# Subversion

von Stefan Arndt, Christian Autermann und Dustin Demuth

4. November 2009

## Inhaltsverzeichnis

# 1 Versionierung

## 1.1 Zweck von Versionierung

Versionierung - eigentlich Versionsverwaltung - beschäftigt sich mit der Erfassung von Änderungen an Dokumenten. Hierfür werden, in dem zu verwaltendem Archiv, für jede Datei Zeitstempel und Nutzerdaten erfasst, welche zu jedem Zeitpunkt wiederhergestellt werden können. Somit wird eine lückenlose Nachvollziehbarkeit von Textänderungen gewährleistet. Versionsverwaltung ist somit eine Art des Variantenmanagements, welches hauptsächlich in der Industrie eingesetzt wird. Dieses Konzept hat sich auf die Softwareentwicklung durchgesetzt.

## 1.2 Geschichtliches

SVN entstand als Ersatz für CVS (Content Versioning System). Jedoch hatte CVS immense Einschränkungen, da es nach den Orten der Änderungen Versionierte und nicht nach der zeitlichen Abfolge. Das brachte die SVN-Entwickler dazu „Schönheitsreperaturen“ an CVS durchzuführen. Dies begann im Februar 2000. Daraufhin verbreitete sich Subversion selbst, nachdem die Entwickler von CVS auf SVN umstellten. Die erste stabile Version erschien im Jahr 2004.

## 1.3 Aufbau der Versionierung

In der Architektur von Subversion wurde das alte Schema P: L-T (Projectarchive: Location-Time) von CVS zu P: T-L umgewandelt. Dies schaffte den Vorteil, dass SVN anschaulicher wurde und somit die Erfahrungsgemäße Realität an Änderungen besser nachstellen konnte. Hierbei bezieht sich das Versionsschema nicht mehr auf einzelne Dateien - wie in CVS - sondern auf das ganze Projekt. Somit lässt sich einfacher eine Konkrete Version beschreiben.

# 2 Wichtige Begriffe

## 2.1 Repository

Unter einem Repository versteht man ein zentrales, auf einem Server lagerndes Archiv, das über die gesamte Versionsgeschichte jeder Datei, die im Repository abgelegt wurde, verfügt.
Da man in der Regel für jedes Projekt ein eigenes Repository benutzt, ist ein Subversion-Server in der Lage mehrere Repositories zu verwalten.
Zum Bearbeiten der versionierten Dateien lädt man sich eine lokale Arbeitskopie aus dem Repository und lädt anschließend die veränderten Dateien in das Repository.

## 2.2 Revision

Im jedem Repository gibt es die sogenannte Revisionsnummer. Beim Anlegen eines Repositories beträgt sie null und wird bei jeder eingereichten Änderung um 1 inkrementiert. Für jede versionierte Datei wird zusätzlich die Revisionsnummer der letzten Bearbeitung gespeichert.
Durch Angabe einer Nummer lässt sich die Version einer Datei eindeutig bestimmen und so auch der Zustand einer Datei oder des gesamten Projektes zu einem bestimmten Zeitpunkt wiederherstellen.
Die aktuellste des Repository nennt man auch *head* und die lokale, noch nicht eingereichte Revision, heißt *base*.

## 2.3 Verzeichnisstruktur

Unabhängig von der Art des Projektes hat sich eine Verzeichnisstruktur etabliert, die mehrere Entwicklungsstränge ermöglicht.
Der Hauptentwicklungszweig lagert in *trunk*, Nebenzweige werden in *branch* abgelegt und *tag* enthält benannte Versionen.
Da in Subversion beim kopieren einer Datei, nicht die Datei kopiert wird, sondern nur ein neuer Verweis in der Datenbank angelegt wird, ist es möglich ohne zusätzlichen Speicherbedarf Dateien zu kopieren und deren Versionsgeschichte zu erhalten. Möchte man nun eine eine benannte Version (bspw. ein Release Candidate o.ä.) erstellen, reicht es mit Hilfe von Subversion das Projekt aus *trunk* nach *tag* zu kopieren und dort umzubennen. Subversion legt dabei nur eine Verknüpfung unter dem neuen Namen mit der Versionsgeschichte der Originaldateien an.
Gleiches geschieht beim Erstellen eines neuen Entwicklungszweiges in *branch*.

