Flexible Versionskontrolle mit Subversion (SVN)

Größere Entwicklungsprojekte bestehen mitunter aus tausenden Dateien. Diese werden zudem von mehreren Entwicklern zum Teil parallel bearbeitet. Ohne den Einsatz von speziellen Tools ist es nicht möglich, dabei die Übersicht zu behalten. Mit Subversion (SVN) wird in diesem Artikel das derzeit aktuellste Versionskontrollsystem aus dem Open-Source-Bereich vorgestellt.

Die Entstehung

Versionskontrollsysteme gibt es bereits seit geraumer Zeit, sowohl im kommerziellen als auch im Open-Source-Bereich. Lange Zeit galt im Open-Source-Bereich das Tool CVS (Concurrent Version System) als erste Wahl. In vielen Entwicklungsprojekten stellte sich jedoch heraus, dass das System in einigen Belangen zu unflexibel und der Funktionsumfang nicht immer ausreichend war.

Die Intention für die Entwicklung des Versionskontrollsystems Subversion bestand darin, die Einschränkungen und Unzulänglichkeiten von CVS auszuräumen und dem Anwender so viel Flexibilität wie möglich zur Verfügung zu stellen. Aktuell steht Subversion in der Version 1.5.5 zur Verfügung.

Einsatzgebiete

Grundsätzlich lassen sich alle Arten von Dateien mit SVN verwalten. So kommt Subversion durchaus auch für die Verwaltung von Binärdateien, z. B. Office-Dokumenten oder Bildern, zum Einsatz. Wichtige Funktionalitäten, wie zum Beispiel der Vergleich verschiedener Dateiversionen oder das automatische Zusammenführen von verschiedenen Bearbeitungsständen, funktionieren jedoch nur mit textbasierten Dateien. Für den Dateivergleich von Office-Dateien existieren zwar teilweise externe Tools, die verwendet werden können, as Haupteinsatzgebiet für Subversion stellt jedoch ganz klar die Softwareentwicklung bzw. die Verwaltung von Quellcodedateien dar.

Grundsätzliche Funktionsweise

Die grundsätzliche Funktionsweise eines jeden Versionskontrollsystems ist im Prinzip immer gleich. Die verwalteten Daten liegen an einer zentralen Stelle in einem Repository, auf das die beteiligten Entwickler von ihren lokalen Rechnern aus zugreifen können. Die Bearbeitung einer Datei wird dann lokal durchgeführt und Änderungen oder neu erstellte Dateien werden anschließend in das Repository übertragen, so dass sie den anderen Entwicklern zur Verfügung stehen.

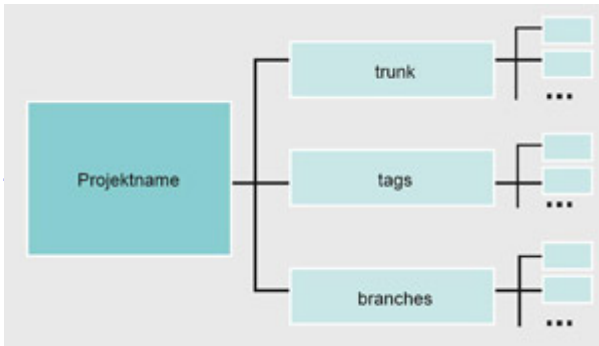

Abb. 1: Aufbau des Repository.
Vergrößern

Das Repository

Das Repository kann auf dem Subversion-Server entweder im Filesystem des Servers oder als BerkleyDB-Datenbank angelegt werden. Beide Alternativen unterstützen Transaktionen, so dass Änderungen immer atomar durchgeführt werden. Eine Netzwerkunterbrechung oder ein ähnliches Problem, das während der Übertragung zwischen Workingcopy und Repository auftreten kann, führt demnach nicht zu einem inkonsistenten Datenbestand, da nur ein Teil der Dateien übertragen wurde.

