Versionskontrollsysteme in der Softwareentwicklung

Baerisch, Stefan

Veröffentlichungsversion / Published Version
Arbeitspapier / working paper

GESIS

IZ InformationsZentrum Sozialwissenschaften

IZ-Arbeitsbericht Nr. 36

Versionskontrollsysteme in der Softwareentwicklung

Stefan Baerisch

Juni 2005

InformationsZentrum
Sozialwissenschaften

Lennéstraße 30
D-53113 Bonn
Tel.: 0228/2281-0
Fax.: 0228/2281-120
E-Mail: iz@bonn.iz-soz.de
http://www.gesis.org

ISSN:	1431-6943
Herausgeber:	Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI)
Druck u. Vertrieb:	Informationszentrum Sozialwissenschaften, Bonn Printed in Germany

Das IZ ist Mitglied der Gesellschaft Sozialwissenschaftlicher Infrastruktureinrichtungen e.V. (GESIS). Die GESIS ist Mitglied der Leibniz-Gemeinschaft.

Inhalt

Zusammenfassung

Bei der Entwicklung von komplexen Softwaresystemen spielt die Erfassung und mögliche Rücknahme von Änderungen an bearbeitetem Quellcode und Dokumenten eine entscheidende Rolle. Dies gilt im Besonderen bei der Koordination mehrerer Entwickler über einen längeren Zeitraum. So genannte Versionskontrollsysteme dienen der Erfüllung dieser Anforderungen, indem sie es ermöglichen, Änderungen an Dateien zu erfassen und zu verwalten. In diesem Arbeitsbericht werden die Eigenschaften verschiedener Versionskontrollsysteme und ihre Nutzung dargestellt. Einen Schwerpunkt bildet die Anwendung des Versionskontrollsystems CVS durch die Abteilung FuE des Informationszentrums Sozialwissenschaften (IZ) und jene Versionskontrollsysteme, die geeignet scheinen, dieses System in Zukunft zu ersetzen.

1 Einleitung

1.1 Thema und Inhalt der Untersuchung

Versionskontrollsysteme sind heute eine der am meisten verbreiteten Gruppen von Werkzeugen bei der Entwicklung von Software. Solche Systeme erlauben den Umgang mit den unterschiedlichen Revisionen einer Datei und unterstützen insbesondere Gruppen von Entwicklern bei der Koordination ihrer Arbeit an einem gemeinsamen Bestand von Dateien. In diesem Arbeitsbericht soll untersucht werden, welche Eigenschaften unterschiedliche Versionskontrollsysteme, insbesondere bei der Anwendung durch Gruppen, auszeichnen. Anhand einer Reihe von Kriterien werden der potentielle Nutzen und mögliche Schwierigkeiten dargestellt, die sich bei einem Wechsel des verwendeten Versionskontrollsystems ergeben. Dies geschieht vor dem praktischen Hintergrund einer möglichen Migration vom durch die Abteilung FuE verwendeten System CVS zu einer Software mit erweitertem Funktionsumfang.

In Kapitel 1 werden die häufigsten Arbeitsschritte im Umgang mit einem Versionskontrollsystem beschrieben, weiterhin wird das Vokabular eingeführt, das zur Diskussion von Versionskontrollsystemen benötigt wird.

Kapitel 2 stellt am Beispiel der Abteilung FuE des IZ eine Einsatzumgebung für Versionskontrollsysteme im Detail dar. Es wird erläutert, unter welchen Bedingungen Versionskontrollsysteme verwendet werden und welche spezifischen Anforderungen sich aus diesen Einsatzbedingungen ergeben. Der Fokus liegt dabei auf den Erfahrungen, die aus der Verwendung der Software CVS resultieren, und die durch Interviews mit den MitarbeiterInnen der Abteilung ermittelt wurden. Aufbauend auf den Ergebnissen dieses Kapitels werden An-

forderungen dargestellt, die ein zukünftiges Versionskontrollsystem erfüllen soll.

Im Kapitel 3 werden unterschiedliche Versionskontrollsysteme behandelt, die entweder als Alternative für das aktuell verwendete CVS in Frage kommen oder aufgrund ihrer Architektur von besonderem Interesse sind. Neben einer Übersicht der technischen Eigenschaften wird in diesem Kapitel das Versionskontrollsystem Subversion auf seine Eignung bezüglich der in Kapitel 2 dargestellten Anforderungen untersucht.

Anschließend werden im Kapitel 4 die Maßnahmen angesprochen, mittels derer eine eventuelle Migration vom System CVS zu einem Nachfolgesystem vorbereitet und durchgeführt werden können. Einen Schwerpunkt stellt die Überführung der jeweiligen Inhalte des Repositories dar, also des jeweils durch das Versionskontrollsystem verwalteten Datenbestandes. Ein besonderer Fokus wird auf Methoden zur Erstellung und zur Erhaltung der so genannten Historie gelegt. Hierbei handelt es sich um Informationen, die es gestatten, Veränderungen in Form unterschiedlicher Revisionen nachzuvollziehen, eine Veränderung also einem Entwickler oder einem Zeitpunkt zuzuordnen oder mit anderen Revisionen in Beziehung zu setzen.

In Kapitel 5 wird ein Fazit über die bei der Erstellung dieses Arbeitsberichts gewonnenen Erfahrungen gezogen. Basierend auf diesen Erfahrungen wird das geplante weitere Vorgehen beim Einsatz von Versionskontrollsystemen durch die Abteilung FuE des IZ dargestellt.

1.2 Aufgaben von Versionskontrollsystemen

Softwareentwicklung besteht heute in der Regel im Rahmen der Implementierungsphase (siehe Sommerville 2004) in der Erstellung und Bearbeitung von so genanntem Quellcode, von Menschen lesbarem Text in Form von ASCII Dateien, aus denen durch die Entwicklungsplattform ausführbare Programme generiert werden. Hierbei besteht ein Projekt aus einer großen Anzahl solcher Quellcode-Dateien, die über einen längeren Zeitraum erstellt und bearbeitet werden.

Um Veränderungen an Dateien nachvollziehen zu können, werden Versionskontrollsysteme eingesetzt. Diese Programme gestatten den Zugriff auf die unterschiedlichen Revisionen einer Datei und erlauben es somit, Veränderungen zu revidieren oder einem Zeitpunkt oder einer Person zuzuordnen. Während Versionskontrollsysteme in erster Linie in der Softwareentwicklung verwendet werden, bieten sie sich prinzipiell für alle Arten von Textdateien an. Denkbar sind hier die Konfigurationsdateien eines Servers ebenso wie Textdokumente. Insbesondere wenn die Bearbeitung dieser Dateien durch

mehrere Person erfolgt, erweisen sich Versionskontrollsysteme als überaus nützlich.

Dem Benutzer eines solchen Versionskontrollsystems fällt hierbei in der Regel lediglich die Aufgabe zu, festzulegen, wann der aktuelle Zustand einer Datei als neue Version betrachtet werden soll. Ein solches Einstellen kann in zeitlichen Intervallen, wie am Ende jedes Arbeitstags, geschehen. Ebenso denkbar ist aber auch ein Einstellen der Veränderungen aus sachlichen Überlegungen, wenn beispielsweise umfangreiche Veränderungen vorgenommen werden sollen. In der Regel unterstützt das Versionskontrollsystem die Vergabe von Kommentaren zu den jeweiligen Veränderungen. Dies gestattet es auch lange nach einer Veränderung festzustellen, mit welcher Intention, wann und von wem diese vorgenommen wurde.

Versionskontrollsysteme bieten einem Benutzer also vereinfacht gesagt zwei Dienste an:

- Eine Dokumentationsfunktion, die es ermöglicht, Veränderungen und die jeweiligen Gründe für diese Veränderungen mittels einer Historie und Kommentaren über einen beliebigen Zeitraum nachzuvollziehen. Mittels dieser Hilfestellung kann oft festgestellt werden, wann eventuell Fehler in einen Programmtext eingeflossen sind.
- Eine Wiederherstellungsfunktion, mittels derer es möglich ist, unerwünschte Veränderungen zu revidieren. Dies kann nötig sein, wenn Veränderungen nicht vorhersehbare Konsequenzen hatten und rückwirkend isoliert und beseitigt werden sollen. Auch ist es möglich, dass bei der Bearbeitung der Datei selbst Fehler aufgetreten sind, z.B. durch versehentliches Löschen oder fehlerhaftes Editieren.[1]

Die oben beschriebenen Dienste werden dabei von heutigen Systemen nicht nur für einzelne Dateien angeboten, wie dies bei früheren Versionskontrollsystemen der Fall war[2], sondern auch für Projekte aus mehreren Dateien. So lässt sich auch ein komplexes Projekt in seiner Entwicklung einheitlich über einen Zeitraum nachvollziehen. Die Menge aller Dateien der Projekte die durch ein Versionskontrollsystem verwaltet werden und deren Informationen zur Versionierung werden hierbei als Repository bezeichnet.

In der Praxis kommt es häufig vor, dass ein Projekt nicht durch eine Person, sondern durch eine Arbeitsgruppe bearbeitet wird. Hier stellt sich das Prob-

---

[1] Während diese Sicherungsfunktion Parallelen zur Datensicherung aufweist, muss darauf verwiesen werden, dass ein Versionskontrollsystem sich in Anwendung und Architektur stark von Datensicherungssystemen unterscheidet und diese keinesfalls ersetzen kann.

[2] Siehe z.B. RCS unter http://www.gnu.org/software/rcs/rcs.html

lem, wie jene Veränderungen, die durch die einzelnen Teilnehmer eines Projektes an den einzelnen Dokumenten des Projektes vorgenommen wurden, zu koordinieren sind. Ziel ist es hier, einerseits allen Benutzern zu gestatten, die für sie relevanten Bestandteile des Projektes zu bearbeiten, andererseits aber Konflikte beim Zugriff auf Dateien zu vermeiden. Ein Konflikt kann unter anderem auftreten, wenn zwei Personen innerhalb eines sich überschneidenden Zeitraums an einer Datei arbeiten, sich aber den Änderungen des anderen nicht bewusst sind. Unter diesen Umständen würde ein Abspeichern eines lokal bearbeiteten Dokumentes alle Veränderungen überschreiben, die seit dem Öffnen dieser Datei von einer anderen Person an der zentral zugänglichen, ursprünglichen Datei angestellt wurden. Wird hingegen ein Versionskontrollsystem verwendet, so bietet dieses die Möglichkeit, alle Zugriffe auf den Datenbestand eines Projektes durch die bearbeiteten Personen zu koordinieren. Dies kann auf unterschiedliche Weisen geschehen.

- Durch das generelle Ausschließen von potentiellen Konfliktfällen.
- Durch den Versuch, Konfliktfälle automatisch zu beheben.
- Durch die Unterstützung bei der Auflösung von Konflikten durch die Benutzer.

Allen diesen Ansätzen ist gemeinsam, dass irrtümlich und unbewusst hervorgerufene Datenverluste weitestgehend vermieden werden. Ein weiteres Einsatzgebiet von Versionskontrollsystemen ist speziell in der Softwareentwicklung die Behandlung von so genannten Branches, also Verzweigungen. (siehe hierzu auch Abbildung 1) Von einer Verzweigung wird in diesem Zusammenhang gesprochen, wenn sich die Entwicklungslinie einer Software gleich einer Astgabel aufspaltet, also zwei Versionen der Software nebeneinander weiterentwickelt werden. Denkbar ist beispielsweise ein Zweig, in den nur noch dringend notwendige Fehlerkorrekturen eingespielt werden, während in einen anderen Entwicklungszweig alle umfangreichen Veränderungen und Erweiterungen einfließen. Ohne ein Versionskontrollsystem würde hier üblicherweise von den Beteiligten eine Kopie des Projektes angefertigt werden. Diese Kopie würde im Weiteren getrennt vom Original weiterbearbeitet werden.

Den Umgang mit Branches unterstützt ein Versionskontrollsystem, indem es erlaubt, gezielt Veränderungen bestimmten Zweigen zuzuordnen und auch aus einem Zweig Veränderungen in einen anderen Zweig zu übernehmen. So ist es möglich, mit Hilfe von Versionskontrollsystemen zwei über einen längeren Zeitraum getrennt fortgeführte Entwicklungslinien wieder zusammenzuführen oder Veränderungen aus einem Ast gezielt in einen anderen Ast zu übernehmen.

Die aktuell verfügbaren Versionskontrollsysteme unterscheiden sich mitunter stark in Bezug auf Architektur, Ausgereiftheit, Verbreitung und den Annahmen über die Nutzung des Systems. Ihnen allen liegen aber die weitgehend gleichen Arbeitsschritte zugrunde. Diese Arbeitsschritte sollen nun anhand der grundlegenden Aufgaben von Versionskontrollsystemen dargestellt werden; hierbei wird auch notwendige Terminologie eingeführt.

1.3 Verwendung von Versionskontrollsystemen

Dieser Unterabschnitt verdeutlicht anhand einiger Szenarien die wesentlichen Aktionen bei der Verwendung von Versionskontrollsystemen.

1.3.1 Einstellen und Entnehmen von Daten mittels Versionskontrollsystemen

Bei der Nutzung eines Versionskontrollsystems wird zwischen mindestens zwei unterschiedlichen Datenbeständen unterschieden: Die aktuelle Arbeitskopie des Benutzers und das Repository zur Bereitstellung aller Revisionen, (siehe hierzu auch Abbildung 1) Bei der Arbeitskopie handelt es sich um alle Projektdateien in einer Revision mit den Veränderungen des Benutzers und ergänzt um Arbeitsinformationen des Versionskontrollsystems.

Diesen Dateien gegenüber steht das Repository;. dieses umfasst alle Dateien des Projekts in allen in das Versionskontrollsystem eingestellten Revisionen. Stellvertretend diesen beiden Datenbeständen stehen als Softwarekomponenten die beiden Bestandteile des Versionskontrollsystems, Server und Client. Der Server ist für die Verwaltung des Repositories verantwortlich und kooperiert mit einer oder mehreren Instanzen des Clients, dessen Aufgabe darin besteht, die lokalen Arbeitskopien mit dem Server zu koordinieren und dem Anwender die Funktionen des Versionskontrollsystems zur Verfügung zu stellen.

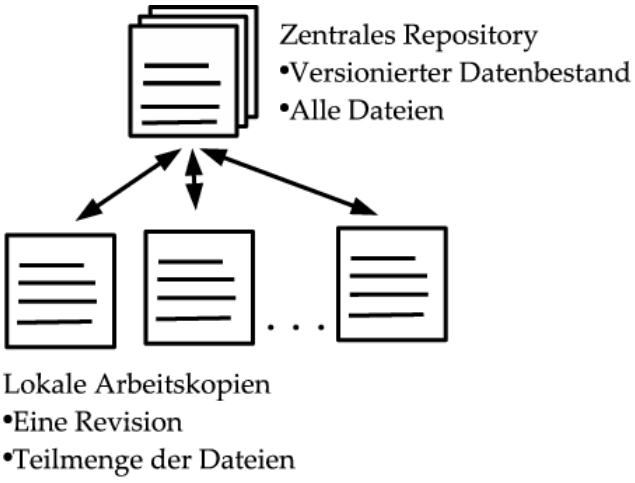

Abbildung 1: Repository und Arbeitskopien

Dabei muss es sich beim Server nicht notwendigerweise um eigenständige Software handeln, verteilte Versionskontrollsysteme verschmelzen Client und Server. Sollen Dokumente der Kontrolle des Versionskontrollsystems übergeben werden, muss als erster Schritt ein Projekt mit allen relevanten Dateien beim Versionskontrollsystem angelegt werden. Bei der Erstellung eines Projektes wird unter anderem festgelegt, wo das Repository des Versionskontrollsystems angelegt werden soll; weiterhin werden für das Projekt gültige Einstellungen, beispielsweise Zugriffsrechte, bestimmt. Nachdem ein Projekt erstellt wurde, werden in einem weiteren Schritt, dem so genannten Import, die zu verwaltenden Dateien eingestellt. Es wird also vereinfacht ausgedrückt eine Kopie der Projektdateien im Repository erstellt. Soll nun mit den in das Projekt eingestellten Dateien gearbeitet werden, ist ein so genannter Checkout notwendig. Hierbei wird eine Kopie der Projektdateien aus dem Repository entnommen. Es wird also eine Arbeitskopie für den Anwender erstellt. Bei einem solchen Checkout können prinzipiell beliebige Revisionen der Dateien des Projektes aus dem Repository entnommen werden, aber meist wird die aktuellste Revision verwendet. Ein Benutzer kann nun mit der lokalen Kopie der Dateien arbeiten. Wenn die angestellten Veränderungen als neue Revision aufgefasst werden sollen, ist ein entsprechender, als Commit bezeichneter Aufruf des Versionskontrollsystems notwendig. Bei diesem Commit werden durch das Versionskontrollsystem alle Unterschiede erfasst, die der Benutzer an seiner lokalen Kopie seit dem Checkout vorgenommen hat. Alle hier behandelten Versionskontrollsysteme bieten die Möglichkeit, zu einem bestehenden Repository neue Dateien hinzuzufügen. Dabei wird nicht von einem Import, sondern von einem Add gesprochen. Ansonsten werden keine Unterschiede zwischen einer beim Import einstellten und einer später hinzugefügten Datei gemacht. Wird eine Revision des Repositories abgefragt, die vor dem Add datiert, sind entsprechende Dateien nicht Teil des Checkouts.

Zusammenfassung:

- Um für die Dateien eines Projektes ein Versionskontrollsystem nutzen zu können, muss dieses in das Repository eingestellt werden; dieser Vorgang wird als Import bezeichnet.
- Der Begriff Checkout bezeichnet den Vorgang, eine komplette Kopie eines Projekts zur lokalen Bearbeitung aus dem Versionskontrollsystem zu entnehmen.
- Als Commit wird die Übergabe von neuen Änderungen an der lokalen Kopie an das Versionskontrollsystem bezeichnet.
- Ein Add ist das Hinzufügen neuer Dateien in ein bestehendes Repository.

1.3.2 Nutzung eines Versionskontrollsystems durch mehrere Anwender

Werden die Dateien eines Projektes unter Versionskontrolle durch mehrere Personen zur gleichen Zeit bearbeitet, verwendet jeder Anwender eine lokale Kopie. Diese kann vom Repository und von den lokalen Kopien der anderen Anwender abweichen. Möchte ein Benutzer seinen Bestand von Dateien mit dem Repository synchronisieren, so führt er ein so genanntes Update durch.

Bei einem Update wird wie bei einem Commit ein Vergleich zwischen lokaler Kopie und Repository durchgeführt, jedoch hier mit dem Ziel, die Arbeitskopie auf den aktuellen Stand im Repository zu aktualisieren. Die aktuellste Version im Repository wird dabei als Head bezeichnet, die ursprüngliche Arbeitskopie, auf der alle Arbeiten des Benutzers stattfanden, als Base. Hat ein anderer Benutzer zwischenzeitlich ein Commit durchgeführt; entspricht also die Head Version nicht mehr der Base Version, werden die Unterschiede zwischen Head und Base Version an den Nutzer übertragen. Während bei einem Commit das Repository auf den Stand der Arbeitskopie aktualisiert wird, werden also beim Update neuere Änderungen aus dem Repository in die Arbeitskopie übertragen.

Zusammenfassung:

- Jeder Benutzer erhält eine eigene Kopie der Dateien des Projektes zur Bearbeitung.
- Ein Update bezeichnet das Aktualisieren der lokalen Kopie durch zwischenzeitlich von anderen Benutzern in das Repository eingestellte Änderungen.
- Die aktuellste Revision im Repository wird als Head bezeichnet.
- Die Revision, die ein Benutzer zuletzt aus dem Repository entnommen hat wird als Base bezeichnet.

1.3.3 Auflösung von Konflikten bei der Bearbeitung von Dateien

Die vorangegangene Schilderung setzt voraus, dass keine beteiligten Benutzer zu einem gegebenen Zeitpunkt mit der gleichen Datei arbeiten, dies ist jedoch kein Regelfall. Wenn mehrere Benutzer ein Projekt gemeinsam bearbeiten, ergibt sich hieraus das Risiko von Konflikten. Als Konflikt wird in diesem Zusammenhang ein Zustand bezeichnet, in dem sich die Änderungen, die unterschiedliche Nutzer im gleichen Zeitraum an einer Datei vorgenommen haben, widersprechen und ohne Intervention des Versionskontrollsystems eine Änderung durch die jeweils nachfolgende überschrieben worden wäre. Der folgende Vorgang stellt ein einfaches Beispiel für einen Konflikt dar:

Zwei Benutzer entnehmen mittels Checkout die gleiche Datei aus dem Repository.

- Beide Benutzer verändern die jeweilige lokale Version der Datei, der erste Benutzer übergibt seine Änderungen ins Repository.
- Der zweite Benutzer übergibt seine Version der Datei ins Repository. Das Versionskontrollsystem greift ein, da ohne Eingriff die späteren Änderungen die vorhergehenden überschreiben würden.

Der einfachste Ansatz zur Vermeidung einer solchen Situation besteht darin, jede Datei für einen bestimmten Zeitraum einem Benutzer zuzuordnen, der sie exklusiv bearbeiten kann, und alleine Schreibrechte auf diese Datei im Repository hat. Erst wenn dieser Benutzer seine Bearbeitung beendet und einen Commit durchgeführt hat, erhalten andere Benutzer Gelegenheit, die Datei in ihrer nun vorliegenden Version zu verändern. Dieses Vorgehen hat sich in der Praxis als zu restriktiv herausgestellt.[3] Überwiegend wird daher durch Versionskontrollsysteme versucht, die unterschiedlichen Änderungen verschiedener Nutzer an einer Datei automatisch zu integrieren oder einen menschlichen Benutzer bei dieser Integration zu unterstützen.

Zur Verdeutlichung soll nun ein einfacher Konfliktfall und seine Auflösung dargestellt werden: Zwei Benutzer haben zeitgleich innerhalb einer Datei denselben Bereich bearbeitet. Nachdem der erste Benutzer nun seine Änderungen bereits eingestellt hat, führt der zweite Benutzer ebenfalls ein Commit durch. Dieses Commit wird vom Versionskontrollsystem zurückgewiesen, da es Änderungen im Repository gibt, die noch nicht in die zum Commit anstehende Arbeitskopie eingeflossen sind.

Würde dieses zweite Commit zugelassen, gingen die zuerst eingestellten Änderungen verloren, da immer eine komplette Datei ins Repository eingestellt oder daraus entnommen wird. Vor einem Commit wird daher durch das Versionskontrollsystem ein Update erzwungen; der lokale Datenbestand muss also mit dem Zustand im Repository abgeglichen werden. Fanden die Änderungen an unterschiedlichen Punkten der Datei statt und kam es zu keinerlei Überschneidungen, so werden die Änderungen beider Benutzer meist automatisch zusammengefasst und ein Commit, an dessen Ende alle Änderungen gemeinsam im Repository stehen, kann stattfinden. Eine weitere Beteiligung des Anwenders ist hier nicht notwendig.

---

[3] Eine Ausnahme stellen Dateien dar, bei denen eine Integration mehrerer konkurrierender Änderungen nicht möglich ist, ein Beispiel sind hier Binärdateien, z.B. von Office Programmen.

Kam es jedoch zu sich überschneidenden Veränderungen innerhalb einer Datei, so ist ein solches automatisches Vorgehen nicht möglich und ein Konfliktfall ist aufgetreten. In diesem Fall muss ein Benutzer entscheiden, wie die widersprüchlichen Veränderungen zu integrieren sind oder welcher Veränderung der Vorrang zu geben ist. Dies wird durch das Versionskontrollsystem unterstützt indem die gegensätzlichen Änderungen in die zu bearbeitende Datei eingespielt und gekennzeichnet werden. Dem Benutzer obliegt es in diesem Fall, die widersprüchlichen Änderungen zu integrieren. Ist dies geschehen kann die Datei mit einem Commit eingestellt werden.

Zusammenfassung:

- Konflikte können entstehen, wenn die gleichen Dateien im gleichen Zeitraum durch mehrere Personen bearbeitet werden.
- Liegen potentielle Konflikte vor, so müssen diese mit einem Update vor einem Commit beseitigt werden; hierdurch werden die Änderungen aus dem Repository in den lokalen Datenbestand aufgenommen.
- Die Integration von lokalen Änderungen in das Repository kann automatisiert erfolgen, wenn die Bearbeitungen in unterschiedlichen Bereichen einer Datei stattfanden, ist dies nicht der Fall, muss ein Benutzer entscheiden, welche Änderungen übernommen und welche verworfen werden sollen.

1.3.4 Parallele Entwicklung verschiedener Zweige eines Projekts

In den bisherigen Szenarien wurde stets davon ausgegangen, dass es bei der Bearbeitung des Datenbestandes unter der Kontrolle des Repositories nur einen Zweig des Datenbestandes unter aktiver Entwicklung gibt, in den alle Änderungen einfließen. Dabei blieben Aufspaltungen in der Entwicklung, so genannte Branches, unberücksichtigt. Einen Branch anzulegen ist dann sinnvoll, wenn die Bearbeitung der Dateien des Projektes für einen bestimmten Zeitraum in getrennten Linien parallel nebeneinander fortgeführt werden soll (eine Darstellung der zeitgleichen Existenz einer Datei in mehreren Branches bietet die Abbildung 2). Dieser Fall kann auftreten, wenn das bearbeitete Projekt veröffentlicht werden soll, gleichzeitig aber neue Bestandteile hinzugefügt werden. Es liegen nun zwei Äste des Projektes vor, die sich weitgehend getrennt voneinander weiterentwickeln. Während in die zur Veröffentlichung bestimmte Version in erster Linie nur Fehlerkorrekturen einfließen, werden weitergehende Neuerungen in den anderen Ast eingefügt.[4]

[4] Prinzipiell wäre es denkbar, jeden der Äste als eigenes Projekt aufzufassen und isoliert zu bearbeiten, wobei die unterschiedlichen Versionen beider Projekte in keinem explizi-

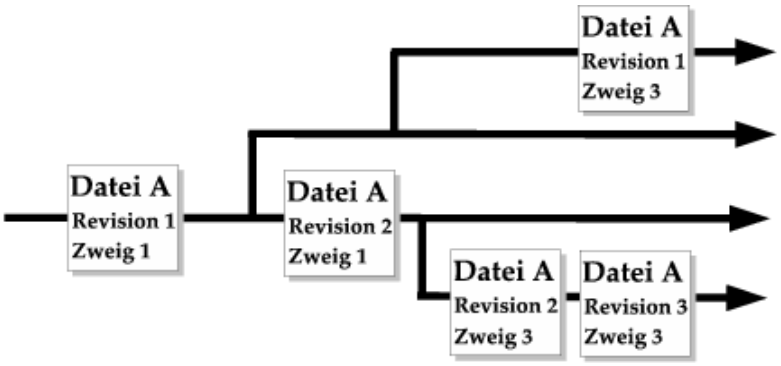

Abbildung 2: Verwendung von Branches

Die explizite Unterstützung der Erstellung und Pflege von Ästen, also das Branching, ist daher Bestandteil aller verbreiteten Versionskontrollsysteme. Um einen Ast anzulegen, legt ein Benutzer dabei eine Version im Versionskontrollsystem als Ursprung des Astes fest. Wie dies geschieht, und wie der Ast angesprochen werden kann, unterscheidet sich zwischen unterschiedlichen Versionskontrollsystemen stark. Es ist nun möglich, einen neuen Branch zu beginnen und ohne Konflikte mit anderen Ästen am eigenen Ast zu arbeiten.

Aus der Sicht der Benutzer handelt es sich bei den Ästen um allein stehende Projekte[5]. Sollen Äste wieder vereinigt werden, oder sollen Veränderungen aus einem Ast in einen anderen Ast einfließen, so werden alle Veränderungen in den zu vereinigenden Ästen auf Vereinbarkeit verglichen; sich widersprechende Veränderungen werden als Konflikte behandelt. Das Zusammenführen von Veränderungen aus unterschiedlichen Ästen wird dabei nicht als Update, sondern als Merge bezeichnet, da hier nicht eine Aktualisierung stattfindet, sondern das Verschmelzen eigenständig geführter Entwicklungslinien.

Zusammenfassung:

- Als Branching wird das Abspalten einer parallelen Entwicklungslinie bezeichnet.
- Vor der Erstellung eines Branches wird der Ausgangspunkt der Veränderung markiert, eine solche Markierung wird als Tag bezeichnet.
- Die neue entstandene Entwicklungslinie wird dabei als Branch bezeichnet.

---

ten Verhältnis stehen. Ein solches Vorgehen erschwert es allerdings, Veränderungen aus einem Ast in den anderen Ast zu übernehmen.

[5] Die vorliegende Darstellung von Branches und Tags orientiert sich an Subversion und CVS, bei anderen Versionskontrollsystemen kann das Vorgehen stark abweichen.

- Versionskontrollsysteme unterstützen das Erstellen, den Austausch zwischen und die Wiedervereinigung von Branches.
- Die Methodik der Unterstützung und die Darstellung von Branches variiert stark zwischen den Versionsverwaltungssystemen.

Die vier in diesem Unterabschnitt dargestellten Szenarien sind allen betrachteten Versionskontrollsystemen gemeinsam. In den Details der Anwendung und in der Architektur gibt es jedoch eine Reihe von Unterschieden mit erheblichem Einfluss auf die Nutzbarkeit eines Versionskontrollsystems in einem gegebenen Arbeitsumfeld. Vor diesem Hintergrund wird im folgenden Abschnitt anhand der Abteilung FuE des IZ ein Arbeitsumfeld für Versionskontrollsysteme exemplarisch vorgestellt.

2 Verwendung von Versionskontrollsystemen am IZ

In diesem Kapitel wird die Verwendung von Versionskontrollsystemen durch die Abteilung FuE des IZ vorgestellt. Es wird auf die Ist-Situation eingegangen, die Verwendung der Versionskontrollsysteme CVS und in geringerem Maß des Visual Age Repositories. Hierbei soll festgestellt werden, welche Funktionen eines Versionskontrollsystems in welchem Umfang in der Abteilung FuE genutzt werden. Insbesondere sollen die Aussagen der Benutzer zum verwendeten System erfasst werden, schwerpunktmäßig in Bezug auf vermisste oder fehlerhafte Funktionalität von CVS und auf das Nutzungsverhalten im Umgang mit CVS.

2.1 Softwareprojekte der Abteilung Forschung und Entwicklung des IZ

Die Abteilung FuE des IZ beschäftigt sich an den Standorten Bonn und Berlin mit der Untersuchung Informationswissenschaftlicher Themenstellungen und mit der Durchführung von Projekten. Forschungs- und Arbeitsgebiete der Abteilung sind im Einzelnen:

- Informationssysteme und Digitale Bibliotheken,
- Heterogenitätsbehandlung,
- Softwareergonomie,
- Werkzeuge zur Datenbearbeitung und -weiterverarbeitung,
- Evaluation von IT-Produkten.

In der Abteilung sind zurzeit 12 Personen beschäftigt, die jeweils selbstständig oder in einem Team in der Projektarbeit arbeiten. Ein Schwerpunkt bei der Softwareentwicklung durch die Abteilung FuE liegt auf der Arbeit mit Java und Web Technologien. Projekte werden durch kleinere Arbeitsgruppen über einen Zeitraum von durchschnittlich 2 Jahren bearbeitet, wobei nach der eigentlichen Entwicklungsphase noch eine mehrjährige Nutzung und die Pflege der Projekte folgen.