# 3 Konflikte

Subversion erlaubt das gleichzeitige Arbeiten mehrerer Personen an der selben Datei. Dies führt zwangsläufig zu Konflikten wie Abbildung 1 zeigt. Subversion bietet hierfür zwei Lösungen an.

## 3.1 Merge

In Abbildung 1 kann B nicht einfach seine Änderungen commiten, da dies alle Änderungen, die A gemacht hat, verwerfen würde. Daher ist B gezwungen die im Konflikt stehenden Dateiversionen zu verschmelzen (zu mergen). Dies kann automatisch (bsplw. wenn sich die Veränderungen nich gegenseitig beeinflussen) oder von Hand geschehen. Um dies zu Vereinfachen gibt es eine Vielzahl Tools die einen synoptischen Vergleich der Dateien ermöglichen und die Unterschiede visualisieren.

## 3.2 Locks

Locks eigenen sich besonders für binäre Dateien, bei denen ein synoptischer Vergleich der beiden in Konflikt stehenden Dateien nicht möglich ist.
Um Konflikte zu Verhindern wird bevor die Datei bearbeitet wird eine Sperre (der sogenannte

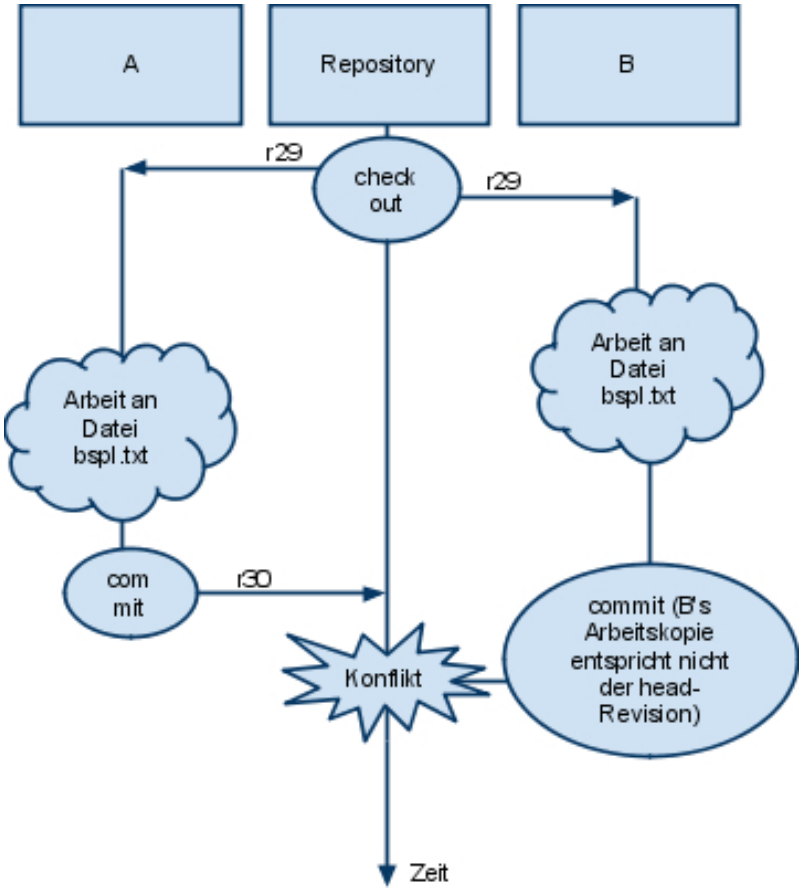


Abbildung 1: Konflikt beim gleichzeitigen Bearbeiten einer Datei (Quelle: eigene Darstellung).

Lock) für die Datei angefordert. Alle anderen Personen können ab diesem Zeitpunkt zwar die Datei lesen und bearbeiten, allerdings nicht commiten. Erst mit dem nächsten Commit des Lock-Inhabers wird die Datei wieder für andere freigegeben.
Da man eine von jemand Anders gesperrte Datei in seiner Arbeitskopie beliebig verändern kann und über ein Lock nur bei einem versuchten Commit informiert wird, empfiehlt es sich vor jedem Bearbeiten den Status einer Binärdatei zu überprüfen um unnötige Arbeit zu verhindern.

# 4 Die Kommandozeile

# 5 Graphical Frontends