19

Werkzeuge zur Verwaltung von Projekten

Dieses Kapitel behandelt die folgenden Themen:

- Dokumentation und Strukturanalyse von Programmen
- Versionsverwaltung
- Projektverwaltung, Kommunikation mit Wiki

Es gibt viele Werkzeuge, um das Management von Projekten zu erleichtern. Dieses Kapitel bietet eine subjektive Auswahl von Open Source-Tools, die sich an vielen Stellen der Welt bewährt haben und deren Einsatz in kleinen und mittleren Projekten sinnvoll ist.

19.1 Dokumentation und Strukturanalyse mit doxygen

Ein Programm sollte gut dokumentiert sein, damit nicht nur der Autor, sondern auch andere, die mit ihm zusammenarbeiten oder irgendwann an seine Stelle treten, das Programm gut verstehen. Es ist es sehr mühevoll, eine separate Programmdokumentation

zu erstellen und zu pflegen. Insbesondere wird häufig vergessen, Änderungen des Programms in der Dokumentation nachzuziehen. Aus diesem Grund hat es sich eingebürgert, die Dokumentation direkt im Programm selbst vorzunehmen (single-source-Prinzip) und mit einem Werkzeug zu extrahieren. Die Dokumentation wird mitsamt den steuernden Anweisungen innerhalb von Kommentaren der jeweiligen Programmiersprache untergebracht, damit der Compiler sie ignoriert.

Doxygen (*http://www.doxygen.org*) ist ein sehr bekanntes und für diesen Zweck hervorragend geeignetes Open Source-Werkzeug, das für die Sprachen C++, C, Java, PHP und mehr ausgelegt ist. Doxygen führt eine statische Analyse der Quelldateien durch, sucht dabei nach Dokumentations-Schlüsselwörtern und erzeugt eine gut lesbare Ausgabe, die als Programmdokumentation geeignet ist. Wenn auf Ihrem System außer Doxygen auch Graphviz, ein Tool zur Visualisierung von Graphen (*http://www.graphviz.org*), installiert ist, erzeugt Doxygen auch Klassendiagramme und Graphen, die die gegenseitigen Abhängigkeiten zeigen. Ein *Aufrufgraph* für eine Funktion zeigt, welche anderen Funktionen von ihr aufgerufen werden. Ein *Aufrufergraph* zeigt hingegen, welche anderen Funktionen diese Funktion aufrufen.

Sowohl Doxygen als auch Graphviz sind für die gängigen Betriebssysteme erhältlich. In vielen Linux-Distributionen sind sie bereits enthalten, und für Windows gibt es leicht installierbare Binär-Dateien. Doxygen erzeugt auf Knopfdruck aus den Quelltexten eine aktuelle Dokumentation für ein ganzes Projekt, wahlweise im RTF-, HTML-, XML- oder TeX-Format.

Für die Dokumentation von Schnittstellen gibt es viele Möglichkeiten, die sich vom Konzept aber nicht unterscheiden, sodass ich mich hier auf eine beschränke:

- Die Dokumentation von Schnittstellen geschieht durch Voranstellen eines (geeigneten) Kommentars.
- Doxygen arbeitet ähnlich wie Javadoc (und umgekehrt). Es sei JAVADOC AUTOBRIEF=YES gesetzt. Damit wird die erste mit einem Punkt endende Zeile als Kurzbeschreibung aufgefasst.
- Ich verwende einen Stil, der auch für Java verwendet wird.
- Als Ausgabeformat wähle ich HTML.

Weitere Möglichkeiten bitte ich Sie, dem Doxygen-Manual zu entnehmen. Das Beispiel einer Musterlösung der Übungsaufgabe vier am Ende des Kapitels 4 zeigt, wie einfach die Kommentierung ist und was sich mithilfe von Doxygen daraus ergibt. Der Kommentar befindet sich *direkt vor* der jeweiligen Methode oder Klasse und beginnt mit `/**`, also zwei Sternchen. Die Schlüsselworte `@param` und `@returns` sind selbsterklärend. Es gibt einige andere, wie etwa `@author` und `@date`.

Listing 19.1: Methoden der Klasse Taschenrechner

```
// Auszug aus cppbuch/loesungen/k4/6/taschenrechner.h
  /** gibt das Ergebnis der Auswertung eines Summanden zurueck.
   * @param c erstes auszuwertendes Zeichen
   * @returns Ergebnis
   */
  long summand(char& c);

  /** gibt das Ergebnis der Auswertung eines Faktors zurueck.
```

```
   * @param c erstes auszuwertendes Zeichen
   * @returns Ergebnis
   */
  long faktor(char& c);
```

Die Abbildung 19.1 gibt einen Überblick über die Klasse, während die Abbildung 19.2 Einzelheiten einschließlich eines Aufrufgraphen zeigt.