Um einen Einblick in die Arbeit der Abteilung zu ermöglichen, sollen nun einige Projekte kurz vorgestellt werden. Nach der Vorstellung der Projekte wird im folgenden Unterabschnitt näher auf die Details der Softwareentwicklung eingegangen.

Informationsverbund Pädagogik – Sozialwissenschaften – Psychologie:

Dieses Projekt[6] stellt eine Internetplattform zur Verfügung, durch die die Versorgung von Wissenschaftlern und Privatpersonen mit Volltexten aus den Bereichen Pädagogik, Sozialwissenschaften und Psychologie ermöglicht werden soll. Einem interessierten Benutzer soll es ermöglicht werden, mit Hilfe einer über das Internet zentral angebotenen Plattform Texte zu den jeweiligen Fachbereichen mittels einer eigens angebotenen Suchfunktionalität zu finden und mit einem Pay-per-View Verfahren oder im Rahmen eines Abonnements zu beziehen. Die unter www.infoconnex.de erreichbare Website fasst für den Benutzer transparent Ressourcen der unterschiedlichen Anbieter und Fachgebiete zusammen.

DBClear:

Datenbankbasierte Clearinghouses sind Inhalt des Projektes DBClear[7]. Ziel ist es, vor dem Hintergrund der Vielzahl der im World Wide Web verfügbaren Inhalte die Orientierung von an Fachinformationen interessierten Benutzern zu erleichtern. Dies wird durch DBClear geleistet, indem die Pflege und Erstellung von unter fachlichen Kriterien erstellten Linksammlungen unterstützt wird. Schwerpunkt von DBClear ist die Unterstützung und teilweise Automatisierung von arbeitsteiligen Abläufen bei der Pflege eines Clearinghouses und die Anpassungsfähigkeit der Darstellung an die jeweiligen Erfordernisse des Anbieters eines solchen Clearinghouses.

---

6 http://www.gesis.org/Forschung/Informationstechnologie/volltextserver.htm
7 http://www.gesis.org/Forschung/Informationstechnologie/DBClear.htm

ISSP Data Wizard:

Im Rahmen dieses Projektes[8] wurde eine Software gleichen Namens entwickelt, die zur Unterstützung der im Rahmen des ISSP[9] Programms ausgeführten Untersuchungen dient. In Zusammenarbeit mit dem Zentralarchiv für Empirische Sozialforschung (ZA) ist ein Werkzeug entwickelt worden, das bei der Auswertung von empirischen Umfragen anfallende Arbeitsschritte durch Prüfung und teilweise Automatisierung unterstützt und für den Benutzer effizienter und weniger fehlerträchtig gestaltet.

Virtuelle Fachbibliothek Sozialwissenschaften:

Dieses auch abkürzend als ViBSoz bezeichnete Projekt[10] hat das Ziel, eine integrierte Plattform bereitzustellen, die bei der Suche nach sozialwissenschaftlicher Literatur verwendet werden kann. Motivation für das Projekt ist die Vielzahl möglicher Informationsquellen, die von einem Sozialwissenschaftler bei der Recherche verwendet werden können und sich in Inhalten und Methoden des Zugriffs unterscheiden. Das Projekt ViBSoz stellt für diese unterschiedlichen Sacherschließungssysteme eine einheitliche Benutzerschnittstelle für Präsentation und Suchanfragen sowie Transferkomponenten zur Verfügung, welche Suchanfragen auf die jeweils lokalen Systeme umsetzt.

Cross-Language Evaluation Forum:

Das Projekt CLEF[11] wird in Kooperation mit dem DELOS Network of Excellence on Digital Libraries betrieben und hat den Status eines eigenständigen EU-Projektes. Das Projekt stellt eine Testumgebung für mehrsprachige Retrievalsysteme zur Verfügung. Hierbei sollen Anfragen in einer beliebigen europäischen Sprache erfolgreich auf eine Menge von Dokumenten angewendet werden können, die ebenfalls in einer beliebigen europäischen Sprache vorliegen.

2.2 Verwendung von Versionskontrollsystemen in FuE

Nachdem im vorangehenden Abschnitt die Abteilung FuE des IZ und einige aktuelle Projekte vorgestellt wurden, soll im Anschluss vertiefend auf die Softwareentwicklung mit Hilfe von Versionskontrollsystemen eingegangen werden.

---

8 http://www.gesis.org/Forschung/Informationstechnologie/ISSPWizard.htm
9 http://www.issp.org/homepage.htm
10 http://www.gesis.org/Forschung/Informationstechnologie/ViBSoz.htm
11 http://www.gesis.org/Forschung/Informationstechnologie/CLEF-DELOS.htm

Verwendete Entwicklungsumgebungen und Projekte

Bis zum Jahr 1998 fand die Entwicklung von Software durch die Abteilung FuE mit dem Sybase Power-Builder statt, wobei einige Komponenten von Software jeweils in C++ implementiert wurden. Projekte die in diesem Zusammenhang entwickelt wurden, sind beispielsweise das Programm Schildkröte zur Verwaltung aller Arten von Periodika, das Programm COGET zum datenbankbasierten Publizieren sowie das Elektronische Verbandsinformations-, Recherche- und Analysesystem - ELVIRA. Diese Software wird aktuell sowohl am IZ als auch durch externe Stellen verwendet.

Ab 1998 wurde die Programmiersprache Java für neue Projekte verwendet, wobei die Entwicklungsumgebung IBM Visual Age zum Einsatz kam. Unter Einsatz dieser Software fand die Arbeit an den Projekten DBClear, ISSP-Data-Wizard, Daffodil und ViBSoz statt. Die durchschnittliche Größe eines Teams im Rahmen dieser Projekte betrug zwei bis drei Personen, wobei im Fall der Funktionsbibliothek FIOLA auch bis zu sieben Entwickler zur gleichen Zeit auf die Inhalte Zugriff hatten. FIOLA enthält oft genutzte Funktionen zur internen Darstellung von Benutzeranfragen in Informationssystemen. Die in diesem Zusammenhang entwickelten Algorithmen und Datenstrukturen werden durch die Mehrzahl der entstandenen Projekte verwandt.

Zur Koordination des Zugriffs auf die unterschiedlichen Quelldateien kam bei allen Projekten das integrierte Repository der Visual Age Entwicklungsumgebung zum Einsatz. Dieses Repository arbeitet nach dem Lock-Modify-Unlock Prinzip, gestattet also stets nur einem Entwickler schreibenden Zugriff auf eine Datei. Diese Restriktion stellte sich als die Produktivität mindernd heraus. Insbesondere ist der Fall problematisch, bei dem versäumt wurde den exklusiven Zugriff auf eine Datei nach deren Bearbeitung wieder aufzugeben. Hierbei war es notwendig, durch einen Administrator die Datei explizit freizugeben, wenn der entsprechende Entwickler nicht verfügbar war.

In der zweiten Jahreshälfte 2002 fand ein Wechsel zur Entwicklungsumgebung Eclipse statt. Ausschlaggebend für diese Entscheidung war die mangelnde Unterstützung der Visual Age Entwicklungsumgebung für neuere Versionen der Java Laufzeitumgebung, was sowohl Entwicklung als auch Fehlersuche erschwerte. Mit der Einführung der Eclipse Entwicklungsumgebung wurde auch das Versionskontrollsystem CVS eingeführt. Diese Software basiert im Vergleich zum Visual Age Repository auf dem so genannten Copy-Modify-Merge Prinzip, erlaubt also die parallele Bearbeitung einer Datei durch mehrere Entwickler. Die unter Verwendung von Eclipse und CVS erstellten Projekte umfassen unter anderem DBClear, die durch mehrere Projekte genutzten Programmbibliotheken CORA und FIOLA und Infoconnex.

Aktuelle Verwendung von Versionskontrollsystemen

Sowohl im Fall des Visual Age Repositories als auch bei Eclipse ist die Nutzung des Versionskontrollsystems in die Benutzungsoberfläche der Entwicklungswerkzeuge integriert. Das CVS-Plugin für Eclipse, eine Komponente zur Erweiterung von Eclipse um Funktionalität zum Umgang mit CVS, unterstützt dabei sowohl den Umgang mit versionierten Dateien in der Arbeitskopie und im Repository als auch eine intuitive Darstellung der Unterschiede zwischen den verschiedenen Revisionen einer Datei. Somit steht jedem Entwickler durch Eclipse das Versionskontrollsystem direkt zur Verfügung und ist vollständig in den Arbeitsablauf integriert. Aktuell wird die überwiegende Mehrheit aller Projekte mit Eclipse und CVS entwickelt und gepflegt; Visual Age kommt zur Pflege bestehender Projekte zum Einsatz, wird also weiterhin verwendet. Neben den Mitarbeitern der Abteilung FuE in Bonn wird das Repository auch mittels Fernzugriff verwendet. Eine externe Partei mit Zugriff auf das CVS sind studentische Hilfskräfte mit lesendem Zugriff auf das Repository. Diese Gruppe von Nutzern umfasst in der Regel ein oder zwei Personen. Weiterhin findet eine Nutzung des Repositories durch die Mitarbeiter der GESIS in Berlin statt; diese Mitarbeiter haben sowohl lesenden als auch schreibenden Zugriff auf den versionierten Datenbestand.

Versionierter Datenbestand und Systemarchitektur

Das CVS Repository befindet sich auf einem lokalen Server und wird über das hausinterne Netzwerk angesprochen. Dabei kommt serverseitig die CVS-Variante CVSNT zum Einsatz. Insgesamt hat das Repository einen Umfang von mehreren hundert Megabyte und enthält den versionierten Datenbestand von mehr als 20 Projekten, bestehend aus etwa 8000 einzelnen Dateien. Die Historien der Projekte spielen eine wichtige Rolle bei Entwicklung und Pflege der Software und sollen bei einem eventuellen Wechsel auf ein neues Versionskontrollsystem definitiv übernommen werden.

2.3 Befragung der Mitarbeiter zur Nutzung von Versionskontrollsystemen

Im Folgenden sollen nun die Ergebnisse dargestellt werden, die bei einer Befragung der Mitarbeiter der Abteilung FuE gewonnen wurden. Diese Befragung hatte das Ziel, die Anforderungen und Muster bei der Nutzung zu ermitteln, die sich aus Sicht der individuellen Entwickler an ein Versionskontrollsystem stellen. Dabei standen zwei Aspekte im Vordergrund:

Es soll jene Teilmenge der Funktionalität eines Versionskontrollsystems identifiziert werden, die in der Abteilung FuE tatsächlich verwendet wird.

Es sollen Anlass und Frequenz für die Nutzung des Versionskontrollsystems festgestellt werden.

Inhalt und Durchführung der Umfrage

Die Umfrage geschah im Rahmen eines Gesprächs mit jeweils einem Mitarbeiter der Abteilung FuE. Verwendet wurde ein für diese Reihe von Interviews erstellter Fragenkatalog. Die Antworten wurden handschriftlich notiert, die Auswertung fand in Form einer Zusammenfassung aller zu einer Frage gegebenen Antworten statt. Nach den in (siehe Hegner 2003) dargestellten Kriterien handelte es sich also um ein halb strukturiertes, direktes Einzelinterview.

Ergebnisse der Befragung

Welches Versionskontrollsystem wird verwendet?
Die überwiegende Mehrheit der Entwickler nutzte das CVS Versionskontrollsystem für die jeweils bearbeiteten Projekte, lediglich im Rahmen eines Projektes wurde noch Visual Age und die in dieser Software integrierte Kontrolle von Revisionen verwendet.

Wie oft wird das Versionskontrollsystem verwendet?
Die meisten der Befragten orientierten sich in der Nutzung des Versionskontrollsystems an ihren Arbeitsfortschritten, wobei Ergebnisse jeweils in das Versionskontrollsystem eingestellt werden, wenn ein Bearbeitungsschritt beendet wurde oder bevor tiefer gehende Änderungen am Quellcode begonnen werden. Die Revisionen im Versionskontrollsystem werden also vor dem Hintergrund der Relevanz der Änderungen erstellt und nicht vor einem zeitlichen Hintergrund. Jede Revision entspricht einem abgeschlossenen Arbeitsvorgang.

Wird Eclipse für den Umgang mit dem Versionskontrollsystem verwendet? Für welche Aufgaben?
Alle Anwender der Eclipse Entwicklungsumgebung nutzen diese auch für den Umgang mit dem Versionskontrollsystem. Unterschiede gab es bei der jeweils genutzten Funktionalität. Teilweise wurde lediglich das Einstellen und Entnehmen von Revisionen aus dem Repository verwendet, in anderen Fällen auch die Darstellung von Unterschieden zwischen Revisionen oder die Unterstützung bei der Behebung von Konflikten.

Welche anderen Werkzeuge werden für die Nutzung von CVS verwendet?
Neben der Nutzung von Eclipse wurde in einigen Fällen das Programm TortoiseCVS für den Umgang mit dem Versionskontrollsystem verwendet.

Wurden eigene Skripte oder Werkzeuge implementiert, um bestimmte Funktionalität zu erzielen?

Skripte zur Erweiterung der Funktionalität des Clients wurden von keinem der Befragten verwendet, ebenso wenig selbst implementierte Werkzeuge. Auf dem Server wird ein System verwendet, um bei Änderungen an den versionierten Inhalten eine Nachricht an eine Mailingliste zu verschicken.

Wie oft treten Konflikte auf?
Konflikte bezeichnen in diesem Kontext sich widersprechende Änderungen, die durch unterschiedliche Entwickler in die gleiche Datei eingestellt worden sind. Konflikte treten laut den Befragten selten auf, was nach Meinung der Befragten zum einen der recht kleinen Gruppengrößen und der räumlichen Nähe aller beteiligten Entwickler zuzurechnen ist, andererseits aber auch der klaren Abgrenzung von Aufgabenschwerpunkten und der Zuständigkeit von Entwicklern für bestimmte Module.

Wie oft und wie werden Branches verwendet?
Branches werden nur in seltenen Fällen verwendet, beispielsweise um eine Entwicklungslinie zu schaffen, in die experimentelle Änderungen eingestellt werden. Bei der Befragung wurde als Argument gegen die Verwendung von Branches die Anforderung genannt, auf Änderungs- und Erweiterungswünsche schnell reagieren zu müssen.

Besteht die Notwendigkeit, Dateinamen von Dateien im Repository zu ändern?
Der Wunsch nach der Möglichkeit zur Versionierung von Verzeichnissen, also dem Umbenennen und Verschieben von Dateien, wurde von etwa zwei Dritteln der Entwickler genannt; diese Entwickler empfanden diese Funktionalität übereinstimmend als sehr wünschenswert. In erster Linie wurde der Wunsch nach einem nachträglichen Refactoring als ausschlaggebend für den Bedarf einer Versionierung von Verzeichnissen genannt.

In welchem Umfang wird das Repository für Binärdaten verwendet?
Binärdaten werden nur in geringem Maß im Repository abgelegt. Bei den abgelegten Binärdaten handelt es sich meist um Bilder oder Icons, die Benutzungsoberflächen der jeweils entwickelten Software sind.

Besteht eine Notwendigkeit für Locks, also die Möglichkeit, Dateien nur für die Bearbeitung durch einen Nutzer zuzulassen?
Eine Mehrheit der Befragten hielt das Vorhandensein von exklusiven Locks für nicht notwendig oder sogar für störend. Insbesondere wenn zu Beginn eines Projektes viele Entwickler auf die grundlegenden Klassen der Klassenstruktur zugreifen kam es bei der Verwendung von Visual Age und damit von exklusiven Locks oft zu Problemen. Einige Befragte äußerten in diesem Zusammenhang, dass es durch die Notwendigkeit, Locks manuell zu lösen, wenn dies vom Inhaber des Locks nach der Bearbeitung versäumt wurde, zu erheblichen Zeitverlusten kam. Eine kleinere Gruppe empfand exklusive

Locks als nützliche Eigenschaft eines Versionskontrollsystems, um komplexe Veränderungen vornehmen zu können, ohne das Risiko von Konflikten einzugehen.

Wie oft wird extern, also von außerhalb des Bonner Standortes, auf das Repository zugegriffen?
Zugriffe von außerhalb wurden durch studentische Hilfskräfte vorgenommen, diese hatten meist nur lesenden oder auf bestimmte Inhalte eingeschränkten Zugriff auf das Repository. Weitere Zugriffe werden durch die Mitarbeiter in Berlin durchgeführt. Diese Zugriffe wurden wegen Problemen mit der Leistungsfähigkeit und Verfügbarkeit der notwendigen Netzwerkverbindung als problematisch bezeichnet. Während des Gesprächs äußerte ein Mitarbeiter den Vorschlag, einen lesenden Zugang mittels einer Webschnittstelle einzurichten, um auch ohne Entwicklungswerkzeuge, beispielsweise von einer Tagung aus, auf das Repository zugreifen zu können.

Wie wichtig ist eine detaillierte Vergabe von Rechten für das Repository?
Eine detaillierte Vergabe von Rechten wurde primär im Zusammenhang mit der Arbeit von studentischen Hilfskräften als wichtig empfunden. Es wurde mehrmals geäußert, dass die Möglichkeit nützlich sei, Rechte auch auf der Ebene der Verzeichnisse zu vergeben. Die Möglichkeit, Rechte zur Administration zu delegieren, wurde überwiegend als nicht notwendig erachtet.

Gab es in der Vergangenheit Probleme bei der Verwendung von CVS?
Die meisten Befragten hatten keine Beschwerden über den Umgang mit CVS, lediglich in der Anfangsphase der Nutzung des Systems gab es Probleme wegen der fehlenden Möglichkeit, Dateien umzubenennen. Kleinere Probleme gab es mit der Handhabung von Binärdateien und der von CVS betriebenen Expansion von Schlüsselwörtern, beispielsweise das automatische Einfügen der Revisionsnummer.

Welche Meinung besteht zur Verwendung eines binären Datenformats für das Repository?
Die Verwendung eines binären Formates zur Datenspeicherung im Repository wurde überwiegend nicht als Problem angesehen. Es wurde jedoch die Notwendigkeit genannt, den Datenbestand im Rahmen eines Backups sichern zu können.

Welche Meinung besteht zur Verwendung einer einheitlichen, für das gesamte Repository gültigen Revisionsnummer?
Eine einheitliche Revisionsnummer die jeder Datei, unabhängig von tatsächlichen Änderungen, bei einem Commit zugewiesen wird, stellt einen Unterschied zum Verhalten von CVS dar. Hier wird die Revision einer Datei mit einem Commit nur dann erhöht, wenn es in der Datei zu Änderungen kam. Die Befragten empfanden keines der beiden Verhalten als dem anderen inhä-

rent überlegen; beiden Vorgehensweisen wurden Vorteile zugebilligt, wenn bestimmte Informationen des Versionskontrollsystems abgefragt werden sollen. Während Revisionen auf der Ebene von Dateien als nützlich empfunden wurden, um sich schnell einen Überblick über die Häufigkeit von Änderungen an einer Datei zu verschaffen, wurden Revisionsnummern auf Ebene des Repositories als vorteilhaft empfunden, wenn alle Dateien einer spezifischen Revision schnell identifiziert werden sollen.

Welche Funktionalität, die nicht durch CVS geboten wird, sollte durch ein Versionskontrollsystem verfügbar sein?
In diesem Zusammenhang wurden zwei Punkte genannt, die automatische Durchführung von Backups und ChangeSets. ChangeSets bezeichnen in diesem Zusammenhang die Möglichkeit, gemeinsam erfolgte Veränderungen auch in der History des Repositories explizit als atomare Veränderungen identifizieren zu können.

2.4 Fazit

Aus den Interviews mit den Mitarbeitern der Abteilung FuE lassen sich zahlreiche Informationen über die Nutzung von Versionskontrollsystemen gewinnen:

- Der wichtigste Punkt in der täglichen Anwendung ist die Einbindung von Versionskontrollsystemen in die verwendete Entwicklungsumgebung. Alle Befragten haben jeweils ihre Entwicklungsumgebung Eclipse für den Zugriff auf die durch das Versionskontrollsystem verwalteten Datenbestände verwendet; dies geschah sowohl beim Einsatz von Eclipse als auch bei Visual Age.
- Die Auflösung von Konflikten durch das Versionskontrollsystem wird nur relativ selten benötigt. Eine zeitgleiche Arbeit an einer Datei durch mehrere Entwickler fand nur selten statt. In den Fällen in denen eine Datei durch mehrere Personen zeitgleich bearbeitet wurde, war es oft möglich, Konflikte durch entsprechende Absprachen zu vermeiden. Diese Form der Koordination wird durch die räumliche Nähe aller Mitarbeiter bedingt.
- Die Arbeit an einem Projekt findet überwiegend in kleinen Gruppen von 2 oder 3 Personen statt. Dabei arbeitet jede Person in erster Linie auf einem eigenen Bestand von Klassen, lediglich in der Anfangsphase eines Projektes arbeiten mehrere Entwickler gemeinsam an einer Datei oder einem Java Package.
- Branches werden selten verwendet; dies bedingt zusammen mit der geringen Nachfrage nach Funktionen zur Konfliktauflösung, dass entsprechende Funktionalität des Versionskontrollsystems auf dem durch CVS angebotenen Niveau vollkommen ausreichend ist.

- Veränderungen werden ins Versionskontrollsystem eingestellt, wenn der Entwickler selbst sie jeweils für beendet hält.
- Der Wunsch nach der Möglichkeit zur Versionierung von Verzeichnissen, also der Möglichkeit beispielsweise Dateien umzubenennen, wird von mehreren Entwicklern genannt. Dies ist auch der Grund, aus dem ein neues Versionskontrollsystem in Betracht gezogen wird.
- Vor dem Hintergrund der Erkenntnisse dieses Kapitels sollen im Anschluss eine Reihe unterschiedlicher Versionskontrollsysteme betrachtet werden.

3 Vergleich verschiedener Versionskontrollsysteme

In diesem Kapitel werden unterschiedliche Versionskontrollsysteme vorgestellt. Im ersten Teil des Kapitels wird auf die grundlegende Architektur eingegangen, im Anschluss auf eine Reihe von Versionskontrollsystemen, die sich zur Zeit im Einsatz oder in der Entwicklung befinden.

3.1 Eigenschaften und Architektur von Versionskontrollsystemen

Im Folgenden sollen einige Kriterien vorgestellt werden, anhand derer Versionskontrollsysteme unterschieden und bewertet werden können. Die Kriterien sollen dabei weniger dazu dienen, die unterschiedlichen Versionskontrollsysteme gegeneinander abzugrenzen oder zu bewerten, sondern sollen im Hinblick auf eine spätere Evaluation helfen, die Bandbreite der zu beachtenden Punkte darzustellen. Weiterhin sollen diese Kriterien dazu dienen, die technische Vielfältigkeit von Versionskontrollsystemen zu verdeutlichen. Bewusst verzichtet wird - sowohl bei der Vorstellung dieser Kriterien wie auch bei der späteren Darstellung konkreter Versionskontrollsysteme, auf die detaillierte Besprechung und wertende Gegenüberstellung technischer Details. Dies ist der Fall, da die Bewertung eines Versionskontrollsystems stets in hohem Maß von den persönlichen Vorlieben und Arbeitsgewohnheiten abhängt.

Einrichtung und Administration

Bei der Betrachtung dieses Kriteriums wird untersucht, mit welchem Arbeits- und Lernaufwand Einrichtung und Betrieb des Versionskontrollsystems verbunden sind. Dies umfasst die Einführung des Systems, die Administration von Nutzern und deren Rechten, sowie die Pflege des Datenbestandes in Repositories. Relevante Punkte zur Bewertung dieser Fragen sind das Format der Konfigurationsdaten und das Vorhandensein von Werkzeugen, die bei der

Konfiguration und Überwachung des Systems genutzt werden können. Weitere Punkte sind der Aufwand für die Datensicherung, beispielsweise angesichts der Frage, ob diese im laufenden Betrieb durchgeführt werden kann. Ein weiterer Aspekt ist die Möglichkeit das System zu automatisieren, beispielsweise in Form der Durchführung von automatischen Prüfungen der Integrität in festgelegten Abständen oder die Ausführung von Skripten unter zuvor festgelegten Bedingungen. Bei all diesen Punkten muss die Frage, welche Möglichkeiten ein Administrator besitzt, um ein System den jeweiligen Anforderungen anzupassen, getrennt von der Komplexität und Fehlerträchtigkeit der Administration betrachtet werden.

Möglichkeiten zur Rechtevergabe und Authentifizierung

Bei der Betrachtung dieses Punktes sind zwei Aspekte relevant, die Granularität der Rechtevergabe und die Authentifizierungsmechanismen, auf deren Basis die Rechtevergabe erfolgt. Unter dem Stichpunkt der Granularität wird die Frage behandelt, auf welcher Ebene Rechte für ein unter Versionskontrolle stehendes Projekt vergeben werden können. Einige Systeme bieten hier nur eine generelle Vergabe von Lese- und Schreibrechten für ein ganzes Projekt, während in anderen Fällen eine feinere Rechtevergabe auf Verzeichnisebene möglich ist.

Bei den Authentifikationsmechanismen existieren Ansätze, die eine eigene Datenbasis mit Benutzerinformationen für das Versionskontrollsystem pflegen und solche, die sich in ein bestehendes System, etwa die Benutzerverwaltung des Betriebssystems, einbetten lassen.

Nutzung in Netzwerken

Bei der Anwendung durch mehrere Benutzer wird durch Versionskontrollsysteme ein Netzwerkprotokoll zur Kommunikation und zum Austausch von Nachrichten verwendet[12]. Bei der Nutzung von Netzwerken durch das Versionskontrollsystem müssen mehrere Fragen untersucht werden. Einerseits muss die Frage betrachtet werden, welche Bandbreite für die Arbeit mit dem System notwendig ist. Während die Bandbreite eines LANs für alle hier vorgestellten Systeme ausreichend ist, wird bei einer Nutzung über ein WAN, beispielsweise eine Einwählleitung, der effiziente Umgang mit der bereitstehenden Bandbreite relevant. Hier muss beachtet werden, für welche Operationen auf das zentrale Repository zugegriffen wird und welche zu übertragende

[12] Ausnahmen sind hier die Nutzung in einem lokalen oder verteilten Dateisystem, wobei eine Nutzung des Netzwerkes transparent für das Versionskontrollsystem erfolgen würde.

Datenmenge für eine Synchronisation notwendig ist. Dies wird maßgeblich dadurch beeinflusst, ob nur Änderungen an Dateien oder die Dateien selbst übertragen werden und welche Art von Komprimierung verwendet wird. Neben der Effizienz bei der Verwendung des Netzwerkes ist ebenfalls wichtig, welche Protokolle durch das Versionskontrollsystem verwendet werden. Hier ist der Aspekt der Sicherheit der mittels des Netzes übertragenen Informationen zu beachten; das verwendete Protokoll sollte sowohl eine sichere Authentifikation als auch eine geschützte Übertragung gewährleisten.

Notwendige Laufzeitumgebung

Bei der Bewertung eines Versionskontrollsystems muss die Frage beachtet werden, welche Laufzeitumgebung durch die Software vorausgesetzt wird. Die Laufzeitumgebung umfasst dabei sowohl benötigte Software, also das Betriebssystem und weitere zur Nutzung des Versionskontrollsystems benötigte Programme, als auch die für einen performanten Betrieb notwendige Hardware.

Hardwareseitig ist neben den Laufzeitanforderungen an Rechner und Netzwerk die Frage interessant, wie viel Speicherplatz durch Daten im Repository benötigt wird. Durch die Kapazität aktueller Festplatten ist der Betrieb eines auch mehrere Gigabyte großen Repositories zwar kein prinzipielles Problem, allerdings sollte die Effizienz der Speicherung vor dem Hintergrund von Backups und Replikation eines Repositories weiterhin beachtet werden.

Ein wichtiger, allerdings meist schwer zu bewertender Punkt ist die Frage nach der Skalierungsfähigkeit des Systems, also das Verhältnis, in dem sich die Anforderungen an das Laufzeitsystem mit erhöhter Arbeitslast des Versionskontrollsystems verändern und die Frage, wie hoch die maximale durch die Architektur bedingte Arbeitslast des Systems ist.

Verbreitung und Ausgereiftheit

Versionskontrollsysteme sind in der Regel integraler Bestandteil bei der Entwicklung von Software, entsprechend wichtig ist die technische Zuverlässigkeit dieser Systeme. Ein zentraler Punkt ist hierbei die Absicherung gegen Datenverlust, sowohl durch Fehlbedienung hervorgerufenen als auch durch eine Beschädigung des Repositories verursachten[13]. Ein zusätzlicher Aspekt der Ausgereiftheit stellt die Stabilität der Schnittstellen des Systems dar. Dies

[13] Während der Datenbestand eines Versionskontrollsystems in eine regelmäßige Datensicherung aufgenommen werden sollte, kann eine solche Sicherung nur bedingt gegen eine Korruption der Historie helfen, bei der sich Fehler über einen längeren Zeitraum kumulieren.

betrifft sowohl die Benutzerschnittstellen und deren Repräsentation in den jeweiligen Entwicklungsumgebungen als auch die Programmierschnittstellen Erstere würden ein Umlernen von Anwendern erfordern und damit sowohl erhöhten Zeitaufwand bedingen als auch die Akzeptanz durch die Benutzer des Systems gefährden. Die Umstellung von Programmierschnittstellen hingegen kann eine Anpassung von eigener Software und Programmen Dritter erfordern, die mit dem Versionskontrollsystem verwendet werden. Ein Versionskontrollsystem sollte für den Einsatz in der Produktion prinzipiell keine kritischen Fehler mehr aufweisen, es soll also ein hohes Maß an Qualität (siehe Sommerville 2004) besitzen. Weiterhin sollten ausreichende Dokumentation und Erfahrungen im Einsatz des Systems vorliegen. Diese Aspekte der Ausgereiftheit eines Systems werden durch dessen Verbreitung positiv beeinflusst. Bei einem weit verbreiteten System werden Fehler schneller gefunden, auch sind in der Regel mehr Dokumentation und ein größerer Erfahrungsschatz verfügbar.

Benutzerfreundlichkeit und Softwareergonomie

Durch die enge Einbindung in den Arbeitsprozess und die teilweise hohe Komplexität der zu lösenden Aufgaben stellen sich unter dem Gesichtspunkt der Softwareergonomie hohe Anforderungen an die Benutzerschnittstelle von Versionskontrollsystemen. Die vom Anwender verwendete Software eines Versionskontrollsystems sollte grundsätzlich die durch die ISO 9141 definierten sieben Anforderungen erfüllen, wie sie im Folgenden erläutert werden. Der Aspekt der Fehlertoleranz ist wegen der hohen Wichtigkeit, die den verwalteten Daten oft zukommt und der teilweise hohen Komplexität der durchgeführten Handlungen von besonderer Bedeutung. Hier sollte auf jeden Fall sichergestellt werden, dass es keinerlei Möglichkeit gibt, durch Fehlbedienung die Integrität des Repositories zu beeinträchtigen. Bei der Bewertung der Lernförderlichkeit und der Selbstbeschreibungsfähigkeit spielt neben der eigentlichen Dialog- und Kommandogestaltung auch wieder das zugrunde liegende Anwendungsmodell eine Rolle. Die Darstellung von Primitiven des Versionskontrollsystems wie Branches und Revisionen hat großen Anteil daran, wie schwer es einem Anwender fällt, neben den einfachen Grundaktionen Checkout, Update und Commit auch mit komplexeren Operationen zu arbeiten. Die Auffassung von Branches als virtuelle Verzeichniskopien, wie sie von einigen Versionskontrollsystemen verwendet wird, führt über die Analogie zum nicht versionierten Dateisystem auch unerfahrene Nutzer an die Verwendung von Entwicklungszweigen heran. Ein wichtiger Aspekt um das Erlernen des Systems zu fördern ist das Vorhandensein von Dokumentation, bei den im weiteren Verlauf dieses Textes untersuchten Programmen gab es hier, was Qualität, aber auch die generelle Verfügbarkeit betrifft, große Unterschiede.

