Kapitel 1

Einführung

Dieses Buch zeigt Ihnen, wie Sie Ihren Softwareentwicklungsprozess mit Versionsverwaltung effektiver gestalten können.

Versionsverwaltung, die manchmal auch Quelltextverwaltung genannt wird, ist der erste Teil unserer Trilogie zur Projektunterstützung. Unserer Meinung nach ist der Einsatz einer Versionsverwaltung bei allen Projekten zwingend notwendig.

Versionsverwaltung bietet viele Vorteile sowohl für Teams als auch für einzelne Entwickler:

- Das Team hat die Möglichkeit, Änderungen projektweit rückgängig zu machen: Nichts ist endgültig und bei einem Fehler geht man einfach auf den vorherigen Stand zurück. Stellen Sie sich vor, Sie verwenden die ausgefeilteste Textverarbeitung der Welt. Diese besitzt jede nur vorstellbare Funktion außer einer: Aus irgendeinem Grund wurde die ENTFERNEN-Taste vergessen. Wie sorgfältig und langsam müssten Sie nun tippen, besonders wenn Sie sich dem Ende eines großen Dokumentes nähern. Ein Fehler und Sie müssen von vorn beginnen. Mit einer Versionsverwaltung ist es das gleiche. Die Möglichkeit, jederzeit eine Stunde, einen Tag oder eine Woche zurück zu gehen, wirkt befreiend. Ihr Team kann deutlich schneller arbeiten, denn es hat immer die Sicherheit, dabei entstehende Fehler schnell beheben zu können.
- Mehrere Entwickler können auf kontrollierte Weise zeitgleich an den gleichen Quelltexten arbeiten. Das Team verliert keine Änderungen mehr, weil jemand die Änderungen eines anderen überschreibt.
- Die Versionsverwaltung protokolliert die gemachten Änderungen und ihre zeitliche Abfolge. Wenn Sie über „seltsamen Quell-

text“ stolpern, ist es leicht herauszufinden, wer die Änderungen wann gemacht hat und mit etwas Glück auch, warum.

- Mit einer Versionsverwaltung ist die Arbeit an mehreren Releases Ihrer Software zur gleichen Zeit möglich, während Sie parallel dazu die Entwicklung auf dem Hauptzweig fortsetzen. Für das Team gibt es keinen Grund mehr, während eines *Code Freeze*[1] kurz vor einem Release die Arbeiten zu unterbrechen.
- Eine Versionsverwaltung ist eine projektweite *Zeitmaschine*. Sie erlaubt uns, ein Datum vorzugeben und den Projektstand an genau diesem Datum zu betrachten. Für Nachforschungen ist das hilfreich. Für die Wiederherstellung eines vorangegangenen Releases, mit dem ein Kunde Probleme hat, ist es notwendig.

Im Mittelpunkt dieses Buches steht Versionsverwaltung im Rahmen von Projekten. Anstatt einfach alle Befehle aufzulisten, betrachten wir die Aufgaben in einem erfolgreichen Projekt und sehen uns dann an, wie eine Versionsverwaltung dabei helfen kann.

Wie funktioniert Versionsverwaltung in der Praxis? Fangen wir mit einer kleinen Geschichte an ...

1.1 Versionsverwaltung in Aktion

Fred kommt motiviert ins Büro, um seine Arbeit am neuen Buchbestellsystem Orinoco fortzusetzen. (Warum Orinoco? Die Firma von Fred benennt interne Projekte nach Flüssen.) Nachdem er seine erste Tasse Kaffee geholt hat, bringt Fred seine lokale Kopie des Projektquelltextes mit Hilfe der zentralen Versionsverwaltung auf den aktuellen Stand. Im Protokoll werden die aktualisierten Dateien aufgelistet und Fred bemerkt, dass Wilma Änderungen an der Basisklasse `Order` (dt. Bestellung) vorgenommen hat. Fred ist besorgt, dass diese Änderung auch seine Arbeit beeinflussen könnte. Wilma ist heute jedoch beim Kunden und installiert das aktuelle Release, er kann sie also nicht direkt fragen. Stattdessen

[1] Der Zeitpunkt, ab dem nur noch Fehler im Quelltext behoben werden und keine neue Funktionalität hinzukommt.

lässt sich Fred den Kommentar anzeigen, der zur Änderung an `Order` gehört. Wilmas Anmerkung trägt aber nicht sehr zu seiner Beruhigung bei:

```
* deliveryPreferences zur Klasse Order hinzugefügt
```

Um mehr zu erfahren, lässt sich Fred von der Versionsverwaltung die tatsächlichen Änderungen am Quelltext anzeigen. Er findet heraus, dass Wilma einige Instanzvariablen hinzugefügt hat. Sie werden jedoch mit Standardwerten initialisiert und später nicht geändert. Das könnte durchaus in Zukunft ein Problem werden, hält Fred aber nicht vom Weiterarbeiten ab.