Obwohl die Struktur des Repositories völlig frei wählbar ist, wird ein Repository in der Regel immer nach dem gleichen Schema aufgebaut (siehe Abbildung 1). Es wird also zwischen der Hauptentwicklungslinie (trunk), den Nebenentwicklungslinien (branches) und gegebenenfalls den Lieferständen (tags) unterschieden. Mehr dazu finden Sie im Abschnitt "Branches und Merges".

Die einzelnen Arbeitsversionen werden innerhalb des Repository mit einer Revisionsnummer gekennzeichnet. Jede Änderung, die per Commit-Befehl aus einer lokalen Workingcopy in das Repository gespielt wurde, führt dazu, dass die Revisionsnummer des gesamten Repository hochgezählt wird. Dabei spielt es keine Rolle, ob eine Datei im Trunk oder in einem Branch geändert wurde. Die Revisionsnummer bezieht sich immer auf das gesamte Repository.

Die Workingcopy

Um die verwalteten Dateien bearbeiten zu können, müssen die Entwickler zunächst eine Workingcopy des Repository-Standes auf ihrem lokalen Rechner erstellen. In dieser können dann die Änderungen vorgenommen werden. Dieser Checkout wird in der Regel nur ein einziges Mal vorgenommen. Anschließend werden nur noch die Unterschiede zwischen der Workingcopy und dem Repository (Commit) und umgekehrt (Update) übertragen.

In jedem Ordner der Workingcopy wird ein verstecktes Verzeichnis mit dem Namen .svn angelegt. Darin befinden sich alle Informationen über die Datei, wie zum Beispiel der Dateipfad innerhalb des Repository, der Dateistatus oder die Revisionsnummer.

Des Weiteren befindet sich innerhalb des Verzeichnisses .svn auch eine Kopie der Dateiversion zum Zeitpunkt des Checkouts bzw. des letzten Updates. Dies ermöglicht es dem SVN-Client zum einen, alle lokal vorgenommenen Änderungen rückgängig zu machen (Revert). Zum anderen kann der Client auch feststellen, ob eine Datei lokal verändert wurde oder nicht. Beides ist möglich, ohne sich mit dem Repository verbinden zu müssen.

Durch die Sicherung der ausgecheckten Version verbraucht die Workingcopy immer ungefähr doppelt so viel Speicherplatz wie der Repository-Stand.

Verschiedene Dateistatus

Eine Datei innerhalb einer Workingcopy kann verschiedene Status haben:

- Normal: Datei, die seit dem letzten Update nicht verändert wurde.
- Modified: Datei, die seit dem letzten Update geändert wurde.

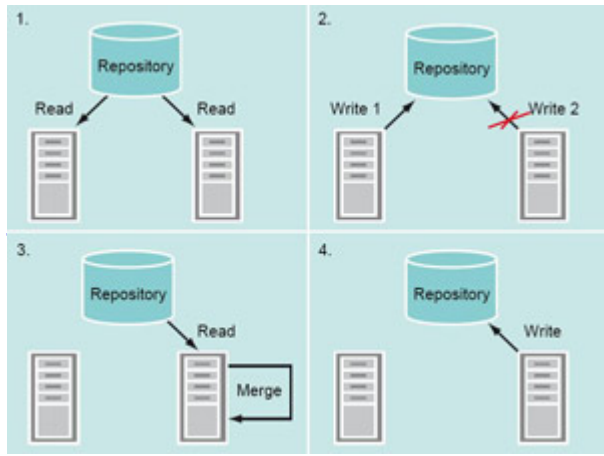

Abb. 2: Copy-Modify-Merge-Prinzip.
Vergrößern

```
Datei svnserve.conf
--------------------
[general]
anon-access = read
auth-access = write
password-db = passwd
authz-db = authz

realm = Repositoryname
[sasl]
# use-sasl = true


Datei users
-------------------
[users]
user1 = Passwort1
user2 = Passwort2


Datei authz
------------------
[/Repository Pfad]
user1 = rw
user2 = rw
* = r
```

Abb. 3: Konfigurationsdateien für den SVN-Zugriff.