Bei der Bewertung der Steuerbarkeit und der Individualisierbarkeit müssen alle Möglichkeiten betrachtet werden, mit dem System zu interagieren. Viele Systeme bieten sowohl die Möglichkeit, mit der Kommandozeile zu arbeiten als auch mit unterschiedlichen graphischen Benutzungsoberflächen, die jeweils getrennt betrachtet werden müssen.

Wurde von den zukünftigen Anwendern eines Versionskontrollsystems bereits ein verwandtes System verwendet, so kann sich der Punkt der Erwartungskonformität als besonders kritisch erweisen. Dies ist der Fall, da sich unterschiedliche Versionskontrollsysteme in ihrem Anwendungsmodell und in Darstellung und Verwaltung teilweise grundlegend unterscheiden, so dass selbst bei einem für sich betrachtet softwareergonomisch gut gestalteten System durch notwendiges Neuerlernen Probleme auftreten können.

Abschließend zu diesem Punkt muss noch einmal betont werden, dass eine Bewertung von Versionskontrollsystemen auch im Hinblick auf softwareergonomische Kriterien nur unter Beachtung der tatsächlichen Einsatzumgebung und Arbeitsabläufe erfolgen kann. Hilfreich ist hier sicher eine Orientierung an Nutzungsszenarien.

Unterstützung von atomaren Operationen

Unter diesem Punkt wird untersucht, ob ein Versionskontrollsystem atomare Commits und Checkouts unterstützt und damit die Integrität von Arbeitsumgebungen und Repository sicherstellt, auch wenn diese Operationen unterbrochen werden.

Ein Commit wird als atomar bezeichnet, wenn die Änderungen aller beteiligten Dateien entweder vollständig in das Repository eingehen oder im Fall eines Scheiterns vollständig verworfen werden. Diese Eigenschaft ist notwendig, wenn es während eines Zugriffs auf das Repository zu Störungen eines der beteiligten Rechner oder der zugrunde liegenden Netzwerkverbindung kommt und die Übertragung unterbrochen wird. Durch ein atomares Commit wird unter diesen Umständen verhindert, dass es im Repository zu einem undefinierten Zustand kommt. Auch wird bei atomaren Commits das Einstellen von veränderten Dateien durch einen Benutzer erst ermöglicht, wenn alle Konflikte gelöst sind; ein separates Einstellen von konfliktfreien und konfliktenthaltenden Dateien wird unterbunden.

Neben atomaren Commits sind auch atomare Updates und Checkouts wünschenswert. Diese Eigenschaft verhindert Szenarien, in denen ein unvollständiger Checkout oder ein unvollständiges Update die Arbeitskopie des Benutzers beeinträchtigt und sie entweder in einen nicht nutzbaren oder mit dem Repository inkonsistenten Zustand bringt.

Umgang mit Branches und Auflösung von Konflikten

Die Unterstützung beim Umgang mit konkurrierenden Änderungen ist eine der wichtigsten Eigenschaften eines Versionskontrollsystems. Besonders relevant wird diese Eigenschaft dann, wenn mehrere Personen an den gleichen Dateien arbeiten und insbesondere, wenn Branches zusammengeführt werden. Eine gute Unterstützung bei Konflikten sollte in der Lage sein, diese möglichst selbstständig zu beheben. So sollten Dateien, an denen mehrere Änderungen an unterschiedlichen Stellen vorgenommen wurden, automatisch integriert werden. Ist eine automatische Behandlung nicht möglich, so sollte das System in der Lage sein, eine weitestgehende Unterstützung zu gewährleisten. Bei der Darstellung von Konflikten gibt es eine Reihe unterschiedlicher Ansätze; ein Beispiel ist der so genannte 2-Way Merge, bei dem die durch den Benutzer gewünschte Veränderung mit dem bereits durch einen anderen Benutzer veränderten Original verglichen wird (siehe hierzu auch die folgende Abbildung 3) oder der 3-Way Merge, bei dem zusätzlich die Ursprungsdatei einbezogen wird.

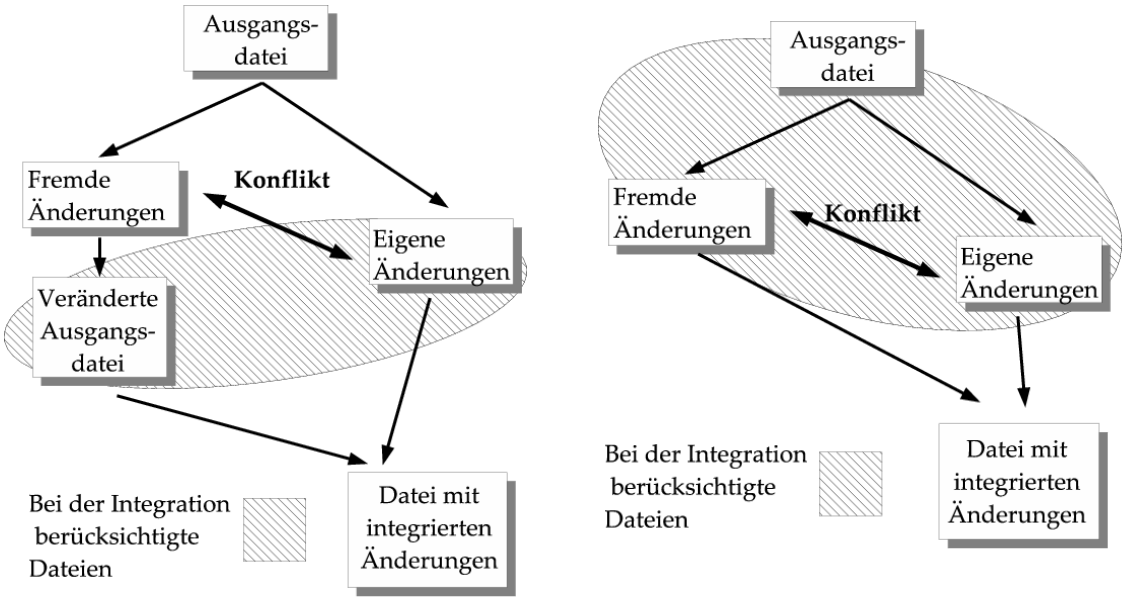

Abbildung 3: 2-Wege Merge und 3-Wege Merge

Dieses Vorgehen bietet mehr Informationen über die jeweiligen Änderungen am Original und erleichtert die Entscheidungen, wie zwei konkurrierende Veränderungen integriert werden sollen.

Neben der Behandlung und Darstellung von Konflikten ist im Umgang mit Branches die Frage wichtig, wie diese durch das jeweilige Versionskontrollsystem dargestellt werden. So können Branches als eine separate Dimension, also orthogonal zu Dateinamen und Revision, aufgefasst werden. Alternativ dazu steht eine Behandlung eines Branches als Kopie der jeweiligen Verzeichnisse mit den Projektdateien, also eine Behandlung eines Branches auf der Dimension des Namens der Datei. Da sich aus der gewählten Darstellung allein noch keine qualitativen Unterschiede im Umgang mit Branches ableiten lassen, ist diese Unterscheidung in erster Linie vor dem Hintergrund von Präferenzen der Benutzer und bisherigen Erfahrungen relevant. Besondere Beachtung verdient die Frage, inwieweit der Umgang mit Branches, etwa die Übernahme von Änderungen zwischen unterschiedlichen Ästen, durch die Software unterstützt wird. Wünschenswert ist hier eine Darstellung der bereits zwischen zwei Ästen übernommenen Änderungen, da hierdurch das häufig auftretende Problem doppelt übernommener Änderungen vermieden wird.

Unterstützung der Versionierung von Verzeichnissen

Neben der Versionierung von Dateiinhalten bieten einige Versionskontrollsysteme auch eine Versionierung von Dateinamen und Verzeichnissen an. Diese Eigenschaft gestattet es, eine Datei umzubenennen, zu kopieren und zu löschen und gleichzeitig die Historie dieser Datei zu erhalten. Bei Unterstützung dieser Funktionalität werden Name und Pfad einer Datei ebenso wie deren Inhalt als zu versionierende Eigenschaft betrachtet. Insbesondere beim Refactoring von Quellcode wird diese Eigenschaft oft nachgefragt. Neben der reinen Unterstützung dieser Funktion muss auch beachtet werden, wie die eigentliche Implementierung im Versionskontrollsystem zu gestalten ist. Einige Versionskontrollsysteme nutzen zur Darstellung einer Umbenennung eine Kopie mit einem anschließenden Löschen des Originals; die resultierende Aufteilung in zwei separate Schritte kann sich in einigen Fällen, beispielsweise bei der Verwendung von Branches, als ungeeignet erweisen.

Funktionsumfang des Systems und Unterstützung durch externe Werkzeuge

Ergänzend zum Umgang mit Revisionen gibt es eine Reihe ergänzender Funktionalität, die sich einem Versionskontrollsystem zuordnen lässt. Ein Beispiel ist das Bugtracking[14], also die Protokollierung und Verwaltung von Fehlern von ihrem Auffinden bis zu ihrer Behebung. Durch die Integration von Bugtracking und Versionskontrollsystem können Fehler anhand spezifi-

[14] Teils auch als Defect- oder Featuretracking bezeichnet.

scher Revisionen des zugrunde liegenden Quellcodes verfolgt werden. Ein anderes Beispiel ist die Integration des Versionskontrollsystems in das Buildmanagement[15]; so können automatische Tests durchgeführt werden, um die Funktionalität der Software, etwa durch UnitTests, sicherzustellen. Neben der reinen Verfügbarkeit dieser Funktionalität muss auch die Frage geklärt werden, in welcher Form sie verfügbar ist. Denkbar ist hier, dass die beschriebenen Funktionen integraler Bestandteil der Software sind, ebenso ist aber auch eine Realisierung als separate Erweiterung des Versionskontrollsystems denkbar. Bei einem externen Produkt muss beachtet werden, ob es durch den Anbieter des Versionskontrollsystems oder durch externe Anbieter verfügbar ist und welche Kosten mit einer Anschaffung eventuell verbunden sind. In jedem Fall muss die Frage einbezogen werden, inwieweit zusätzliche Funktionalität eines Versionskontrollsystems im Rahmen des geplanten Einsatzes der Software sinnvoll ist oder ob durch die gestiegene Komplexität mit Nachteilen zu rechnen ist.

Lizenz und Kosten der Software

Anhand der verwendeten Lizenz lassen sich aktuelle Versionskontrollsysteme grob in zwei Gruppen aufteilen. Eine Gruppe stellen hier kommerziell vertriebene Softwaresysteme dar. Für ein solches System ist in der Regel für jeden Nutzer, in einigen Fällen auch zusätzlich für einen zentralen Server, eine Lizenzgebühr zu entrichten. Die Preise variieren mit dem jeweiligen Produkt, der Anzahl der Lizenzen und teilweise auch mit dem Kauf oder der Miete einer Lizenz. In der Praxis sind etwa 300 Euro bis 8000 Euro für eine Lizenz zu zahlen. In den Lizenzkosten ist in der Regel eine Form von Unterstützung durch den Hersteller bei der Einrichtung und Nutzung der Software enthalten, beispielsweise in Form einer Hotline oder moderierten Foren.

Eine zweite Gruppe stellen Programme aus dem Open Source Bereich[16] dar. Diese Programme sind ohne Zahlung von Lizenzkosten nutzbar, dafür fehlt hier oft die Möglichkeit, professionellen Support zu erhalten.

Bei Lizenzverträgen sollte beachtet werden, in welchem Umfang sie über die reinen Kosten hinaus noch weitere Bedingungen für die Nutzung festlegen. In der Praxis sind obligatorische Updates auf die jeweils neueste Version einer Software als Teil einer Lizenzvereinbarung bekannt.

---

[15] http://www.martinfowler.com/articles/continuousIntegration.html
[16] http://www.opensource.org

Speicherung der versionierten Daten

Bei der Speicherung der versionierten Daten durch ein Versionskontrollsystem kommen entweder reguläre Dateien oder eine Form einer Datenbank zum Einsatz, wobei dieser Begriff in diesem Zusammenhang sehr frei verwendet wird und nicht mit der Definition nach Codd (siehe Codd 1970) übereinstimmt. Ein Beispiel für verwendete Dateiformate sind etwa RCS Dateien. Das Dateiformat wird im Folgenden beschrieben. Durch die Verwendung einer Datenbank lassen sich durch diese bereitgestellt Dienste, etwa die Unterstützung für Transaktionen, ohne größeren Implementierungsaufwand für das Versionskontrollsystem nutzen. In beiden Fällen sollte besonders die Frage beachtet werden, wie die Migration eines bestehenden Repositories auf einen anderen Rechner durchgeführt werden kann und welche Möglichkeiten zur Datensicherung angeboten werden. Eine weitere Frage betrifft den verwendeten Algorithmus zur Isolation von Veränderungen zwischen zwei Versionen einer Datei. Dieser sollte in der Lage sein, auch beim Umgang mit Binärdateien nur die Unterschiede zu speichern, die zwischen zwei unterschiedlichen Revisionen einer Datei bestehen. Einige Systeme erstellen stattdessen eine Kopie für jede Revision einer Binärdatei.

Zentrale und verteilte Architekturen

Die Unterschiede zwischen einem verteilten und einem zentralisierten Versionskontrollsystem betreffen sowohl die Architektur des Systems als auch das zugrunde gelegte Entwicklungsmodell. Vom Blickwinkel der Architektur her unterscheiden sich beide Ansätze darin, dass bei einem verteilten System jeder Nutzer des Versionskontrollsystems ein eigenes Repository verwaltet, während bei einem zentralisierten System nur ein Repository existiert, aus dem jeder Benutzer seine Arbeitskopie entnimmt und nach der Bearbeitung wieder einstellt. Da dieses zentrale Repository in den meisten Fällen durch einen eigenständigen Serverprozess verwaltet wird, wird bei zentralisierten Systemen auch oft von Client-Serversystemen gesprochen. Aus dem Blickwinkel der Nutzung werden Systeme mit einem zentralen Repository auch als auf Snapshots basierend bezeichnet. Diese Bezeichnung rührt vom Bild eines zentralen Repositories her, von dem bei jeder Veränderung eine Aufnahme genommen wird. Der Zustand einer Datei im Repository in der jeweils aktuellen Revision ist also Referenzpunkt für alle anderen Versionen dieser Datei. Verteilte Versionskontrollsysteme werden auch als auf ChangeSets basierend bezeichnet; diese Systeme arbeiten auf logischer Ebene nicht mit zentralen Repositories und in diese eingespielte Veränderungen, sondern betrachten ih-

ren versionierten Datenbestand als Aggregierung von Veränderungen[17]. Deutlich wird der Unterschied zwischen einer ChangeSet und einer Snapshot Fokussierung bei der zugrunde gelegten Verwendung der Systeme: bei einem System mit einem zentralen Repository, werden Dateien nach der lokalen Bearbeitung stets wieder in das Repository eingestellt, wobei Konflikte nötigenfalls behoben werden, maßgeblich ist der Zustand im Repository. Bei einem verteilten System gibt es kein zentrales Repository über das die beteiligten Entwickler ihre Änderungen koordinieren. Sollen hier Veränderungen anderen Nutzern zur Verfügung gestellt werden, geschieht dies durch den expliziten Austausch von ChangeSets. Jeder Benutzer kann selbst entscheiden, aus welchen ChangeSets sich seine Arbeitskopie zusammensetzt. Aus der Sicht der Befürworter von verteilten Versionskontrollsystem gestatten diese Systeme Arbeitsweisen, die mit zentralisierten Systemen nicht zu verwirklichen sind. In erster Linie wird hier die Verwendung von so genannten Staging Areas genannt[18], Repositories, in denen Änderungen gesammelt und integriert werden, bevor sie in ein weiteres Repository übernommen werden, dessen Inhalt der Hauptentwicklungslinie des jeweiligen Projektes entspricht. Als weiterer Vorteil wird die Möglichkeit genannt, über längere Zeiträume isoliert vom Repository zu arbeiten, auch dies vor dem Hintergrund, dass durch die Fokussierung auf ChangeSet eine Synchronisation unterschiedlicher Repositories vereinfacht wird. Diesen Vorteilen wird zugunsten der zentralisierten Versionskontrollsysteme oftmals entgegengehalten, dass durch Verwendung von Branches viele dieser Eigenschaften verteilter Systeme auch mit herkömmlichen Client- Server Systemen erreicht werden können[19]. Weiterhin muss beachtet werden, dass zur tatsächlichen Nutzung der Vorteile, die sich aus der Verwendung von ChangeSets ergeben, weiter entwickelte Werkzeuge zur Integration von Veränderungen notwendig sind: von dem Vorhandensein solcher Werkzeuge würden aber zentralisierte Systeme in gleichem Maß profitieren. Die Frage, ob und welcher Ansatz dem anderen überlegen ist, lässt sich nur schwer beantworten; die Diskussion wird sehr kontrovers geführt und wird oft von der persönlichen Arbeitsweise mit dem Versionskontrollsystem bestimmt.

---

[17] Die Unterscheidung zwischen ChangeSet und Snapshot ist explizit nicht technischer Natur; die interne Repräsentation eines Versionskontrollsystems ist unabhängig von dieser logischen Sichtweise.

[18] http://www.bitkeeper.com/Products.BK_Pro.Feature_.html

[19] http://web.mit.edu/ghudson/thoughts/bitkeeper.whynot

3.2 Darstellung relevanter Versionskontrollsysteme

Auf den folgenden Seiten sollen einige aktuelle Versionskontrollsysteme kurz darstellt werden. Ziel ist es, einen Überblick über konkrete Implementierungen zu bieten, nachdem im vorangehenden Kapitel bereits eine Darstellung von technischen Eigenschaften erfolgt ist. Die betrachteten Systeme sollen also in erster Linie vorgestellt werden, ein Vergleich der unterschiedlichen Eigenschaften der Systeme wird nicht durchgeführt. Die Auswahl der betrachteten Systeme geschah unter mehreren Gesichtspunkten. Mit CVS und Visual Source Safe werden zwei zur Zeit weit verbreitete Systeme vorgestellt, wobei CVS auch vor dem Hintergrund von Interesse ist, dass dieses System zur Zeit innerhalb der Abteilung FuE überwiegend verwendet wird.
Arch und Bitkeeper sind verteilte Versionskontrollsysteme und stehen stellvertretend für diesen relativ jungen Entwicklungszweig.
Perforce ist ein recht verbreitetes System, das für eine Vielzahl von Plattformen angeboten wird.
Subversion ist ein System, dessen offen vertretene Zielsetzung es ist, Fehler in CVS zu beseitigen und CVS langfristig zu ersetzen. Aufgrund der Vielzahl von verfügbaren Versionskontrollsystemen ist es schwer, eine wirklich repräsentative Übersicht zu bieten. Bewusst verzichtet wurde auf die Darstellung von Systemen, die aus technischen Gründen[20], wegen ihrer geringen Verbreitung[21]oder wegen der hohen technischen und finanziellen Aufwendungen[22] nicht für eine Verwendung durch die Abteilung FuE geeignet sind.

3.2.1 CVS

CVS[23], das Concurrent Versioning System, ist ein aus dem Open Source Bereich stammendes Versionskontrollsystem mit einer sehr hohen Verbreitung, es basiert auf einer Client-Server Architektur[24].

Unterstützte Plattformen

Bei CVS handelte es sich ursprünglich um eine Sammlung von Unix-Skripten, die das Versionskontrollsystem RCS um die Möglichkeit erweiterten, durch mehrere Nutzer gleichzeitig verwendet zu werden. Die erste Veröffentlichung dieser Skripte fand in einem Newsgroup Posting im Jahr 1986

---

[20] RCS und SCCS
[21] SourceJammer, MetaCVS
[22] Clearcase
[23] https://www.cvshome.org/
[24] Bei Verwendung eines Repositories im lokalen Dateisystem besteht die Möglichkeit, auf einen separaten Serverprozess zu verzichten.

statt (siehe Bar/Fogel 2003) Die Funktionalität dieser Skriptsammlung wurde im Jahr 1990 erweitert, gleichzeitig wurde eine Portierung des Programms nach C vorgenommen. Heute ist CVS unter Windows, Unix und Mac OS X sowie einer Reihe weiterer Plattformen verfügbar. Die Entwicklung der einzelnen Programme auf den jeweiligen Plattformen findet dabei durch eine Reihe von Projekten statt, einen einzigen zentralen Anbieter gibt es nicht. Insgesamt hat die weite Verbreitung von CVS dazu geführt, dass es eine große Auswahl unterschiedlicher Software gibt, die mit CVS zusammenarbeitet. Dies beinhaltet zahlreiche Benutzungsoberflächen, mehrere Implementierungen von Webschnittstellen und eine Reihe von Plugins, die den Zugriff auf CVS Repositories mittels Entwicklungsumgebungen oder Werkzeuge zur Verwaltung von Dateien gestatten.

Architektur

CVS ist eine Client-Server Architektur mit einem zentralen Repository. Neben dem Zugriff auf Repositories durch das Dateisystem wird auch die Verwendung in Netzwerken unterstützt, wobei mehrere Protokolle bereitstehen, die sich in erster Linie durch die verwendete Authentifizierung und die Absicherung der Datenkommunikation unterscheiden. Um einen Schutz der Datenübertragung zu gestatten, können SSH sowie Kerberos 4 und 5 eingesetzt werden. Hierbei wird sowohl die Anmeldung eines Benutzers als auch der Datentransfer zwischen Client und Server durch Verschlüsselung gesichert. Die Authentifizierung der Nutzer kann weiterhin über die Benutzerdaten des Betriebssystems oder eine eigene Datei mit Informationen zur Authentifikation erfolgen. CVS nutzt für sein Repository Textdateien im RCS Dateiformat und spezielle Dateien mit Verwaltungsinformationen. Dies ermöglicht es prinzipiell, an der Struktur des Repositories Veränderungen mittels eines Texteditors vorzunehmen. Ein von Seiten der Entwickler des Programms oft geäußerter Kritikpunkt an CVS ist die begrenzte Erweiterbarkeit des Quellcodes des Servers, was eine Anpassung und Erweiterung erschwert.

Eigenschaften und Nutzung

CVS bietet keine Unterstützung für das Umbenennen oder Verschieben von Dateien oder Verzeichnissen. Eine Umbenennung oder ein Verschieben von Dateien ist zwar durch eine Veränderung des Repositories selbst möglich (entweder durch einen Skript oder durch das Editieren von Hand), ein solcher Eingriff wird aber durch die Historie nicht berücksichtigt. Es ist somit im Nachhinein nicht feststellbar, wann eine Datei umbenannt wurde und aus welchem Grund dies geschehen ist. Auch beim Wiederherstellen eines vorherigen Zustands des Repositories mit dem ursprünglichen Namen der Datei treten Probleme auf. CVS unterstützt keine atomaren Commits oder Checkouts. Dies kann dazu führen, dass ein Zugriff auf das Repository nur unvollständig

erfolgt und einige Dateien nicht aktualisiert werden, was das jeweilige Projekt in einen inkonsistenten Zustand bringt. Bei einem Abgleich von Daten zwischen Repository und lokaler Kopie werden alle veränderten Dateien zwischen Client und Server übertragen.

ChangeSets werden von CVS nicht unterstützt; es ist also nicht möglich, eine Menge von veränderten Dateien als logische Einheit aufzufassen; Revisionen können jedoch mittels eines Tags gekennzeichnet werden. CVS unterstützt das Anlegen und Verwenden paralleler Entwicklungslinien; Branches werden mittels einer speziellen Versionsnummer dargestellt, die sich aus der Versionsnummer der Ausgangsrevision, erweitert um eine eigene Versionsnummer des jeweiligen Branches, zusammensetzt. Branches werden über einen Tagging Mechanismus verwirklicht, also Funktionalität, die auch zur Markierung eines spezifischen Zustands des Repositories verwendet wird. Beim Ausführen eines Commits gibt es keine Kopie des lokalen Datenbestands,, mit dessen Hilfe die Änderungen festgestellt werden könnten, die seit dem letzten Entnehmen aus dem Repository angestellt wurden. Daher müssen für einen Abgleich des lokalen Datenbestandes und des Repositories die Inhalte aller veränderten Dateien übertragen werden. CVS bietet keine integrierte Unterstützung bei der Auflösung von Konflikten. Bei konkurrierenden Veränderungen innerhalb einer Datei werden beide Veränderungen in die entsprechende Datei eingetragen und markiert, die Auflösung der Konflikte muss nun durch den Benutzer erfolgen. Problematisch ist, dass keine Sicherung vorhanden ist, die das irrtümliche Commit einer solchen Datei mit eingestellten Konflikten in das Repository verhindern würde. Beim Umgang mit binären Dateien müssen diese beim ersten Einstellen in das Repository explizit gekennzeichnet werden, da ansonsten CVS versucht Transformationen durchzuführen, die im Falle einer Binärdatei deren Unlesbarkeit bewirken. Trotz der oben genannten Einschränkungen ist CVS immer noch eines der am häufigsten genutzten Versionskontrollsysteme. Dies lässt sich einerseits durch die große Verbreitung und Unterstützung von CVS erklären; das Programm ist kostenlos auf jeder Plattform erhältlich und wird durch zahlreiche Werkzeuge Dritter unterstützt. Ein weiterer Punkt ist aber auch, dass die meisten der genannten Einschränkungen durch den richtigen Umgang mit dem Programm umgangen werden können, eine wichtige Ausnahme stellen das Fehlen atomarer Commits und von Möglichkeiten zur Versionierung von Verzeichnissen dar. CVS ist in der Lage, bei Operationen auf dem Repository externe Programme aufzurufen. Diese Programme können den Inhalt einer Operation, beispielsweise eines Commits, beeinflussen und eine Operation nötigenfalls abbrechen. Aufgrund seiner weiten Verbreitung ist CVS ausgezeichnet dokumentiert. Es stehen neben zahlreichen Quellen im Web mindestens vier Bücher (siehe Bar/Fogel 2003) zur Verfügung, die sich ausschließlich mit dieser Software beschäfti-

gen, für zumindest zwei dieser Bücher gestatten die Lizenzbedingungen auch einen kostenlosen Bezug über das Internet.

CVSNT

CVSNT[25] entstand aus einer Portierung des CVS Servers auf das Windows Betriebssystem, das Programm wurde dabei um einige Funktionen erweitert, die im eigentlichen CVS nicht vorhanden sind. CVSNT in seiner aktuellen Form kann als eigenständige Variante von CVS aufgefasst werden. Die Software beinhaltet neben dem Server auch einen angepassten Client. Unterstützt werden neben der ursprünglichen Windows Plattform auch verschiedene Unix Varianten und Mac OS X. Gegenüber CVS verfügt CVSNT über die Möglichkeit, eine Datei mit einem exklusiven Lock zu belegen, also die gleichzeitige Bearbeitung einer Datei durch mehrere Benutzer explizit auszuschließen. Weiterhin verfügt CVSNT über eine Unterstützung für binären Diffs, die Fähigkeit, bei Veränderungen an binären Dateien wie Bildern nur die tatsächlichen Veränderungen zu speichern. Das ursprüngliche CVS musste in solchen Fällen für jede Revision der Datei eine separate Kopie anlegen. Andere Erweiterungen betreffen einen vereinfachten Umgang mit Branches und Merges und die Erweiterung der Möglichkeiten zur Authentifikation. Insbesondere bei Verwendung mit dem Windows Betriebssystem ist es möglich, die Vergabe von Rechten für Projekte unter Versionsverwaltung auf der Ebene von Verzeichnissen durchzuführen und weiterreichende Möglichkeiten sowie automatische Aktionen beim Zugriff auf das Repository ausführen zu lassen.

Zusammenfassung:

- CVS ist ein frei erhältliches Versionskontrollsystem auf Client-Server Basis.
- Die unterstützten Betriebssysteme umfassen unter anderem Windows, Unix und Mac OS X.
- CVS wird durch eine Vielzahl externer Software ergänzt, graphische Benutzerschnittstellen und Webschnittstellen stehen zur Verfügung.
- CVS bietet Branch Support und ist in der Lage, Konflikte festzustellen.
- Atomare Checkouts und Commits und Verzeichnisversionierung werden nicht unterstützt.

[25] http://www.cvsnt.com/cvspro/

3.2.2 Visual Source Safe

Visual Source Safe ist das Versionskontrollsystem des Microsoft Developer Studios und mit diesem eng integriert.

Unterstützte Plattformen und Architektur

Visual Source Safe setzt als Plattform das Microsoft-Betriebssystem Windows voraus. Für die Nutzung der Software stehen ein Client für die Kommandozeile und eine graphische Oberfläche zur Verfügung. Eine weitere Zugriffsmöglichkeit besteht durch die direkte Integration in die Programme des Developer Studios. Durch die Veröffentlichung einer Programmierschnittstelle ist die Ansprache des Visual Source Safe Repositories auch durch Programme von anderen Anbietern möglich. So gibt es Implementierungen sowohl aus dem kommerziellen als auch aus dem Open Source Bereich. Visual Source Safe verwendet Clients, die auf ein gemeinsames Repository ohne Nutzung eines zentralen Servers zugreifen; die Kommunikation der Clients geschieht mittels SMB, dem für Windows Dateifreigaben verwendeten Protokoll. Dies erschwert den Einsatz von Visual Source Safe außerhalb einer LAN-Umgebung, da von einer Verwendung von SMB über öffentliche Netze aus Sicherheitsgründen abzuraten ist. Abhilfe für die Einschränkung ist durch die Verwendung eines virtuellen privaten Netzwerks (VPN) oder von Gateway Servern möglich, die speziell für diesen Anwendungsfall von Drittanbietern angeboten werden. Zur Speicherung im Repository kommt ein binäres Datenformat zum Einsatz. Microsoft empfiehlt eine maximale Größe von 5 GByte für diesen Datenbestand, und regelmäßigen Einsatz von mitgelieferten Werkzeugen zur Analyse und Pflege, um das Risiko von Datenverlust auszuschließen.

Eigenschaften und Nutzung

Da ein zentraler Servers fehlt, setzt die Durchführung von automatischen Aktionen beim Zugriff auf das Repository eine entsprechende Konfiguration aller Clients separat voraus. Atomare Commits werden durch Visual Source Safe nicht unterstützt. Die Software bietet Unterstützung für das Umbenennen von Dateien, ebenso gibt es einen Branch und Merge Support. Visual Source Safe unterstützt exklusive Locks auf Dateien. Eine Verwaltung von Rechten ist auf der Ebene von Verzeichnissen möglich Die Dokumentation des Programms wird als durchaus angemessen bezeichnet.

Zusammenfassung:

- Visual Source Safe ist ein kommerzielles Versionskontrollsystem.

