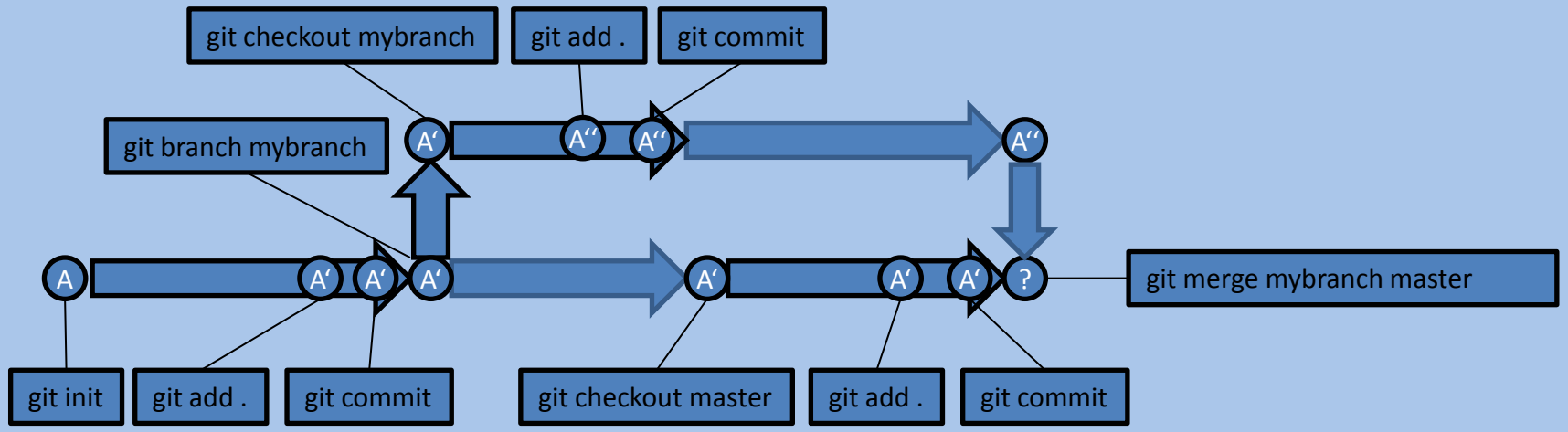
Arbeitsweise - Spezifisch

- Initialisiere/Klone ein Repository
  - *git init [<Ziel>] / git clone <Quelle> [<Ziel>]*
- Arbeitskopie modifizieren
- Index aktualisieren (staging)
  - *git add [<Datei> {<Datei>}| .]*
- Repository aktualisieren
  - *git commit*