Beim Programmieren legt Fred eine neue Klasse und einige Testklassen im System an. Fred macht die Namen der neuen Dateien sofort der Versionsverwaltung bekannt. So kann er sie später nicht vergessen. Die Dateien selbst werden aber so lange nicht hinzugefügt, bis er seine Änderungen eincheckt.

Einige Stunden später hat Fred den ersten Teil der neuen Funktionalität fertig gestellt. Dieser Teil besteht seine Tests und beeinflusst auch den Rest des Systems nicht. Also entscheidet Fred, alles in die Versionsverwaltung einzuchecken und es so dem Team zugänglich zu machen. Über die Jahre hat Fred gelernt, dass häufiges Ein- und Auschecken bequemer ist, als tagelang ohne Abgleich weiter zu arbeiten. Es ist wesentlich leichter, gelegentlich auftretende Konflikte zu beseitigen, die nur ein paar Dateien betreffen, als alle Änderungen einer Woche des gesamten Teams zusammenzuführen.

Warum man niemals ans Telefon gehen sollte ...

Gerade als Fred wieder mit dem Programmieren beginnen will, klingelt sein Telefon. Es ist Wilma, die vom Kunden aus anruft. Es sieht so aus, als ob in dem Release, das sie gerade installiert hat, ein Fehler steckt: Ausgedruckte Rechnungen berücksichtigen keine Mehrwertsteuer bei den Versandkosten. Der Kunde regt sich sehr auf und verlangt eine sofortige Fehlerkorrektur.

... es sein denn, man benutzt eine Versionsverwaltung

Fred überprüft noch einmal den Release-Namen mit Wilma und checkt dann alle Dateien dieser Version aus der Versionsverwaltung aus. Er legt die Dateien in ein temporäres Verzeichnis auf seinem PC, da er sie nach getaner Arbeit wieder löschen möchte. In diesem Moment hat er zwei Kopien des Quelltextes des Systems auf seinem Computer, den Hauptstrang und die Release-Version des Kunden. Weil er dabei ist, einen Fehler zu beheben, markiert er in der Versionsverwaltung seinen Quelltext mit einem *Tag*[2] (dt. Etikett, Markierung). (Er wird ein weiteres Tag setzen, wenn er den Fehler beseitigt hat. Diese Markierungen sind Kennzeichen für signifikante Punkte während der Entwicklung. Durch die Benutzung konsistent benannter Tags können andere Leute in seinem Team genau sehen, was geändert wurde, sollten sie später einmal einen Blick auf den Quelltext werfen.)

Um das Problem einzugrenzen, schreibt Fred zuerst einen Test. Es sieht tatsächlich so aus, als ob bis dahin niemand die Mehrwertsteuerberechnung der Versandkosten geprüft hätte, da der Test sofort das Problem zeigt. (Fred macht eine Notiz für das Review-Meeting der aktuellen Iteration. Dieser Fehler hätte niemals das Haus verlassen dürfen.) Seufzend fügt Fred die Quelltextzeile hinzu, um die Versandkosten zur Nettosumme (dem zu versteuernden Betrag) zu addieren, kompiliert und überprüft, ob sein Test jetzt funktioniert. Anschließend lässt er die gesamte Test-Suite zur Sicherheit noch einmal laufen und checkt den fehlerbereinigten Quelltext in die zentrale Versionsverwaltung ein. Schließlich ergänzt er den Release-Zweig um ein Tag, dass dieser Fehler behoben ist. Fred sendet eine Nachricht an das QM-Team, das für die Auslieferung von Notfall-Releases an den Kunden zuständig ist. Mit Hilfe seines Tags können Sie den Build-Prozess so parametrisieren, dass eine CD mit einer fehlerbereinigten Version entsteht. Fred ruft Wilma zurück und teilt ihr mit, dass die neue Version in den Händen des QM-Teams ist und bald bei ihr ankommen sollte.

---

[2] Ein Tag ist ein selbst definierter Name für die Version einer oder mehrerer Dateien zu einem bestimmten Zeitpunkt.