- Vom Hersteller wird nur Software für das Windows Betriebssystem angeboten, Dritthersteller bieten kompatible Produkte auch für andere Betriebssysteme.
- Das System basiert auf Clients, die direkt auf ein gemeinsames Repository zugreifen; einen Server gibt es nicht.
- Zur Speicherung der versionierten Daten wird ein binäres Datenformat verwendet.

3.2.3 Arch

Arch[26] ist eine aus dem Open Source Bereich stammende verteilte Versionsverwaltung. Zurzeit befindet sich die Software noch in aktiver Entwicklung, diese Beschreibung erfolgt daher weniger aufgrund des derzeitigen Zustandes des Projekts, als vielmehr aufgrund seines Potentials.

Unterstützte Plattformen

Es gibt mehrere Implementierungen der Arch Architektur, die am weitesten entwickelte Implementierung ist Gnu Arch, auch unter dem Namen TLA[27]bekannt. Diese Version ist die einzige zurzeit weiterentwickelte, hat aber genügend interessierte Nutzer und Programmierer angezogen, um eine weitere Entwicklung zu gewährleisten. Ziel der Entwicklung ist es, eine Alternative zum kommerziell vertriebenen Bitkeeper Versionskontrollsystem darzustellen. Die Gnu Arch Software, im Folgenden kurz Arch genannt, ist zur Zeit nur auf Unix Plattformen nativ lauffähig. Eine Portierung auf Windows ist in nächster Zeit nicht absehbar, so dass eine Nutzung lediglich durch den Einsatz der Cygwin[28] Umgebung, einer Kompatibilitätsschicht für den Einsatz von Unix Programmen unter Windows, möglich ist. Parallel zur Entwicklung der eigentlichen Arch Software gibt es eine Reihe von Open Source Projekten, die grafische Benutzungsoberflächen und Webschnittstellen für Arch bereitstellen. Arch ist in der Lage, zahlreiche Netzwerkprotokolle zu verwenden, zur Zeit werden HTTPS, SFTP und WebDAV unterstützt.

Architektur, Eigenschaften und Nutzung

Ebenso wie Bitkeeper, aber im Gegensatz zu den restlichen hier vorgestellten Programmen handelt es sich bei Arch um ein verteiltes Werkzeug zur Versionskontrolle; es gibt also weder einen dedizierten Server noch ein zentrales

---

[26] http://www.gnu.org/software/gnu-arch/
[27] Tom Lord Arch, benannt nach dem Entwickler dieser Version
[28] http://www.cygwin.com/

Repository. Diese Eigenschaft erlaubt einen sehr freien Umgang und Austausch von Änderungen zwischen den jeweiligen lokalen Dateibeständen. Arch unterstützt Verzeichnisversionierung, also beispielsweise das Umbenennen von Dateien, durch die Vergabe einer eindeutigen Identifikationsnummer für jeden Dateinamen. Zur Speicherung der versionierten Daten wird das Dateisystem verwendet, atomare Commits werden unterstützt. Arch ist zur Zeit noch in einer relativ frühen Entwicklungsphase und erscheint für den produktiven Einsatz noch nicht geeignet. Dieses Urteil beruht weniger auf dem Reifegrad des eigentlichen Versionskontrollsystems als auf der Verfügbarkeit einer adäquaten graphischen Benutzungsoberfläche und einer Verfügbarkeit unter Windows. Ein weiteres Hemmnis bei der Anwendung von Arch ist die Strukturierung des Befehlssatzes, der viele in anderen Versionskontrollsystemen als allein stehende Befehle implementierte Aktionen als Folge von Kommandos darstellt. Dies erfordert eine weitgehende Anpassung von bestehenden Nutzungsgewohnheiten. Erschwert wird die Anwendung von Arch auch durch den Mangel an aktueller Dokumentation. Zusammengefasst stellt Arch ein sehr interessantes System dar, wenn ein verteiltes Versionskontrollsystem erprobt werden soll. Vor einem produktiven Einsatz sollte allerdings die Stabilisierung und weitere Verbreitung abgewartet werden.

Zusammenfassung:

- Arch ist ein Open Source Projekt der GNU Foundation.
- Arch ist ein verteiltes Versionskontrollsystem.
- Es werden eineVielzahl von Netzwerkprotokolle unterstützt.
- Zur Zeit ist eine Nutzung nur auf Unix Betriebssystemen nativ möglich.
- GUI und Webschnittstelle befinden sich in Entwicklung.
- Die Bedienung mittels der Kommandozeile erfordert gegenüber anderen Versionskontrollsystemen eine Umstellung.

3.2.4 Bitkeeper

Bitkeeper[29]ist ein verteiltes Versionskontrollsystem unter kommerzieller Lizenz. Es stellt zurzeit eines von wenigen verteilten Versionskontrollsystemen dar, die in größeren Projekten[30] verwendet werden.

[29] http://www.bitkeeper.com/

[30] Bitkeeper kam über einen längeren Zeitraum zur Pflege des Linux Kernels zum Einsatz.

Unterstützte Plattformen

Bitkeeper unterstützt Windows Betriebssysteme, Mac OS X und Unix Plattformen inklusive Linux. Das Softwarepaket beinhaltet jeweils das eigentliche Versionskontrollsystem und eine graphische sowie eine auf Text basierende Benutzerschnittstelle. Unter Microsoft Windows besteht die Möglichkeit, Bitkeeper in den Windows Explorer zu integrieren. Ebenso ist durch eine Unterstützung des SCC Protokolls eine Einbindung in das Visual Studio möglich. Weitere Bestandteile des Programms sind eine Webschnittstelle und eine Komponente zur Registrierung und Behandlung von Fehlern. Wegen der dezentralen Struktur des Programms beinhaltet jede Installation des Programms eigene Repositories und deren Verwaltungssoftware; einen dedizierten Server gibt es nicht. Die Preise für eine Lizenz werden von dritter Seite auf bis zu 8000$ angegeben, sinken jedoch mit wachsender Anzahl eingekaufter Lizenzen. Neben einem Kauf werden auch zeitlich auf ein Jahr begrenzte Lizenzen vergeben, diese liegen preislich zwischen einem Drittel und einem Viertel des Kaufpreises. Die Bitkeeper Lizenzbedingungen sehen ein obligatorisches Update auf neu erscheinende Versionen und deren jeweilige Lizenzbedingungen vor.

Architektur

Bitkeeper ist ein verteiltes Versionskontrollsystem, somit arbeitet jeder Benutzer auf seinem eigenen lokalen Repository. Die Speicherung der versionierten Text-Dateien erfolgt in Form von SCCS[31] Dateien, ergänzt um Metadaten zur Versionsverwaltung. Zur Sicherung gegen Datenverlust werden bei der Speicherung und beim Datentransfer automatisch Kontrollen auf Korrektheit durchgeführt. Die Kommunikation zwischen Bitkeeper Instanzen kann sowohl über ein lokales Dateisystem als auch über das Netzwerk erfolgen. Bei der Verwendung des Netzwerkes stehen mehrere Protokolle bereit, neben dem proprietären Protokoll BKD finden auch die Protokolle SSH, RSH, SMTP und HTTP Anwendung.

Eigenschaften und Nutzung

Aufgrund der verteilten Struktur von Bitkeeper gestaltet sich die Benutzung der Software anders, als dies bei einem zentralen Repository der Fall wäre. Veränderungen an einer Datei werden nicht in ein zentrales Repository eingestellt, stattdessen verfügt jeder Nutzer über ein eigenes Repository. Das Repository wird als eine Sammlung von ChangeSets betrachtet, wobei die ChangeSets zwischen den Repositories frei ausgetauscht werden können. Eine

[31] http://cssc.sourceforge.net/old-cyclic/sccs.html

Anwendung der verteilten Repositories ist die Etablierung von Hierarchien von Repositories, hierbei werden Änderungen jeweils auf einer Ebene gesammelt, integriert, geprüft und an eine darüber liegende Ebene weitergereicht. Im Zusammenhang mit der Entwicklung des Linux Kernels hat sich die Verwendung einer solchen Hierarchie insofern bewährt, als eine effizientere Integration eingehender Änderungen durch Linus Torvalds, dem Entscheidungsträger für die zu übernehmenden Änderungen, ermöglicht wurde[32], Die Verwendung von Bitkeeper für die Entwicklung der Linux-Kernels wurde im Frühjahr 2005 aus Lizenzgründen eingestellt.

Eine Konsequenz der verteilten Architektur ist die Unterstützung für die Integration von Änderungen, auch wenn es zu längeren Zeiträumen ohne Synchronisation kam. Dies ist bedingt durch die an ChangeSets orientierte Arbeitsweise verteilter Systeme, siehe hierzu auch die Darstellung verteilter Systeme unter 3.1.

Die Unterstützung für Branches und Merges ist in Bitkeeper weit entwickelt; insbesondere verfügt Bitkeeper über einen 3-Way Merge Algorithmus, also über die Möglichkeit, zwei in Konflikt stehende Änderungen mit ihrer gemeinsamen Ursprungsversion in Beziehung zu setzen (siehe hierzu auch Punkt 3.1). ChangeSets, also die Möglichkeit, eine Menge von Änderungen zusammenzufassen, werden ebenso unterstützt wie die Möglichkeit, aus einem ChangeSet unter Berücksichtigung aller vorliegenden Veränderungen einen Zustand des Repositories zu rekonstruieren. Bitkeeper unterstützt atomare Commits und Checkouts ebenso wie Verzeichnisversionierung. Als verteiltes Versionskontrollsystem gestaltet sich die Vergabe von Rechten zum Einstellen von Änderungen anders als bei Client-Server Systemen. Ein Einstellen von Veränderungen erfolgt in Form eines ChangeSets, die Integration dieser ChangeSets in ein anderes Repository kann von beliebigen Bedingungen abhängig gemacht werden. Möglichkeiten sind die Authentifikation und Autorisierung des Einstellenden oder eine erfolgreiche Überprüfung des eingestellten Codes. Bitkeeper bietet die Möglichkeit, Kommentare zur Erläuterung einer Änderung konkret einer Datei zuzuordnen. Bitkeeper ist in der Lage, bei Aktivitäten auf dem Repository beliebige Operationen ausführen zu lassen, diese werden durch Skripte implementiert, die bei Aktivitäten auf einem Repository aufgerufen werden.

Neben der Versionskontrolle bietet Bitkeeper auch Funktionalität zum Bugtracking. Die Nutzung dieses Programmteils ist sowohl mittels der graphischen Benutzerschnittstelle als auch mit einer Webschnittstelle möglich.

---

[32] http://www.linuxworld.com/story/32722_p.htm

Diese erlaubt es, Fehlerbeschreibungen mit einer Beschreibung und Einordnung der Schwere des Fehlers einzustellen, Entwicklern zuzuweisen und in Kontext mit spezifischen Änderungen zu setzen. Die Dokumentation des Programms ist auf der Webseite des Herstellers frei verfügbar und durchaus erschöpfend, weiterhin existiert dort ein archiviertes Forum mit zahlreichen, in der Regel beantworteten, Nutzeranfragen.

Zusammenfassung:

- Bitkeeper ist ein verteiltes, kommerzielles Versionskontrollsystem.
- Unterstützte Plattformen sind Windows, Mac OS X und Unix.
- Neben der Kommandozeile können eine grafische Benutzeroberfläche und eine Webschnittstelle genutzt werden.
- Als verteiltes System eignet sich Bitkeeper gut zur hierarchischen Verwaltung von Repositories.
- Neben der Versionskontrolle beinhaltet Bitkeeper auch Bugtracking Funktionalität.

3.2.5 Perforce

Perforce[33] ist ein kommerzielles Versionskontrollsystem mit Schwerpunkten auf der hohen Geschwindigkeit aller Operationen in Netzwerkumgebungen und einer weit entwickelten Behandlung von Konflikten.

Unterstützte Plattformen

Perforce unterstützt Windows, Mac OS X und eine Reihe von Unix Plattformen, so auch Linux und Solaris. Die graphische Benutzungsoberfläche ist auf allen unterstützten Plattformen verfügbar und bietet auf Benutzerseite Unterstützung für die grundlegende Verwaltung von Veränderungen im Rahmen von Checkin, Checkout und Update. Über diese Funktionen hinaus gibt es Hilfsmittel zur Behebung von Konflikten und zur Verwaltung von Merges und unterschiedliche Darstellungen für Änderungen an den eingestellten Dateien. Neben der grafischen Benutzungsoberfläche besteht die Möglichkeit, sowohl clientseitig als auch zur Steuerung des Servers mit der Kommandozeile zu arbeiten. Eine Webschnittstelle erlaubt es, mittels eines Browsers Veränderungen und deren Erstellungsdatum einzusehen; eine Ansicht der Veränderungen selbst ist jedoch nicht möglich. Perforce kann durch Plugins in Entwicklungsumgebungen, beispielsweise das Microsoft Developer Studio, integriert werden. Ebenso ist eine Verwendung mit Eclipse und Visual Age

[33] http://www.perforce.com/

möglich. Neben der Unterstützung von Entwicklungswerkzeugen bietet Perforce auch eine Integration in die Microsoft Office Produkte und einen FTP-Gateway an, der in erster Linie für den Einsatz mit HTML Editoren gedacht ist. Zusätzlich zu diesen Zugriffsarten existiert eine C++ Programmierschnittstelle zur Nutzung von Perforce und eine Unterstützung der von Microsoft Source Safe verwendeten API. Die Kosten einer Perforce Lizenz betragen 750 $, der Preis nimmt mit steigender Zahl der eingekauften Lizenzen ab.

Architektur

Perforce ist eine Client-Server Architektur; die Kommunikation zwischen Server und Clients geschieht mittels eines eigenen Protokolls, wobei zur Entlastung der Netzwerkverbindung ein spezieller Proxy ohne weitere Kosten angeboten wird. Dieser Proxy soll die Replikation von Repositories bei räumlich getrennten Teams von Entwicklern und einer langsamen Netzwerkverbindung unnötig machen. Zur Absicherung des Datentransfers wird die Verwendung eines SSH Tunnels empfohlen, eine Verschlüsselung durch Perforce selbst findet nicht statt. Die Speicherung der unter Versionskontrolle stehenden Dateien erfolgt in einem RCS kompatiblen Format im Dateisystem, Metadaten zur Kontrolle von Transaktionen werden in einer im Server integrierten Datenbank gespeichert. Zum Abgleich von Veränderungen an Dateien und der Verwaltung einer lokalen Arbeitskopie werden keine Hilfsdaten verwendet, stattdessen werden Veränderungen in der lokalen Arbeitskopie durch den Perforce Client überwacht.

Eigenschaften und Nutzung

Perforce unterstützt alle grundsätzlichen Funktionen eines Versionskontrollsystems für den Einsatz durch mehrere Nutzer über ein Netzwerk. Branches werden durch Perforce als virtuelle Pfade im Dateisystem dargestellt; die Erstellung eines Branches kann hierbei sowohl durch die Kopie eines Verzeichnispfades im Repository erfolgen als auch mittels einer speziellen Semantik zur Behandlung von Branches, was den Benutzer bei der späteren Integration von Veränderungen entlastet. Der Perforce Client überwacht permanent alle Veränderungen der Dateien im lokalen Dateisystem. Dies ermöglicht es, Information über vorliegende Konflikte und anstehende Updates ohne explizites Aufrufen des Versionsverwaltungssystems bereitzustellen[34]. Perforce beinhaltet ein integriertes Defect Tracking System, Perforce Jobs. Mittels dieses Programms ist es möglich, Fehler und deren Beschreibung durch Perforce zu er-

[34] Dies steht im Unterschied zu Systemen wie Subversion und CVS; diese Werkzeuge sind inaktiv bis zur Aktivierung durch den Nutzer.

fassen, verfolgen zu lassen und den Status eines Fehlers zu verändern. Hierbei kann die Behebung eines Fehlers mit einer erfolgten Veränderung assoziiert werden. Perforce unterstützt Atomic Commits und die Vergabe von exklusiven Locks, die Integration von Veränderungen erfolgt mittels eines 3-Way Merge Verfahrens. Die Software kann serverseitig durch Skripte erweitert werden, dies geschieht durch die Definition von Triggern für Aktionen auf dem Repository.

Perforce unterstützt die Versionierung von Verzeichnissen, also das Umbenennen und Verschieben von Dateien unter Beibehaltung der Historie für diese Dateien, allerdings muss hierzu ein manuelles Kopieren und anschließendes Löschen vorgenommen werden. Die Vergabe von Rechten kann auf der Ebene einzelner Dateien erfolgen; hierbei kann festgelegt werden, welche Aktionen ein Nutzer oder ein Rechner auf der jeweiligen Datei ausführen kann.

Die Verwendung von Perforce ist auf der Website des Herstellers ausführlich dokumentiert.

Zusammenfassung:

- Perforce ist ein kommerzielles Versionskontrollsystem mit Client-Server Architektur.
- Windows, Unix und Mac OS X werden unterstützt.
- Neben einem Kommandozeilen Client werden auch eine GUI und eine Webschnittstelle angeboten, Perforce kann mit Programmen zusammenarbeiten, die Microsofts API für Visual Source Safe unterstützen.
- Die Versionierung von Verzeichnissen wird indirekt unterstützt, atomare Commits und Checkouts werden angeboten.
- Die Software bietet 3-Way Merge Unterstützung und fortschrittliche Methoden zur Konfliktbehebung.

3.2.6 Subversion

Subversion[35] ist ein Open Source Projekt mit dem Ziel, bei größtmöglicher Ähnlichkeit zu CVS dessen Fehler zu beheben. Das Projekt besteht seit dem Jahr 2000 und wird durch Collabnet, einen Anbieter für Software zur kooperativen Softwareentwicklung, finanziell unterstützt. Wie auch CVS ist Subversion eine Client-Server Architektur[36] [37].

---

[35] subversion.tigris.org

[36] Ebenso wie bei CVS besteht die Möglichkeit, beim Zugriff auf ein lokales Repository keinen separaten Prozess auf dem Server einzusetzen.

Unterstützte Plattformen

Das eigentliche Subversion Projekt umfasst mehrere Implementierungen eines Servers und einen Client für die Kommandozeile. Die Software ist auf Microsoft Windows, verschiedenen Unix Plattformen und Mac OS X verfügbar. Bei der Entwicklung von Subversion wird auf einen Satz von Programmbibliotheken aus dem Apache Projekt zurückgegriffen; durch die Verwendung dieser Bibliotheken ist die parallele Entwicklung der Software für unterschiedliche Betriebssysteme mit geringem Portierungsaufwand möglich. Die Entwicklung von Software zur Unterstützung und Erweiterung von Subversion, beispielsweise durch graphische Benutzungsoberflächen oder Webschnittstellen, findet ebenfalls in einer Reihe von Open Source Projekten statt.

Architektur

Bei Subversion handelt es sich um eine sehr modular gestaltete Software, für die zurzeit unterschiedliche Implementierungen sowohl des Servers als auch des Repositories vorliegen. Neben der Möglichkeit, auf ein Repository direkt mittels des lokalen Dateisystems zuzugreifen, stehen zwei Implementierungen des Servers zur Verfügung. Die als SVNServe bezeichnete Implementierung verwendet zur Kommunikation zwischen Client und Server ein für die jeweilige Anwendung spezifisches Protokoll, das bei Bedarf durch einen SSH-Tunnel geleitet werden kann[38]. Dieser Server zeichnet sich durch eine einfache Administration und geringen Ressourcenbedarf aus, bietet aber nur eingeschränkte Funktionalität. Die Authentifizierung ist nur auf der Ebene von ganzen Repositories möglich und geschieht mittels einer Datei mit Nutzer- und Passwortdaten, eventuell ergänzt um die Mechanismen, die von SSH zur Absicherung und Authentifizierung bereitgestellt werden. Alternativ zur Verwendung von SVNServe kann als Serverprozess auch ein Modul für den HTTP-Server Apache in der Version 2 verwendet werden. Diese Lösung ist anspruchsvoller in Bezug auf die benötigten Ressourcen und in der Konfiguration, bietet aber eine Reihe von Eigenschaften, die über SVNServe hinausgehen.

So ist es möglich, die Authentifikationsmechanismen des Apache Servers auch für den Zugang auf die Repositories zu verwenden. Weiterhin wird die

---

[37] Mit SVK befindet sich ein verteiltes Versionskontrollsystem in Entwicklung, das Subversion zur Speicherung von Revisionen verwendet.

[38] Die Unterstützung für den SSH-Tunnel ist in die Software integriert und findet für den Nutzer mit Ausnahme der Repository URL weitgehend transparent statt.

Vergabe von Benutzerrechten auf der Ebene von Verzeichnissen unterstützt[39]. Die Kommunikation zwischen dem Subversion Client und dem Subversion Apache Modul geschieht mittels der Protokolle WebDAV und DeltaV. Hierbei handelt es sich um Erweiterungen des HTTP Standards, um Operationen für den schreibenden Zugriff und um Versionierung, ab der Version 1.2 unterstütz Subversion den Zugriff mit beliebigen WebDAV Clientprogrammen Um eine Verbindung abzusichern kann SSL (siehe Viega/Chandra/Messier 2002) verwendet werden. Für die Speicherung von Dateien in Repositories verwendet Subversion ein binäres Datenformat; in Version 1.0 wird ausschließlich die Berkeley DB verwendet, eine Sammlung von Programmbibliotheken, die das effiziente Speichern von Binärdaten und Unterstützung für Transaktionen bietet. Ab der Version 1.1 ist alternativ auch eine Speicherung in Form von Binärdateien im Dateisystem direkt möglich. Mit der aktuellen Version 1.2 ist diese Form der Speicherung der Standard. Die Gründe für diese Entwicklung liegen in erster Linie in der vereinfachten Administration und Pflege des Repositories und im Wunsch, ein von der Berkeley DB unabhängiges Repository zu erhalten[40]. Unabhängig von der Verwendung von Datenbank oder Dateisystem wird bei der Speicherung ein binärer Differenzalgorithmus verwendet, so dass nur die tatsächlichen Unterschiede zwischen Dateien abgespeichert werden müssen. Die Datenspeicherung von Subversion unterstützt virtuelle Kopien von Verzeichnissen und Dateien innerhalb des Repositories. Bei Verwendung dieser Kopien wird nur Speicherplatz für die Unterschiede zwischen Kopie und Original benötigt. Diese Eigenschaft gestattet Subversion die Platz sparende Implementierung von Branches. Subversion unterstützt atomare Commits und stellt den konsistenten Zustand des Repositories somit auch sicher, wenn ein Zugriff auf das Repository vorzeitig beendet wird.

Eigenschaften und Nutzung

Subversion orientiert sich in der Nutzung sehr stark an CVS, Unterschiede gibt es in der Darstellung von Branches und bei der Vergabe von Revisionsnummern. Durch eine Reihe von Erweiterungen geht Subversion über den Funktionsumfang von CVS hinaus, in erster Linie ist hier die Möglichkeit der Verzeichnisversionierung zu nennen.

Subversion vergibt für jede Veränderung im Repository eine Versionsnummer, mittels dieser Versionsnummer sind alle eingestellten Dateien einer Revision eindeutig zu identifizieren. Die vergebene Revisionsnummer lässt sich

---

[39] Die Vergabe der Rechte in einem Verzeichnis kann bis auf die Ebene der erlaubten HTTP/WebDAV/DeltaV Befehle festgelegt werden.

[40] http://svn.collab.net/repos/svn/trunk/notes/fsfs

als ChangeSet interpretieren, da jede innerhalb einer Transaktion durchgeführte Änderung sich eindeutig identifizieren lässt.

Der exklusive Zugang auf Dateien, also die Vergabe von Locks, wird durch Subversion ab der Version 1.2 unterstützt. Eine Datei kann im Normalfall von mehreren Nutzern zur gleichen Zeit bearbeitet werden, die Vergabe eines Locks auf eine Datei kann explizit festgelegt werden.

Subversion fasst Branches als Kopien eines Projektes im Repository auf; ein Branch wird also nicht, wie im Falle von CVS, mittels einer speziellen Versionsnummer oder seines Tags angesprochen. Der Zugriff geschieht, vergleichbar mit einem regulären Dateisystem, mittels eines Pfades. Bei einem Merge werden die Äste über ihren Pfad und ihre Revisionsnummer angesprochen. Das Zusammenführen unterschiedlicher Entwicklungslinien wird also wie ein Update behandelt. Eine Unterstützung von 3-Way Merges oder anderen weitergehenden Methoden zur Konfliktbehebung wird durch Subversion selbst nicht geboten, wird allerdings unterstützt indem bei einem Konflikt neben einer Datei mit markierten Differenzen auch eine Ursprungsdatei bereitgestellt wird.

Subversion beinhaltet keine Sicherung gegen ein unbeabsichtigtes mehrfaches Einspielen der Änderungen eines Astes, generell werden keine Mechanismen angeboten, die Äste auf einer höheren Abstraktionsebene als der erwähnten Verzeichniskopie darzustellen. Verzeichnisversionierung, also das Umbenennen von Dateien und Verzeichnissen, wird durch Subversion unterstützt, eine Einschränkung ist allerdings die intern verwendete Implementierung. Hierbei wird eine Kopie mit dem neuen Namen der Datei erstellt und die ursprüngliche Datei gelöscht. Dieses Vorgehen hat den Nachteil, dass bei einer Umbenennung einer Datei in einem Ast der Bezug zur Datei im anderen Ast verloren geht. Die Unterstützung von Merges und die Auflösung von Konflikten werden ähnlich gehandhabt wie im Falle von CVS: In Dateien, in denen Konflikte bestehen, werden die widersprüchlichen Änderungen entsprechend markiert eingetragen und müssen vom Benutzer aufgelöst werden. Dieser muss also entscheiden, welche Änderungen übernommen werden sollen. Subversion unterscheidet sich hier insofern von CVS als überprüft wird, ob tatsächlich auf Dateien mit Konflikten zugegriffen wurde, bevor diese wieder ins das Repository eingestellt werden können. Dieses Verhalten vermindert das Risiko, Dateien einzustellen, die sich noch in Konflikt befinden. Um unnötige Zugriffe auf das Repository zu vermeiden, beinhaltet die lokale Arbeitskopie alle verwendeten Dateien zweimal, wobei eine Datei sich auf dem Stand der letzten Synchronisierung mit dem Repository befindet und als Referenz für alle bisher angestellten Änderungen dient. Dies gestattet es, bei einem Commit nur tatsächlich erfolgte Veränderungen zu übertragen und somit den Datentransfer über eine eventuelle Netzwerkverbindung auf das Notwen-

dige zu beschränken. Eine interessante Eigenschaft von Subversion sind Properties. Im Rahmen von Subversion ist unter einem Property die Möglichkeit zu verstehen, beliebige Daten unter einem Schlüsselwort zu einer Datei, einem Verzeichnis oder einer Revision abzulegen. Durch Properties kann ein Subversion Repository prinzipiell um beliebige Metadaten erweitert werden. Praktisch werden Properties verwendet, um das Locking von Dateien und die Verwaltung von Dateien, die bei der Versionierung nicht berücksichtigt werden sollen, zu implementieren.

Subversion ist sehr gut dokumentiert, zurzeit gibt es mehrere Bücher die das System zum Thema haben, besonders hervorzuheben ist "Version Control with Subversion“ (siehe Pilato/Collins-Sussman/Fitzpatrick; 2004). Das Buch ist online kostenfrei zu beziehen. Weiterhin verfügt das Kommandozeilen-Werkzeug `svn` über eine ausführliche Hilfsfunktion mit detaillierter Erläuterung jedes Befehls: Ähnliches lässt sich über TortoiseSVN sagen, dieses Programm bietet ebenfalls eine sehr gute Online Dokumentation.

Zusammenfassung:

- Subversion ist ein Open Source Versionskontrollsystem mit dem Ziel, die Fehler in CVS bei größtmöglicher Ähnlichkeit zu vermeiden.
- Unterstützte Plattformen sind Windows, Unix und Mac OS X.
- Graphische Benutzeroberflächen und Webschnittstellen werden durch weitere Open Source Projekte bereitgestellt.
- Die Architektur beruht auf dem Client-Server Prinzip, wobei verschiedene Implementierungen des Servers verwendet werden können.
- Subversion unterstützt atomare Commits und Verzeichnisversionierung.
- Durch Properties können Metadaten für ein Repository vergeben werden.

3.2.7 Eigenschaften der untersuchten Versionskontrollsysteme

Abschließend werden in Tabelle 1 die Eigenschaften aller betrachteten Versionskontrollsysteme dargestellt.

Name	Arch	Bitkeeper	CVS	Subversion	Source Safe	Perforce
Lizenz	GPL	Kommerziell	GPL	Apache	Kommerziell	Kommerziell
Verteilte Architektur	Ja	Ja	Nein	Nein	Nein	Nein
Atomare Commits	Ja	Ja	Nein	Ja	Nein	Ja
Versionierung von Verzeichnissen	Ja	Ja	Nein	Ja	Nein	Ja

ChangeSets	Ja	Ja	Nein	Ja	Nein	Ja
Serverseitige Skripte	Nein	Nein	Ja	Ja	Nein	Ja
Nutzbar über Internet	Ja	Ja	Ja	Ja	Nein	Ja
Webschnittstelle	Ja	Ja	Ja	Ja	Nein	Ja
GUI verfügbar	Ja	Ja	Ja	Ja	Ja	Ja
Kommandozeile	Ja	Ja	Ja	Ja	Ja	Ja
Plugin für Eclipse	Nein	Nein	Ja	Ja	Nein	Ja
Umstellung zu CVS	Hoch	Hoch	Keine	Gering	Hoch	Mittel
Windows Version	Nein	Ja	Ja	Ja	Ja	Ja
Unix Version	Ja	Ja	Ja	Ja	Nein	Ja

Tabelle 1: Eigenschaften unterschiedlicher Versionskontrollsysteme

3.3 Fazit

Anhand der zuvor erfolgten Untersuchung der technischen Eigenschaften unterschiedlicher Versionskontrollsysteme soll nun dargestellt werden, welches System für die Verwendung in der Abteilung FuE ausgewählt wurde. Dabei wird auf die in Kapitel 2 gemachten Ergebnisse zur Verwendung von Versionskontrollsystemen eingegangen werden.

Bei der Befragung der Mitarbeiter der Abteilung FuE stellte sich heraus, dass diese überwiegend mit CVS als Versionskontrollsystem zufrieden waren; lediglich die Möglichkeit einer Verzeichnisversionierung wurde vermisst. Vor diesem Hintergrund und unter Einbezug der Arbeitsabläufe in der Abteilung FuE, bei denen üblicherweise nur kleine Teams gemeinsam an einem Projekt arbeiten, erscheint die Verwendung von verteilten Versionskontrollsystemen nicht sinnvoll.

Die Systeme Arch und Bitkeeper würden somit eine nicht unerhebliche Umstellung von Seiten der Anwender erfordern, ohne dass die potentiellen Vorteile eines verteilten Versionskontrollsystems zum Tragen kämen. Visual Source Safe kann durch den Schwerpunkt auf die Verwendung mit dem Microsoft Developer Studio nicht überzeugen, zumal technische Gründe, wie die fehlende Möglichkeit zur Automatisierung, vor allem aber die verwendete Datenspeicherung und deren oft zitierte Anfälligkeit zusammen mit dem Fehlen von atomaren Commits gegen das System sprechen.

Am ehesten geeignet erscheinen somit Perforce und Subversion, beide Systeme bieten die CVS fehlenden technischen Eigenschaften wie die Versionierung von Verzeichnissen und atomare Commits. Jedes der beiden Programme bietet darüber hinaus eine Vielzahl unterschiedlicher Möglichkeiten zum Zugriff auf das Repository.

