Institut für Physikalische Chemie

Vorlesung Physikalische Chemie V: Programmierkonzepte in der Physikalischen Chemie im Wintersemester 2018/19

Dr. Till Biskup

— Glossar zu Lektion 06: „Versionsverwaltung“ —

Hinweis: Die nachfolgend genannten Begriffe und Definitionen erheben keinen Anspruch auf formale Korrektheit, sondern dienen lediglich dem besseren Verständnis der in der Vorlesung behandelten Themen und sind im jeweiligen Kontext zu sehen. Mehrfache, voneinander abweichende Definitionen in unterschiedlichen Kontexten sind daher möglich. Englische Begriffe werden zwar nach Möglichkeit übersetzt, erscheinen aber ggf. unter ihrem englischen Namen in der Liste. Verweise untereinander sind durch ↑ gekennzeichnet.

Arbeitskopie Lokale Kopie eines Zustandes eines ↑Repositorys, auf der gearbeitet (entwickelt) bzw. die produktiv eingesetzt wird.

Branch Zweig, Abspaltung von einer anderen Version (↑Revision). Branches können parallel weiterentwickelt und später wieder miteinander verbunden (↑merge) werden.

checkout Holen einer Version (↑Revision) aus dem ↑Repository.

commit Übertragen einer Version (↑Revision) in das ↑Repository.

CVS *Concurrent Versions System*, freies, quelloffenes, aber nicht mehr weiterentwickeltes zentrales (↑Server-Client-Konzept) ↑Versionsverwaltungssystem, das sich ursprünglich großer Beliebtheit für die Entwicklung von Software erfreute. Der direkte Nachfolger ist ↑Subversion.

diff Vergleich zweier Versionen (↑Revisionen); ggf. bieten Editoren gleichzeitig den Vergleich und die Möglichkeit der Zusammenführung (↑merge). Es gibt ein gleichnamiges Kommandozeilenprogramm in der Unix-Welt, das genau diese Aufgabe erfüllt.

Entwicklerversion Eine ↑Arbeitskopie der Software, an der aktiv entwickelt wird und die im Gegensatz zu ↑Produktivversionen *nicht* für den produktiven Einsatz gedacht ist. Entwicklerversionen sollten entsprechend (z.B. im Versionsnummernschema) klar gekennzeichnet werden. Ein Problem solcher Entwicklerversionen und gleichzeitig der Grund, warum sie *nie* produktiv für (bleibende) Datenauswertung verwendet werden sollten, ist der, dass die Eineindeutigkeit zwischen Versionsnummer und Zustand der Software nicht gewährleistet werden kann, da die Versionsnummer frühestens beim nächsten ↑commit inkrementiert wird, die Software aber zwischendurch ggf. aktiv verändert und weiterentwickelt.

Feature-Branch ↑Branch zur Entwicklung neuer (größerer) Funktionalität. Besonders hilfreich bei der Entwicklung größerer Projekte mit einer aktiven Nutzerbasis, da so die aufwendigere Entwicklung ungestört stattfinden kann. Ist die Entwicklung abgeschlossen, wird der Feature-Branch mit dem Hauptentwicklungszweig zusammengeführt (↑merge).

Git freies, quelloffenes, verteiltes ↑Versionsverwaltungssystem, ursprünglich binnen Tagen für die Entwicklung des Linux-Kernels von dessen Hauptentwickler Linus Torvalds entwickelt, erfreut es sich mittlerweile großer Be-

liebtheit nicht nur für die Entwicklung freier Software.

Hotfix Kurzfristige Behebung eines (den Entwicklern) bekannt gewordenen kritischen Fehlers in einer ↑Produktivversion der Software. Wird in einer Produktivversion ein Fehler bekannt, der zügig behoben werden soll, ohne auf die nächste Produktivversion zu warten, bietet sich die Behebung in einem zugehörigen kurzlebigen ↑Branch an, der direkt von der letzten Produktivversion abzweigt, den Fehler behebt, und anschließend wieder mit dem Hauptzweig zusammengeführt wird (↑merge). Wichtig ist dabei, die Fehlerbehebung auch in den Entwicklungszweig einfließen zu lassen – und ggf. entsprechende Tests zu entwickeln, die ihn automatisch detektieren können.

mercurial freies, quelloffenes, verteiltes ↑Versionsverwaltungssystem, ähnlich wie ↑Git.

merge Zusammenführen unterschiedlicher Versionen. Oft helfen die Werkzeuge von ↑Versionsverwaltungssystemen dabei, die Unterschiede lassen sich mittels ↑diff anzeigen.

Produktivversion Im Gegensatz zur ↑Entwicklerversion eine für den Produktiveinsatz geeignete ↑Revision eines Programms. Sie zeichnet sich u.a. durch eine eineindeutige Versionsnummer aus (d.h. die Versionsnummer entspricht genau einem Zustand des Programms und umgekehrt ein Zustand des Programms genau einer Versionsnummer).

pull Holen der Historie aus einem anderen (ggf. entfernten) ↑Repository; relevant bei verteilten Versionsverwaltungssystemen.

push Übertragen der Historie in ein anderes (ggf. entferntes) ↑Repository; relevant bei verteilten Versionsverwaltungssystemen.

Repository Versionsdatenbank, (zentraler) Speicherort der versionierten Dateien

Revision einzelner der ↑Versionsverwaltung bekannter Zustand eines ↑Repositorys

Routine 1. eingeschliffener, ggf. strukturierter Arbeitsablauf; 2. (Software) Funktion: der Programmiersprache unter einem festen Namen bekannte Liste von Anweisungen, die eine bestimmte Aufgabe erfüllt.

Server-Client-Konzept arbeitsteiliger Aufbau, der ursprünglich in der Frühzeit der Computer entwickelt wurde, wo Rechenkraft noch teuer und auf wenige, zentrale Rechner beschränkt war. Mittlerweile verallgemeinert und nicht notwendiger Weise auf unterschiedliche Hardware bezogen: Server- und Client-Prozess (d.h. Programm) können auf derselben Hardware laufen. Im Kontext von ↑Versionsverwaltungssystemen geht damit oft eine Aufteilung auf unterschiedliche Hardware und insbesondere die Abhängigkeit des Klienten (*client*) von einer zentralen Infrastruktur (*server*) einher. Abhilfe schaffen hier dezentrale ↑Versionsverwaltungssysteme wie ↑Git oder ↑mercurial.

Subversion *Apache Subversion*, freies, quelloffenes, zentrales (↑Server-Client-Konzept) ↑Versionsverwaltungssystem und Nachfolger von ↑CVS.

SVN siehe ↑Subversion

Tag „Etikett“, frei wählbarer Bezeichner für eine ↑Revision. Ein Beispiel wäre eine nach außen kommunizierte Versionsnummer eines Projektes.

VCS siehe ↑Versionsverwaltung

Version siehe ↑Revision

Versionsdatenbank siehe ↑Repository

Versionsverwaltung engl. *version control system*, VCS; Software zur Verwaltung unterschiedlicher Versionen von Dateien und Programmen, die den Zugriff auf beliebige ältere als Versionen (↑Revision) gespeicherte Zustände ermöglicht. Gleichzeitig ein wichtiges Werkzeug für die Softwareentwicklung und wesentlicher Aspekt einer Projektinfrastruktur.

Zweig siehe ↑Branch