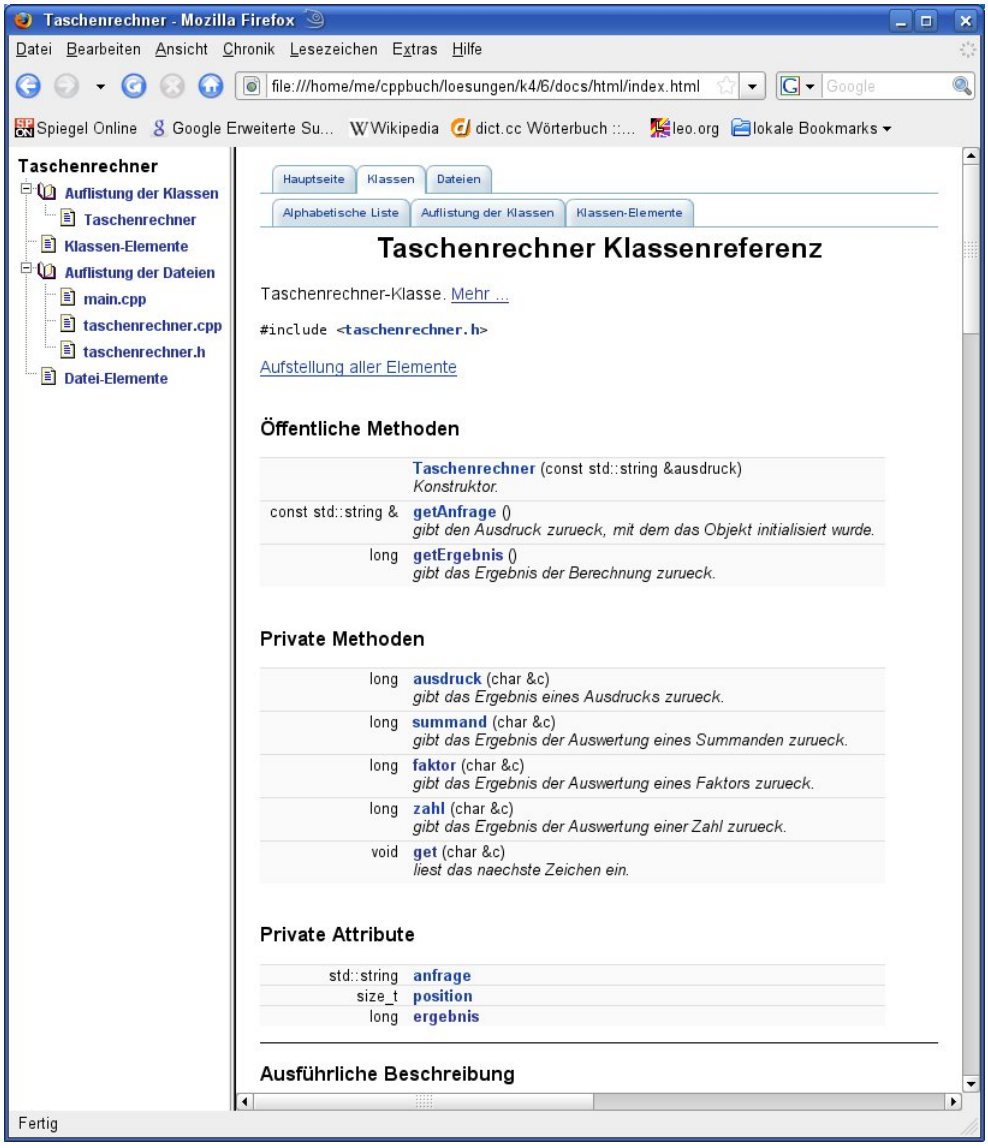

Abbildung 19.1: Auszug der HTML-Ausgabe von doxygen: Klassenüberblick

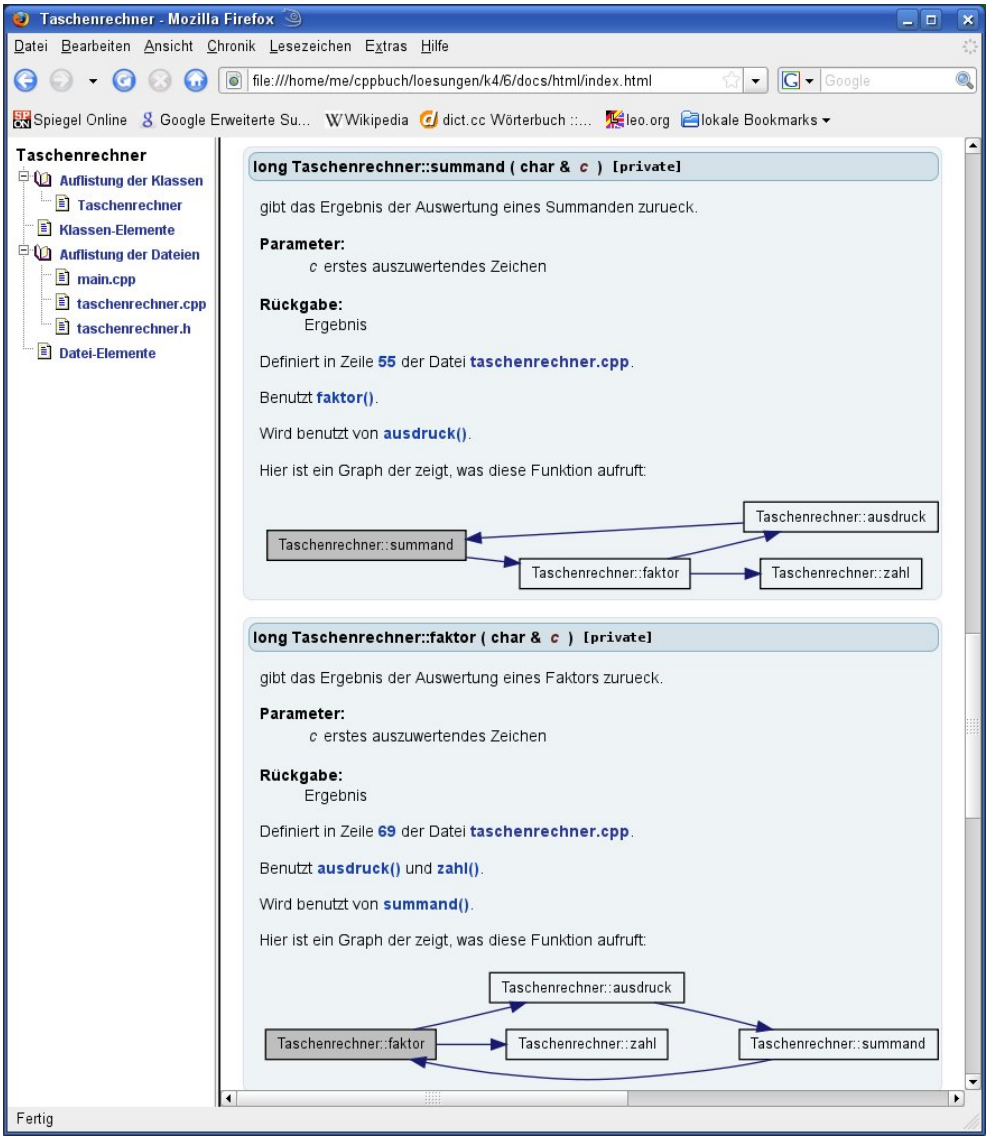

Abbildung 19.2: Auszug der HTML-Ausgabe von doxygen: Detail

Die Ausgabe wird erzeugt, indem im Verzeichnis *cppbuch/loesungen/k4/6/* das Kommando `doxygen` eingegeben wird. Doxygen wertet eine Konfigurationsdatei aus, die normalerweise vorher mit `doxywizard` oder mit `doxygen -g Doxyfile` erzeugt werden muss (im genannten Verzeichnis ist sie vorhanden). Diese Datei kann mit dem Wizard oder mit einem Editor bearbeitet werden. Der große Vorteil ist die Verlinkung aller Elemente, sodass man sehr leicht navigieren kann; auch der Quellcode wird mit einem Klick angezeigt. Die in der Header-Datei vorhandenen Kommentare brauchen in der Implementationsdatei nicht wiederholt zu werden. Dort können sich aber `/**` ... `*/`-Kommentare befinden, deren Inhalt zusätzlich angezeigt wird. Die Erzeugung von Graphen wird in der Konfigu-

rationsdatei (voreingestellter Name Doxyfile) durch den Schlüssel `HAVE_DOT` gesteuert, der mit dem Wert `YES` versehen werden muss. Der Name des Schlüssels rührt von der Komponente `dot` des Tools Graphviz her. Die Tabelle 19.1 enthält die wichtigsten Einstellungen der Konfigurationsdatei für eine Code-Analyse. Der Wert ist nach Wunsch jeweils auf `YES` oder `NO` zu setzen.

