

Software Engineering in der Praxis

Praktische Übungen

Versionskontrolle

Matthias Meitner Marc Spisländer

Lehrstuhl für Software Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

- 1 Inhalt
- 2 Versionskontrolle
 - Einführung
 - Paradigmen der Versionskontrolle
- 3 Subversion
 - Einführung
 - Architektur
 - Subversion auf der Serverseite einrichten
 - Subversion auf der Clientseite
 - Die Versionsverwaltung von Subversion
 - Konflikte
 - Projekt-Branching
- 4 Literatur

Was ist Versionskontrolle?

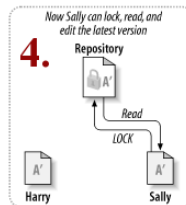
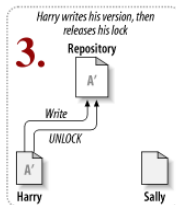
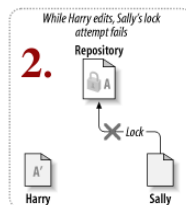
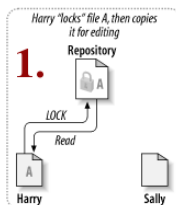
System zur:

- Kontrolle von gleichzeitigem Zugriff auf gemeinsam genutzte Dateien eines Projektes
- Verwaltung verschiedener Projekt-Versionen
- Verwaltung von Projekt-Aufspaltungen (Forks)

Zwei Paradigmen

- *Lock-Modify-Unlock*
- *Copy-Modify-Merge*

Das Lock-Modify-Unlock-Paradigma

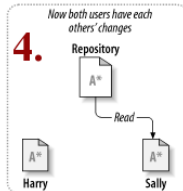
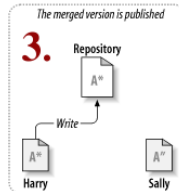
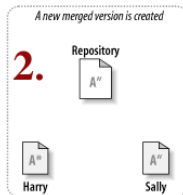
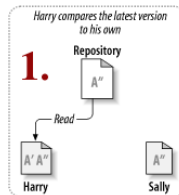
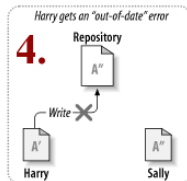
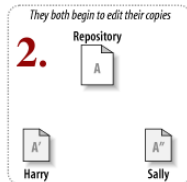
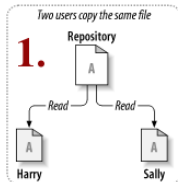


Das Lock-Modify-Unlock-Paradigma

Probleme

- Gleichzeitiges Arbeiten an der gleichen Datei ist nicht möglich.
- Ein Benutzer kann die Arbeit am ganzen Projekt blockieren, wenn er die von ihm gesperrte Datei nicht mehr frei gibt.

Das Copy-Modify-Merge-Paradigma



Das Copy-Modify-Merge-Paradigma

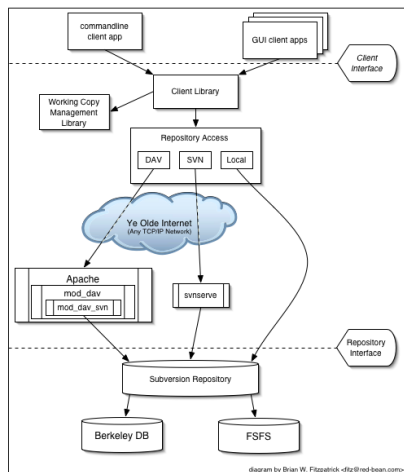
Probleme

- Es wird vorausgesetzt, dass das Zusammenführen von Änderungen an einer Datei einen »gültigen« Dateiinhalt ergibt. Das gilt meistens nur für Textdateien.
- Änderungen an der gleichen Datei können sich überlappen. ⇒ *Konfliktsituation*

Subversion

- Copy-Modify-Merge-System
- Unterstützt auch Sperren von Dateien
- Open Source
- Als vollwertiger Ersatz für CVS gedacht
- Abkürzung: SVN
- `http://subversion.tigris.org/`

Subversion-Architektur



Subversion auf der Serverseite einrichten

1 Repository anlegen:

```
svnadmin create Repository-Pfad
```

2 Server starten. Zwei Möglichkeiten:

- Apache mit entsprechenden Modulen starten
- `svnserve` starten

Im Praktikum verwenden wir `svnserve`.

