NewElements

IT-Schulungen.com

# Subverison - Seminar

Dr.sc.nat. Michael J.M. Wagner*

Revision 247

*michael@wagnertech.de

# Inhaltsverzeichnis

## IT-Schulungen.com
## Portfolio

IT-Schulungen.com ist eines der führenden, herstellerunabhängigen Seminarportale von Schulungen rund um die Informationstechnologie (IT) und das IT-Management. Seit über 15 Jahren ist IT-Schulungen.com eine anerkannte Anlaufstelle für viele Unternehmen und Behörden, wenn es um die Durchführung von DACH-weiten Schulungen geht.

- Applikationsserver / Middleware
- Business Intelligence
- Business-Skills und Führung
- Cloud
- CRM
- Datenbanken
- eBusiness
- ERP-Systeme
- IT Management
- IT-Recht / Lizenzierung
- ITIL
- Mobile
- Multimedia
- Office
- Open Source
- Portale
- SAP®
- Security
- Serversysteme
- Softwareentwicklung
- Systemmanagement



## Zertifizierungen & Partnerschaften

# 1 Grundlagen

### Konfigurationsmanagement

- Ein zentrales Repository für gemeinsamen Zugriff
- Koordination der gemeinsamen Arbeit in diesem Repository („einchecken", „auschecken")
- Dokumente werden versioniert
- Die Möglichkeit der Attributierung einzelner Dokumentversionen (Kommentar, Datum der Änderung, Autor, etc.)
- Die Rückverfolgbarkeit des Arbeitsverlaufs (Die Versionskette beantwortet die Frage: Wer hat was, wann verändert?).
- Bildung von Konfigurationen („Markieren" zusammengehöriger Dokumentversionen)

### Marktübersicht

- OpenSource
  - RCS (Revision Control System)
    - 1981 an der Perdue University
    - Arbeitet auf einzelnen Dateien (pessimistisches Sperren)
  - cvs (concurrent versioning system)
    - 1989 aus RCS weiterentwickelt
    - Erweiterung auf Verzeichnisbäume (optimistisches Sperren)
    - Netzwerkfähig
  - svn (Subversion)
    - Entwicklung seit 2000
    - Atomarer Checkin von Verzeichnisbäume -> Revisionierung von Projekten statt Dateien
    - Webfähig
  - git[1]
    - 2005 von Linus Torwalds für die Linux-Kernelentwicklung entwickelt
    - Unterstützung verteilter Arbeitsabläufe

[1] https://de.wikipedia.org/wiki/Git (19.6.2018)

        * Sehr hohe Sicherheit gegen sowohl unbeabsichtigte als auch böswillige Verfälschung
        * Hohe Effizienz

- Proprietäre Software
  - Microsoft
    Visual Souce Safe (VSS) / Team Foundation Server
  - IBM
    Rational ClearCase
    * Entwicklung seit 1985
    * Multi Site: Verteilte Repositories
    * Build Tools integrierbar

### Grundlagen der Versionskontrolle

| | |
|---|---|
| Hauptpfad - Trunc | Hauptpfad der Entwicklung |
| Zweig - Branch | Seitenpfad: Nachträgliche Änderungen an älteren Softwareständen |
| Top | Jüngste Version in einem Pfad (Hauptpfad oder Zweig) |
| Head | Top-Version im Hauptpfad |
| Marke - Tag | Ausgezeichnete Version in einem Pfad |

- Projektarchiv und Arbeitskopie → CFP1.7[2]
- Versionierungsmodelle → CFP1.7[3]
- Subversion verwendet Globale Revisionsnummern → CFP1.7[4]

### Lokale Installation[5]

Um sich mit Subversion vertraut zu machen, kann eine lokale Installation verwendet werden.

- Anlegen eines Projektarchivs (*repository*):
  `$ svnadmin create /path/to/repos`
- Erstellen einer Arbeitskopie:
  `$ svn checkout file:///path/to/repos local_name`
- Neue Dateien/Verzeichnisse unter Versionskontrolle nehmen:
  `$ svn add file_or_dir`

  Anmerkung: Verzeichnisse werden so samt ihrem Inhalt dem Projektarchiv hinzugefügt. Soll ein Verzeichnis ohne Inhalt hinzugefügt werden, ist die Option `-N` zu verwenden.

[2] http://svnbook.red-bean.com/de/1.7/svn.basic.version-control-basics.html (19.6.2018)
[3] http://svnbook.red-bean.com/de/1.7/svn.basic.version-control-basics.html#svn.basic.vsn-models (19.6.2018)
[4] http://svnbook.red-bean.com/de/1.7/svn.basic.in-action.html#svn.basic.in-action.revs.dia-1 (19.6.2018)
[5] CFP2004: S. 7f.

- Änderungen einchecken:
  `$ svn commit [-m "Kommentar"]`

Aufgabe:

- Legen Sie ein lokales Projektarchiv an.
- Erstellen Sie eine Arbeitskopie und fügen Sie Dateien hinzu.
- Legen Sie eine zweite Arbeitskopie an und erzeugen Sie durch entsprechendes Editieren und Einchecken Konflikte.