Tabelle 19.1: Die wichtigsten Einstellungen in der Konfigurationsdatei

Schlüssel	Bedeutung
`CALL_GRAPH`	Aufrufgraph erzeugen.
`CALLER_GRAPH`	Aufrufergraph erzeugen.
`CLASS_DIAGRAMS`	Klassen- und Vererbungshierarchiediagramme erzeugen.
`EXTRACT_ALL`	Alle vorhandenen Informationen extrahieren.
`GENERATE_HTML`	HTML-Dateien erzeugen.
`GENERATE_LATEX`	Latex-Dateien erzeugen.
`HAVE_DOT`	Grafiken erzeugen (setzt Graphviz voraus).
`SOURCE_BROWSER`	Vollständige Verlinkung erzeugen.

19.1.1 Strukturanalyse

Doxygen analysiert Programmcode und erzeugt eine Ausgabe, auch wenn keinerlei Dokumentationsanweisungen enthalten sind. Damit ist Doxygen ein hervorragendes Tool zur Analyse von Programmen, auch wenn sie schlecht oder nicht dokumentiert sind.

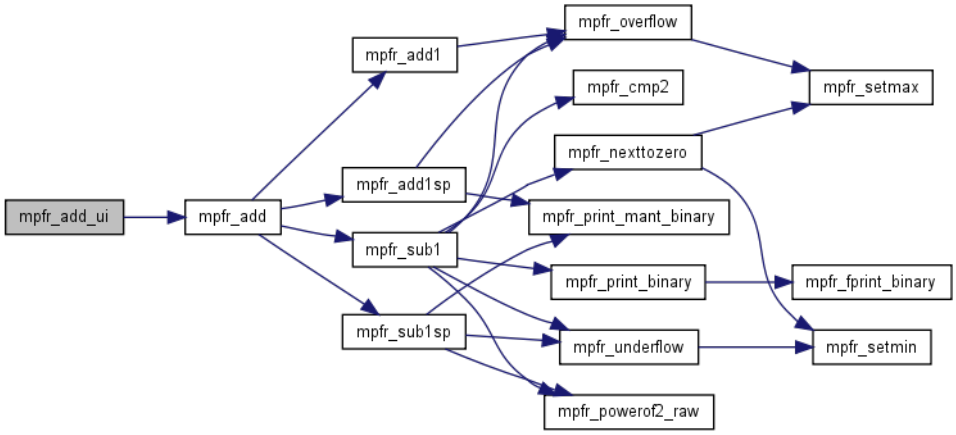

Abbildung 19.3: Aufrufgraph der MPFR-Library für die Funktion mpfr_add_ui

Wer sich jemals in ein unbekanntes Programm mit vielen Dateien einarbeiten musste, wird einen Aufruf- und einen Aufrufergraphen schätzen. Das obige Programmbeispiel ist zu einfach, daher zeigen die Abbildungen 19.3 und 19.4 als Beispiel die mit Doxygen erzeugten Graphen für die Funktion `mpfr_add_ui` der MPFR[1]-Bibliothek, die vom G++-Compiler benötigt wird.