Svnserve

- Starten mit `svnserve -d -r Repo-Pfad`
- Konfigurierbar über `Repo-Pfad/conf/svnserve.conf`:

```
[general]
password-db = userfile
realm = repository name

# Specific access rules
authz-db = authzfile
```

Svnserve

Userfile

Definiert Benutzernamen und Passwörter für den Zugriff auf das Repository:

```
[users]
harry = foopassword
sally = barpassword
```

Svnserve

Authzfile

Definiert Benutzerrechte für den Zugriff auf das Repository:

```
[groups]
calc-developers = harry, sally, joe
paint-developers = frank, sally, jane
allusers = @calc-developers, @paint-developers

[/projects/calc]
@calc-developers = rw
jane = r

[/projects/paint]
@paint-developers = rw
jane = r
```

Subversion auf der Clientseite

Die wichtigsten Operationen

- Neues Projekt ins Repository aufnehmen:

```
svn import PfadZumProjekt svn://Servername/Pfad
```

- Projekt aus dem Repository kopieren:

```
svn checkout svn://Servername/Pfad LokalerPfad
```

Der Inhalt von `svn://Servername/Pfad` wird nach `LokalerPfad` kopiert.

- Lokale Änderungen ins Repository übertragen:

```
svn commit -m "Beschreibung der Änderungen"
```

- Lokale Kopie (*Working Copy*) auf Repository-Version (*HEAD-Version*) aktualisieren:

```
svn update
```

- Alte Änderungen rückgängig machen:

```
svn merge -r 303:302
```

```
svn://svn.example.com/repos/calc/trunk
```


Änderungen an der Working Copy vornehmen

- Vorhandene Dateien oder Verzeichnisse ins Versionierungssystem aufnehmen:
`svn add Pfad`
- Dateien oder Verzeichnisse löschen:
`svn delete Pfad`
- Dateien oder Verzeichnisse kopieren:
`svn copy Quelle Ziel`
- Dateien oder Verzeichnisse verschieben:
`svn move Quelle Ziel`
- Änderungen widerrufen:
`svn revert Pfad`

Die Änderungen werden erst nach `svn commit` im Repository widergespiegelt.

Status der Working-Copy abfragen

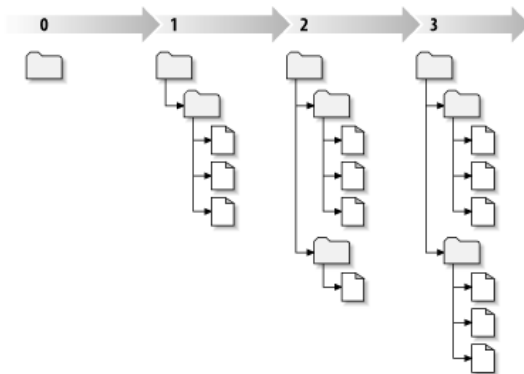
Den Status der Working-Copy im Vgl. zum letzten Update liefert `svn status`. Diese Operation listet Dateien/Verzeichnisse aus der Working-Copy auf, die wie folgt markiert sein können:

- A Zum Versionierungssystem hinzugefügt
- D Aus dem Versionierungssystem entfernt
- C Inhalt konnte nicht konfliktfrei zusammengeführt werden
- ? Nicht zum Versionierungssystem hinzugefügt

Die Versionenverwaltung von Subversion

- Der Status bzw. die Version (Revision) einer Repository ist definiert durch die Verzeichnisstruktur und den Inhalt der Dateien in der Repository.
- Subversion verwaltet pro Repository eine ganze Zahl, die den aktuellen Status der Repository repräsentiert.
- Diese Zahl wird bei jedem erfolgreichen Commit um eins erhöht.
- Zugriff auf ältere Versionen möglich