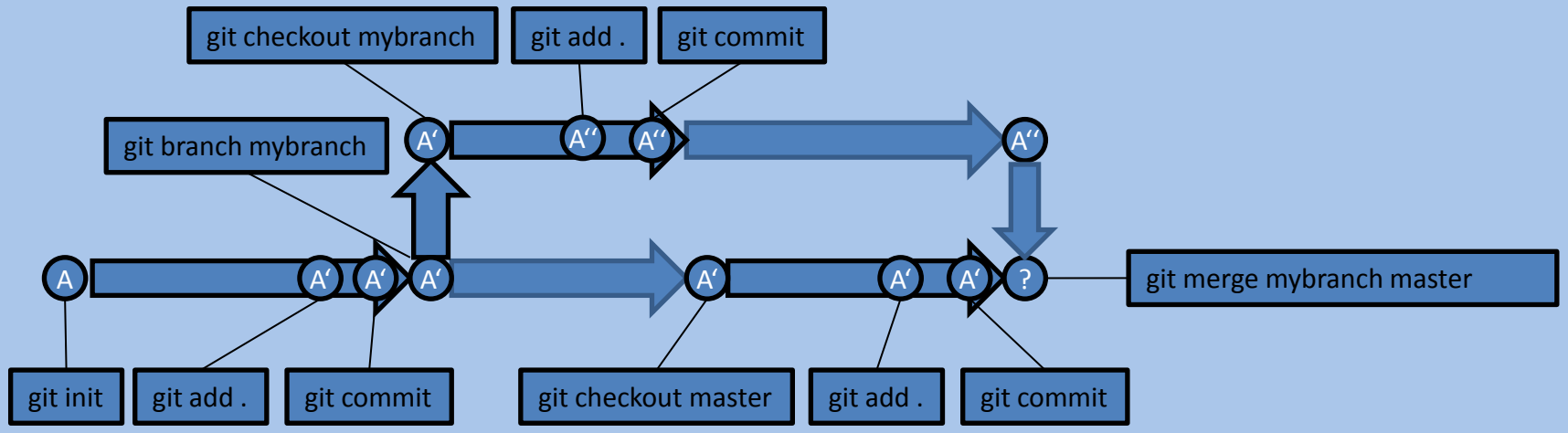
Branching / Merging

- Das Repository als Graph
  - Commits und Branches als Knoten
- Beim Mergen ggf. Konflikte beheben
- Branch nur nach commit wechseln