[1] Multiple-Precision Floating-point computations with correct Rounding

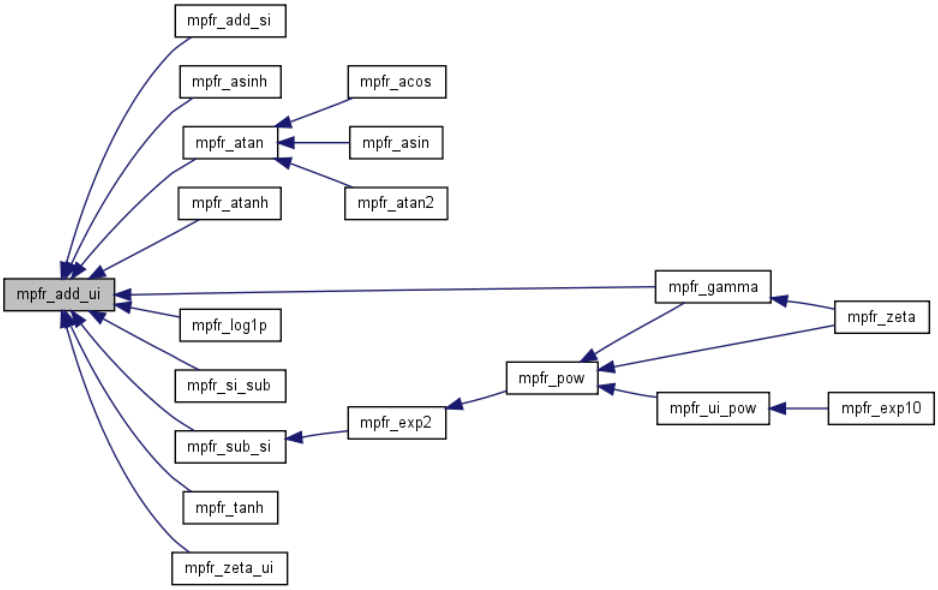

Abbildung 19.4: Aufrufergraph der MPFR-Library für die Funktion mpfr_add_ui

19.2 Versionskontrolle

Ebenso wichtig wie eine gute Dokumentation ist eine Versionskontrolle. Sie erlaubt es, einen vergangenen Stand einer Datei zu rekonstruieren oder kundenspezifische Varianten des Programmcodes zu verwalten. CVS (Concurrent Version System) war früher das Open Source-Werkzeug der Wahl. Vor einigen Jahren wurde CVS nach und nach durch Subversion abgelöst, ebenfalls Open Source-Software [svn]. Subversion verwaltet alle Dateien in einem sogenannten Repository, welches datenbank- oder dateisystembasiert sein kann. Dabei geht es nicht nur um Programmcode, sondern um alle zum Projekt gehörenden Dateien einschließlich Bilddateien. Wenn geänderte Dateien in das Repository eingespeist werden, erhöht sich die Änderungsnummer, Revision genannt, um 1. Einer Revison ist ein Datum und eine Uhrzeit zugeordnet. Subversion erlaubt es, die Unterschiede zwischen Revisionsständen festzustellen.

Typischerweise wird ein ganzes Projekt in Form eines Verzeichnisbaums im Repository abgelegt. Beim ersten Mal wird dieser Baum vollständig kopiert. Bei späteren Änderungen jedoch werden nur noch die *Unterschiede* nach einem gut durchdachten Schema abgelegt. Jeder Revision ist eine geschickt verlinkte Baumstruktur zugeordnet. Dadurch bedingt wird viel Speicherplatz gespart.

Die Abbildung 19.5 zeigt die Subversion-Architektur. Als Repository ist sowohl eine Datenbank als auch ein Filesystem möglich. Es gibt mehrere Wege der Kommunikation des Clienten mit dem Subversion-Server, gekennzeichnet durch das Protokoll:

- `file://` Direkter Zugriff auf ein Repository, das sich auf einer lokalen Festplatte befindet. Zum Ausprobieren ist dieser Weg gut geeignet. Er hat aber den Nachteil, dass

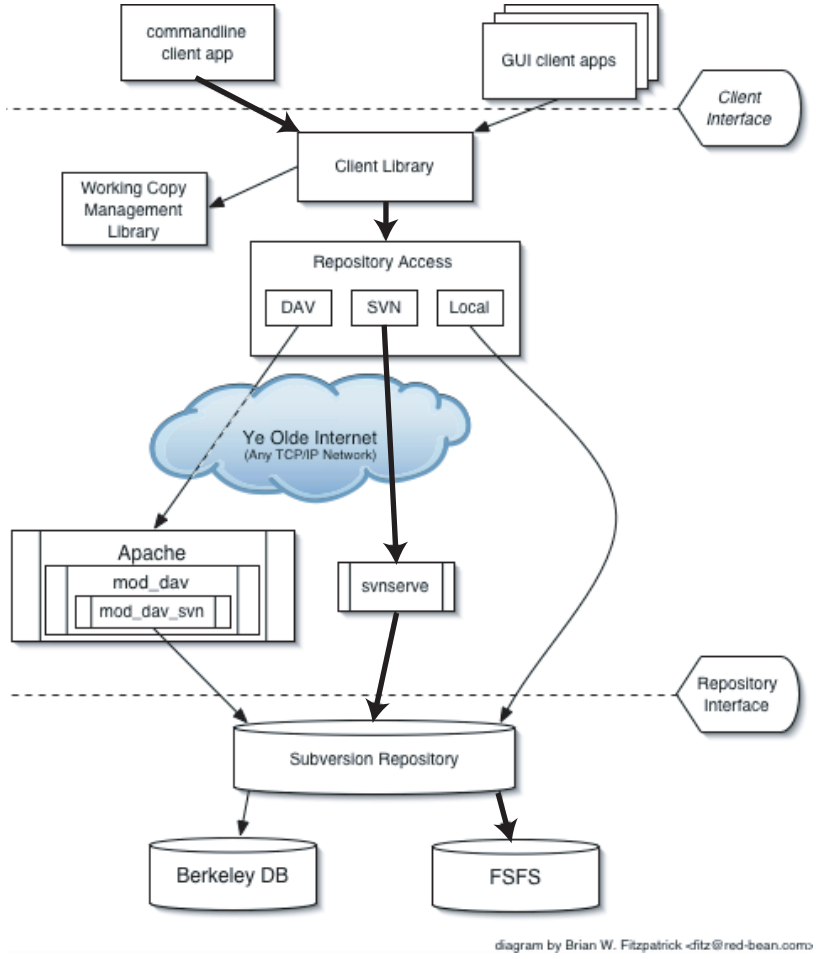

Abbildung 19.5: Subversion-Architektur (aus [SFP], einige Pfeile verstärkt)

es keine Datensicherheit bei einem Ausfall der Festplatte gibt. Wenn man eine andere Festplatte wählt, besteht die Möglichkeit des Ausfalls des Festplatten-Controllers. Immerhin kann man den ersetzen oder die Platte in einen anderen Rechner einbauen, um wieder an die Daten zu kommen.