- Conflicted: Datei, die sowohl in der lokalen Workingcopy als auch im Repository verändert wurde.
- Deleted bzw. added: Datei, die in der lokalen Workingcopy gelöscht oder hinzugefügt wurde.

Die Änderungen in der Workingcopy werden erst durch ein Commit zum SVN-Server wirksam.

Weitere Status sind "locked", "readonly" und "ignored":

- Locked: Eine Datei kann durch einen Anwender für andere Benutzer gesperrt werden, um eine Bearbeitung zu unterbinden. Sie erhält dann den Status "locked".
- Readonly: Dateien, die manuell mit einem Schreibschutz versehen sind, werden als "readonly" verwaltet.
- Ignored: Auch der Status "ignored" wird manuell gesetzt. Dies ist nützlich für Dateien, die zwar bei einem Checkout aus dem Repository in die lokale Workingcopy übernommen, deren Änderungen jedoch nicht ins Repository zurückgespielt werden sollen. Verwendet wird dieser Mechanismus beispielsweise für lokal unterschiedliche Konfigurationsdateien.

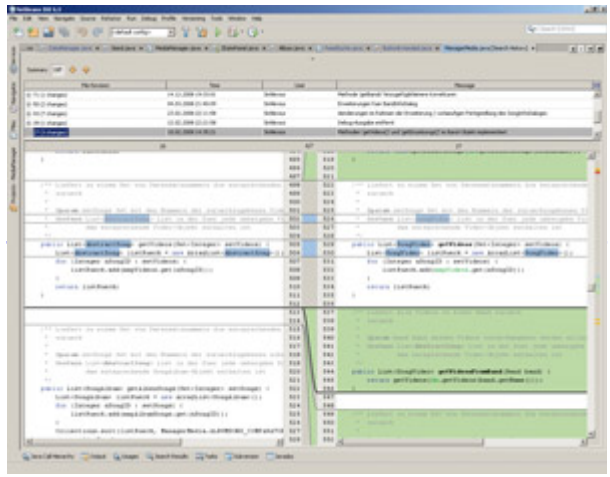

Abb. 4: Vergleich zweier Source-Dateien in NetBeans. Vergrößern

Das Copy-Modify-Merge-Prinzip

Um die parallele Bearbeitung einer Datei zu ermöglichen, greift SVN auf das so genannte Copy-Modify-Merge-Prinzip zurück (siehe Abbildung 2). Im Gegensatz zu vielen anderen Versionskontrollsystemen, wird eine Datei, die sich in Bearbeitung befindet, nicht für die Bearbeitung durch andere Benutzer gesperrt. Das Lesen und Bearbeiten ist also für jeden Nutzer zu jeder Zeit möglich (siehe Schritt 1).

Ein Benutzer kann eine lokal geänderte Datei nicht in das Repository spielen, wenn diese in der Zwischenzeit bereits von einem anderen Anwender geändert wurde (siehe Schritt 2). Die Datei erhält den Status "conflicted".

Er muss zunächst ein Update vornehmen und die beiden geänderten Versionen innerhalb seiner Workingcopy zusammenfügen (siehe Schritt 3). Wenn sich die vorgenommenen Änderungen nicht wiedersprechen, wird dieser Vorgang bei textbasierten Dateien automatisch vorgenommen. Dieser Mechanismus arbeitet zeilenbasiert, d. h. Dateien, in denen Änderungen von unterschiedlichen Benutzern in derselben Zeile vorgenommen wurden, können nicht automatisch bearbeitet werden. In diesem Fall muss der Benutzer die Dateiversionen manuell zusammenfügen. Die meisten SVN-Clients bieten aber auch hier Unterstützung an.

Nachdem die beiden Versionsstände zusammengeführt wurden, kann der Dateistatus auf "modified" gesetzt und die Datei anschießend über ein Commit in das Repository gespielt werden (siehe Schritt 4).