Eine Arbeit mittels der Kommandozeile ist ebenso möglich wie die Verwendung einer graphischen Oberfläche, sowohl als allein stehendes Programm, als auch in Form eines Plugins für die Eclipse Entwicklungsumgebung. Das für Subversion verfügbare Plugin bietet hier den großen Vorteil, sich in der Anwendung sehr eng an das CVS Plugin anzulehnen, so dass nutzerseitig nur geringe Umstellungen notwendig sind.

Sowohl für Subversion als auch für Perforce sind Webschnittstellen verfügbar. Der Hauptvorteil von Perforce ist die Vereinigung aller wesentlichen Bestandteile eines Versionskontrollsystems in einem Paket, so sind leistungsfähige Werkzeuge zum Umgang mit Konflikten Bestandteil des Programms; vergleichbare Funktionalität ist bei Subversion allerdings durch externe Projekte verfügbar. Ein weiterer Vorteil von Perforce ist die Integration von Perforce Jobs, also einem Werkzeug zum Bugtracking. Da solche Funktionalität allerdings durch externe Software auch für Subversion verfügbar, ist relativiert sich dieser Vorteil. Der größte Vorteil von Subversion bei einer möglichen Anwendung in der Abteilung FuE ist die enge Orientierung an CVS. Für mit CVS vertraute Anwender ist nur eine geringe Umstellung notwendig. Diese Aussage betrifft sowohl Subversion selbst - hier besteht die wichtigste Änderung in der veränderten Revisionsbenennung - als auch die in Zusammenhang mit Subversion eingesetzten Programme, insbesondere also alternative Benutzerschnittstellen. Besonders die graphische Benutzungsoberfläche TortoiseSVN als auch das für die Entwicklungsumgebung Eclipse verwendete Plugin unterscheiden sich in der Anwendung nur minimal von den entsprechenden Programmversionen für CVS, die in der Abteilung FuE zur Zeit genutzt werden. Eine Unterstützung der Konvertierung bestehender CVS Repositories wird für beide Produkte angeboten, dies geschieht jeweils von Form von Programmen zur Überführung bestehender Repositories. Im Hinblick auf die notwendige Laufzeitumgebung unterscheiden sich Subversion und CVS kaum, in beiden Fällen werden alle benötigten Betriebssysteme und Hardwareplattformen unterstützt. Subversion hat gegenüber Perforce den Vorteil, kostenlos verfügbar zu sein. Da Perforce jedoch mit einem Preis von etwa 700 Euro für eine Lizenz im Vergleich zu anderen Systemen vergleichsweise kostengünstig ist, wiegt dieser Vorteil nicht schwer.

Unter dem Gesichtspunkt der technischen Ausgereiftheit hinterlassen beide Produkte einen guten Eindruck, insbesondere was die Datensicherheit der Repositories und die jeweilige Grundsoftware betrifft. Im Fall von Subversion

wirkten auch alle externen Programme als durchaus für die ihnen zugedachte Aufgabe geeignet, dies wird jedoch noch im Detail zu überprüfen sein.

Ein Punkt, der für die Verwendung von Subversion gegenüber Perforce spricht, ist die Möglichkeit, die serverseitigen Komponenten an die jeweiligen Erfordernisse anzupassen. Durch die Wahlmöglichkeit zwischen dem allein stehenden Subversion Server und der Verwendung mit Apache kann Subversion sehr gut an das vorliegende IT-Umfeld und die jeweiligen Erfordernisse angepasst werden. Ein Vorteil gegenüber Perforce ist hier insbesondere eine weitergehende Möglichkeit der Absicherung mittels SSL. Gegen den Einsatz von Perforce spricht in erster Linie, dass sich dieses in der Darstellung von lokalen Arbeitskopien von CVS und Subversion unterscheidet; Perforce definiert lokale Arbeitskopien von Nutzern als Sichten auf ein oder mehrere zentrale Repositories, was bei der Verwendung des Systems eine gewisse Umstellung erforderlich macht.

Unter Einbeziehung all dieser Punkte erscheint Subversion am ehesten geeignet, ein verwendetes CVS System zu ersetzen. In erster Linie, weil bei Erfüllung aller technischen Erfordernisse größtmögliche Kontinuität auf der Anwenderseite gewährleistet werden kann. Gleichzeitig ist serverseitig eine sehr flexible Anpassung an die jeweiligen Anforderungen möglich. Im folgenden Kapitel soll nun eine weitergehende Evaluation von Subversion als Ersatz für CVS durchgeführt werden. Ziel hierbei ist in erster Linie die Überprüfung von Subversion und der mit Subversion zusammen verwendeten Software unter Betrachtung von Anwendungsszenarien. Im Rahmen dieser Überprüfung und auf dieser basierend wird auch ein Plan für die Migration von CVS nach Subversion erarbeitet.

4 Vorgehen bei der Migration zu Subversion

In den vorangegangenen Kapiteln wurden unterschiedliche Versionskontrollsysteme anhand ihrer technischen Eigenschaften vorgestellt, des Weiteren wurde die Verwendung des Versionskontrollsystems CVS in der Abteilung FuE des IZ vorgestellt. Dabei wurde besonders auf die Erwartungen der Anwender an ein Versionskontrollsystem eingegangen. Es wurde entschieden, dass unter den untersuchten Systemen Subversion am ehesten für den Einsatz in der Abteilung FuE geeignet ist. In diesem Kapitel soll nun untersucht werden, wie bei einer Einführung von Subversion vorgegangen werden kann, welches Vorgehen also bei einer Migration von CVS sinnvoll ist. Ein Aspekt ist auch eine detaillierte Überprüfung der technischen Eignung von Subversion zum Einsatz in der Abteilung.

Insbesondere sollen die folgenden Fragen geklärt werden:

Wie soll Subversion in die bestehende Infrastruktur zur Softwareentwicklung integriert werden? Diese Frage betrifft sowohl die Auswahl und die Konfiguration eines Servers zur Verwaltung des Subversion Repositories, als auch eine Auswahl der Software auf Seite der Benutzer. Unter diesem Punkt soll auch betrachtet werden, welche Erweiterungen - etwa zum automatischen Versand von Mail beim Einstellen einer neuen Datei in das Repository oder zum Zugriff auf das Repository mittels eines Browsers - verwendet werden sollen.

Welche Arbeitsschritte sind notwendig um den Einsatz von Subversion zu ermöglichen? Es muss eine Konfiguration der eingesetzten Software entworfen werden, zum Beispiel die Vergabe von Rechten und die verwendeten Mechanismen, die Benutzern zur Authentifikation angeboten werden. Weiterere Aspekte sind das Vorgehen und die Werkzeuge zur Konversion der Inhalte des CVS Repositories.

Wie soll eine Migration zeitlich und organisatorisch ablaufen? Es muss entschieden werden, in welchem Umfang und Zeitrahmen Subversion in der Abteilung eingeführt werden soll.

4.1 Nutzung und Konfiguration von Subversion

Die Subversion-Architektur lässt sich in mehrere Schichten aufteilen, die in Abbildung 4 dargestellt sind. Einem Benutzer stehen unterschiedliche Programme zum Zugriff auf ein Subversion Repository zur Verfügung. Zur Nutzung der durch Subversion bereitgestellten Dienste wird eine Funktionsbibliothek bereitgestellt, alternativ verwenden einige Programme Aufrufe des `svn` Programms. Diese Bibliothek bietet Dienste zum Umgang mit versionierten Dateien und zur Verwaltung der lokalen Arbeitskopie. Mittels spezieller URLs wird entweder auf Server oder auf das Repository direkt zugegriffen. Unterschiedliche Serverimplementierungen verfolgen unterschiedliche Implementierungsziele, alle leisten eine zentrale Authentifikation und Autorisation und erlauben eine Verschlüsselung der übertragenen Daten. Das Repository bietet nach außen hin ein versioniertes Dateisystem an, die tatsächliche Datenspeicherung erfolgt entweder mittels einer BerkeleyDB Datenbank oder in speziellen Dateien im Dateisystem.

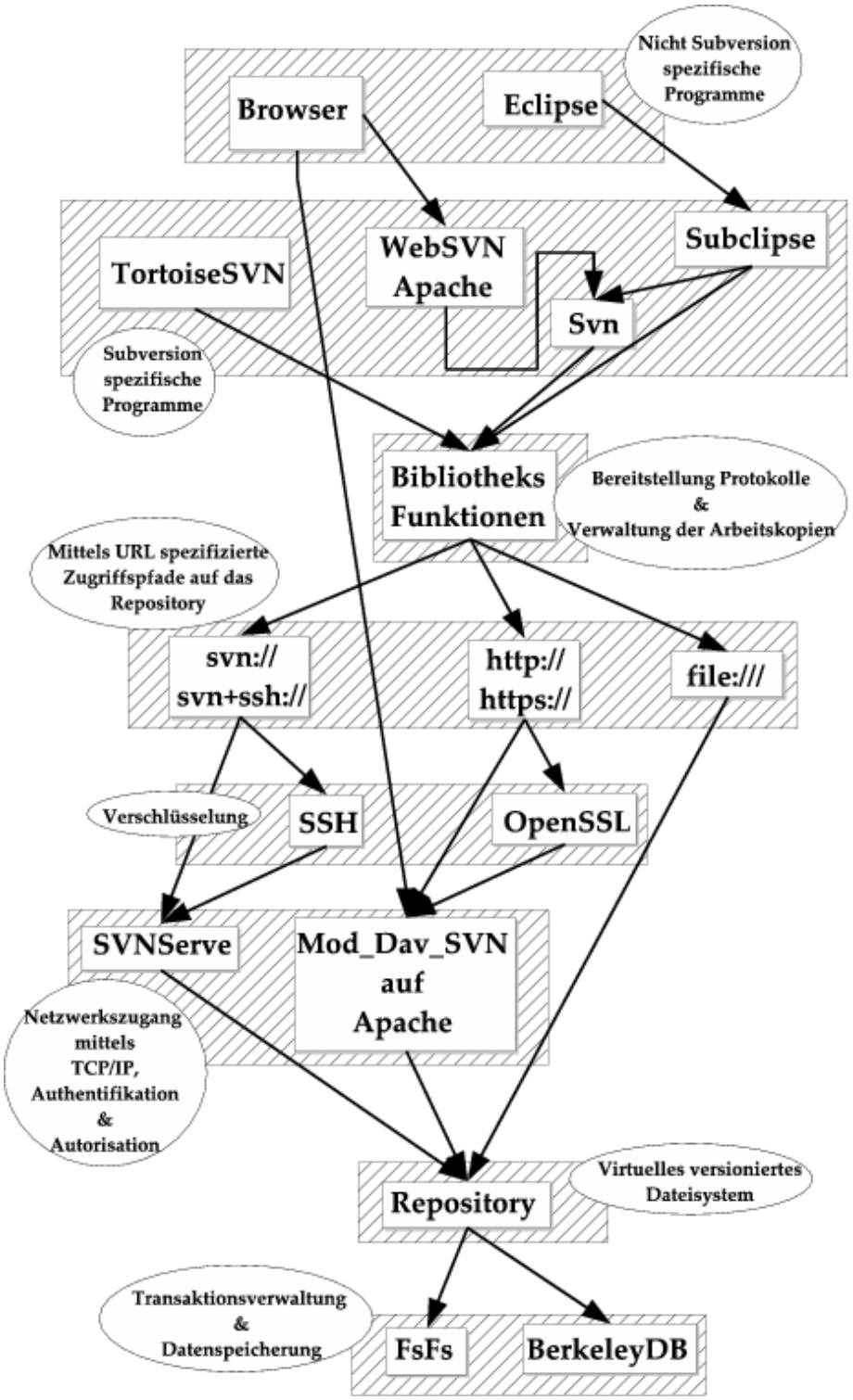

Abbildung 4: Unterschiedliche Schichten der Subversion Architektur

Serverseitige Architektur

Zum aktuellen Zeitpunkt gibt es bei der Auswahl der Subversion Serversoftware zum Einsatz im Netzwerk zwei Alternativen, die Verwendung des SVNServe Programms oder den Einsatz als Apache Modul. Bei der Entscheidung zwischen beiden Programmen muss zwischen benötigtem Funktionsumfang und Komplexität abgewogen werden. SVNServe zeichnet sich durch eine sehr einfache Inbetriebnahme aus, bietet aber nicht alle Fähigkeiten, die durch das Zusammenspiel von Subversion und Apache erzielt werden können. Insbesondere ist keine Vergabe von Rechten auf Ebene von Dateien möglich, die kleinste Einheit bei der Vergabe von Zugriffen ist das Repository. Bei Verwendung des Apache Servers ist eine Vergabe von Rechten auf der Ebene von Verzeichnissen möglich; weiterhin können zur Authentifikation der Nutzer alle Mechanismen verwendet werden, die hier von Apache angeboten werden.

Der Verwendung von Apache in Zusammenhang mit Subversion steht ein höherer Aufwand bei der Einrichtung des Systems entgegen. Hier ist das Aufsetzen beziehungsweise Anpassen eines Apache Servers aus dem 2.0 Entwicklungszweig notwendig, während sich SVNServe mit einem Aufruf der Kommandozeile starten lässt. Der tatsächliche Aufwand für eine Konfiguration von Apache und der zum Betrieb notwendigen Module ist, insbesondere wenn bereits Erfahrung mit der Syntax der Apache Konfigurationsdatei vorliegt, vergleichsweise gering. Der Vorgang beläuft sich im einfachsten Fall auf das Einfügen eines kurzen Konfigurations-Blocks für die zu verwaltenden Subversion Repositories und das Laden der entsprechenden Module (siehe Pilato/Collins/Sussman/Fitzpatrick 2004). Bei Verwendung von Apache wird für die Kommunikation zwischen Subversion-Client und Server ein mit der DeltaV Erweiterung um Versionierung ergänztes WebDAV Protokoll verwendet. Bei WebDAV handelt es sich um ein Protokoll zur Bearbeitung von entfernten Dokumenten. Ab der Version 1.2 ist der Subversion-Server weitestgehend WebDAV kompatibel und kann von allen Programmen mit WebDAV Unterstützung zum Speichern von Dokumenten genutzt werden. Somit können etwa Office-Programme oder Werkzeuge zur Webseitengestaltung ein Subversion Repository zur Speicherung nutzen, wobei automatisch Revisionen der Dateien gespeichert werden.

Vor diesem Hintergrund wurde entschieden, Subversion in Kombination mit Apache als Grundlage zu verwenden. Ausschlaggebend ist hierfür die höhere Flexibilität bei der Vergabe von Nutzerrechten. Auch hat die Verwendung von Apache den Vorteil, dass unabhängig vom tatsächlichen Nutzer nur der

Apache Server Rechte zum Zugriff auf die Berkeley DB benötigt; dies vermeidet einige Fehlerszenarien, die bei Nutzung des SVNServe Prozesses auftreten können[41] [42].

	Apache	SVNServe
Rechtevergabe	Auf Verzeichnisebene	Auf Repositoryebene
Authentifizierung	Alle Apache Mechanismen	Passwort und SSH Authentifikation
Absicherung der Übertragung	SSL	SSH
Protokoll	WebDAV / DeltaV	anwendungsspezifisch
Hardwareansprüche	tendenziell höher	tendenziell niedriger
Einrichtungsaufwand	tendenziell höher	tendenziell geringer

Tabelle 2: Eigenschaften von Apache und SVNServe

Apache Grundkonfiguration

Die Integration von Subversion in den Apache Server erfolgt durch drei Module, dies sind externe Programmbestandteile zur Erweiterung der Funktionalität des Apache Servers. Subversion nutzt das Modul `mod_dav` aus dem Apache Paket zur Bereitstellung grundlegender WebDAV Funktionalität, das Modul `mod_dav_svn` erweitert dies um für Subversion notwendige Funktionen zum Umgang mit den Repositories, während das Modul `mod_authz_svn` optional ist und eine feingranulare Rechtevergabe ermöglicht. Die Einbindung der Module erfolgt wie bei Apache üblich (siehe Coar; Bowen 2003) . Zur Bereitstellung eines Repositories durch Apache wird ein Konfigurationsblock in die Hauptkonfigurationsdatei des Apache Servers eingebunden, dieser Block hat dabei das folgende Format

```
<Location /svn>
  DAV svn
  SVNPath /pfad/zum/repository
</Location>
```

Dieser Block veranlasst das Subversion Modul von Apache dazu, das im lokalen Dateisystem unter dem durch SVNPath beschriebenen Pfad zu findende

[41] http://svnbook.red-bean.com/en/1.1/ch05.html
[42] http://svn.haxx.se/users/archive-2004-10/1291.shtml

Repository unter der URL `http://<serveradresse>/svn` für jeden Subversion Client bereitzustellen.

Konfigurationsdateien und Nutzung der Registry

Die primäre Konfiguration des Subversion Servers erfolgt über einen Block in der Apache Konfigurationsdatei httpd.conf. Aus dieser Datei heraus werden eventuell weitere Dateien referenziert, mittels derer die Vergabe von Rechten und die Zuordnung von Gruppenzugehörigkeiten festgelegt werden. Einen weiteren Bestandteil zur Beeinflussung des Verhaltens des Subversion Servers sind Skripte, die beim Zugriff auf das Repository ausgeführt werden. Das Verhalten der Subversion Client Software lässt sich durch die beiden Konfigurationsdateien servers und config beeinflussen; in der servers Datei werden hierbei Optionen für den Zugriff auf entfernte Netzwerke festgelegt, während in der config Datei das Verhalten des lokalen Clients festgelegt wird. Optionen für die Konfiguration der Netzwerke sind die Festlegung von zu verwendenden Proxy-Servern oder die Spezifizierung der eventuell notwendigen SSL Zertifikate. Die Konfigurationsdatei für das Verhalten der Laufzeitumgebung umfasst Punkte wie das Caching von Passwörtern oder Pfade zu externen Programmen wie Werkzeuge zur Darstellung und Bearbeitung von Differenzen in Textdateien oder Editoren. Sowohl die servers als auch die config Datei wird für den Benutzer automatisch beim ersten Aufruf des Programms generiert. Unter Unix sind diese Dateien unter dem Pfad `.subversion` im Nutzerverzeichnis zu finden.

Unter Windows werden sie für jeden Benutzer in dem Verzeichnis `C:\Dokumente und Einstellungen\NUTZERNAME\Anwendungsdaten\Subversion` abgelegt. Alternativ ist auch eine Konfiguration über die Windows Registry möglich.

Authentifikation und Autorisierung

Zur Authentifikation von Benutzern können alle Mechanismen verwendet werden, die von Apache selbst bereitgestellt oder durch den Einsatz von Modulen hinzugefügt werden können. So stehen verschiedene Methoden zur Verfügung, um eine Authentifikation über eigens vergebene Passwörter zu realisierten, für deren Speicherung von Textdateien bis relationalen Datenbanken ein weites Feld an Möglichkeiten zur Verfügung steht, (siehe Laurie/Laurie2004 und Coar/Bowen 2003). Auch ist eine Authentifikation über die Nutzerverwaltung des jeweiligen Betriebssystems oder Netzwerkes möglich. Als Grundlage kann hier eine Windows Authentifizierung mittels Kerbe-

ros[43] [44] ebenso dienen, wie eine bestehende Unix Shadow Datei[45]. Weitere Möglichkeiten für die Nutzung einer netzwerkweiten Authentifikation nutzen LDAP Verzeichnisse (siehe Carter 2003) oder die Kerberos Infrastruktur. (siehe Garman 2003) ,

Eine Autorisierung oder Vergabe von Rechten ist bei Einsatz des im Subversion Paket standardmäßig enthaltenen Moduls `mod_authz_svn` auf Verzeichnisebene möglich, ohne Verwendung des Moduls ist lediglich eine Vergabe von Rechten auf der Ebene von Repositories möglich. Die Definition von Rechten erfolgt durch eine externe Datei, deren Position im Konfigurationsblock des Apache Servers festgelegt wird. In dieser Datei können für Repositories und in ihnen angelegte Pfade Rechte für einen Nutzer oder für Gruppen von Nutzern festgelegt werden. Hierbei wird so vorgegangen, dass ein Nutzer die Rechte, die er in einem Verzeichnis - eventuell auch dem gesamten Repository - inne hat, für den Zugriff auf Unterverzeichnisse dieses Verzeichnisses erbt. Alternativ können Rechte für Unterverzeichnisse aber auch explizit gesetzt werden, um die Zugriffmöglichkeiten eines Nutzers zu erweitern oder zu beschränken. Zugriffsrechte werden auf drei Ebenen vergeben, ein Benutzer kann Leserechte oder Schreibrechte auf ein Verzeichnis haben oder auch vom Zugriff auf ein Verzeichnis gänzlich ausgeschlossen werden. Der folgende Ausschnitt aus einer Konfigurationsdatei definiert zwei Gruppen, Entwickler und Tester. Entwickler haben vollständigen Zugriff auf das Repository `repo` sowohl in dem Verzeichnis `programm` als auch in dem Unterverzeichnis `bibliothek`. Tester haben lesenden Zugriff auf das Verzeichnis `programm`, jedoch keinen Zugriff auf das Unterverzeichnis `bibliothek`. Der Nutzer David hat nur lesende Rechte im Verzeichnis `programm`, aber Schreibrechte im Verzeichnis `bibliothek`. Der Nutzer Mike hat nur Rechte im Verzeichnis `bibliothek`.

```
[groups]
Entwickler = tim, tom, thorsten
Tester = tim, andi, albert

[repo:/programm/]
@Entwickler = rw
@Tester = r
david=r

[repo:/programm/bibliothek/]
```

[43] http://modauthkerb.sourceforge.net/
[44] http://support.microsoft.com/?kbid=555092
[45] http://mod-auth-shadow.sourceforge.net/

```
@Entwickler = rw
@Tester =
david=rw
mike=rw
```

Eine weitere Möglichkeit ist es, den Zugriff auf Repositories oder auf Teile von diesen anonym zu gestatten, also ohne die Nutzung jeglicher Mechanismen zur Authentifikation; hierbei kommt die Satisfy[46] Direktive der Apache Konfigurationsdatei zum Einsatz. Dies gestattet es beispielsweise, Leserechte ohne jede Authentifikation zu vergeben.

Absicherung mittels SSL

Die Verwendung von SSL[47] erlaubt sowohl die Absicherung der Datenübertragung zwischen einem Subversion Server und einem Client als auch eine Authentifikation von Nutzern. Dies wird durch den Austausch von Zertifikaten zwischen Server und Client und die Verschlüsselung sämtlicher Nachrichten erreicht. Die Verwendung von Zertifikaten stellt sicher, dass sowohl Client als auch Server diejenigen Parteien sind, als die sie sich ausgeben: ein so genannter Man-in-the-middle Angriff[48] wird auf diese Weise prinzipiell ausgeschlossen. Zertifikate können sowohl von Zertifizierungsstellen bezogen als auch selbst erstellt werden. Der Vorgang dabei ist, dass ein kryptographischer Schlüssel erzeugt wird; diesem Schlüssel nun werden Informationen über die zugrunde liegende Partei (z.B. geographischer Ursprung, Personen oder Name einer übergeordneten Organisation) hinzugefügt. Im Anschluss wird der Schlüssel, nach einer Überprüfung der enthaltenen Informationen, von einer Zertifizierungsstelle kryptographisch signiert. Ein signierter Schlüssel ist damit beglaubigt; es wird also ausgedrückt, dass die Angaben, die mit dem Schlüssel zusammen verteilt werden, durch die Zertifizierungsstelle überprüft wurden. Für die Verwendung von Subversion soll im Folgenden davon ausgegangen werden, dass keine externe Zertifizierungsstelle einbezogen wird.

Zur Verwendung von SSL wird eine Version des Apache Servers mit SSL Unterstützung benötigt; wegen der weltweit stark unterschiedlichen Rechtslage in Bezug auf die Nutzung und die Ein- beziehungsweise Ausfuhr von kryptographischen Produkten wird diese Variante des Servers nicht in binärer Form von der Apache Foundation selbst angeboten. Bei der Verwendung ei-

---

[46] http://httpd.apache.org/docs/howto/auth.html
[47] http://www.openssl.org/
[48] http://www3.tsl.uu.se/ micke/ssl_links.html

nes Betriebssystems aus der Unix-Familie ist eine Übersetzung der entsprechenden Quellcodes aber in der Regel unproblematisch. Bei Einsatz eines Betriebssystems aus dem Microsoft Windows Umfeld können installationsfertige Pakete von mehreren inoffiziellen Quellen bezogen werden. Die Einrichtung eines SSL-fähigen Apache Servers ist in beiden Fällen in zahlreichen Dokumenten dargestellt (siehe Laurie;Laurie;2004 und Coar;Bowen 2003) . Um SSL mit Subversion einsetzen zu können, müssen serverseitig ein kryptographischer Schlüssel und ein Zertifikat für diesen Schlüssel vorliegen, der Client benötigt ebenfalls ein Zertifikat; um seine Identität zu bestätigen. Die Position der serverseitigen Dateien für Apache wird in der `httpd.conf` Datei abgelegt, die Konfiguration unterscheidet sich hier nicht von der Konfiguration beim Angebot einer Website mittels SSL. Auf der Seite des Clients muss in der Konfigurationsdatei `server` (siehe hierzu auch Punkt 4.2) der Pfad des zu verwendenden clientseitigen Zertifikates angegeben werden, ebenso ist es bei Verwendung von selbst erstellten Zertifikaten ratsam, hier einen Verweis auf die verwendete Identifikationsdatei der erstellten Zertifizierungsstelle zu hinterlegen. Dieses Vorgehen ermöglicht die Verwendung des selbst erstellten serverseitigen Zertifikates ohne eine weitere Rückfrage an einen Nutzer bei der Verbindungsaufnahme. Eine solche Rückfrage erfolgt ansonsten immer dann, wenn die angegebene Zertifizierungsstelle eines Zertifikats unbekannt ist, wie es bei selbst erstellten Zertifikaten zwangsläufig der Fall ist.

Bei entsprechender Konfiguration geschieht die Verwendung von SSL durch Subversion also nahezu transparent und bietet einen hohen Zugewinn an Sicherheit bei der Authentifikation und Übertragung von Daten.

Zugriff mittels WebSVN

Im Zusammenwirken mit dem Apache Webserver bietet Subversion bereits standardmäßig eine einfache Webschnittstelle auf die Inhalte des Repositories, diese beschränkt sich allerdings lediglich auf eine Ansicht der letzten eingestellten Revision. Es existieren eine Reihe unterschiedlicher Projekte,[49] [50] [51] die es gestatten, mittels eines Browsers lesend auf die Inhalte eines Repositories zuzugreifen; hier soll stellvertretend WebSVN vorgestellt werden. Dieses Projekt zeichnet sich gegenüber anderen Projekten in erster Linie durch die hohe Anzahl möglicher Einstellungen und Optionen bei der Konfiguration aus. WebSVN[52] erlaubt das Einsehen von beliebigen Revisionen in-

[49] http://viewcvs.sourceforge.net/
[50] http://viewsvn.berlios.de/
[51] http://freshmeat.net/projects/svnweb/
[52] http://WebSVN.tigris.org/

nerhalb eines Subversion Repositories und der jeweils mit diesen Revisionen abgelegten Kommentare. Weiterhin ist es möglich, sich die Differenzen zwischen unterschiedlichen Revisionen anzeigen zu lassen. Eine besonders interessante Darstellung zeigt die Veränderungen einer Datei mit der jeweils letzten Revision an, die in der Zeile verändert worden ist. Neben diesen Darstellungen bietet WebSVN auch die Möglichkeit, aus einer Revision jeweils ein Paket mit Quellcode zu generieren oder für eine Datei oder ein Verzeichnis einen RSS-Feed (siehe Hammersley 2003) zu abonnieren, durch den Veränderungen am Repository verfolgt werden können.

WebSVN ist in Form einer Reihe von PHP Skripten[53] implementiert, die Installation beschränkt sich also im Wesentlichen auf das Aufsetzen des PHP Interpreters und Apache Moduls[54] und das Kopieren der Skripte an eine entsprechende Stelle in Apaches `htdocs` Verzeichnis. Bei der Konfiguration und Installation der Software die im WebSVN Dokumentationsverzeichnis ausführlich beschrieben ist, werden die Optionen ausgewählt, die durch die Webschnittstelle angeboten werden sollen. Weiterhin werden hier die Pfade im Dateisystem festgelegt, unter denen das Subversion Repository zu finden ist. Ein wichtiger Punkt bei der Konfiguration des Programms ist das Setzen von Pfaden auf Hilfsprogramme, verwendet werden unter anderem Sed, Diff, Gzip, Tar und enscript. Diese Programme sind auf den meisten Unix Installationen im Lieferumfang enthalten, unter Windows können sie über die Webseite des GnuWin32 Projektes[55] bezogen werden.

Die Authentifikation und Autorisierung für die durch WebSVN bereitgestellten Inhalte wird durch den Apache Server geleistet und muss hier entsprechend konfiguriert werden. Besonderheiten in Bezug auf Subversion gibt es hier keine. Es ist noch anzumerken, dass WebSVN auf ein Subversion Repository zugreift, indem es Kommandozeilen-Werkzeuge aufruft. Ein eventuell vorhandener Subversion Serverprozess wird also nicht benutzt.

Datensicherung des Repositories

Subversion verwendet zur Speicherung seines versionierten Datenbestandes ein Binärformat und das Programm BerkeleyDB[56] zur Verwaltung der Daten[57]

---

[53] http://www.php.net/

[54] WebSVN kann als PHP Skript auch mit anderen Webservern verwendet werden. Da ein Test nur unter Apache stattfand, soll dies hier aber nicht weiter berücksichtigt werden.

[55] http://gnuwin32.sourceforge.net/

[56] http://www.sleepycat.com/

Es existieren unterschiedliche Methoden zur Datensicherung, die einfachste Methode ist eine Replikation der gesamten Repositories. Dies ist durch den Befehl `svnadmin hotcopy` möglich. Wie der Name bereits andeutet, kann der Befehl jederzeit im laufenden Betrieb erfolgen und erstellt eine vollständige Kopie der BerkeleyDB Dateien des Repositories. Eine Alternative zur Replikation der BerkeleyDB im Rahmen eines Backups ist die Durchführung eines Dumps des Repositories, also die Extraktion der eingestellten Daten. Dieser Vorgang wird durch den Befehl `svnadmin dump` durchgeführt, ebenso wie beim oben beschriebenen `svnadmin hotcopy` ist eine Durchführung im laufenden Betrieb möglich. Für einen Dump wird ein spezielles Textformat verwendet. Eine Dumpdatei enthält standardmäßig alle Revisionen des Repositories, kann aber auch auf einzelne Revisionen beschränkt werden oder inkrementell erstellt werden[58].

Das verwendete Dumpformat ist im Gegensatz zu den Binärdateien plattformunabhängig, somit können auf einem Rechner erstellte Dumpdateien auf einer anderen Rechnerplattform wieder hergestellt werden.

Die Sicherung eines Subversion Repositories wird also durch mehrere Werkzeuge unterstützt, wobei der Befehl `svnadmin dump` durch eine Vielzahl von Optionen eine genaue Anpassung an die jeweiligen Anforderungen erlaubt. Sehr von Vorteil ist, dass eine Datensicherung im laufenden Betrieb möglich ist.