- `http://` Zugriff über das WebDAV-Protokoll, das vom Apache-Server unterstützt wird. Der Apache-Server muss natürlich eingerichtet sein.
- `https://` Wie `http://`, aber mit SSL-Verschlüsselung (Secure Sockets Layer).
- `svn://` Subversion stellt einen eigenen Server zur Verfügung. Damit kann die Installation des Apache-Servers entfallen, falls er nicht für andere Zwecke gebraucht wird. `svn://` ist ein Subversion-eigenes Protokoll. Wie bei `http://` gehen alle Informationen im Klartext über die Leitung, auch Passwörter. `svn://` läuft über Port 3690.
- `svn+ssh://` Wie `svn://`, aber über SSH (Secure Shell). SSH ist ein Protokoll, das nicht nur die Verbindung verschlüsselt, sondern auch eine Authentifizierung verlangt. SSH

ist gleichzeitig auch der Name eines Programms, mit dem man sich über eine sichere Verbindung auf einem anderen Rechner einloggen kann. Das Kommando ist `ssh -X -l username hostname`. Der Parameter `-X` ist optional. Er sorgt dafür, dass man Anwendungen (Editor ...) des entfernten Rechners aufrufen kann, die das X-Window System nutzen. SSH benutzt den Port 22.

Wenn aus Datensicherheitsgründen ein entfernter Rechner mit einer verschlüsselten Verbindung und gleichzeitig eine möglichst einfache Konfiguration gewählt werden soll, bietet sich für kleine Teams die zuletzt genannte Möglichkeit an, wobei das Filesystem des Servers verwendet wird. Dem entspricht der dick in Abbildung 19.5 eingezeichnete Weg. Er wird im Folgenden exemplarisch dargestellt. Etwas ansprechender in der Benutzung als die Kommanozeile sind die verschiedenen für Subversion erhältlichen GUI-Clients wie TortoiseSVN oder RapidSVN. Für manche Entwicklungsumgebungen gibt es Plug-ins. SVN ist die Kurzform von Subversion.

Hinweis

Aus Platzgründen wird in diesem Abschnitt nur kurz auf die wichtigsten Möglichkeiten von Subversion eingegangen. Mehr dazu können Sie im SVN-Buch [SFP] lesen, das sich auf der zu diesem Buch gehörenden DVD befindet. Auf der Internetseite [svn] gibt es Links zu weiteren hilfreichen Informationen.

Arbeitsweise aus Benutzersicht

Normalerweise sind an einem Projekt mehrere Personen beteiligt, denen bestimmte Bereiche des Projekts zugeordnet sind. Am Ende des Projekts könnte die Integration der Bereiche zum Beispiel ein neues Produkt ergeben. Nachdem das Projekt der Versionsverwaltung unterliegt, kann sich jeder Softwareentwickler die aktuelle Kopie herunterladen, die damit seine Arbeitskopie wird. Dieser Vorgang wird `checkout` genannt. Nach der Arbeit an einem Modul, einschließlich Integration und Test (sonst gibt es Schelte von den Kollegen ...), wird es mit `commit` in das Repository eingespeist. Jedem anderen steht dann das geänderte Modul nach einem `update` zur Verfügung.

19.2.1 Einrichtung des Servers

In diesem Abschnitt beschreibe ich die Einrichtung auf einem Linux-System, weil Linux-Rechner als Server verbreitet sind. Die Einrichtung als Windows Server bitte ich Sie, der Literatur zu entnehmen, zum Beispiel [SFP], Kapitel 6. Zunächst einmal müssen SSH und Subversion installiert sein. Dies ist bei den meisten Linux-Distributionen bereits der Fall oder mithilfe des Systemverwaltungstools leicht nachzuholen. Eine weitere Möglichkeit ist das Herunterladen der Quellen. Nach Entpacken wird Subversion wie üblich mit `./configure` und `make` übersetzt und mit `make install` installiert. Bitte achten Sie darauf, dass die auf dem Server installierte Version von Subversion mit der auf dem Clienten wenigstens in der Ziffer nach dem Punkt übereinstimmt, d.h. nicht auf dem einen Rechner Version 1.4 und auf dem anderen 1.5 verwenden.

Bevor Sie an die Einrichtung des Servers gehen, probieren Sie bitte aus, ob der Login auf dem Server mit `ssh` funktioniert. Dies kann direkt am Server geschehen, wenn als Hostname `localhost` oder `127.0.0.1` eingegeben wird. Falls die Verbindung nicht zustan-

de kommt, liegt es vermutlich daran, dass die Firewall eingeschaltet ist und den entsprechenden Port (siehe oben) sperrt. Gegebenenfalls muss der Port freigeschaltet werden. Bei Computern, die über einen Router mit dem Internet verbunden sind, kann die Firewall ganz abgeschaltet werden, wenn der Router die Firewall-Aufgabe übernimmt.

Wegen der von SSH verlangten Authentifizierung sollten auf dem Server spezielle Benutzerkonten nur für den SVN-Zugriff angelegt werden. Der Grund: Weil diese User Lese- und Schreibrechte im SVN-Bereich haben, können sie Informationen anderer SVN-User lesen. Danach kann der Server mit den folgenden Schritten eingerichtet werden:

- Als Root-User das Wurzelverzeichnis der Repositories anlegen, z. B. */var/svn/repos*. Für das zu verwaltende Projekt mit dem Namen `projekt` (es kann hier ein anderer Name gewählt werden) wird das Repository mit
  ```
  svnadmin create /var/svn/repos/projekt
  ```
  angelegt. Svnadmin erzeugt dabei verschiedene Unterverzeichnisse.
- In */var/svn/repos/projekt/conf* wird die Datei *passwd* modifiziert, sodass die Benutzer mit den Passwörtern unter der Gruppe `[users]` aufgeführt werden. Die Syntax ist `username = passwort`. Wegen der späteren Authentifizierung ist es am einfachsten, wenn hier das Login-Passwort verwendet wird. Dies ist nicht problematisch, wenn diese User nur für SVN verwendet werden. Inhalt zum Beispiel:
  ```
  [users]
  hansi = hansipasswort
  leila = leilapasswort
  # usw.
  ```
- In die Datei *svnserve.conf* im selben Verzeichnis wird unter `[general]` eingetragen:
  ```
  # Anonyme duerfen nichts
  anon-access = none
  # Autorisierte duerfen lesen und schreiben
  auth-access = write
  # Verweis auf die Passwortdatei:
  password-db = passwd
  # realm ist ein Namensbereich fuer die Authentifizierung.
  realm = Projekt Wunderbar
  ```
- Die Ausführung der SVN-Programme sollte durch einen speziellen User svn nur für diesen Zweck erfolgen, um andere Bereiche zu schützen. Ebenfalls wird eine Gruppe svn angelegt:
  ```
  groupadd svn
  useradd -G svn svn
  ```
- Den Usern auf dem System, die SVN benutzen, diese Gruppe zusätzlich zuordnen. Mit `usermod -A svn username` wird `svn` dem User hinzugefügt. Falls die Option `-A` nicht zur Verfügung steht (`man usermod` und `usermod -help` geben bei meinem System nicht dieselben Auskünfte), hilft der folgende Umweg: `groups username` gibt alle Gruppen aus. Mit `usermod -G svn,... username` werden `svn` und alle anderen Gruppen dem Benutzerkonto `username` hinzugefügt. Die Ellipse `...` ist dabei durch die mit `groups username` ermittelten Gruppen zu ersetzen, falls außer `svn` weitere Gruppen gewünscht sind, für den Fall, dass es sich doch nicht nur um reine SVN-User handeln sollte. Die Gruppen sind ohne Leerzeichen nach dem Komma aufzuführen.

- Dann werden von Root die Rechte gesetzt, wobei die letzte Zeile den Zugriff von anderen verhindert (o = others).

```
cd /var
chown –R svn.svn svn/
chmod –R ug+rw svn/
chmod –R o–rwx svn/
```

- Den Netzwerkdienst mit einem Hilfsprogramm des Betriebssystem (yast2 bei SusE-Linux) einstellen, damit bei einer SVN-Anfrage der Dienst `svnserve` gestartet wird. In der Datei */etc/xinetd.d/svnserve* (SuSe-Linux) könnte danach Folgendes stehen (für Ihr System anpassen):

```
service svnserve
{
   socket_type    = stream
   protocol       = tcp
   wait           = no
   user           = svn
   group          = svn
   server         = /usr/bin/svnserve
   server_args    = svnserve –i –r /var/svn
}
```

  `/usr/bin/svnserve` ist das aufzurufende Programm, der Ort könnte je nach Installation auch zum Beispiel `/usr/local/bin/svnserve` sein. Der Parameter `-i` steht für die Xinet-Einbindung, der Wert nach dem Parameter `-r` gibt das Wurzelverzeichnis für SVN an.

- Um in jedem Fall Probleme mit Zugriffsrechten zu vermeiden, empfehlen die Autoren von [SFP], die svn-Programme umzubenennen und über ein Skript aufzurufen. Beispiel: *svnserve* wird in *svnserve-real* umbenannt. Dann wird ein Skript *svnserve* mit folgendem Inhalt angelegt:

```
#! /bin/sh
umask 002
/usr/bin/svnserve–real "$@"
```

  Dieses Skript wird mit `chmod ug+x` ausführbar gemacht und tritt damit an die Stelle des ursprünglichen Programms. Der Sinn ist, mit `umask 002` (= user mask) die Schreibrechte anderer grundsätzlich zu verbieten, egal welche Rechte der ausführende User hat. `umask` verwendet dieselben Zahlenkombinationen wie der Befehl `chmod`, nur dass `umask` die angegebene Berechtigung ausblendet. Wer keine Lust hat, mit Oktalzahlen zu rechnen, kann sich des Links *http://javascriptkit.com/script/script2/chmodcal.shtml* bedienen.

- Zum Schluss kann mit `svn list svn+ssh://127.0.0.1/var/svn/repos/projekt` auf dem Server geprüft werden, ob `svn+ssh` funktioniert. Nach Passwort-Eingabe muss das Kommando ohne Fehlermeldung zurückkommen. Eine Ausgabe gibt es nicht – das Verzeichnis ist ja noch leer.

19.2.2 Exemplarische Benutzung

Vor dem ersten Import des Projektes sollten alle Dateien, die nicht der Versionsverwaltung unterliegen sollen, entfernt werden. Das Verzeichnis *projekt/trunk* enthalte den

Hauptentwicklungsstamm (zu den Verzeichnissen *branches* und *tags* siehe [SFP]). Der erste Import geschieht mit:

```
svn import lokalesVerzeichnis svn+ssh://username@hostname/var/svn/repos/projekt \
  -m "erster Import"
```

`lokalesVerzeichnis` ist das zu übertragende Projektverzeichnis, statt `username` muss der Name eines auf dem Server eingerichteten Benutzers verwendet werden, und `hostname` ist der Rechnername, der auch eine IP-Adresse sein kann, etwa 192.168.1.20 in einem Intranet. Der mit dem Parameter `-m` übergebene Kommentar ist zwingend erforderlich; wird er vergessen, wird automatisch der Standardeditor des Betriebssystems zwecks Eingabe gestartet. Natürlich wird auch das Passwort verlangt, bei manchen SVN-Kommandos auch zweimal (Login- und SVN-Passwort). Der Einfachheit halber werden Passworteingaben im Folgenden weggelassen.

```
svn -v list svn+ssh://username@hostname/var/svn/repos/projekt
```

zeigt die unter *projekt* existierenden Verzeichnisse an, zum Beispiel *trunk*. Die Option `-v` sorgt für eine ausführliche Ausgabe mit Revision, User und Datum, was bei späteren Aufrufen interessant sein kann. Das Versionsverwaltungssystem ist jetzt fertig eingerichtet.