Branches und Merges

Um den Entwicklungszweig (trunk) zu duplizieren, bietet Subversion, wie auch die meisten anderen Versionskontrollsysteme, die Möglichkeit an, einen so genannten Branch (zu Deutsch: Ableger) zu erstellen. Technisch gesehen stellt ein Branch lediglich eine Kopie dar. Die im Branch vorgenommenen Änderungen können in die Hauptentwicklungslinie übernommen werden und umgekehrt. Das Übernehmen der gesamten Änderungen oder Teilen davon in eine andere Entwicklungslinie bezeichnet man als "Merge". Für die Erstellung eines Branches kann es mehrere Gründe geben.

Bug-Fix-Branch

Am häufigsten wird ein so genannter Bug-Fix-Branch angelegt. Dies geschieht immer dann, wenn ein bestimmter Verionsstand erreicht ist, der in den Test- oder Produktionsbetrieb geht. Somit kann in der Hauptentwicklungslinie die

Entwicklung des nächsten Release beginnen und im Branch können Fehler behoben oder Änderungen vorgenommen werden, die während des Test- oder Produktionsbetriebs aufgefallen sind. Diese Änderungen können aus dem Branch in den Trunk übernommen werden. Aus dem Branch kann dann zu jeder Zeit ein Programmstand erzeugt werden, der zwar die Fehlerkorrekturen, nicht jedoch die neuen Funktionen für das nächste Release enthält.

Entwicklungs-Branch

Wenn für eine Entwicklung einer bestimmten Funktionalität oder eines bestimmten Moduls sehr weitreichende Änderungen an der Software vorgenommen werden müssen, kann dies die Arbeit anderer Entwickler beeinträchtigen. In diesem Falle kann es sinnvoll sein, für diese Entwicklung einen eigenen Branch anzulegen. Änderungen aus der Hauptentwicklungslinie würde man regelmäßig in den Branch übernehmen. Wenn die Entwicklung des Moduls abgeschlossen ist, übernimmt man diese dann wieder in den Trunk.

Sonderversions-Branch

Sonderversions-Branch Wenn neben der Standardversion einer Software eine oder mehrere Sonder- oder Spezialversionen existieren, kann es sinnvoll sein, auch hierfür einen separaten Branch zu erstellen. In diesen können dann alle Erweiterungen und Bugfixes aus dem Trunk bzw. anderen Branches übernommen werden. Die spezifischen Sonderfunktionen können jedoch in dem Branch separat verwaltet werden.

Administratives

Der Zugriff auf das Repository kann auf vielerlei Arten und über verschiedene Protokolle realisiert werden.

Der gängigste Weg ist die Verwendung eines Apache Webservers, über den dann auch die Authentifizierung stattfindet. Als Protokoll kommt in diesem Fall http/https zum Einsatz. Der Webserver muss um 2 Moduldateien für die Einbindung von WebDAV und den SVN-Zugriff erweitert werden.

In der Datei httpd.conf des Servers kann dann die Authentifizierungsmethode festgelegt und konfiguriert werden. Die Authentifizierung lässt sich dabei beispielsweise über einen LDAP, eine Datenbank oder die Kombination einer Passwort- und Berechtigungsdatei realisieren. Durch die Verwendung des WebDAV-Protokolls können die Repository-Informationen auch in einem ganz normalen Browser angezeigt werden.

Alternativ kann auch der SVN eigene Server "svnserv" verwendet werden. Dieser kommuniziert über ein eigenes Protokoll namens "svn" mit dem Client. Für die Authentifizierung und Autorisierung werden die Dateien svnserve.conf, users und authz im Verzeichnis conf des Repositories angelegt (siehe Abbildung 3).

In der Datei svnserve.conf wird zunächst festgelegt, welche Berechtigungen ein authentifizierter Benutzer hat (Wert des Parameters auth-access) und welche Rechte ein nicht-authentifizierter Benutzer hat (Wert des Parameters anon-access). Die Werte für die Parameter password-db und auth-db verweisen auf die Dateien, in denen die User-Passwort-Kombinationen (password-db) bzw. die Berechtigungen der User (auth-db) hinterlegt sind. Es können lediglich die Werte r für lesenden und rw für lesenden und schreibenden Zugriff vergeben werden.