Automatisierung durch Hook-Skripte

Das Verhalten des Subversion Servers kann durch so genannte Hookskripte weiter angepasst werden. Es handelt sich dabei um Programme, die bei bestimmten Aktionen durch den Server selbstständig ausgeführt werden. Ein gängiges Beispiel ist der Versand einer Mail bei erfolgten Commits.

Die Bezeichnung Hookskripte ist insofern irreführend, als es sich um beliebige ausführbare Programme handeln kann; jedoch ist der Begriff Skript in den Quellen so verbreitet, dass er auch hier beibehalten werden soll. Diese Programme werden jeweils mit einer Reihe von Parametern aufgerufen, die die zugehörige Aktion detailliert beschreiben. Die Skripte können beliebige Akti-

---

[57] Mit der Version 1.1 von Subversion wird eine Speicherung der Daten nativ im Dateisystem ohne Verwendung der Berkeley DB als Alternative eingeführt. siehe auch http://svn.collab.net/repos/svn/trunk/notes/fsfs

[58] Im Fall eines inkrementellen Dumps werden nicht komplette Revisionen in die Ausgabedatei geschrieben, sondern lediglich die Veränderungen gegenüber der ersten Datei im Dump.

onen auf dem Server ausführen und durch ihren Rückgabewert gegebenenfalls den Abbruch der auslösenden Aktion veranlassen.

Im Einzelnen können folgende Skripte definiert werden:

- start-commit

Dieser Skript stellt in erster Linie eine weitere Möglichkeit der Autorisierung von Nutzern dar. Dem aufgerufenen Programm wird der Pfad übergeben auf den ein Commit erfolgen soll, ebenso der Nutzername, der den Commit durchführen möchte; Durch einen entsprechenden Rückgabewert kann die Durchführung des Commit an dieser Stelle unterbunden werden, bevor durch den Server überhaupt Datenstrukturen zur Verwaltung des Commits erzeugt worden sind.

- pre-commit

Dieser Skript wird ausgeführt, wenn bereits eine Transaktion erstellt wurde, diese aber noch nicht endgültig in das Repository eingestellt wurde. Der Skript erhält als Parameter den Pfad des Repositories und den Namen der Transaktion. Dieser Name kann verwendet werden; um etwa durch die Verwendung von svnlook Einblick in die Änderungen zu nehmen, die mit dem Commit eingestellt werden sollen. Durch den Rückgabewert des Skriptes kann der Commitvorgang unterbrochen werden, wenn die Inhalte des Commits als nicht akzeptabel betrachtet werden.

- post-commit

Das für diesen Fall definierte Programm erhält als Übergabewerte einen Repository Pfad und die Revisionsnummer dieses Commits. Dies erlaubt beispielsweise den Versand von Benachrichtigungen oder das Durchführen von Backups.

- pre-revprop-change

Dieses Programm wird aufgerufen wenn ein Subversion Property verändert werden soll. Properties sind Name-Wert Paare, die einem Subversion Objekt, beispielsweise einer Datei, zugeordnet werden können. Genutzt werden diese Properties beispielsweise zur Speicherung von dem Nutzer vergebenen Kommentaren zu einer Revision. Im Gegensatz zu Dateien und Verzeichnissen sind Properties nicht versioniert, bei einer Veränderung geht also der alte Zustand verloren. Dieser Skript übergibt vier Parameter:

  - Die Kennung des Nutzers der die Veränderung durchführen möchte.
  - Den Pfad des Repositories
  - Die Revisionsnummer
  - Den Namen des zu ändernden Properties.

Durch einen entsprechenden Rückgabewert kann eine Veränderung des Properties unterbunden werden. Um unbeabsichtigte Veränderungen der nicht versionierten Properties zu unterbinden, ist das Vorhandensein eines pre-revprop-change Skriptes Voraussetzung, um überhaupt Veränderungen durchzuführen.

- post-revprop-change

Dieser Skript entspricht pre-revprop-change, wird jedoch nach einer erfolgten Änderung eines Properties ausgeführt und dient in erster Linie dazu, eine Benachrichtigung mit dem neuen Wert des Properties zu versenden.

- pre-lock und pre-unlock

Diese Hookskripte sind ab der Version 1.2 der Software verfügbar und dienen dazu, das Setzen und Aufheben von Locks auf Dateien den Anforderungen des jeweiligen Repositories anzupassen. Durch einen entsprechenden Rückgabewert kann die Verwendung von Lock generell unterbunden werden.

- post-lock und post-unlock

Diese Skripte dienen der Protokollierung und bieten die Möglichkeit, Benachrichtigungsemails zu versenden, wenn sich der Lockingstatus einer Datei ändert.

Clientseitige Architektur

Aus der großen Anzahl der für Subversion verfügbaren Clientprogramme sollen hier drei vorgestellt werden, die für die Verwendung von Subversion von besonderer Bedeutung sind.

Verwendung der Kommandozeile

Ein Kommandozeilen-Client ist im Subversion-Paket enthalten, somit kann das Vorhandensein dieses Werkzeugs bei jeder Installation als gegeben betrachtet werden. Die Arbeit mittels der Kommandozeile wird in erster Linie mit zwei Programmen durchgeführt:

Der Befehl `svn` ist für die eigentliche Versionsverwaltung zuständig (also für die Synchronisation mit dem zentralen Repository), das Hinzufügen von Dateien unter Versionskontrolle und den Vergleich zwischen Revisionen. Auch wird dieses Programm verwendet, um die durch Subversion angebotenen Properties zu editieren beziehungsweise zu setzen. Mittels `svn` können alle von Subversion angebotenen Zugriffsmethoden auf Repositories genutzt werden. Das Verhalten des Werkzeugs lässt sich weitgehend an Nutzerwünsche anpassen. So ist beispielsweise eine Speicherung von Passwörtern für den Zugriff auf das Repository möglich oder die Definition von Dateien, die durch das Versionskontrollsystem ignoriert werden können. Die Nutzung des Befehls `svn` erfolgt nach dem Muster `svn <Befehl> <Optionen>`; durch den

Aufruf `svn help` wird eine Übersicht der verfügbaren Befehle angezeigt, `svn help <Befehl>` liefert eine umfangreiche Darstellung des Befehls und seiner Optionen und Parameter.

Der Befehl `svnadmin` dient der Verwaltung von Subversion-Repositories; im Gegensatz zu `svn` muss dieser Befehl auf dem Rechner ausgeführt werden, auf dem sich das entsprechende Repository befindet. Ebenso wie der `svn` Befehl vereint `svnadmin` eine Reihe unterschiedlicher Funktionen, der Aufruf eines Kommandos erfolgt auch hier in der Form `svnadmin <Befehl>`.

Integration in Eclipse mittels Subclipse

Subclipse ist die zur Zeit am weitesten entwickelte Implementierung einer Integration von Subversion in Eclipse, ein weiteres Projekt unter dem Namen Svn4Eclipse befindet sich zur Zeit noch in der Entwurfsphase. Subclipse[59] ist als Plugin für Eclipse[60] implementiert und kann in Eclipse ab der Version 3.0 über den integrierten Update und Erweiterungsmechanismus bezogen werden. Subclipse nutzt dieselben Ansichten und Dialoge des standardmäßig in Eclipse enthaltenen CVS Plugins; für Benutzer sind also keine tief greifenden Umstellungen notwendig. Nach einer Installation des Subclipse Plugins wird die Eclipse Arbeitsumgebung im Wesentlichen in zwei Punkten erweitert:

Es werden neue Sichten und Perspektiven bereitgestellt, die den CVS Dialogen zur Ansicht von Revisionen von Dateien und den zugrunde liegenden Repositories entsprechen, aber für den Umgang mit Subversion angepasst sind.

Das Kontextmenü von Dateien, Verzeichnissen und Projekten wird erweitert; hier besteht nun die Möglichkeit, die entsprechenden Ressourcen mittels Subversion unter Versionskontrolle zu stellen.

Der Zugriff auf ein Subversion Repository kann auf zwei unterschiedliche Arten erfolgen: Standardmäßig verwendet Subclipse im Programm integrierte Subversion Bibliotheksfunktionen, die mit Hilfe des Java Native Interfaces angesprochen werden; alternativ kann Subclipse auch eine bestehende Subversion Installation nutzen, indem Aktionen an das `svn` Kommandozeilen-Werkzeug delegiert werden. Dieses Vorgehen macht Subclipse weniger abhängig von der verwendeten Subversion Version, ist allerdings weniger performant als die Verwendung der eigenen Bibliothek und zurzeit auch weniger gut getestet.

---

[59] http://subclipse.tigris.org/
[60] http://www.eclipse.org/

Verwendung von TortoiseSVN

TortoiseSVN ist ein Programm, um aus Windows Betriebssystemen heraus auf Subversion Repositories zugreifen zu können. Dabei bindet sich die Software als Erweiterung des Windows Explorers ein. Verzeichnisse und Dateien unter Versionskontrolle werden durch Symbole kenntlich gemacht, über Kontextmenüs können Aktionen der Versionsverwaltung durchgeführt werden. TortoiseSVN bietet zahlreiche Möglichkeiten zum Umgang mit dem versionierten Datenbestand, so lassen sich Differenzen und Kommentare zu Revisionen einsehen. Neben der Verwendung der eigenen Programmteile bietet TortoiseSVN auch die Möglichkeit, externe Werkzeuge zur Durchführung von Merges oder zur Darstellung von Diffs zu definieren.

4.1.1 Abschließende Anmerkungen zur Architektur

Mit der hier dargestellten Architektur und der verwendeten Client Software lassen sich alle Anforderungen erfüllen, die sich in der Abteilung FuE stellen. Insbesondere können alle Funktionen, die aus der bisherigen Anwendung von CVS aus den Entwicklungswerkzeugen heraus bekannt sind, weiterhin angeboten werden. Serverseitig ist mit der Verwendung des Apache Server und unter Einbezug von SSL eine feingranulare Rechtevergabe und Absicherung von Datenverbindungen möglich. Dies ermöglicht es zum Beispiel, studentischen Mitarbeitern selektiv den Zugriff auf die von ihnen benötigten Projektteile freizugeben - wenn dies erwünscht ist, auch von außerhalb des lokalen Netzwerks. Mit WebSVN ergeben sich weitere Möglichkeiten zum Zugriff auf das Repository. Da der Zugriff keine spezielle Clientsoftware erfordert, sondern mit jedem Browser möglich ist, sind die Anforderungen für die Verwendung dieses Werkzeugs sehr niedrig; auch ist ein Zugriff möglich wenn lediglich ein Browser zur Verfügung steht, beispielsweise während einer Tagung.

Nachdem im vorangehenden Text die Bestandteile der Architektur dargestellt worden sind wie sie in der Abteilung FuE zum Einsatz kommen, werden im Folgenden die konkreten Erfahrungen dokumentiert, die bei der Erprobung der Software gemacht wurden.

4.2 Konversion der Inhalte des CVS Repositories zur Nutzung mit Subversion

In diesem Kapitel wird die Überführung von CVS Repositories in Subversion Repositories beschrieben. Dazu wird dargestellt welche unterschiedlichen Arten der Konvertierung grundsätzlich denkbar und für welche Fälle sie jeweils geeignet sind.

4.2.1 Arten der Konversion von Repositories

Ziel einer Konvertierung zwischen den unterschiedlichen Repositories von Versionskontrollsystemen ist es, den bisher von einem System verwalteten Datenbestand in ein anderes System zu übernehmen, wobei die Historie zumindest in Teilen erhalten bleiben soll. Im Vorfeld einer Konvertierung muss entschieden werden, welche Projekte übernommen werden und inwieweit dabei die Historie der jeweiligen Projekte erhalten bleiben soll. Bei der Übernahme der Historie gibt es unterschiedliche Ansätze, die sich jeweils in ihrem Umfang unterscheiden; generell handelt es sich um ein Abwägen zwischen Vollständigkeit der Konvertierung und deren Komplexität.

Übernahme nur der aktuellsten Revision
Dies ist keine Konversion im eigentlichen Sinne; der Datenbestand eines Repositories des bisher verwendeten Versionskontrollsystems wird lediglich exportiert und als Quelldateien in das neue Versionskontrollsystem importiert. Ein Vorteil dieser Methode ist seine Einfachheit: es werden keinerlei Werkzeuge zur Konversion benötigt. Der Nachteil ist ein Verlust der Historie und Metadaten der eingestellten Dateien.

- Übernahme der Hauptentwicklungslinie
  Bei diesem Vorgehen wird nur der zentrale Ast der Entwicklung übernommen, Branches werden ignoriert. Dieses Vorgehen erzeugt ein einfach strukturiertes Repository, allerdings können Branches nicht nachvollzogen werden.
- Teilweise Übernahme einzelner Branches
  Hier werden nur explizit ausgewählte Branches in das neue Repository übernommen; das so entstehende Repository ist also genau an die aktuellen Bedürfnisse der Entwickler angepasst. Dieser Vorteil wird durch eine erhöhte Komplexität der Konversion erkauft, bietet sich aber an, um nicht mehr aktive Branches gar nicht erst in das neue Repository einfließen zu lassen.
- Vollständige Übernahme der gesamten Historie
  Es wird das gesamte Repository übernommen. Ein Nachteil ist der erhöhte Bedarf an Platz und Rechenzeit den dieses Vorgehen fordert, schwerer kann aber wiegen, dass das entstehende Repository mehr Informationen erhält als aktuell verwendet werden und damit weniger übersichtlich ist, als wünschenswert wäre. Größter Vorteil ist demgegenüber die Gewissheit, keinerlei Informationen zu verlieren.

Für die Abteilung FuE wurde entschieden, die Konvertierung der Projekte jeweils auf den Hauptentwicklungsast zu beschränken, Branches also zu ignorieren. Dieses Vorgehen bietet sich an, da Branches nur in einem geringen Umfang genutzt wurden. Weiterhin gibt es spezifische technische Gründe die

für dieses Vorgehen sprechen und auf die im Weiteren noch eingegangen wird.

4.2.2 Speicherung versionierter Daten durch CVS

Ein CVS-Repository lässt sich in zwei Bereiche unterteilen, die Informationen zur Verwaltung des Repositories und die versionierten Benutzerdaten. Zuerst soll nun auf das Datenformat eingegangen werden, das zur Speicherung der versionierten Daten zum Einsatz kommt. Hierbei handelt es sich um RCS Dateien, also das Format jenes Programms, als dessen Erweiterung CVS ursprünglich entwickelt wurde.

Das RCS-Dateiformat

Bei einer RCS Datei handelt es sich um eine Textdatei, die den versionierten Datenbestand jeweils einer Datei repräsentiert. Der Name einer RCS-Datei im Repository entspricht dem Namen der nicht versionierten Datei mit dem Anhang `,v`. Eine RCS ist in drei Bereiche aufgeteilt.

Der erste Bereich ist der Headerbereich. In diesem werden die aktuellste Revisionsnummer und Informationen über Tags und Branches abgelegt, die für diese Datei definiert wurden.

Daran schließt eine Folge von Einträgen für jede Revisionsnummer der Datei an. Innerhalb des Blocks für jede Revisionsnummer wird auf Branches verwiesen, die von dieser Nummer ausgehen; ebenso sind hier der Autor und das Datum der Revision aufgeführt.

Den letzten und in der Regel größten Teil einer Datei machen die eigentlichen versionierten Inhalte aus. Diese liegen in einem Format vor, bei dem zuerst die aktuellste Version einer Datei unter der höchsten Revisionsnummer inklusive der zugehörigen Kommentare abgelegt ist. Alle anderen Revisionen folgen in der Reihenfolge der sinkenden Revisionsnummer. Die Inhalte der jeweiligen Revision sind nicht komplett in der Datei abgelegt, sondern jeweils als Differenz zur nächst höheren Revision. Neben dieser Differenz sind in der Datei auch die Kommentare zu der jeweiligen Revision abgelegt.

Die Verwaltungsdaten im CVS Repository

Neben den eigentlichen RCS Dateien mit den zur Versionierung notwendigen Informationen nutzt CVS noch eine Reihe von weiteren Dateien und Verzeichnissen, die auch bei einer Konvertierung der Repositories teilweise von Belang sind. Für eine Konvertierung des CVS Repositories ist das `Attic` Unterverzeichnis von größter Relevanz. In diesem Verzeichnis werden sämtliche Dateien gespeichert, die unter Versionskontrolle gelöscht wurden, also nicht mehr Bestandteil der aktuellen Arbeitskopie sein sollen, dennoch aber

für ältere Revisionen des Repositories verfügbar sein müssen. Bei diesen Dateien handelt es sich intern um reguläre RCS Dateien, sie können also bei einer Konvertierung wie die anderen RCS Dateien behandelt werden. Ein weiteres für CVS spezifisches Verzeichnis ist das CVS Unterverzeichnis; dieses enthält keine versionierten Daten, sondern Informationen, ob beispielsweise Watches (siehe Bar/Fogel 2003) auf Dateien bestehen. Für das gesamte Repository existiert ein `CVSROOT` Verzeichnis; dieses enthält primär Skripte, die automatisch bei Aktionen auf dem Repository ausgeführt werden. Ein letzter Bestandteil eines CVS Repositories sind Lock-Dateien. Mittels dieser Dateien wird verhindert, dass mehrere Nutzer zeitgleich auf Dateien zugreifen, also beispielsweise ein Lese- und ein Schreibzugriff zeitgleich stattfinden.

4.2.3 Vorgehen zur Umwandlung des CVS Repositories

Im Folgenden sollen die Schritte dargestellt werden, die von einem Algorithmus zur Umwandlung eines CVS Repositories in das Datenformat von Subversion durchlaufen werden müssen.

Unterschiede bei Speicherung von Daten durch CVS und Subversion

Die Datenspeicherung im CVS Repository und die Datenspeicherung von Subversion unterscheiden sich stark. Der erste Unterschied besteht darin, dass Subversion ein Binärformat für sein Repository nutzt, während CVS auf RCS Dateien beruht. Um bei einer Konvertierung den direkten Umgang mit Binärdaten zu vermeiden, wird durch die im Weiteren hier betrachteten Werkzeuge eine Konvertierung des CVS Repositories in jenes Dateiformat durchgeführt, das Subversion auch zur Sicherung seiner Datenbank verwendet. Eine solche Datei wird auch als Dump des Subversion-Repositories bezeichnet, es handelt sich hierbei um eine Textdatei, die wiederum in ein Subversion-Repository importiert werden kann. Der zweite Unterschied ist der unterschiedliche Umgang mit Revisionsnummern durch Subversion und CVS. Das einem CVS Repository zugrunde liegende RCS Dateiformat verwaltet Revisionsnummern auf Ebene einzelner Dateien, während Subversion Revisionsnummern jeweils für das gesamte Repository vergibt. Hieraus ergibt sich bei CVS das Problem, dass Dateien, die gemeinsam verändert eingestellt wurden, nur über den Umweg des Autorenkommentars und des Einstellungsdatums identifiziert werden können.

Schritte und Datenstrukturen zur Umwandlung eines CVS Repository

Diese Darstellungen helfen, den Aufbau der jeweiligen Repositories noch einmal zu verdeutlichen; der beschriebene Ablauf orientiert sich dabei an dem Vorgehen des `cvs2svn` Skripts[61]. Die Umwandlung des CVS Repositories wird in mehreren aufeinander aufbauenden Schritten vollzogen. Diese Schritte sollen hier im Anschluss skizziert werden, eine ausführlichere Darstellung bietet die Dokumentation des `cvs2svn` Programms.

Parsing aller CVS Commits
In diesem Schritt werden für jede Datei unter Kontrolle des CVS sämtliche Revisionen ermittelt und zusammengefasst. Ziel ist es, eine nicht geordnete Übersicht sämtlicher Revisionen im Repository zu erhalten. Dies geschieht, indem jede RCS Datei im Repository durchlaufen und für jede Revision innerhalb dieser Datei ein Eintrag in einer zentralen Datei generiert wird.

Unter anderem werden die folgenden Daten für jeden Eintrag berücksichtigt:

- Zeitpunkt der Revisionserstellung
- Hashwert des Namens des Autors und der Lognachricht
- Revisionsnummer der Revision
- Zeitpunkt der vorangehenden Revision
- Revisionsnummer der folgenden Revision
- Art der Änderung, möglich sind Hinzufügen, Veränderung oder Löschen
- Information ob die Datei gelöscht ist, also nicht mehr Teil der aktuellsten Revision der Arbeitskopie
- Informationen ob es sich um eine ausführbare Datei handelte
- Die Größe der Datei
- Der Branch in dem diese Verzweigung stattfand
- Der Pfand im CVS Repository

Es kann hierbei vorkommen, dass innerhalb einer RCS Datei eine jüngere Revision ein früheres Datum hat als eine spätere Revision. In diesem Fall wird das Erstellungsdatum der älteren Revision derart angepasst, dass es kurz vor der jüngeren Revision liegt. Außerdem wird ein Verweis auf diese Revision zur späteren Verwendung abgelegt. Um Revisionen der Dateien einander zuordnen zu können, die gemeinsam in das Repository eingestellt wurden, wird ein Hash aus dem Namen des Autors der Revision und der abgegebenen Beschreibung der Revision verwendet.

---

[61] http://cvs2svn.tigris.org/

Behandlung von fehlerhaften Zeitangaben und Aufbereitung der Daten
In diesem Schritt findet eine Nachbearbeitung der im ersten Schritt erzeugten Datei statt. Ziel dieses Schrittes ist es, Unregelmäßigkeiten zu entfernen, die durch die Anpassung des Einstellungsdatums einer Revision in Schritt 1 aufgetreten sein können. Es wird dabei derart vorgegangen; dass eine Gruppierung von Revisionen anhand des Hashs von Autor und Kommentierung der Revision erfolgt. Innerhalb einer solchen Gruppe werden alle Revisionen mit einem nahe beieinander liegenden Erstellungsdatum als gemeinsam ins Repository eingestellt betrachtet; eventuell wird ihr Erstellungsdatum derart verändert, dass die Differenz des Einstellungsdatums innerhalb einer Gruppe möglichst gering ist.

Zusammenführung von gemeinsamen CVS Commits zu ChangeSets
In diesem Schritt sollen ChangeSets rekonstruiert werden, also Gruppen von Dateien und Veränderungen an diesen Dateien, die gemeinsam in ein Repository eingestellt wurden. Um diese ChangeSets zu finden; wird die in Schritt 1 erzeugte und in Schritt 2 modifizierte Liste aller Revisionen nach dem Datum der Revision und dem Hash für Autor und Nachricht sortiert. Da eine Identifizierung von gemeinsamen CVS Revisionen nur auf der Grundlage dieses Hashs möglich ist, kann es zu fehlerhaften Zuordnungen kommen, insbesondere wenn derselbe Autor eine Nachricht mehrmals verwendet hat oder wenn es serverseitig Probleme mit der Zeitnehmung gab.

Identifikation von CVS Symbolen
In diesem Schritt werden Datenstrukturen erstellt, die eine Zuordnung von Revisionen zu symbolischen Namen ermöglichen. Symbolische Namen bezeichnen in diesem Kontext die von CVS verwendeten Bezeichner für Tags und Branches. Am Ende dieses Schritts können alle symbolischen Namen identifiziert werden, die jeweils mit dieser Revision assoziiert sind. Dazu wird die in Schritt 3 erzeugte Datenstruktur durchlaufen und für jeden Eintrag auf Branches und Tags hin untersucht, diese werden einer Revisionsnummer zugeordnet.

Erstellung der Subversion Revisionsnummern für CVS ChangeSets
Hier findet eine Zuordnung von in den vorherigen Schritten identifizierten CVS ChangeSets, also Gruppen von gemeinsam erfolgten Revisionen, und Subversion Revisionsnummern statt. Neben dieser Zuordnung werden auch symbolische Namen der Subversion-Revisionsnummer zugeordnet, die in dem Bereich für das erste und letzte Auftreten dieses Namens liegen. Als Ergebnis dieser Zuordnung wird eine Datei erstellt, die für jede untersuchte RCS Datei für jeden symbolischen Namen je einen Eintrag mit der niedrigsten und höchsten Subversion Revisionsnummer enthält, zu dem eine solche Zuordnung erfolgen kann.

Sortierung nach Subversion Revisionsnummer
Nun wird die im vorherigen Schritt erstellte Zuordnung von CVS Revision, Subversion Revisionsnummer und symbolischen Namen sortiert. Diese Sortierung findet nach dem symbolischen Namen und innerhalb des symbolischen Namens nach der Subversion Revisionsnummer statt; somit liegen in der resultierenden Datei alle Einträge für einen symbolischen Namen in Abfolge.

Zuordnung von Branches und Tags auf Revisionsnummern
In diesem Schritt werden aus allen Subversion-Revisionsnummern die jeweils als erster beziehungsweise letzter Zeitpunkt für das Auftauchen eines symbolischen Namens in Frage kommenden Revisionsnummern identifiziert. Für jeden symbolischen Namen wird ein Verweis in Form eines Offsets in die zuvor erstellte Datei generiert.

Einstellen ins Subversion Repository
In diesem letzten Schritt wird aus den zuvor generierten Datenstrukturen eine Datei erstellt, die sich in ein Subversion Repository importieren lässt. Dabei wird in der Reihenfolge der identifizierten Subversion-Revisionen vorgegangen. Es werden also jeweils alle Änderungen eingestellt, die einer Revisionsnummer zugeordnet wurden. Bei der Erstellung von Branches und Tags wird dabei die Zuordnung von symbolischen Namen auf Revisionsnummern verwendet. Hierbei muss beachtet werden, dass Subversion Branches und Tags als Kopien[62] innerhalb des versionierten Datenbestandes darstellt. Ziel bei der Darstellung von Branches und Tags ist die Verwendung möglichst weniger Kopien aus Gründen der Übersichtlichkeit. Um dieses Ziel zu erreichen werden Tags so früh wie möglich, Branches so spät wie möglich erzeugt. Tags werden in der ersten Revision erzeugt, nachdem die letzte ursprüngliche CVS Revision den Tag eingeführt hat; Branches werden unter der Revisionsnummer, die direkt vor der ersten Änderung in einem Branch liegt, eingeführt

Die kurze Übersicht über die Schritte, die bei einer Konvertierung eines CVS Repositories durchlaufen werden, lässt bereits auf die Komplexität des Vorganges schließen. Insbesondere wird eine Überführung dadurch erschwert, dass die Erstellung von ChangeSets nur auf der Basis von Autor, Kommentar und Erstellungsdatum einer Änderung möglich ist, es also eine Vielzahl von Sonderfällen und Unregelmäßigkeiten geben kann, die durch einen Algorithmus beachtet werden müssen.

---

[62] Wie bereits angesprochen, handelt es sich um Kopien, bei denen lediglich die Unterschiede zwischen Kopie und Original tatsächlich Speicherplatz benötigen.

4.2.4 Werkzeuge zur Konversion von Repositories

Es gibt eine Reihe von Werkzeugen und Hilfsmitteln, die eine Konvertierung von CVS Repositories nach Subversion anbieten oder unterstützen. Das verbreitetste Werkzeug ist wohl `cvs2svn`, eine Sammlung von Python[63] Skripten. Dieses Programm wird im Rahmen des Subversion Projekts auch aktuell noch weiterentwickelt. Cvs2svn bietet die Möglichkeit, auf die Konvertierung durch eine Reihe von Parametern Einfluss zu nehmen. So ist beispielsweise die Auswahl von Symbolen, also Branches und Tags, möglich, die beim Auslesen der Dateien aus dem CVS Repository berücksichtigt werden sollen.

4.2.5 Umwandlung der CVS Repositories der Abteilung FuE

Für die Umwandlung des CVS Repositories der Abteilung FuE wurden zwei Werkzeuge erprobt: das aktuell weiterentwickelte `cvs2svn` und das bereits ältere refinecvs[64]. Bei refinecvs handelt es sich um einen umfangreichen Skript in der Sprache Perl. Die Leistung des Programms ließ sich nicht beurteilen, da trotz Verwendung mehrerer Versionen und manueller Behebung einiger kleinerer Fehler in den Skripten eine Ausführung nicht möglich war. `cvs2svn` ist in Python implementiert, einer objektorientierten Skriptsprache. Bei einer Konvertierung geht `cvs2svn` nach den oben beschriebenen 8 Schritten vor. Als Ergebnis der Konvertierung werden wahlweise Dump Dateien erstellt oder Revisionen inkrementell in ein Subversion Repository geschrieben.

Neben dem Programm `cvs2svn` ist zur Konvertierung eine Installation des Python Interpreters und einiger Hilfsprogramme notwendig; die sowohl für Linux als auch für das Windows Betriebssystem verfügbar sind `cvs2svn` benötigt weiterhin das Programm co aus dem RCS Paket und in einigen Fällen das Programm CVS in der Version für die Nutzung mittels der Kommandozeile. Erste Tests von `cvs2svn` auf der Windows Plattform verliefen enttäuschend. Sowohl in einer reinen Windows Umgebung als auch bei Verwendung der Unix-Emulationsschicht Cygwin war es nicht möglich, die in `cvs2svn` enthaltene Testsuite auszuführen. Eine Konvertierung von kleineren Projekten[65] im CVS Repository gelang zwar, bei größeren Projekten mit etwa 2000 Revisionen in 1200 Dateien kam das Programm jedoch reproduzierbar zum Stillstand und musste abgebrochen werden. Diese Probleme ließen sich durch den Einsatz von `cvs2svn` unter Linux vermeiden. Hier wurde eine Konvertierung vollständig durchgeführt, und die hier bereits erwähnte Test-

---

[63] http://www.python.org/
[64] http://lev.serebryakov.spb.ru/refinecvs/
[65] weniger als 200 identifiziere Subversion Revisionen, etwa 200 Dateien

suite vollständig und erfolgreich durchlaufen. Diese Erfahrung deckt sich mit Beschreibungen die in mehreren Diskussionsforen gefunden wurden. Nach der Konvertierung wurde der Datenbestand des Repositories mittels des Befehls `svnadmin dump` extrahiert. Der resultierende Datenbestand konnte im Folgenden unter Windows wieder in ein Subversion Repository eingestellt werden.

Probleme gab es sowohl unter Linux als auch unter Windows bei der Konvertierung eines Projektes mit nicht normgerechten RCS Dateien. Diese Dateien enthielten nach dem RCS Dateiformat nicht zulässige Revisionsnummern und Symbole. Um eine Konvertierung dieser Dateien durch `cvs2svn` zu gestatten wurde mittels eines Perl Skriptes vor dem Beginn der Konvertierung durch `cvs2svn` auf die RCS-Dateien zugegriffen, dabei wurden drei Aktionen durchgeführt.

- Sämtliche Symbole wurden aus der Datei entfernt[66].
- Sämtliche Branches und die ihnen zugeordneten Revisionsnummern wurden entfernt[67] .

Die dateiweiten Revisionsnummern wurden neu vergeben. Hierbei wurden die ungültigen Revisionsnummern in einigen RCS Dateien durch neue Revisionsnummern ersetzt. Die Vergabe der Revisionsnummer konnte dabei für jede Datei isoliert erfolgen. Dies ist möglich, da CVS kein Konzept von ChangeSets hat. Gemeinsame Änderungen werden, wie oben bei der Darstellung des Algorithmus zur Konvertierung beschrieben, allein durch ihr Datum und durch den Kommentar identifiziert, somit gehen durch eine neue Vergabe der Revisionsnummern keine Informationen verloren.