Die Versionsverwaltung von Subversion



Die Versionsverwaltung von Subversion

Abstrakte Versionsnamen

- HEAD** Die aktuelle Versionsnummer im Repository
- BASE** Die Versionsnummer nach der letzten Update- oder Checkout-Operation

Konfliktsituationen

Nach einem Update können sich in einer Datei Änderungen aus der Repository mit Änderungen in der Working Copy überschneiden. Subversion behandelt solche Konflikte wie folgt:

- Die Datei wird als *Conflict* markiert.
- Der Inhalt der Datei wird so strukturiert, dass die sich überlappenden Änderungen sichtbar werden.
- Es werden lokal folgende Dateien erzeugt:
 - `.mine` Die lokale Version der Datei
 - `.rOLDREV` Die BASE-Version der Datei
 - `.rNEWREV` Die HEAD-Version der Datei

Konflikte lösen

- Kommunikation zwischen Autoren notwendig
- Nach der Einigung:

```
svn resolve Datei
```

Die *Conflict*-Markierung wird entfernt und die drei erzeugten Dateien werden gelöscht.

Projekt-Branching

Man betrachte folgendes Szenario der Softwareentwicklung:

- **Projekt A wird gestartet**
- Ist die Funktionalität für Release 1 erreicht, wird eine Kopie von A zu Testzwecken erzeugt.
- Während der Testphase von Release 1, wird Projekt A weiterentwickelt.
- Beim Testen von Release 1 beseitigte Fehler sollen auch bei der Weiterentwicklung berücksichtigt werden, und umgekehrt.
- Erreicht Release 1 in der Testphase einen ausreichenden Reifegrad, wird eine Kopie davon erzeugt und diese an den Kunden ausgeliefert.

Projekt-Branching

Man betrachte folgendes Szenario der Softwareentwicklung:

- Projekt A wird gestartet
- Ist die Funktionalität für Release 1 erreicht, wird eine Kopie von A zu Testzwecken erzeugt.
- Während der Testphase von Release 1, wird Projekt A weiterentwickelt.
- Beim Testen von Release 1 beseitigte Fehler sollen auch bei der Weiterentwicklung berücksichtigt werden, und umgekehrt.
- Erreicht Release 1 in der Testphase einen ausreichenden Reifegrad, wird eine Kopie davon erzeugt und diese an den Kunden ausgeliefert.

Projekt-Branching

Man betrachte folgendes Szenario der Softwareentwicklung:

- Projekt A wird gestartet
- Ist die Funktionalität für Release 1 erreicht, wird eine Kopie von A zu Testzwecken erzeugt.
- Während der Testphase von Release 1, wird Projekt A weiterentwickelt.
- Beim Testen von Release 1 beseitigte Fehler sollen auch bei der Weiterentwicklung berücksichtigt werden, und umgekehrt.
- Erreicht Release 1 in der Testphase einen ausreichenden Reifegrad, wird eine Kopie davon erzeugt und diese an den Kunden ausgeliefert.

Projekt-Branching

Man betrachte folgendes Szenario der Softwareentwicklung:

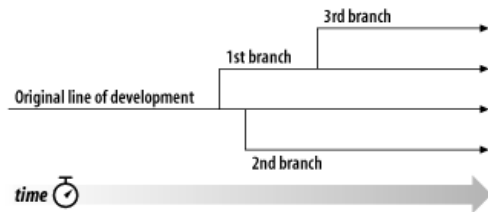
- Projekt A wird gestartet
- Ist die Funktionalität für Release 1 erreicht, wird eine Kopie von A zu Testzwecken erzeugt.
- Während der Testphase von Release 1, wird Projekt A weiterentwickelt.
- Beim Testen von Release 1 beseitigte Fehler sollen auch bei der Weiterentwicklung berücksichtigt werden, und umgekehrt.
- Erreicht Release 1 in der Testphase einen ausreichenden Reifegrad, wird eine Kopie davon erzeugt und diese an den Kunden ausgeliefert.