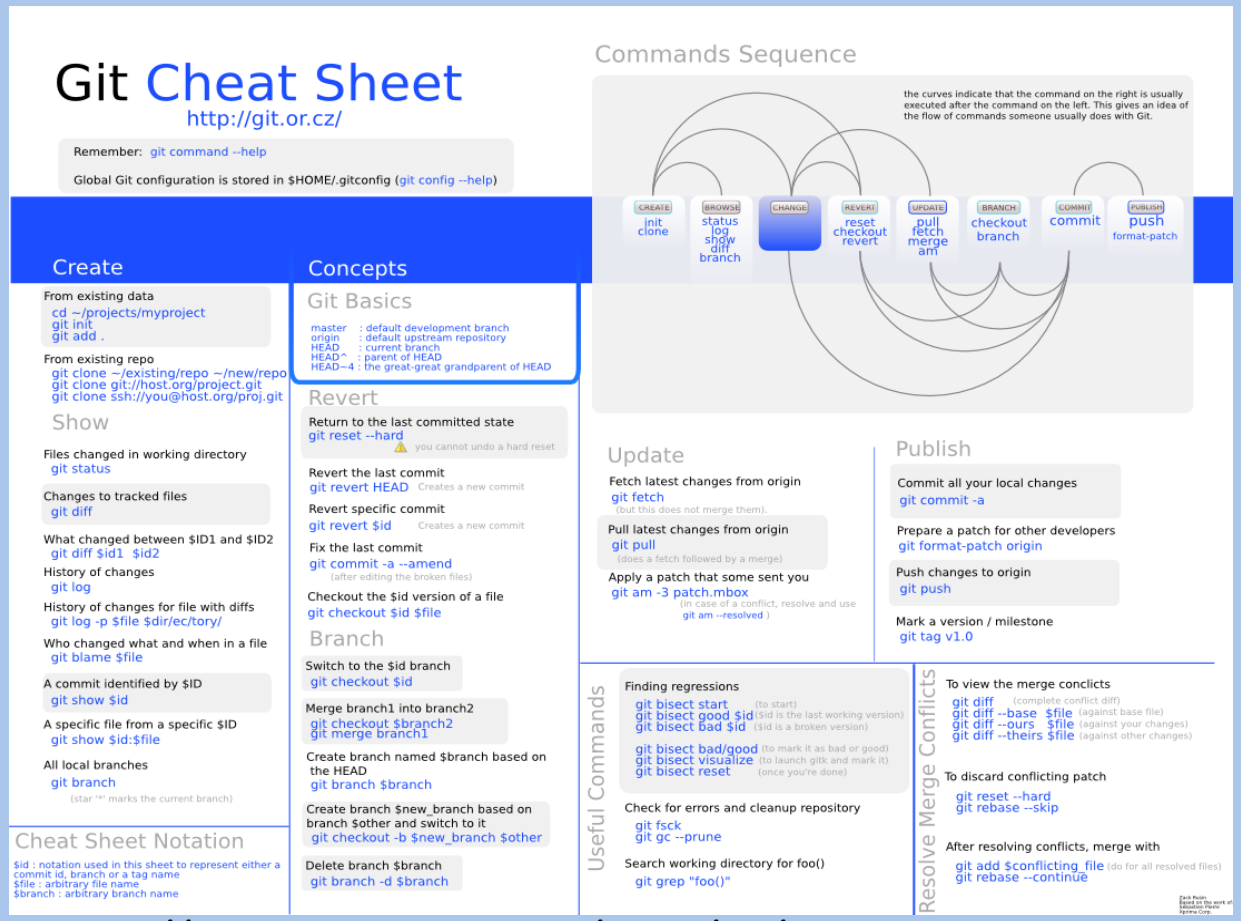


Anwendungsbeispiel

- Live-Anwendung



Hilfe



http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html

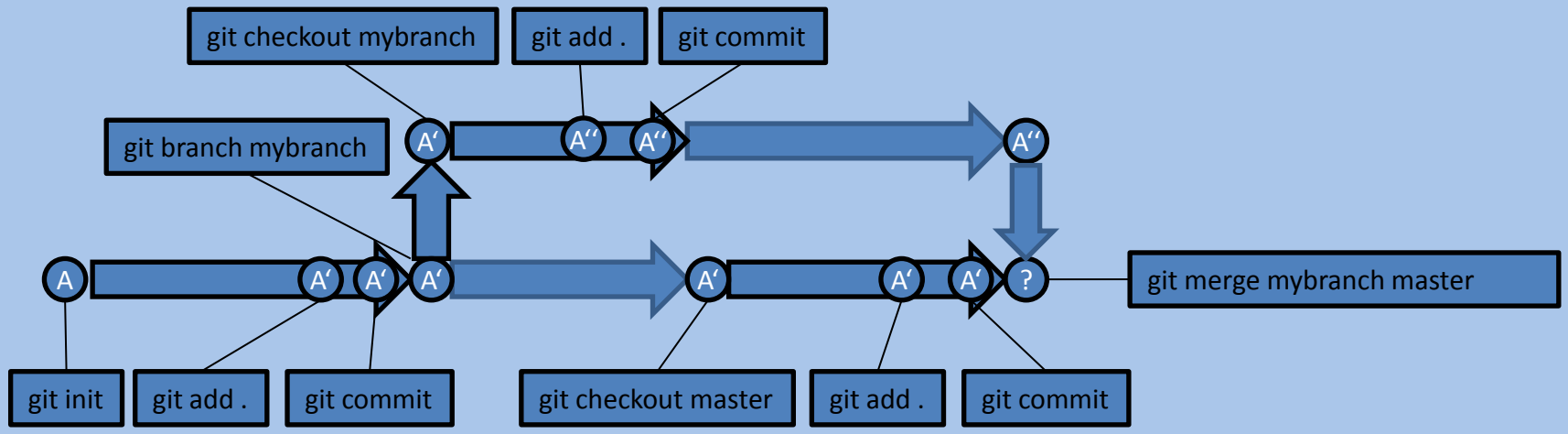
Zusammenfassung

- Zentrale und verteilte Versionierungssysteme
- Git als (gutes) Werkzeug für SIW
- Koordination bleibt wichtig
- Save early – save often
- Erstelle ein Versionierungskonzept…
- …und halte es ein!!!

Literatur

- [LOE09]: Jon Loeliger, 2009, „Version Control with Git“, O‘Reilly
- git-Webseite: http://git-scm.com/
- [BAE05]: Stefan Baerisch, 2005, „Versionskontrollsysteme in der Softwareentwicklung“, Informationszentrum Sozialwissenschaften

Diskussion / Fragen



Optional - Zentrale Versionsverwaltung in Git

HowTo: Zentrale Versionsverwaltung

- Nutze ein Repository ohne Arbeitskopie
  - git init --bare
- Definiere das zentrale Repository
  - git clone <bare> <Ziel> oder
  - git remote add origin <bare>
- Synchronisiere Daten
  - git push <origin [branch]>
  - git pull