Nachdem der Skript zur Konvertierung auf alle nicht-binären Dateien[68] angewendet worden ist, war `cvs2svn` in der Lage eine Konvertierung aller Repositories durchzuführen.

---

[66] Prinzipiell wäre es hier ausreichend gewesen, nur Symbole zu entfernen,die auf eine ungültige Revisionen verweisen, da aber eine weitere Verwendung der Symbole nicht intendiert, ist erscheint dieses Vorgehen gerechtfertigt, zumal diese Entscheidung die Entwicklung des Skriptes entschieden vereinfachte.

[67] Dies ist ein Sonderfall der oben beschriebenen Entfernung aller Symbole; auch dieses Vorgehen ist weniger der Notwendigkeit geschuldet als dem Wunsch, den Skript zur Konvertierung einfach zu halten sowie dem Umstand, dass im Wesentlichen nur ein einziger Vendorbranch vorhanden war, der im Rahmen eines anfänglichen Imports entstanden ist.

[68] Alle RCS Dateien, die Binärdaten wie jar oder jpg Dateien kapseln, wurden vom Skript ignoriert, die Einschränkung auf nicht binäre Dateien erfolgte in hier da sich die Beschädigung der RCS Dateien auf Textdateien beschränkte.

4.2.6 Abschließende Bemerkung zur Konvertierung von CVS Datenbeständen

Einer Überführung von bestehenden versionierten CVS Datenbeständen in ein Subversion Repository steht in der Praxis nichts entgegen. Es stehen zahlreiche Werkzeuge zur Verfügung, mit deren Hilfe eine Konvertierung möglich ist, wovon `cvs2svn`., besonders herauszuheben ist. Dieses bietet eine Konvertierung in mehreren Schritten, was einen Eingriff in den Datenbestand zu jedem Zeitpunkt der Konvertierung erlaubt, um eventuell Änderungen am Datenbestand vorzunehmen oder eigene Änderungen einzupflegen.

Während der Konvertierung des CVS Datenbestandes der Abteilung FuE wurde die Erfahrung gemacht, dass Schwierigkeiten in erster Linie während der ersten Phase einer Konvertierung, dem Parsen der RCS Dateien im CVS Repository, auftreten. Sind die zugrunde liegenden Dateien durch CVS selbst noch verwendbar, so lassen sich eventuelle Probleme durch Vorverarbeitung für eine Konvertierung aufbereiten.

4.3 Überprüfung von Subversion in der praktischen Nutzung

In diesem Kapitel wird dargestellt, mit welchen Methoden die in Unterkapitel 4.1 beschriebenen clientseitigen Softwarekomponenten auf ihre Eignung in der Abteilung FuE hin überprüft werden sollen.

Der Schwerpunkt liegt auf der Überprüfung der Fehlerfreiheit und Ausgereiftheit der Programme sowie auf deren Bedienbarkeit unter softwareergonomischen Gesichtspunkten[69]. Eine bewusst kleine Rolle spielt die Überprüfung der Software unter dem Gesichtspunkt der Performanz. Die Aussparung der Untersuchung dieses Aspekts wird durch zwei Faktoren gerechtfertigt:

Vergleichsweise geringe Anforderungen an die Performanz der verwendeten Software. Unter dem Gesichtspunkt der Leistungsfähigkeit der verwendeten Software stellen sich für die Anwendung in der Abteilung FuE nur sehr geringe Anforderungen. Die Begründung für diese Einschätzung liegt in den Ergebnissen der unter Punkt 2.3 dargestellten Befragung der Mitarbeiter der Ab-

[69] Eine vollständige Evaluation wird dabei nicht durchgeführt, diese Einschränkung lässt sich zum einen damit begründen, dass eine erschöpfende Untersuchung im Rahmen dieser Arbeit und vor dem Hintergrund der zahlreichen beteiligten Softwarekomponenten nicht zu leisten gewesen wäre, in erster Linie spielt aber die Tatsache eine Rolle, dass die die untersuchten Programme in ihrer Bedienung mit den entsprechenden Programmversionen für CVS äquivalent sind.

teilung. Dieses Interview führte zu dem Ergebnis, dass nur kleine Teams von zwei bis drei Personen mit jeweils einem Projekt im Repository arbeiten und dass hier wiederum die Frequenz von Zugriffen auf das Repository relativ gering ist.

Mangelnde Aussagekraft von Benchmarks. Da Umfang und Art der Nutzung eines Versionskontrollsystems unmittelbar von den Arbeitsabläufen abhängt. in die das System eingebettet ist, fällt es schwer, realistische, allgemein anwendbare Benchmarks für ein solches System zu erstellen. Eine denkbare Herangehensweise läge hier in einer Untersuchung der vorliegenden CVS Repositories mit dem Ziel, die Zeitpunkte und den jeweiligen Umfang von Zugriffen auf das Repository quantitativ zu bestimmen und diese Zugriffe mit einem Skript nachzubilden. Während ein solches Vorgehen eine Evaluation des Laufzeitverhaltens von Subversion an realistischen Anwendungsszenarien von CVS ermöglicht, kann auch hier nicht gewährleistet werden, dass eventuelle kritische Punkte im Laufzeitverhalten durch die so erstellten Szenarien zur Leistungsüberprüfung offen gelegt werden.

Vor diesem Hintergrund beschäftigt sich dieses Kapitel in höherem Maß mit qualitativen als mit quantitativen Aspekten von Subversion und den gemeinsam mit Subversion verwendeten Programmen. Es wird dabei vorgegangen, indem relevante Arbeitsschritte beim Einsatz von Subversion auf der unter 4.1 beschriebenen Architektur durchgeführt werden. Ziele sind dabei die Identifikation von Fehlerquellen bei der Durchführung der einzelnen Arbeitsschritte und eine Überprüfung der technischen Eignung der jeweiligen Bestandteile der Software anhand der konkreten Einsatzbedingungen in der Abteilung. Nicht eingegangen wird auf Aktionen, die bei der Befragung der Mitarbeiter der Abteilung FuE als nur selten verwendet bezeichnet wurden. Ein Beispiel für solche Funktionalität ist der Umgang mit Branches.

Szenarien für eine Überprüfung

Ziel dieser Überprüfung ist es, alle bisher beschriebenen Softwarekomponenten, insbesondere aber den Subversion-Serverprozess in Zusammenarbeit mit Apache und das Subclipse-Plugin, anhand realistischer Szenarien einer Überprüfung zu unterziehen. Als weitere Zugriffsmethoden auf das Repository werden die mit Subversion vertriebenen Kommandozeilen-Werkzeuge und das Werkzeug TortoiseSVN untersucht. Im Einzelnen sollen die folgenden Szenarien behandelt werden:

- Erstellen eines neues Projektes,
- Checkout eines bestehenden Projektes aus dem Repository,
- Umgang mit zeitgleich durch mehrere Personen bearbeiteten Dateien ohne Konflikte,

- Verhalten bei Konflikten und Unterstützung bei der Behandlung von Konflikten,
- Entfernen und Umbenennen von Dateien unter Versionskontrolle,
- Betrachtung der Historie einer Datei. Hier soll untersucht werden, welche Sichten angeboten werden, um die Veränderungen einer Datei nachzuvollziehen.

Dabei soll für jedes Szenario überprüft werden, wie sich die unterschiedlichen Softwarekomponenten jeweils in Zusammenarbeit verhalten. Es wird auch jeweils ein grob umrissener Anwendungsfall dargestellt.

Verwendete Software und Hardware

Die für die beschriebenen Tests verwendeten Programmversionen lassen sich der Tabelle entnehmen. Die Verwendung von Subversion 1.1rc2 war notwendig, um das Subclipse Plugin für Eclipse in der Version 3.x mit den nativen Bibliotheken für den Zugriff auf den Subversion Server verwenden zu können.

Server		
Name	Version	Anmerkung
Subversion	1.1rc2	Empfehlung zur Nutzung von Subclipse
Apache	2.0.50	
Repository	BerkeleyDB	Aus Subversion Installation
Betriebssystem	Windows 2000 SP4	
WebSVN	1.61	
PHP	4.3.9	
OpenSSL	0.9.7d	

Tabelle 3a: Bei Tests verwendete Serverprogramme

Client		
Name	Version	Anmerkung
TortoiseSVN	1.1.0 Build 1769	Neuere Version ist seitdem verfügbar
Subclipse	0.9.22	Bezug über Eclipse Update. Es wurden die Subversion Bibliotheken verwandt.
Eclipse	3.1m2, 3.0	Keine Unterschiede feststellbar

Tabelle 3b: Bei Tests verwendete Clientprogramme

4.3.1 Erstellen eines neues Projektes

Hier soll untersucht werden, wie die Erstellung eines neuen Projektes unter Versionskontrolle vonstatten geht. Als Referenz wurde das Project DBClear der Abteilung FuE verwendet; dieses Projekt stellt mit 1800 Dateien bereits einen nicht trivialen Anwendungsfall für das Versionskontrollsystem dar.

Verwendung der Kommandozeile

Die Erstellung eines neuen Projektes erfolgt durch den Befehl `svn import`. Dieser Befehl fügt einen Pfad des Dateisystems und alle untergeordneten Pfade in das Versionskontrollsystem ein. Wegen des rekursiven Verhaltens sollte der verwendete Client entweder derart konfiguriert sein, dass alle nicht unter Versionskontrolle gestellten Dateien automatisch ignoriert werden, oder es sollte anstelle des tatsächlichen Verzeichnisses des Projektes zuerst ein leeres Verzeichnis im Repository erstellt werden, dem anschließend alle relevanten Dateien manuell hinzugefügt werden. Üblicherweise wird nach dem Einstellen eines Projektes in das Subversion Repository die lokale Kopie des Projektes gelöscht; für die weitere Arbeit mit dem Projekt wird eine Arbeitskopie aus dem Subversion Repository per Checkout entnommen. Sollte dieses Vorgehen nicht gewünscht oder nicht möglich sein, ist es alternativ möglich, auch ein so genanntes in-place Import[70] durchzuführen. Hier wird ein leeres Verzeichnis im Subversion Repository erstellt. Nach diesem Checkout ist im Verzeichnis die grundlegende Struktur einer Subversion Arbeitskopie angelegt, alle weiteren Inhalte können mittels `svn add` hinzugefügt werden.

Nutzung von Subclipse

Das Einstellen eines neuen Projektes in ein Subversion Repository durch das Eclipse-Plugin Subclipse erfolgt analog zum Einstellen von Inhalten in ein Repository unter CVS. Für einen mit der CVS Integration in Eclipse vertrauten Nutzer ergeben sich hier also keine größeren Unterschiede. Neu eingestellt werden können nur ganze Projekte; es ist somit nicht möglich, lediglich ein Verzeichnis in das Versionskontrollsystem einzustellen. Wird ein Projekt eingestellt, so sind anfänglich noch keine Ressourcen des Projektes zum versionierten Datenbestand des Verzeichnisses hinzugefügt. Sollen Dateien unter Versionskontrolle gestellt werden, so müssen sie entweder einzeln oder auf der Ebene von Verzeichnissen hinzugefügt werden. Beachtet werden sollte hier, dass für die jeweilige Installation spezifische Dateien, insbesondere die `.project` und die `.classpath` Datei, nicht in das Repository übernommen

[70] http://subversion.tigris.org/project_faq.html#in-place-import

werden. Ein Einstellen des als Referenz betrachteten Projektes gelang bei Verwendung von Subclipse ohne Probleme.

Einsatz von TortoiseSVN

Da es sich bei TortoiseSVN um eine Erweiterung des Windows Explorers handelt, wird das Hinzufügen eines Projektes zu einem Repository durch das Kontextmenü des Windows Explorers ausgelöst. Unter dem Punkt import können die Inhalte eines Verzeichnisses zum versionierten Datenbestand eines Repositories hinzugefügt werden. Der Pfad des Repositories wird dabei in einem Dialog abgefragt; hier können sowohl lokale als auch entfernte Repositories angegeben werden.

Ergebnisse

Das Einstellen des als Referenz verwendeten Projektes unter Versionsverwaltung gelang mit allen drei besprochenen Werkzeugen ohne Schwierigkeiten; der Zeitaufwand betrug dabei stets weniger als eine Minute. Bei der Verwendung der Kommandozeile unterscheidet sich das Vorgehen, um ein bereits bestehendes Projekt unter Versionskontrolle zu stellen, vom Vorgehen mit beiden graphischen Werkzeuge insofern, als dass hier explizit ein leeres Verzeichnis zuerst in das Arbeitsverzeichnis kopiert werden muss, um die notwendigen Verzeichnisse und Dateien eines Subversion Arbeitsbereiches zu erstellen.

4.3.2 Checkout eines bestehenden Projektes

Hier wird untersucht, wie sich der Checkout eines bereits bestehenden Projektes aus dem Versionskontrollsystem heraus vollzieht. Dabei wird ein komplexes Projekt als Grundlage verwendet, das mittels der in 4.5 beschriebenen Schritte in ein Subversion Repository übernommen wurde.

Checkout einer Arbeitskopie mittels der Kommandozeile

Auf der Kommandozeile wird das Erstellen einer lokalen Arbeitskopie durch den Befehl `svn checkout <URL>` angestoßen, wobei die URL für das Repositories steht. Der Vorgang benötigt etwa 2 Minuten, als Ergebnis liegt eine komplette lokale Arbeitskopie vor.

Checkout nach Eclipse mittels Subclipse

Der Checkout des Projektes mittels Subclipse erfolgt über den Repository Browser, einer Ansicht, die durch das Plugin bereitgestellt ist und sich am CVS Repository Browser orientiert. Die Möglichkeit, über den Import-Menüpunkt ein Projekt aus einem Repository direkt zu importieren, wie dies bei Verwendung von CVS möglich ist, besteht nicht. Handelt es sich um ein

Java Projekt, so ist wichtig, dass nicht der Menüpunkt `Checkout Projekt` sondern `Checkout Project as...` verwendet wird. Dies hat zur Folge, dass Eclipse den Datenbestand als Java Projekt erkennt. Während des Checkouts des Projektes kommt es zu einer Warnmeldung, die das Überschreiben des Stammverzeichnisses des Projektes ankündigt; trotz dieser Meldung funktionierte das Erstellen der Projekte ohne Probleme.

Checkout einer Arbeitskopie mit TortoiseSVN

Durch TortoiseSVN wird ein Projekt entnommen, indem in einem beliebigen Verzeichnis das Kontextmenü des Windows Explorers geöffnet wird, hier steht ein Eintrag checkout zur Verfügung. Bei Aufruf dieses Eintrags werden in einem Dialogfeld das Zielverzeichnis und die URL des Repositories abgefragt; ebenso kann hier eine Revisionsnummer angegeben werden, die entnommen werden soll.

Export eines Tarballs mit WebSVN

Da WebSVN nur einen lesenden Zugriff auf ein Projekt im Repository ermöglicht, wird durch die Schnittstelle kein Checkout angeboten, sondern lediglich ein Export als Tarball. Im Gegensatz zum Checkout werden hier nur die eigentlichen Dateien des Projekts entnommen, also kein kompletter Arbeitsbereich generiert. Daten, die zur Verwaltung des Arbeitsbereiches verwendet werden, bleiben unberücksichtigt. Ebenso stellt WebSVN eine Ansicht einzelner Dateien bereit. Diese Funktionen sind dann nützlich, wenn ein Projekt zur Einsichtnahme aus dem Versionskontrollsystem entnommen werden soll. In einem solchen Fall ist die Verwendung eines speziellen Subversion Client nicht notwendig, lediglich ein Browser muss vorhanden sein.

Ergebnisse

Mit allen untersuchten Systemen ist das Abrufen von Dateien aus dem Repository ohne Probleme möglich.

4.3.3 Umgang mit parallelen Änderungen an einer Datei

Bei diesem Punkt steht der Umgang mit sich zeitlich überschneidenden Änderungen im Zentrum der Betrachtung, insbesondere die Frage, inwieweit eine Integration von gemeinsam bearbeiteten Dateien durch die jeweiligen Werkzeuge unterstützt wird. Schwerpunkt ist hier die Frage, inwieweit die Integration von prinzipiell nicht widersprüchlichen Änderungen durch die jeweilige Client-Software unterstützt wird.

Verwendung der Kommandozeile

Bei der Verwendung der Kommandozeile werden die Befehle `svn status`, `svn update` und `svn commit` verwendet. `svn status` dient dazu, die lokale Arbeitskopie mit der ebenfalls lokalen Referenzdatei im `.svn` Verzeichnis des Arbeitsbereichs oder mit der aktuellen Revision auf dem Server zu vergleichen. Werden dabei Unterschiede festgestellt, so lässt sich die Arbeitskopie mittels des Befehls `svn update` auf den aktuellsten Stand bringen. Nachdem Server und der lokale Arbeitsbereich synchronisiert sind, werden die lokalen Änderungen mittels des Befehls `svn commit` auf dem Server verfügbar gemacht. Bei Verwendung der Kommandozeile können parallel erfolgte Änderungen automatisch integriert werden, sofern sie auf unterschiedliche Zeilen innerhalb der Datei erfolgt sind.

Verwendung des Subclipse Plugins

Die Schritte,. die sich bei der Bearbeitung von Ressourcen mittels des Subclipse Plugins ergeben, entsprechen denen auf der Kommandozeile. Alle Aktionen können durch das Kontextmenü der jeweiligen Datei, des Verzeichnisses oder des Projekts ausgelöst werden; hier stehen alle mit der Versionsverwaltung zusammenhängenden Aktionen unter dem Menüpunkt team[71] zur Verfügung. Neben dieser Möglichkeit zum Zugriff wird durch das Plugin auch eine Eclipse-Perspektive zur Synchronisation angeboten, mittels dieser Perspektive können Unterschiede zwischen der Arbeitskopie und dem Server detaillierter betrachtet werden. Ebenso wie bei Verwendung der Kommandozeile werden nicht widersprüchliche Unterschiede innerhalb eines Dokumentes während eines Updates selbstständig integriert.

Verwendung von TortoiseSVN

Wie bei TortoiseSVN üblich, wird auch das Update eines Repositories durch Kontextmenüs des Windows Explorers angeboten. Durch die Verwendung von speziellen Symbolen für Dateien, die nicht mit dem Repository synchronisiert sind, erlaubt das Programm einen schnellen Überblick, welche Dateien aktuell sind und welche vom Zustand im Repository abweichen. Das Vorgehen, um Änderungen im Repository zu veröffentlichen, entspricht jenem auf der Kommandozeile oder bei der Verwendung von Subclipse. Durch den Commit Befehl werden Änderungen in das Repository übertragen. Wurden in der Zwischenzeit Änderungen in das Repository eingestellt, so müssen diese

---

[71] Für Details zum Umgang mit Versionskontrollsystemen sei auf [Hol04a] und [Hol04b] verwiesen. Beide Bände betrachten die Verwendung mit CVS; gleiche Grundsätze gelten aber auch bei der Verwendung von Subversion.

mittels der Update Aktion in die lokale Arbeitskopie übernommen werden; nicht widersprüchliche Unterschiede werden hierbei integriert.

Ergebnisse

Alle drei betrachteten Programme nutzen beim Umgang mit lokalen Änderungen bei parallel erfolgten Änderungen im Repository das gleiche Vorgehensmodell. Die grundlegenden Aktionen sind das Update eines lokalen Repositories und das Commit der darin erfolgten Änderungen: beide Aktionen werden bei allen betrachteten Werkzeugen durch den Benutzer direkt aufgerufen. Alle betrachteten Programme sind in der Lage, nicht widersprüchliche Änderungen zu integrieren und den Benutzer über diese automatische Integration in Kenntnis zu setzen.

4.3.4 Umgang mit widersprüchlichen Änderungen einer Datei

Dieser Punkt erweitert das in Punkt 4.7.3 entworfene Szenario durch die Aufnahme von widersprüchlichen Änderungen innerhalb einer Datei. Hier stehen also die Werkzeuge und Hilfsmittel im Vordergrund, die durch die jeweilige Clientsoftware zur Behandlung von Konflikten angeboten wird.

Auflösung von Konflikten mittels der Kommandozeile

Der Umgang mit Konflikten durch den Subversion Client für die Kommandozeile orientiert sich am Vorgehen von CVS. Liegen Konflikte vor, so werden die widersprüchlichen Zeilen beider Dateien bei einem Update in einer Datei mit Differenz-Darstellung zusammengefasst. Diese Konflikte müssen vom Nutzer manuell integriert werden. Um diese manuelle Integration zu unterstützen, werden neben der Datei mit den aufgetretenen Konflikten drei weitere Dateien in der Arbeitskopie des Nutzers abgelegt.

- Eine Datei mit dem Namen`<Dateiname>.mine` <Dateiname> entspricht dabei dem Namen der Datei mit einem Konflikt. Diese Datei ist eine Kopie der Datei des Nutzers vor dem Ausführen des Updates.
- Eine Datei mit dem Namen `<Dateiname>.rNR0`. NR0 ist hierbei die Revisionsnummer, die mit dem Erstellen der aktuellen Arbeitskopie aus dem Repository entnommen wurde.
- Eine Datei mit dem Namen `<Dateiname>.rNR1`. NR1 ist dabei die Revisionsnummer, die zum Zeitpunkt des Updates im Repository aktuell war.

Eine Erweiterung gegenüber der Auflösung von Konflikten durch CVS ist, dass nach der Behebung der Konflikte für jede Datei beziehungsweise jedes Verzeichnis der Befehl `svn resolved` aufgerufen werden muss. Dieser Befehl signalisiert dem Versionskontrollsystem die Behebung aller Konflikte vom Standpunkt des Benutzers. Daraufhin ist ein Commit der entsprechenden

Ressourcen möglich und die oben beschriebenen Dateien zur Unterstützung des Anwenders bei der Auflösung von Konflikten werden entfernt.

Unterstützung von Subclipse bei der Behebung von Konflikten

Die Behebung von Konflikten durch Subclipse geht nur wenig über deren reine Erkennung hinaus, die auch auf der Kommandozeile angeboten wird. Wird nach der Feststellung eines Konfliktfalls ein Update durchgeführt, so muss der Anwender die entsprechende Datei manuell editieren, um die Konflikte zu beheben und die Markierung zu deren Kennzeichnung entfernen. Nachdem die Konflikte in der Datei entfernt sind, kann über den Team-Eintrag im Kontextmenü der Status der Datei auf resolved gesetzt werden; dies gestattet es, die Datei wieder in das Repository einzustellen. Eine Hilfestellung ist das Versehen von Ressourcen im Konfliktstatus mit zusätzlichen Icons

Neben dieser Methode gibt es unter dem Menüpunkt `Team->Synchronize` umfangreiche Werkzeuge in Form einer speziellen Perspektive mit dem Ziel, mögliche Konflikte zu integrieren. Der Aufruf dieses Werkzeugs muss allerdings vor dem eigentlichen Update erfolgen. Bei Nutzung dieser Funktionalität wird durch Eclipse eine 3-Way oder 2-Way Differenz-Darstellung angeboten, mittels derer die Unterschiede der Dateien integriert werden können.

Insgesamt bietet Eclipse mit dem Subclipse Plugin eine gute Unterstützung bei der Behebung von Konflikten; allerdings ist das Vorgehen hier insofern vom üblichen Arbeitsablauf abweichend, als dass eine Behandlung von Konflikten optimalerweise nicht nach einem Update erfolgt, sondern davor. Werden vor einem Abgleich mit dem Repository mittels der Perspektive zur Synchronisierung alle potenziellen Konflikte ausgeräumt, so steht eine mit zahlreichen Möglichkeiten ausgestattete GUI zur Verfügung.

Vorgehen bei Konflikten bei Verwendung von TortoiseSVN

Bei der Nutzung von TortoiseSVN wird das Vorliegen eines Konflikts durch das Scheitern eines Commits festgestellt. Die Integration der widersprüchlichen Dateien wird nun durch ein Update eingeleitet: dabei werden wie oben beschrieben drei zusätzliche Dateien in das Arbeitsverzeichnis kopiert. TortoiseSVN unterstützt die Auflösung von Konflikten auf zweierlei Weise:

Für Dateien, die sich miteinander in Konflikt befinden, wird ein spezielles Icon verwendet; dies erlaubt eine schnelle Identifikation solcher Dateien.

Im Kontextmenü für eine solche Datei wird ein Menüpunkt `Edit Conflict` dargestellt; mittels dieses Eintrags wird ein von TortoiseSVN bereitgestelltes Diff-Werkzeug gestartet.

Dieses Werkzeug stellt die Veränderung der Datei im Repository den Veränderungen gegenüber, die lokal in der ursprünglichen Datei vorliegen, es wird

also ein 3-Way Merge durchgeführt. Mittels dieses Werkzeugs können für jeden Konfliktfall aus den widersprüchlichen Dateien die Änderungen ausgewählt werden, die in eine endgültige Version übernommen werden sollen. Nachdem eine konfliktfreie Datei erstellt worden ist, kann diese mittels eines weiteren Menüpunktes als resolved gekennzeichnet und daraufhin mittels eines Updates eingestellt werden. In der Testanwendung konnte TortoiseSVN durchaus überzeugen, sowohl die Darstellung von Konflikten als auch die Behebung erfolgte nachvollziehbar und stieß auf keinerlei technische Probleme. Einen Schwachpunkt stellte lediglich die Tatsache dar, dass Icons, die den Zustand einer Datei darstellten, nicht immer aktualisiert werden wenn sich der Zustand einer Datei verändert, dies kann etwa dann zur Verwirrung beitragen, wenn nach einem aufgelösten Konflikt weiterhin ein Icon zur Kennzeichnung von Konflikten für die Datei dargestellt wird.

Fazit

Der Client für die Nutzung mittels der Kommandozeile verfügt über keine direkte Unterstützung bei der Behebung von Konflikten, die über das Einstellen von Markierungen zur Kennzeichnung hinausgehen. Durch das Bereitstellen der drei angesprochenen zusätzlichen Dateien wird jedoch die Grundlage geschaffen, mit externen Werkzeugen eine Integration zu unterstützen. Eine solche Unterstützung wird durch TortoiseSVN in Form eines 3-Way Diffs geboten. Die hier angebotenen Werkzeuge erlauben eine einfache und wirkungsvolle Integration durch zeilenweise Auswahl von jeweils zu übernehmenden Änderungen. Subclipse unterscheidet sich von TortoiseSVN insofern, als dass eine Behebung von Konflikten optimalerweise vor einem Update erfolgt, bietet ansonsten aber vergleichbare Funktionalität. Es bleibt somit festzuhalten, dass für Subversion durchaus adäquate Werkzeuge zur Behebung von Konflikten vorliegen, sowohl unter technischen Gesichtspunkten als auch unter dem Aspekt der Benutzerfreundlichkeit.

4.3.5 Entfernen und Umbenennen von Verzeichnissen und Dateien

In diesem Punkt soll die jeweilige clientseitige Implementierung der Entfernung und der Umbenennung von Dateien und Verzeichnissen betrachtet werden. Ein kritischer Punkt bei dieser Betrachtung ist die Integration der versionierten Umbenennung von Dateien in TortoiseSVN und Eclipse. Hier stellt sich das Problem, dass der Unterschied zwischen versionierten und regulären Operationen mit Dateien nicht immer klar ersichtlich ist.

Versionierte Verzeichnisoperationen durch die Kommandozeile

Für versionierte Operationen mit Dateien können die Befehle `svn delete`, `svn copy` und `svn move` verwendet werden. Mittels dieser Befehle ist es möglich, eine Datei oder ein Verzeichnis zu löschen, also ab der aktuellen Revision aus dem Repository zu entfernen. Auch besteht die Möglichkeit, den Namen einer solchen Ressource zu ändern oder eine Kopie zu erstellen, die zukünftig getrennt bearbeitet werden kann.

Subclipse Unterstützung für Verzeichnisversionierung

Für die oben beschriebenen versionierten Operationen mit Dateien bietet Subclipse in der SVN-Repository-Perspektive Aktionen im Kontextmenü an. Diese erlauben es, die entsprechenden Aktionen direkt serverseitig durchzuführen und mittels eines Updates in die Arbeitskopie zu übernehmen. Neben dieser expliziten Möglichkeit, auf das Repository zuzugreifen, werden auch Operationen, die auf der Arbeitskopie direkt ausgeführt werden, in versionierte Operationen umgesetzt. Im Einzelnen werden Aktionen wie das Verschieben einer Java-Datei mittels Drag and Drop in ein anderes Package oder die Veränderungen des Namens einer Klasse durch die Refactoring-Funktionalität von Eclipse in entsprechende versionierte Operationen umgesetzt. Diese Unterstützung des Refactorings ist jedoch nicht vollständig, so muss beim Refactoring einer Inner Class (siehe Flanagan 2002) zu einer vollwertigen Klasse die neue Datei mit der Klassendefinition explizit in die Versionskontrolle eingefügt werden.

Integration von Verzeichnisoperationen in TortoiseSVN

TortoiseSVN bietet versionierte Operationen auf Verzeichnissen und Dateien über ein Kontextmenü des Windows Explorers an. Hier können Dateien oder Verzeichnisse umbenannt, kopiert oder verschoben werden. Veränderungen, die direkt mittels des Windows Explorers, also außerhalb des TortoiseSVN Kontextmenüs ausgelöst werden, sind nicht versioniert.

Fazit

Alle drei untersuchten Werkzeuge sind in der Lage, die Möglichkeiten von Subversion zur Versionierung von Dateinamen und Verzeichnissen zu unterstützen. Am elegantesten geschieht der Umgang mit den versionierten Aktionen Umbenennen, Kopieren und Löschen von Datei durch das Subclipse Plugin. Hier werden entsprechende Aktionen im Repository direkt versioniert umgesetzt. Im Fall von TortoiseSVN werden versionierte Aktionen durch einen Untereintrag im Kontextmenü ausgelöst; daraus ergibt sich ein gewisses Risiko, irrtümlich die vom Windows Explorer verwendeten nicht versionierten Daten zu verwenden. Bei der Verwendung der Kommandozeile besteht

dieses Risiko in der Regel nicht; hier muss jeder Befehl explizit eingegeben werden, was Verwechslungen erschwert.

4.4 Betrachtung von Änderungen einer Datei

Unter diesem Punkt wird untersucht, welche Mittel und Ansichten jeweils bereitgestellt werden, um Änderungen an einer Datei nachzuvollziehen. Es wird dabei auf unterschiedliche Zielsetzungen bei der Betrachtung einer Datei eingegangen. Relevant sind bei der Betrachtung einer Datei vor allem drei Aspekte:

Die unterschiedlichen Logmessages die für diese Datei abgelegt wurden.

Die Unterschiede zwischen Dateien in unterschiedlichen Revisionen und unterschiedlichen Branches.

Zeitpunkte, an denen die Veränderungen in eine Datei eingestellt wurden.

Es wird nun untersucht, wie und in welcher Aufbereitung diese Ansichten durch die unterschiedlichen Softwaresysteme angeboten werden.

Information über Änderungen mittels der Kommandozeile

Die Kommandozeile bietet für die drei beschriebenen Ansichten jeweils einen separaten Befehl an. Die Abfrage jener Nachrichten, die ein Autor jeweils mit einer Revision eingestellt hat, erfolgt mittels des Befehls `svn log` für die jeweilige Ressource. Mit dem Befehl `svn blame` kann nachvollzogen werden, in welcher Revision und von wem Veränderungen an Dateien vorgenommen worden sind. Hier wird zu jeder Zeile einer Datei die letzte Revision angezeigt, zu der eine Änderung stattgefunden hat, weiterhin wird die Kennung des Autors vermerkt. Der Befehl `svn diff` schließlich erlaubt die Darstellung der Unterschiede zwischen beliebigen Revisionen einer Datei, auch wenn sich diese auf unterschiedlichen Ästen befindet. Mittels der Kommandozeile lassen sich alle Informationsbedürfnisse lösen, die zu einer Datei und ihren Revisionen bestehen können, hierbei ist gerade auch wegen der Fülle an Optionen eine gewisse Einarbeitung unerlässlich.