SASL

Seit der Version 1.5 unterstützt SVN auch die Verwendung einer SASL-Bibliothek. SASL ist ein Standard, der verschiedene Authentifizierungsmechanismen für unterschiedliche Protokolle zur Verfügung stellt.

Der Einsatz von SASL in Verbindung mit Subversion ermöglicht beispielsweise die Verwendung einer Verschlüsselung bei der Authentifizierung oder die Anbindung eines LDAP, ohne dafür einen Apache Webserver installieren und einrichten zu müssen. Sinnvoll ist der Einsatz also vor allem dann, wenn kein Apache installiert werden kann/soll, oder wenn WebDAV-Kommandos nicht akzeptiert werden.

Verschiedene SVN-Clients

Für alle gängigen Betriebssysteme existiert ein Commandline Client, über die SVN-Befehle direkt an den SVN-Server abgeschickt werden können. Mit einem Commandline Client lassen sich zwar sämtliche Funktionen von SVN nutzen, der Komfort eines Kommandozeilen-Tools ist aber in der Regel nicht sehr hoch. Vor allem der Vergleich von

verschiedenen Dateiversionen lässt sich in einer Kommandozeile nur sehr spartanisch darstellen.

Aus diesem Grunde gibt es eine ganze Reihe von Tools oder grafischen Benutzeroberflächen, die das Arbeiten mit Subversion deutlich erleichtern. Teilweise verwenden diese Tools einen Commandline Client als Grundlage, teilweise beinhalten sie die Funktion auch selbst. Das wahrscheinlich bekannteste Tool für Windows-Umgebungen ist das Programm "Tortoise". Dieses integriert sich direkt in den Windows Explorer und erlaubt das Arbeiten daraus.

Integration in IDEs

Zwar lassen sich mit Subversion prinzipiell alle Arten von Dateien verwalten, der Hauptanwendungszweck ist jedoch sicherlich die Softwareentwicklung und die Verwaltung und Versionierung von Quellcode-Dateien. Ein wichtiger Aspekt bei der Auswahl eines Versionskontrollsystems ist also die Kompatibilität bzw. die Integration in die Entwicklungsumgebung, mit der gearbeitet wird.

Für alle gängigen IDEs wie Netbeans, Eclipse, IntelliJ oder Visual Studio existieren Plugins, die den Zugriff auf ein SVN-Repository direkt aus der Entwicklungsumgebung heraus ermöglichen. Die meisten Plugins bieten neben den Grundfunktionen auch weitere Funktionalitäten an, wie zum Beispiel die Überprüfung von im Repository geänderten Dateien oder den kompletten Abgleich der lokalen Workingcopy mit dem Repository-Stand.

Besonders nützlich ist auch der Vergleich zweier Versionsstände einer Datei. Hier bieten die meisten IDE-Integrationen grafische Übersichten an, in denen geänderte, gelöschte und neu hinzugefügte Dateifragmente farbig unterschiedlich dargestellt werden (siehe Abbildung 4). Dadurch können Unterschiede auf einen Blick erkannt und Änderungen über mehrere Versionen hinweg schnell und einfach nachverfolgt werden.

Fazit

Subversion überzeugt mit durchdachten Konzepten und einer ausgereiften Umsetzung. Nicht zuletzt sind auch die zahlreichen Tools und grafischen Bedienoberflächen sowie die sehr gute Integration in verschiedene Entwicklungsumgebungen ein Grund dafür, dass sich SVN zum meistgenutzten Versionskontrollsystem entwickelt hat.

Subversion lässt kaum Wünsche offen und erleichtert die tägliche Arbeit in großen und kleinen Software-Projekten erheblich.

Alexander Zeller (info@ordix.de).