Aus- und Einchecken

Jedes berechtigte Projektmitglied kann sich jetzt seine aktuelle Arbeitskopie mit

```
svn co svn+ssh://username@hostname/var/svn/repos/projekt projekt
```

herunterladen. `co` ist die Abkürzung für `checkout` Das Arbeitsverzeichnis wird am Ende der Zeile angegeben (*projekt*); es kann auch anders heißen. In jedem Verzeichnis der Arbeitskopie befinden sich Verzeichnisse mit dem Namen *.svn*. Sie dienen der SVN-Verwaltung und dürfen nicht geändert oder gelöscht werden. Nun kann im Verzeichnis *trunk* gearbeitet werden. Nach Abschluss der Arbeiten wird der neue Stand mit

```
svn commit projekt -m "erstes Commit"
```

gesichert. Es wird hier dabei ausgegangen, dass das zu sichernde Arbeitsverzeichnis *projekt* heißt, entsprechend dem `checkout` von oben. Dabei werden nur die geänderten Daten übertragen. Die Begründung für die Änderung muss wie oben nach `-m` angegeben werden. Wenn nun ein anderes Projektmitglied, das bereits einmal das Projekt ausgecheckt hat, `svn up projekt` aufruft, wird der aktualisierte Stand auf seinen Rechner übertragen. Das Wort `up` steht für update.

Wenn versehentlich etwas gelöscht wurde (oder aus anderen Gründen), kann es helfen, einen alten Stand des Projektes zu rekonstruieren.

```
svn co -r 44 svn+ssh://username@hostname/var/svn/repos/projekt rev44
```

legt zum Beispiel eine vollständige Kopie der Revision 44 im Verzeichnis *rev44* an. Es können auch einzelne Dateien rekonstruiert oder anstelle der Revision kann ein Datum gewählt werden ([SFP]).

Dateien hinzufügen und entfernen

Neue Dateien werden nicht automatisch der Versionsverwaltung hinzugefügt, um nicht temporäre oder andere weniger wichtige Dateien zu speichern. Neue Dateien müssen

explizit mit `add` angemeldet und zu löschende Dateien mit `rm` (= remove) abgemeldet werden:

```
svn add projekt/trunk/neuedatei
svn rm projekt/trunk/altedatei
```

Beide Aktionen wirken ab dem nächsten `commit`.

Schlüsselwortersetzung

Es ist sinnvoll, in einer Datei selbst den aktuellen Stand zu dokumentieren. Dazu bietet SVN einige Schlüsselwörter an: `Date` liefert Datum und Uhrzeit, `Author` den User-Namen, `HeadURL` die Adresse des Dokuments, und `Rev` ist die Revision. `Id` ist eine Kurzfassung von allem. Dazu werden die Schlüsselworte, durch Dollarzeichen begrenzt, in den Kommentarbereich einer Datei eingetragen. Beispiel einer C++-Datei *main.cpp*:

```
// $Id$
// $Rev$
// $Date$
// $Author$
// $HeadURL$
// ...
{
 // Programmcode
}
```

Der Wunsch nach einer Schlüsselwortersetzung in allen .cpp-Dateien wird dem SVN-System einmalig mit

```
svn propset svn:keywords "Date Rev Author Id HeadURL" projekt/trunk/*.cpp
```

mitgeteilt. Die Eintragungen werden beim nächsten `commit` wirksam und beim nächsten `update` expandiert. Danach sieht der obige Programmauszug etwa so aus:

```
// $Id: main.cpp 6 2008-11-04 18:07:48Z username $
// $Rev: 6 $
// $Date: 2008-11-04 19:07:48 +0100 (Di, 04. Nov 2008) $
// $Author: username $
// $HeadURL: svn+ssh://username@hostname/var/svn/repos/projekt/trunk/main.cpp$
// ...
{
  // Programmcode
}
```

Die Zahl 6 nach main.cpp im Id-Bereich ist die Revision. Die Zusammenstellung zeigt nur die verschiedenen Möglichkeiten, in der Praxis wird man entweder `$Id$` *oder* die anderen Schlüsselwörter nehmen. Damit sind die wichtigsten Möglichkeiten von SVN beschrieben. Es gibt aber noch sehr viele weitere, die hier nicht behandelt werden können und die anderweitig gut dokumentiert sind.

Mehr zur Versionsverwaltung mit Subversion lesen Sie im SVN-Buch auf der DVD.

19.3 Projektverwaltung

Außer Versionen zu verwalten gibt es in einem Projekt noch andere Aufgaben: Organisation der Kommunikation im Projekt, Termine verfolgen, Kosten kontrollieren, Personaleinsatz planen. In diesem Abschnitt stelle ich Open Source-Werkzeuge vor, die dabei helfen können.

19.3.1 Projektmanagement

Seit mehr als fünf Jahren gibt es eine leistungsfähige Open Source-Konkurrenz zu den kommerziellen Werkzeugen für die Dokumentation und Steuerung von Zeitplänen und Finanzen eines Projekts: Open-Project (*http://openproj.org/openproj/*) und Project-Open (*http://www.project-open.com/*). Beide Software-Pakete laufen auf Windows-, Linux- und Mac-Betriebssystemen. Sie bieten die Möglichkeit, Arbeitspakete zu bilden und zu strukturieren. Die Arbeitspakete und ihre Abhängigkeiten werden terminlich eingeplant. Die Terminlage wird mit Balkendiagrammen (Gannt[2]-Diagrammen) visualisiert. Project-Open hat auf der Homepage allerdings einiges mehr an Dokumentation zu bieten und hat nach eigenen Angaben mehr als 1000 Firmen als Nutzer. Wie bei anderen Open Source-Projekten wird nicht an der Software, sondern der begleitenden Beratung verdient.