Nach dieser kleinen Ablenkung löscht Fred den Quelltext des neuen Releases von seiner lokalen Maschine. Warum soll er sich auch jetzt noch damit belasten, denn die Änderungen sind schon auf dem Server gesichert. Aber er kommt ins Nachdenken, ob der Mehrwertsteuerfehler auch in der aktuellen Entwicklungsversion enthalten ist. Der schnellste Weg, das zu überprüfen, ist, den gerade geschriebenen Test aus der Release-Version auch in die Test-Suite der Entwicklungsversion einzubauen. Dazu übernimmt er die entsprechende Änderung im Release-Zweig mit Hilfe der Versionsverwaltung in die dazugehörige Datei im Entwicklungszweig. Beim Zusammenführen (engl. merge) ermittelt die Versionsverwaltung alle Veränderungen der Release-Dateien und führt diese auf der Entwicklungsversion aus. Als Fred nun die Tests startet, schlägt der neue Test fehl: der Fehler ist also tatsächlich noch vorhanden. Anschließend kopiert Fred die Korrektur aus dem Release-Zweig in die Entwicklungsversion. (Er benötigt dafür nicht den Release-Quelltext auf seinem Rechner; alle Unterschiede ermittelt die zentrale Versionsverwaltung.) Wenn alle Tests wieder fehlerfrei laufen, checkt er diese Änderung in die Versionsverwaltung ein. Ein Fehler weniger, über den das Team stolpern kann. Die Krise ist überwunden und Fred macht sich wieder an seine eigenen Aufgaben. Er verbringt einen schönen Nachmittag mit dem Schreiben von Tests und Quelltext und gegen Ende des Tages entscheidet er, dass er fertig ist. Während er gearbeitet hat, haben andere Teammitglieder auch Änderungen gemacht. Also benutzt er die Versionsverwaltung, um ihre Arbeit in seine lokale Kopie des Quelltextes zu integrieren. Er startet die Tests ein letztes Mal, checkt seine Änderungen ein und ist bereit, morgen weiterzuarbeiten.

Der nächste Tag ...

Unglücklicherweise bringt der nächste Tag eine Überraschung. Über Nacht hat Freds Zentralheizung endgültig den Geist aufgegeben. Da Fred in Minnesota lebt und es gerade Februar ist, sollte man damit nicht spaßen. Fred ruft im Büro an und sagt Bescheid, dass er den größten Teil des Tages auf den Heizungsmonteur warten wird.

Dies bedeutet jedoch nicht, dass er nicht weiterarbeiten kann. Mit Hilfe einer sicheren Verbindung über das öffentliche Internet greift Fred auf das Büronetzwerk zu und checkt den letzten Entwicklungsquelltext auf seinen Laptop aus. Da er seinen Quelltext eingecheckt hatte, bevor er gestern nach Hause ging, ist alles vorhanden und auf dem neuesten Stand. Am Kamin sitzend und in eine Decke gehüllt, setzt er seine Arbeit zu Hause fort. Bevor er die Arbeit für den Tag beendet, checkt er alle Änderungen von seinem Laptop ein, damit sie am nächsten Tag auf Arbeit verfügbar sind. Das Leben ist so schön (abgesehen von der Rechnung für die Heizungsreparatur).

Bilderbuchprojekte

Die richtige Verwendung der Versionsverwaltung im Projekt von Fred und Wilma war ziemlich unaufdringlich, aber sie gab dem Team Kontrolle und half bei der Kommunikation, auch als Wilma nicht vor Ort war. Fred konnte Änderungen am Quelltext ermitteln und eine Korrektur in verschiedenen Releases der Anwendung vornehmen. Die Versionsverwaltung unterstützt die Arbeit ohne ständige Netzwerkanbindung, wodurch Fred eine gewisse Ortsunabhängigkeit erlangte: Er konnte von zu Hause aus arbeiten, als seine Heizung Probleme bereitete. Weil Fred und Wilma von Anfang an mit Versionsverwaltung arbeiteten (und auch wussten, wie man das macht), konnten sie eine Reihe von Projektnotfällen meistern. Sie erlebten dabei keineswegs die Panik, die so oft unsere Reaktion auf das Unerwartete prägt.

Die Verwendung einer Versionsverwaltung gab Fred und Wilma die Kontrolle und Flexibilität, mit den Launen der realen Welt umgehen zu können. Darum geht es in diesem Buch.

1.2 Der Aufbau des Buches

Das nächste Kapitel, *Was ist Versionsverwaltung?*, ist eine Einführung in die Konzepte und die Begriffswelt von Versionsverwaltungen. Es gibt eine Vielzahl solcher Systeme, zwischen denen man sich entscheiden muss. In diesem Buch konzentrieren wir uns auf das frei verfügbare CVS (Concurrent Versions System). Ausgehend von unseren täglichen Erfahrungen halten wir CVS für die wohl meistgenutzte Versionsverwaltung.

Kapitel 3, *Erste Schritte*, ist eine einfache Einführung in die Benutzung von CVS. Der verbleibende Teil des Buches besteht aus Rezepten für den Einsatz von CVS. Dieser Abschnitt ist in sechs Kapitel unterteilt, von denen jedes eine Anzahl von Rezepten enthält:

- Verschiedene Zugriffsmethoden auf CVS
- Gebräuchliche CVS-Kommandos
- Verwendung von Tags und Zweigen bei Releases und experimentellem Quelltext
- Anlegen eines Projektes
- Erzeugung von untergeordneten Modulen
- Umgang mit Quelltext und Bibliotheken von Fremdanbietern

Abschließend fassen wir in Anhang A alle Rezepte zusammen und bieten in Anhang B eine kurze Liste anderer Quellen sowie eine Bibliografie.