Projekt-Branching

Man betrachte folgendes Szenario der Softwareentwicklung:

- Projekt A wird gestartet
- Ist die Funktionalität für Release 1 erreicht, wird eine Kopie von A zu Testzwecken erzeugt.
- Während der Testphase von Release 1, wird Projekt A weiterentwickelt.
- Beim Testen von Release 1 beseitigte Fehler sollen auch bei der Weiterentwicklung berücksichtigt werden, und umgekehrt.
- Erreicht Release 1 in der Testphase einen ausreichenden Reifegrad, wird eine Kopie davon erzeugt und diese an den Kunden ausgeliefert.

Projekt-Branching



Projekt-Branching mit Subversion

- Erzeuge im Projektverzeichnis Unterverzeichnisse:
 - trunk** Für die Hauptlebenslinie des Projektes
 - branch** Enthält z. B. Unterverzeichnisse »Release 1« , »Release 2« , »Release 3« , ... , die Nebenzweige der Hauptlinie sind.
 - tag** Enthält z. B. Unterverzeichnisse »Auslieferung 1.0« , »Auslieferung 1.0.1« , ... , die Nebenzweige der »branch« -Unterverzeichnisse sind

- Erzeugung von Nebenzweigen mit

```
svn copy
```

- Übernehmen von Änderungen:

```
svn merge
```

Projekt-Branching mit Subversion

- Erzeuge im Projektverzeichnis Unterverzeichnisse:
 - trunk** Für die Hauptlebenslinie des Projektes
 - branch** Enthält z. B. Unterverzeichnisse »Release 1« , »Release 2« , »Release 3« , ... , die Nebenzweige der Hauptlinie sind.
 - tag** Enthält z. B. Unterverzeichnisse »Auslieferung 1.0« , »Auslieferung 1.0.1« , ... , die Nebenzweige der »branch« -Unterverzeichnisse sind

- Erzeugung von Nebenzweigen mit

```
svn copy
```

- Übernehmen von Änderungen:

```
svn merge
```

Projekt-Branching mit Subversion

- Erzeuge im Projektverzeichnis Unterverzeichnisse:
 - trunk** Für die Hauptlebenslinie des Projektes
 - branch** Enthält z. B. Unterverzeichnisse »Release 1« , »Release 2« , »Release 3« , ... , die Nebenzweige der Hauptlinie sind.
 - tag** Enthält z. B. Unterverzeichnisse »Auslieferung 1.0« , »Auslieferung 1.0.1« , ... , die Nebenzweige der »branch« -Unterverzeichnisse sind

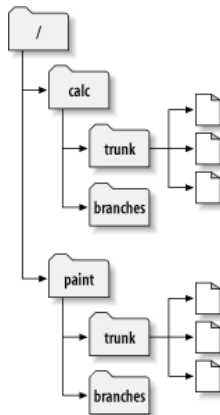
- Erzeugung von Nebenzweigen mit

```
svn copy
```

- Übernehmen von Änderungen:

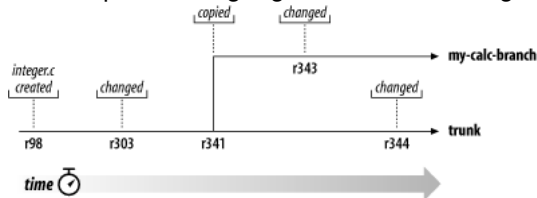
```
svn merge
```


Projekt-Branching mit Subversion



Nebenzweige erzeugen

`svn copy Quelle Ziel` Kopiert *Quelle* nach *Ziel*, wobei sich die Kopie die Vergangenheit mit dem Original teilt:



Änderungen übernehmen

```
svn merge Quelle1[@N] Quelle2[@M] Ziel Vergleicht  
Quelle1 in der Revision N mit Quelle2 in der Revision M  
und wendet die Unterschiede auf Verzeichnis Ziel der  
Working Copy an.
```

Die Unterschiede sind die Liste der Operationen, die durchgeführt werden müssen, um die Verzeichnisstruktur *Quelle1* in die Struktur *Quelle2* zu überführen.

Literatur

Subversion-Handbuch: <http://svnbook.red-bean.com/>