19.3.2 Wiki für Software-Entwicklungsprojekte

Ein Wiki ist eine Online-Plattform zur Kommunikation zwischen Menschen mit ähnlich gelagerten Interessen. Ein Wiki zeichnet sich dadurch aus, dass es als Hypertextsystem aufgebaut ist, in dem seine Benutzer nicht nur lesen, sondern auch schreiben können. Zur Textformatierung wird eine Sprache benutzt, die einfacher als HTML ist. In einem Wiki können Dokumente abgelegt und von mehreren Autoren bearbeitet werden. Es sind Diskussionsforen und vieles andere möglich. Das bekannteste Wiki ist vermutlich Wikipedia, eine beliebte Auskunftsquelle mit zig-tausenden Autoren. Ein Wiki ist eine ideale Plattform zur Kommunikation innerhalb eines Projekts aus folgenden Gründen:

- Die Mitarbeiter eines Projekts müssen sich nicht in räumlicher Nähe befinden, sodass eine weltweit verteilte Entwicklung beinahe so einfach ist, als würden alle am selben Ort arbeiten.
- Der Zugriff ist asynchron, das heißt, dass jeder dann über das Wiki kommuniziert, wenn es der Arbeitsablauf am besten erlaubt. Das soll nicht bedeuten, dass persönliche Treffen überflüssig sind – sie sind nur nicht so oft erforderlich.
- Ein Wiki ist ein Content Management System, in dem Dokumente archiviert werden. Gleichzeitig hat es eine eingebaute Versionsverwaltung für die im Wiki erzeugten Inhalte, sodass Vorgängerversionen eines Artikels gesehen und rekonstruiert werden können.
- Manche Wikis erlauben die leichte Integration einer Versionsverwaltung für Software auf eine Weise, dass das Repository mit dem Browser eingesehen werden kann.

Wikis sind gerade in der Open Source-Entwicklung sehr beliebt. Eine beeindruckende Zahl von Wikis der Apache Community ist unter *http://wiki.apache.org/general/FrontPage*

[2] Henry Gannt hatte diese Art der Darstellung vor etwa 100 Jahren bekannt gemacht.

aufgeführt. Wenn Sie ein Wiki in einem Projekt einsetzen, empfiehlt sich die MoinMoin Wiki Engine (*http://moinmo.in/*), mit dem die erwähnten Apache Wikis und viele andere realisiert wurden.

Eine ebenfalls empfehlenswerte Alternative ist Trac (*http://trac.edgewall.org/*). Trac ist bereits für Softwareprojekte zugeschnitten. Es stellt eine Schnittstelle für Subversion zur Verfügung, wobei sich das Repository allerdings auf derselben Maschine wie Trac befinden muss. Trac erlaubt mithilfe von sogenannten Tickets die Verfolgung von Aufgaben und Terminen. Die Aufgabenverfolgung mit Tickets erlaubt auch eine Fehlerverfolgung, sodass dafür spezialisierte Werkzeuge wie Mantis (*http://www.mantisbt.org/*) nicht notwendig sind. Trac wird von einer großen Anzahl Firmen eingesetzt. Die Abbildung 19.6 zeigt die Trac-Wiki-Titelseite eines studentischen Projekts, in dem auch die genannten Werkzeuge Doxygen und Subversion zum Tragen kamen.

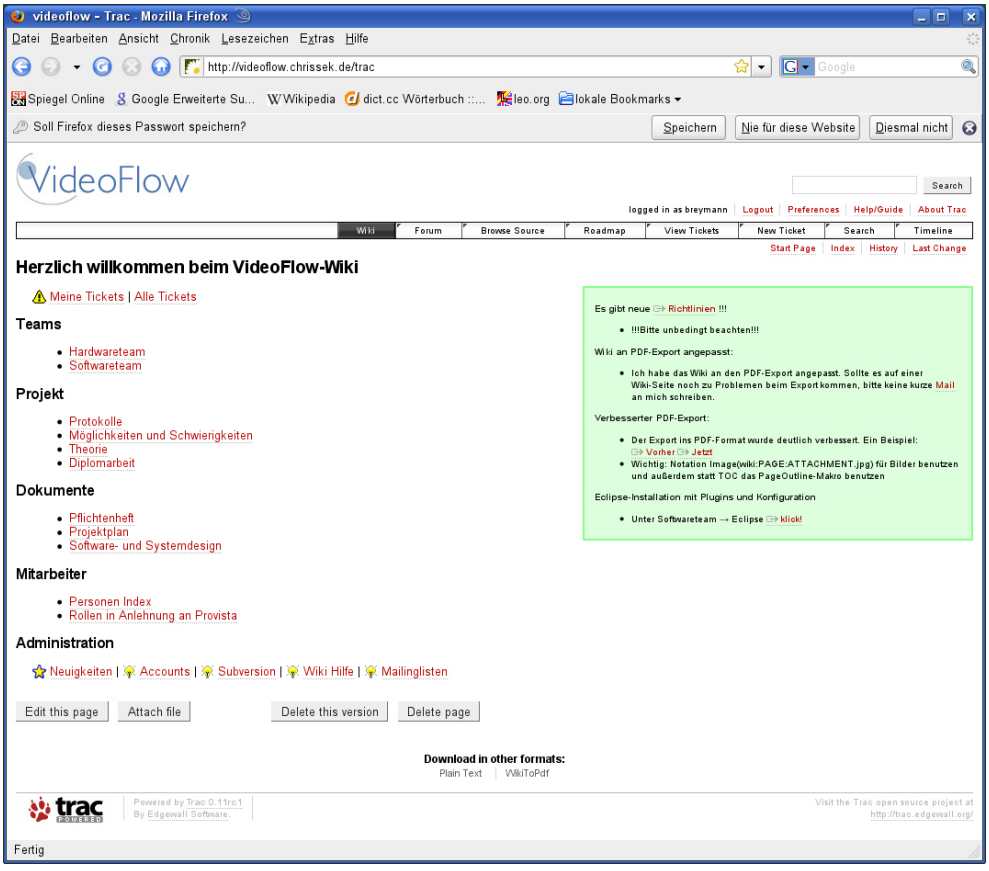

Abbildung 19.6: Trac-Hauptseite eines Projekts