Darstellung von Änderungen durch Subclipse

Subclipse bietet eine Vielzahl unterschiedlicher Ansichten, um die Veränderungen an Dateien nachzuvollziehen. Eine Übersicht über die Veränderungen einer Datei stellt die View SVN Ressource Historie dar. In dieser Ansicht werden alle Revisionen aufgeführt, in denen Änderungen an der Datei vorgenommen wurden. Zu diesen Änderungen wird weiterhin der Kommentar angezeigt, der bei ihrer Einstellung abgegeben wurde. Mittels eines Doppelklicks auf eine Zeile kann die entsprechende Revision der Datei in einem Editorfenster geöffnet werden. Sollen die unterschiedlichen Revisionen einer Da-

tei nicht nur eingesehen, sondern direkt verändert werden, so steht hierfür die Compare View zur Verfügung. Diese Ansicht wird im Kontextmenü einer Datei mit der Auswahl des Punktes `Compare with` geöffnet und erlaubt den Vergleich einer Datei sowohl mit vorherigen Revisionen als auch Einsicht in die Veränderungen seit dem letzten Einstellen einer Revision in das Repository. Zuletzt steht auch eine Ansicht zur Verfügung, die der `blame`-Ansicht der Subversion-Kommandozeile entspricht. Diese Ansicht ist als `annote` bezeichnet. Hier wird eine Datei in einem Editor angezeigt und ergänzend gibt es eine Ansicht, die zu jeder Revision der Datei auf den Autor und die geänderten Zeilen verweist. Wird eine Zeile der Datei mit der Maus selektiert, so werden der Autor der Zeile, die Revision der letzten Änderung und der Umfang der Änderung hervorgehoben. Besonders bemerkenswert ist hier auch die explizite Unterstützung von Java Dateien. Für diese werden Unterschiede nicht nur auf der Ebene des Textes, sondern auch auf der Ebene von Klassen dargestellt, so ist beispielsweise ersichtlich, wann einer Klasse Methoden hinzugefügt worden sind. Die Aufnahme einer Verbindung zum SVN Repository erfolgt für den Benutzer transparent.

Nachvollziehbarkeit von Veränderungen mit TortoiseSVN

TortoiseSVN fasst die Darstellung von Veränderungen unter dem Menüpunkt `ShowLog` zusammen, dieser wird wie alle TortoiseSVN Aktionen durch das Kontextmenü des Windows Explorers für eine Ressource unter Versionskontrolle aufgerufen. Die entsprechende Ansicht zeigt für jede Revision der Ressource das Erstellungsdatum, den Autor und die erste Zeile der bei der Einstellung abgelegten Nachricht an. Wird eine Revision ausgewählt, so wird die vollständige mit der Revision eingestellte Nachricht angezeigt, sowie alle im Rahmen dieser Revision veränderten Dateien. Um Einblick in die detaillierteren Veränderungen innerhalb einer Datei zu erhalten, kann diese Datei zusammen mit der vorherigen Revision in einer speziellen Ansicht zur Darstellung von Differenzen geöffnet werden, Hierzu wird auf ein externes Programm aus dem TortoiseSVN Paket zurückgegriffen.

Navigation von Revisionen mit WebSVN

Mittels der Webschnittstelle WebSVN ist eine Betrachtung der Differenzen zwischen Revisionen und eine Zuordnung der Änderungen innerhalb einer Datei zu einer Revisionsnummer und einem Autor möglich. Ein Projekt wird hierbei als Verzeichnisbaum dargestellt. Für jede Datei besteht die Möglichkeit, direkt eine Anzeige der Differenzen zum Vorgänger zu erhalten; hier werden jeweils veränderte Inhalte einer Datei einander gegenübergestellt. Eine alternative Ansicht auf eine Datei ist auch hier die Blame- oder Annote-Ansicht, bei der für jede Zeile einer Datei die Revision der jeweils letzten Änderung angezeigt wird. Die Betrachtung der Inhalte eines Repositories mit-

tels WebSVN bietet den Vorteil, dass außer einem Browser keine weitere Software benötigt wird und dass eine unbeabsichtigte Veränderung der Inhalte des Repositories durch den nur lesenden Zugriff gänzlich ausgeschlossen ist; auch das Erstellen einer lokalen Arbeitskopie ist nicht notwendig. Diese beiden Faktoren bedingen eine sehr geringe Einstiegsbarriere zur Nutzung der durch diese Software bereitgestellten Darstellung. Ein Nachteil von WebSVN ist die im Vergleich geringe Geschwindigkeit. Diese bewegte sich aber mit Antwortzeiten für nahezu alle Aktionen im Bereich von 2 Sekunden durchaus noch im tolerablen Maß.

Zusammenfassung zur Darstellung von Revisionen

Alle betrachteten Werkzeuge bieten jeweils Möglichkeiten, um die Veränderungen einer Revision und ihres jeweiligen Vorgängers zu betrachten; beim Angebot weitergehender Ansichten unterscheiden sich die Werkzeuge jedoch. Mittels der Kommandozeile lassen sich beliebige Revisionen einer Datei vergleichen, auch ist eine Einsichtnahme in die Änderungen einer Datei ohne weiteres möglich. Diese Flexibilität setzt jedoch ein verhältnismäßig hohes Maß an Einarbeitung voraus. TortoiseSVN bietet sich an, wenn Einblick in die Unterschiede zweiter Revisionen oder deren jeweiligen Nachrichten zur Beschreibung genommen werden soll. Wegen der Integration in den Windows Explorer können solche Einblicke schnell und ohne das Starten weiterer Werkzeuge vorgenommen werden. Einen ebenfalls sehr einfachen Zugang zur Historie eines Repositories bietet WebSVN. Da hier außer einem Browser keinerlei Software oder Daten clientseitig vorausgesetzt werden, bietet sich dieser Zugang an, um sich einen schnellen Überblick über ein nicht lokal bearbeitetes Projekt zu verschaffen. Die umfangreichste Unterstützung für die Verfolgung von Änderungen und den Vergleich von Revision bietet Subclipse im Zusammenspiel mit Eclipse. Hier werden zahlreiche Ansichten geboten, die zudem sehr gut in die Arbeitsumgebung integriert sind. Bei allen untersuchten Systemen kam es während der Nutzung zu keinerlei technischen Problemen.

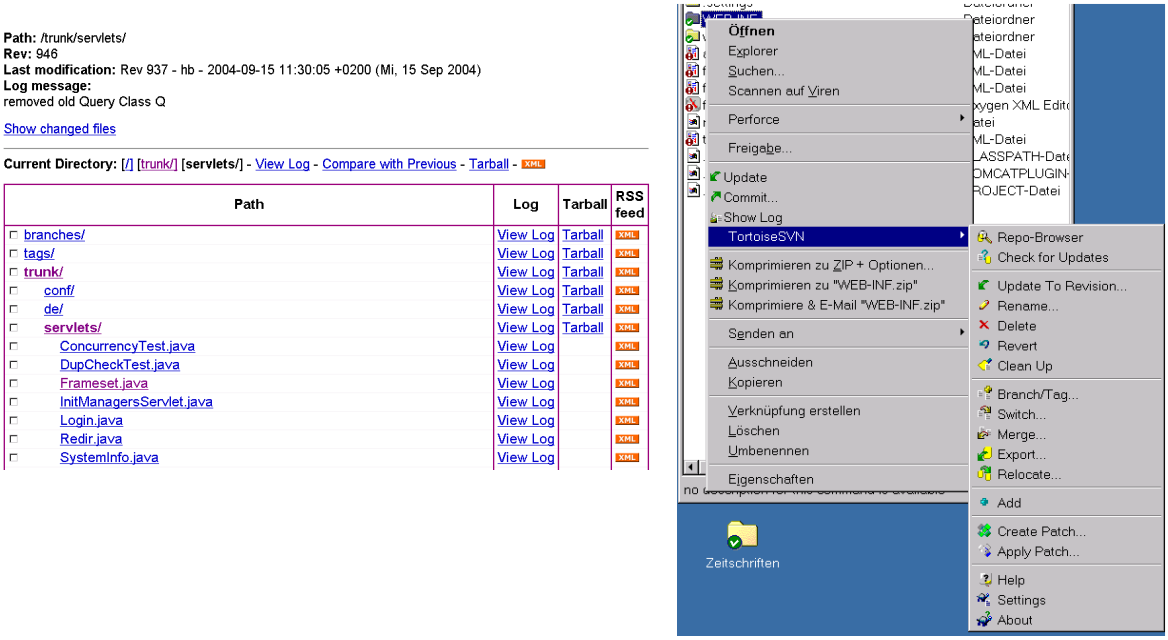

Abbildung 5: WebSVN und TortoiseSVN

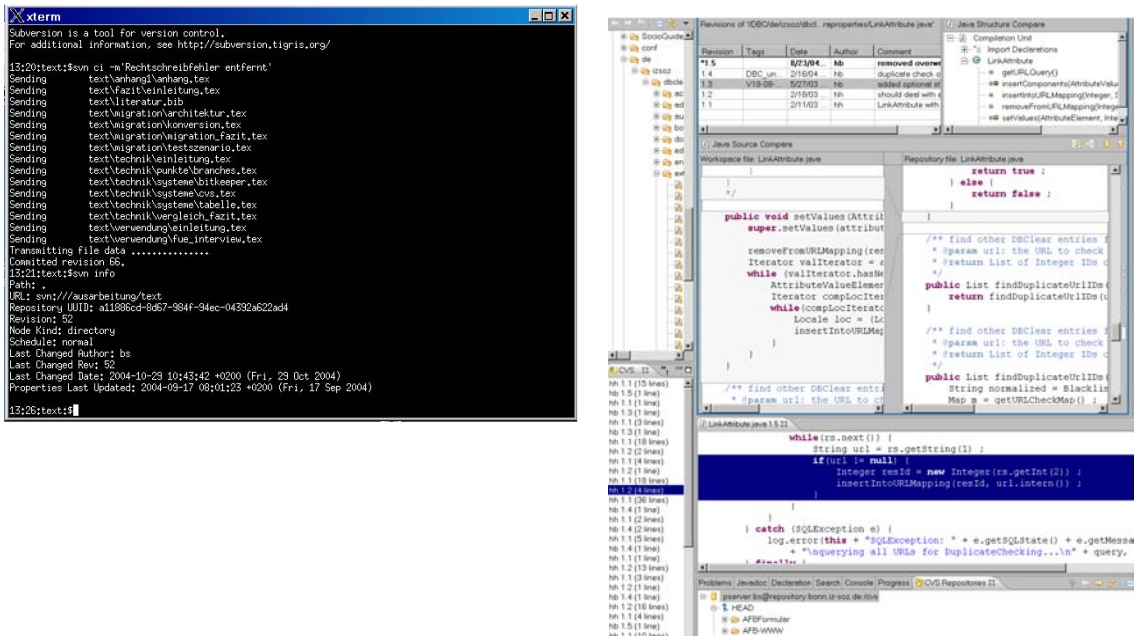

Abbildung 6: Kommandozeilenclient svn und Subclipse

4.4.1 Fazit zur Praxistauglichkeit der untersuchten Software

In diesem Kapitel wurden typische Handlungen von Nutzern mit der Software Subversion anhand unterschiedlicher Clientsoftware nachvollzogen. Dabei wurde festgestellt, dass alle untersuchten Werkzeuge in der Lage sind, den Anforderungen von Nutzern im täglichen Gebrauch gerecht zu werden. Besondere Aufmerksamkeit verdient das Umbenennen und Löschen von Dateien. Hier muss beachtet werden, dass diese Operation nicht auf der Ebene des Betriebssystems ausgeführt wird, sondern durch die jeweiligen Subversion-Befehle. Dies ist in erster Linie bei der Verwendung von TortoiseSVN problematisch, da hier die enge Integration mit dem Windows Explorer dazu verleitet, Operationen mit Dateien durch die hier üblichen Arbeitsschritte durchzuführen, was nicht in die entsprechenden versionierten Operationen umgesetzt wird.

Sehr überrascht hat während der Tests die technische Ausgereiftheit aller verwendeten Werkzeuge. Es kam in keinem Fall zu Abstürzen oder Datenverlusten Hierbei soll aber betont werden, dass auf das bewusste Erzeugen von Fehler auslösenden Situationen verzichtet wurde. Ein kritischer Punkt ist bei TortoiseSVN und Subclipse die Gestaltung der Fehlermeldung, die auf ein Scheitern eines Commits bei einem ausstehenden Update hinweist. Aus dieser wird lediglich ersichtlich, dass ein Update fehlgeschlagen ist, detaillierte Gründe werden nicht angegeben.

Zusammenfassung und Rollen der untersuchten Software bei der Verwendung von Subversion

Im Folgenden sollen alle untersuchten Systeme noch einmal kurz unter Bezugnahme auf softwareergonomische Kriterien vorgestellt werden. Dabei wird auch das optimale Nutzungsszenario der Software dargestellt.

svn

Das Kommando `svn` bietet mit der Kommandozeile vertrauten Nutzern die vielfältigsten Möglichkeiten zum Umgang mit Subversion-Repositories. Unter softwareergonomischen Gesichtspunkten kann dieses Werkzeug als sehr gut beschrieben werden. Das Kommando ist in eine Reihe von Unterkommandos aufgeteilt, die jeweils eine Aufgabe erfüllen; mit `svn commit` werden Änderungen in das Repository eingestellt, eine Liste aller Befehle kann mittels `svn help` eingesehen werden und alle Befehle bieten eine detaillierte Hilfe. Optionen und Benennung der Parameter sind für die unterschiedlichen Unterkommandos weitgehend konsistent, Fehlermeldungen sind meist ausführlich und verständlich. Eine Anpassung an Benutzerwünsche ist mittels Konfigurationsdateien und Umgebungsvariablen möglich, ebenso bietet sich

wie bei allen Kommandozeilen-Werkzeugen eine Integration in Shellskripte oder Alias-Definitionen an.

TortoiseSVN

TortoiseSVN bietet sich an, wenn mit einem graphischen Werkzeug auf ein Repository zugegriffen werden soll, ohne dass explizit andere Programme aufgerufen werden. Durch die Integration in den Windows Explorer und die Hervorhebung von unter Versionskontrolle stehenden Dateien und Verzeichnissen ist der Zugriff auf ein Subversion Repository für jeden Anwender, der mit den grundlegenden Aktionen im Umgang mit einem Versionskontrollsystem vertraut ist, leicht zu erlernen. Eine Anpassung von TortoiseSVN ist möglich; die mittels des Kontextmenüs aufzurufenden Aktionen, die Darstellung des Programms und eine Vielzahl technischer Details lassen sich mittels Menüs festlegen. Problematisch sind zum Teil die verwendeten Fehlermeldungen(siehe hierzu die Erläuterung oben).

Subclipse

Subclipse bietet sich für den versionierten Umgang mit Dateien aus Eclipse-Projekten an. In Darstellung und Verhalten entspricht Subclipse weitestgehend der CVS Darstellung in Eclipse. Kritikpunkte sind die teilweise fehlende Transparenz bei der Nutzung der Verzeichnisversionierung, die Tatsache, dass eine Integration am Besten nicht nach, sondern vor einem Update vollzogen wird und die Gestaltung der Fehlermeldungen.

WebSVN

WebSVN bietet nur lesenden Zugriff auf ein Subversion Repository an. Die Software eignet sich in erster Linie, um ohne spezielle Software und ohne lokale Dateien Einsicht in das Repository zu nehmen. Die Benutzungsoberfläche von WebSVN wird über einen Browser bedient, die Nutzung ist intuitiv möglich.

Zusammengefasst lässt sich sagen, dass für Subversion eine Vielzahl unterschiedlicher Programme zum Zugriff auf ein Repository vorliegt. Diese Programme decken in ihrer Summe ein weites Feld an möglichen Formen ab, auf ein Repository zuzugreifen, so dass für unterschiedliche Erfordernisse und Vorlieben von Anwendern eine Lösung stets zu finden ist.

4.5 Fazit und Darstellung der zukünftigen Nutzung von Subversion

Bisher wurde in diesem Kapitel dargestellt, welche Faktoren bei der Überführung eines CVS Repository in das von Subversion verwendete Datenformat eine Rolle spielen und welche Schritte bei einer solchen Überführung not-

wendig sind. Darüber hinaus wurde dargestellt, wie eine Subversion-Installation serverseitig zu konfigurieren ist. Weiterhin wurde der Beschreibung von clientseitiger Software ein großer Bereich dieses Kapitels gewidmet. Hier wurde anhand des `svn` Kommandos, des Programms TortoiseSVN, der Eclipse Erweiterung Subclipse und der Webschnittstelle WebSVN dargestellt, welche Möglichkeiten es gibt, auf ein Subversion Repository zuzugreifen. Im Folgenden soll ein Konzept zur Einführung von Subversion in der Abteilung FuE dargestellt werden.

Vorgehen bei der Einführung von Subversion

Es erscheint sinnvoll, bei der Einführung von Subversion graduell vorzugehen. In diesem Sinn soll Subversion zuerst ausschließlich für Projekte und Entwickler eingeführt werden, die dessen über CVS herausgehende Funktionalität benötigen. Hierbei ist insbesondere an Projekte gedacht, in deren Rahmen ein Refactoring (siehe Fowler 1999) stattfinden soll. Da hierbei auch Veränderungen auf Verzeichnisebene notwendig sind, ist Subversions Möglichkeit zur Versionierung von Verzeichnissen in diesem Zusammenhang sehr nützlich.

Eine vollständige Einführung von Subversion anstelle von CVS und somit auch eine komplette Konvertierung des gesamten CVS Datenbestandes soll abhängig von den gemachten Erfahrungen zu einem späteren Zeitpunkt erfolgen.

Für diesen Ansatz sprechen in erster Linie zwei Argumente:

- Sollten unvorhergesehene Probleme bei der Verwendung von Subversion auftreten, so werden deren Auswirkungen begrenzt. Insbesondere kann notfalls auf die parallel verwendete CVS Infrastruktur zurückgegriffen werden.
- Eine graduelle Einführung erlaubt einen größeren Spielraum bei der Anpassung der Systemkonfiguration an die tatsächlichen Erfordernisse.

Ein Nachteil der zeitgleichen Nutzung von Subversion und CVS ist die Notwendigkeit, den Umgang mit beiden Systemen zu koordinieren. Dieses Problem wird aber mit dem hier angestrebten Vorgehen relativiert, da sich mit Subversion bearbeitete Projekte von mit CVS bearbeiteten Projekten abgrenzen lassen.

Notwendige Bestandteile einer Subversion-Infrastruktur

Um Subversion einführen zu können, werden theoretisch lediglich die Subversion-Kommandozeilen-Werkzeuge benötigt, um ein Repository zu erstellen und auf dieses zuzugreifen. Für ein ausbaufähiges System bietet sich jedoch die Verwendung eines zentralen Servers an. Dieser bietet einen zentra-

len Zugriffspunkt auf das Repository und vereinfacht somit die Administration des Repositories. Ein weiterer wichtiger Punkt, der bei der Einführung von Subversion bedacht werden muss, ist die Datensicherung.

Als Serverprozess erscheint es sinnvoll, von Anfang an den Apache-Server im Zusammenspiel mit dem Subversion Modul anstelle der Alternative SVNServe zu verwenden. Auch wenn die Vorteile des Apache-Moduls anfangs aller Wahrscheinlichkeit nach nicht nachgefragt werden, erlaubt dieses Vorgehen es doch, mit dem System Erfahrungen zu sammeln. Weiterhin vermeidet eine frühe Einführung einen notwendigen Umstieg von SVNServe, falls dessen Möglichkeiten nicht mehr ausreichen sollten.

Anfangs sollte eine generelle Rechtevergabe auf Projektebene gewählt werden, diese kann später differenzierter ausgestaltet werden um die tatsächlichen Erfordernisse wiederzugeben.

Neben der Einrichtung eines Servers muss die Konvertierung der ersten Projekte für die Verwendung mit Subversion vorbereitet werden. Hier bietet die anfängliche Beschränkung auf ausgewählte Projekte den Vorteil, dass der Konvertierungsprozess auf einem überschaubaren Datenbestand erfolgt. Dies erlaubt eine leichtere Kontrolle im Hinblick auf die Korrektheit der erzielten Ergebnisse. Sollte ein manueller Eingriff in die Konvertierung notwendig sein, bevor eine Anpassung der Konvertierungsskripte erfolgt ist, so wird dieser Vorgang ebenfalls durch einen kompakteren Arbeitsdatensatz erleichtert.

Bei der Installation der clientseitigen Software ist nicht mit Problemen zu rechnen. Das Aufspielen auf die Arbeitsplatzrechner erfolgt bei TortoiseSVN mittels eines Windows-Installationspakets, im Falle von Subclipse über den in Eclipse integrierten Updatemechanismus. Eine Konfiguration ist in beiden Fällen nicht nötig, lediglich die URL des zu verwendenden Repositories muss mitgeteilt werden. (Eine Darstellung der möglichen URL Formate bietet Tabelle 3.)

URL Format	Zugang zum Repository
`file:///`	Verwendung eines lokalen Repositories. Es wird kein Serverprozess verwendet
`http://`	Zugriff auf einen Apache-Server mit Subversion-Modul
`https://`	Repository auf einem Apache-Server mit SSL-Verschlüsselung
`svn://`	Verwendung von SVNServe
`svn+ssh://`	Durch SSH getunnelte Verbindung zu SVNServe

Tabelle 4: Unterschiedliche URL-Formate zum Zugriff auf Subversion Repositories

Neben oben aufgeführten Punkten muss noch eine Strategie zur Datensicherung festgelegt werden. Da Subversion die Möglichkeit bietet, ein Repository im laufenden Betrieb zu sichern und anfänglich nur mit einem vergleichsweise kleinen Datenbestand gearbeitet wird, bieten sich häufige Datensicherungen an. Generell sind Backups sowohl durch Hookskripte möglich als auch in vorgegebenen Zeitintervallen. Die sicherste Lösung liegt in einer automatischen Komplettsicherung mit jedem schreibenden Zugriff auf das Repository, ausgelöst durch einen Hookskript.

Ergänzende Funktionalität

Neben der oben dargestellten Funktionalität, die für die Verwendung von Subversion als Versionskontrollsystem zwingend notwendig, ist soll im Rahmen der Einführung von Subversion auch zusätzliche Software praktisch erprobt werden.

Dies ist zum einen WebSVN; diese Software bietet lesenden Zugriff auf Subversion Repositories. Zwei Aspekte von WebSVN sind im Hinblick auf ihre Nützlichkeit besonders interessant: die Information über Änderungen mittels eines RSS-Feeds und die Navigation von Änderungen. Die Verwendung von RSS-Feeds, die über jede Veränderung am Repository informieren, stellt eventuell eine Alternative zu den bisher automatisch erstellten Mails dar, die im Fall von Veränderungen am Repository versandt werden. Die Navigation auf den versionierten Daten mit WebSVN erlaubt es, unterschiedliche Revisionen in Beziehung zu setzen oder den Ursprung und die Historie einer Quellcodezeile nachzuvollziehen. Die Funktionalität wird dabei durch WebSVN mit einem erheblich geringeren Lernaufwand geboten, als dies durch die anderen Werkzeuge der Fall ist.

Neben der Verwendung WebSVN soll als weiterer Bestandteil einer Installation von Subversion auch erprobt werden, für welche Aufgabe sich Hookskripte anbieten. Zu den offensichtlichsten Aufgaben solcher Programme zählen die automatische Datensicherung und der Versand von Benachrichtigungen bei Veränderungen, während der Einführungsphase sollen aber auch weitere Einsatzmöglichkeiten geprüft werden.

Ein letzter Aspekt ist die Frage, in welcher Form sich die Verwendung der durch Subversion bereitgestellten Properties Funktionalität anbietet.

Weiteres Vorgehen nach einer ersten Erprobung

Sollte sich Subversion im Zusammenspiel mit den in diesem Text dargestellten clientseitigen Werkzeugen im Einsatz anhand ausgewählter Projekte bewähren, so sollte im Rahmen einer Vereinheitlichung des verwendeten Versionskontrollsystems eine Migration aller Projekte auf diese Software durchgeführt werden. Hierbei kann dann auf die konkreten Erfahrungen zurückgegrif-

fen werden, die anhand des Beispielprojektes gemacht wurden. Insbesondere soll bis zu diesem Zeitpunkt eine detaillierte Strategie für die Datensicherung und für die Rechtevergabe erarbeitet werden, ebenso wie für die Benachrichtigung bei Veränderungen. Weiterhin sollte die Frage geklärt werden, in welchem Umfang eine Konvertierung bestehender Repositories zu Subversion erfolgen soll, ob also alle Metadaten aus den CVS Repositories übernommen werden.

5 Fazit und Darstellung der Ergebnisse

Dieses Kapitel fasst die Ergebnisse dieses Arbeitsberichts zusammen. Dabei wird insbesondere auf das Ergebnis zur zukünftigen Nutzung von Subversion in der Abteilung FuE Stellung genommen.

Aufgaben und Besonderheiten von Versionskontrollsystemen

Das Angebot an Versionskontrollsystemen umfasst eine Vielzahl unterschiedlicher Systementwürfe und konkreter Implementierungen. Die grundlegenden Aufgaben von Versionskontrollsystemen wurden in Kapitel 1 dargestellt; sie stellen quasi die Grundfunktionalität dar, die von allen Versionskontrollsystemen erfüllt werden muss. Eine Gemeinsamkeit ist in diesem Sinn das Ziel, Gruppen von Entwicklern die Möglichkeit zu bieten, an einer Menge von Dokumenten kooperativ zu arbeiten und die Veränderungen an diesen Dokumenten dabei nachvollziehbar und reversibel zu gestalten. Abhängig vom jeweiligen Einsatzumfeld und den Anwendern können die praktischen Anforderungen an ein Versionskontrollsystem sehr unterschiedlich sein. Eine Bewertung von Versionskontrollsystemen muss daher stets vor diesem Hintergrund erfolgen.

Als Grundlage für die Auswahl eines Versionskontrollsystems für die Abteilung FuE wurde daher untersucht, unter welchen Rahmenbedingungen die Softwareentwicklung hier erfolgt. Dabei wurde insbesondere auch auf das aktuell verwendete Versionskontrollsystem CVS eingegangen. Es wurde festgestellt, dass das durch CVS vertretene Anwendungsmodell den vorliegenden Anforderungen weitgehend entspricht und dass in erster Linie Funktionalität zur Versionierung von Verzeichnissen vermisst wird. Ein weiteres relevantes Ergebnis war die Tatsache, dass eine Nutzung von CVS in erster Linie mittels der Entwicklungsumgebung Eclipse erfolgte.

Es wurden einige Versionskontrollsysteme exemplarisch unter Berücksichtigung einer Reihe von zuvor definierten Bewertungskriterien betrachtet. Hierbei wurden verbreitete Versionskontrollsysteme ebenso vorgestellt wie Software mit geringer Verbreitung, aber interessanten technischen Eigenschaften.

Gründe für die Auswahl von Subversion

Aus den untersuchten Versionskontrollsystemen wurde Subversion als gut geeigneter Nachfolger für eine auf CVS beruhende Infrastruktur ausgewählt. Die Gründe für diese Entscheidung werden im Kapitel 3.3 ausführlich erläutert, sollen hier aber noch einmal rekapituliert werden.

- Der primäre Punkt für den Einsatz von Subversion ist dessen enge Anlehnung an CVS, also die Absicht der Entwickler, das bestehende Anwendungsmodell größtenteils beizubehalten. Die Umstellung auf Subversion stellt auf Anwenderseite keine großen Anforderungen, hier wird bestehende Funktionalität im Wesentlichen erweitert. Insbesondere sind alle Werkzeuge für den Umgang mit CVS in ähnlicher Form auch für Subversion verfügbar, so dass eine Einbettung in Eclipse möglich ist.
- Eine Abweichung von CVS stellt lediglich die Vergabe der Bezeichnungen von Revisionen dar. Diese erfolgt bei Subversion auf Ebene eines Repositories, CVS vergibt Revisionsnummern hingegen für jede Datei einzeln.
- Serverseitig ist Subversion im Einsatz sehr flexibel. Es stehen zwei unterschiedliche Implementierungen von Servern zur Verfügung. SVNServe zeichnet sich durch höhere Performanz und leichtere Einrichtung aus, demgegenüber bietet eine Implementierung als Apache-Modul weitergehende Möglichkeiten zur Konfiguration.

Fazit

Subversion stellt eine gute Alternative zur Verwendung von CVS dar, wenn das grundsätzliche Anwendungsmodell eines Client-Server Versionskontrollsystems beibehalten werden soll, zugleich aber das Bedürfnis nach Möglichkeiten besteht, die von CVS nicht geboten werden. Subversion selbst, also die eigentliche Serversoftware und die Werkzeuge für die Kommandozeile, machten bei allen Tests einen sehr ausgereiften Eindruck. Insbesondere für den Einsatz in der Abteilung FuE ist die Verwendbarkeit des Subclipse Plugins von großer Relevanz. Zwar konnte dieses Plugin bei ersten Testfällen überzeugen, eine umfangreiche Überprüfung steht jedoch noch aus.

Es wird daher vorgeschlagen, bei der Einführung von Subversion schrittweise vorzugehen. Subversion sollte anfänglich für Projekte verwendet werden, die Bedarf an dessen neuer Funktionalität haben. Abhängig von den hier gemachten Erfahrungen sollte zu einem späteren Zeitpunkt entschieden werden, ob sämtliche Projekte nach Subversion überführt werden.

6 Literatur

Bar, M.; Fogel, K. (2003): Open Source Development with CVS. Paraglyph Press.

Carter, G. (2003): LDAP System Administration. O'Reilly.

Coar, K.; Bowen, R. (2003): Apache Cookbook. O'Reilly.

Codd, E. F. (1970): A relational model of data for large shared data banks In: Communications of the ACM, 13(6):377387.

Flanagan, D. (2002): Java in a Nutshell 4th Edition O'Reilly.

Fowler, M. (1999): Refactoring: Improving the Design of Existing Code Addison-Wesley.

Garman, J. (2003): Kerberos: The Definitive Guide. O'Reilly.

Hammersley, B. (2003): Content Syndication with RSS, O'Reilly.

Hegner, M. (2003): Methoden zur Evaluation von Software. Bonn: IZ-Sozialwissenschaften, Arbeitsbericht Nr. 29.
http://www.gesis.org/Publikationen/Berichte/IZ-Arbeitsberichte/pdf/ab_29.pdf

Laurie, P; Laurie, B. (2002): Apache: The Definitive Guide, O'Reilly.

Pilato, M.; Collins-Sussman, B.; Fitzpatrick, B.W. (2004): Version Control with Subversion, O'Reilly.

Sommerville, I. (2004) Software Engineering 7. Chapter 29. Addison-Wesley.

Viega, J.; Chandra, P.; Messier, M. (2002): Network Security with OpenSSL. O'Reilly