1.2 Versionsverwaltung

Keine verantwortungsvolle Person würde heutzutage ein Projekt starten ohne eine backup Strategie. Denn Daten sind flüchtig und können leicht verloren gehen, entweder druch eine fehlerhafte Änderung oder einen fatalen Festplattenabsturz. Deshalb sollte jede Arbeit am Projekt archiviert werden. Für Software Projekte ist die typische backup Strategie ein Versionskontrollsystem, sprich das Verfolgen und Verwalten von Überarbeitungen und Änderungen. (vgl. Web2)

Ein System zur Versionsverwaltung (VCS) ermöglicht es Änderungen an einer oder mehreren Dateien zu speichern. Wobei nicht nur die aktuellste Version gespeichert wird. Die Speicherung funktioniert wie folgt. Zunächst wird die Initiale Version einer Datei gespeichert. Wird diese Datei nun verändert, wird ebenfalls diese Änderung, sprich die differenz beider Dateien, gespeichert. Somit ist die Initiale Datei wéiterhin vorhanden und von der Änderung seperat gespeichert. Jeder weitere Änderung wird gleichfalls seperat gespeichert. Dadurch ist es jederzeit möglich auf eine frühere Version zurückzugreifen, falls es beispielsweise Probleme mit der aktuellen Version geben sollte. (vgl. Web3)

Das VCS speichert ein Projekt aus solchen Datein in Form eines Verzeichnisbaumes und sorgt gleichzeitig dafür das dieses Projekt mit dem Repository synchronisiert wird. Das Repository ist der Ablageort des gesamten Projekts.
Umgekehrt ist also das VCS essenziell, damit mit den Dateien in dieser Art gearbeitet werden kann. (vgl. Web3)

Folglich gibt es für verschiedene Anwendungsgebiete unterschiedliche VCS. Grob wird zwischen 3 verschiedenen Arten der Versionsverwltung unterschieden:

-

lokal

Lokale VCS funktionieren nur auf einem Computer. Ausserdem wird bei diesem Verfahren meist nur eine einzelne Datei versioniert und nicht mehrere. Somit ist die Arbeit in einem Team nicht wirklich möglich. Verwendet wird diese Art hauptsächlich in Büroanwendungen, indem die Version des Dokuments in der Datei selbst gespeichert wird. Oder bei technischen Zeichnungen, wo jedoch der Änderungsindex gespeichert wird.

-

zentral

Diese Art der Versionsverwaltung ermöglicht die Arbeit in Teams. Das VCS ist als Client-Server-System aufgebaut.Dder Zugriff erfolgt über das Netzwerk auf ein zentrales Repository, wobei mittels Rechteverwaltung sichergestellt werden kann das nur berechtigte Personen zugriff auf das Repository haben oder Änderungen speichern können. Dieses System wird von vielen kommerziellen Anbietern, beispielsweise Subversion (SVN), unter anderem auch konstenfrei zu Verfügung gestellt.

-

verteilt

Bei dem verteilten VCS gibt es kein zentrales Repository im eigentlichen Sinne. Jeder, der an dem Projekt beteiligt ist hat sein eigenes Repository und kann dieses mit jedem beliebigen anderen Repository vergleichen. Die Versionshistorie ist genauso verteilt wie beim zentralen System, jedoch können Änderungen auch lokal und ohne Verbindung zum Server durchgeführt werden.

Doch eines haben diese 3 Verfahren gemeinsam. Sie speichern ausschließlich die Differenz zwischen den Versionen. Was einiges an Speicherplatz spart, sollten jedoch zwei oder mehrere Beteiligte an der gleichen Datei arbeiten, wird es bei der synchronisierung zu einem Konflikt kommen. Unterschiedlichen Versionen können zunächst parallel existieren, was zum Beispiel bei der Entwicklung von unterschiedlichen Features oder teilfunktionalitäten eines Programmes, von Vorteil ist. Am Ende müssen diese Versionen zusammengefÜhrt werden, was in einigen Fällen manuell durch Benutzer durchgeführt werden muss. (vgl. Web3)

Git beispielsweise ist ein frei zugängliches VCS zur verteilten Versionsverwaltung.
Für die anfertigung dieser Bachelorarbeit wurde angenommen das als VCS GitHub verwendet wird.

GitHub bietet einen Cloud-basierten Git Repository Hosting Service. Im Wesentlichen macht es Einzelpersonen und Teams viel einfacher, Git für die Versionskontrolle und die Zusammenarbeit zu nutzen.

Github verwendet einige Fachtermini, die im folgenden verwendet werden und deshalb kurz erläutert werden. Wird eine Version aus dem Repository in die lokale Kopie übertragen, wird von einem Checkout, von Aus-Checken oder von Aktualisieren gesprochen. Die umgekehrte Übertragung von der lokalen Kopie in das Repository hingegen wird als Check-in, als Einchecken oder als Commit bezeichnet. Wird nun das Repository mit dem 'Remote-Repository' auf dem GitHub-Server synchronisiert, wird das Wort push beziehungsweise Hochladen verwendet. Zudem ist es auch möglich von dem Verzeichnisbaum beliebig viele Abzweigungen bewusst einzubauen, was Branch genannt wird. Diese werden dann wie gesagt nach fertigstellung der Arbeit mit dem Hauptverzeichnis (HEAD oder Master) zusammengeführt, sprich gemerged, wird. Hierbei kann es sich um kommandozeilenorientierte Programme oder um solche mit grafischer Benutzeroberfläche handeln oder auch um ein Plugin innerhalb einer IDE oder eines Editors. (vgl. Web3)