# 2 Installation von Subversion

## 2.1 Architektur

Subversion-Architektur → CFP1.7[6]

## 2.2 Subversion über Apache[7]

Üblicherweise wird Subversion über das Netzwerk betrieben. Dazu wird der Zugriff auf das Projektarchiv über einen Server, meist den Apache-Webserver hergestellt. Dafür sind folgende Elemente nötig.

- `mod_dav`: Apache-Erweiterung *Distributed Authoring and Versioning*
- `mod_dav_svn`-Erweiterung

Auf einem Debian/Ububtu-System ist dafür folgendes zu tun:

- Installation des Pakets `libapache2-svn`
- Aktivierung der Module mit `sudo a2enmod MODUL`
- Kontrolle: Das Modul muss im Verzeichnis `/etc/apache2/mods-enabled` erscheinen.

Als nächstes muss der Ort des Projektarchivs dem Apache mitgeteilt werden. Dazu benötigt die Konfiguration einen Eintrag der Form:

```
<Location /repos>
  DAV svn
  SVNPath REPOSIRORY_PATH
</Location>
```

[6] http://svnbook.red-bean.com/de/1.7/svn.intro.whatis.html#svn.intro.architecture.dia-1 (29.6.2018)
[7] http://svnbook.red-bean.com/de/1.7/svn.serverconfig.httpd.html (29.6.2018)

Bei einem Debian/Ubuntu-Server erfolgt die Konfiguration üblicherweise in einer eigenen Datei `SITE.conf` im Verzeichnis `/etc/apache2/sites-available` und werden über den Befehl `sudo a2ensite SITE` aktiviert.

Aufgabe:

- Richten Sie den Apache so ein, dass er auf Ihr bestehendes Projektarchiv zeigt.

  Legen Sie Ihre Konfiguration in eine Datei `svn.conf` und aktieren Sie die *site*.
- Übergeben Sie die Rechte in Ihrem Projektarchiv an den Benutzer `www-data`:
  `chown -R www-data:www-data BASE_DIR`
- Legen Sie eine Arbeitskopie an, dem Sie über den Apache auschecken:
  `svn co http://localhost/repos`
- Machen Sie Änderungen und checken Sie diese ein.

Sie stellen fest, dass mit dieser Installation keinerlei Zugriffskontrolle angelegt wurde. Für diese muss die Konfiguration wie folgt erweiter werden:

```
<Location /repos>
  DAV svn
  SVNPath REPOSIRORY_PATH

  # Authentifizierung: Digest
  AuthName "Subversion repository"
  AuthType Digest
  AuthUserFile DIGEST_FILE

  # Autorisierung: Nur f"ur authentifizierte Anwender
  Require valid-user
</Location>
```

Die Datei `DIGEST_FILE` enthält die Passwörter und wird folgendermaßen angelegt:

```
$ ### Beim ersten Mal: -c zum Erzeugen der Datei verwenden
$ htdigest -c DIGEST_FILE "Subversion repository" harry
Adding password for harry in realm Subversion repository.
New password: *****
Re-type new password: *****
$ htdigest DIGEST_FILE "Subversion repository" sally
Adding user sally in realm Subversion repository
New password: *******
Re-type new password: *******
```

Hierfür ist zusätzlich das Apache-Modul `auth_digest` nötig.

Soll nur der Schreibvorgang autorisiert erfolgen, der Lesevorgang hingegen frei möglich sein, kann der Konfigurationseintrag wie folgt abgeändert werden:

```
<Location /repos>
  ...
  # Autorisierung: Nur authentifizierte Anwender f"ur nicht Nur-Lese
  #                (Schreib-) Operationen; anonymes Lesen zulassen
```

```
  <LimitExcept GET PROPFIND OPTIONS REPORT>
    Require valid-user
  </LimitExcept>
</Location>
```

Aufgabe:

- Ergänzen Sie die Apache-Konfiguration um eine Authentifizierung.

Wird über eine Benutzerüberprüfung (Authentifizierung) hinaus eine Kontrolle der Zugriffsrechte auf Verzeichnisbasis gewünscht (Autorisierung), kann dies über ein `AuthzSVNAcessFile` erfolgen. Auf die Datei wird im `Location`-Block der Konfiguration verwiesen.

Format des `AuthzSVNAcessFile` → CFP1.7[8]

Üblicherweise wird heute eine verschlüsselte Übertragung über `https` erwartet. In diesem Fall erfolgt die Konfiguration in der entsprechenden Datei des Apachen.

## 2.3 Hooks

Während einen Checkin-Vorgangs lassen sich an verschiebenen Stellen Skripte einhängen, um (zuvor) Prüfungen durchzuführen oder (danach) Stakeholders zu informieren. Die Hook-Skripte befinden sich im Projektarchiv im Unterverzeichnis `hooks`. Hook-Skripte werden nur dann ausgeführt, wenn sie nur den Namen des Hooks (ohne `.tmpl`) haben und ausführbar sind. In den Templates sind die Auswirkungen der Rückgabewerte der Hook-Skripte dokumentiert.

Um auf die Daten eines Commits zugreifen zu können steht das Kommando `svnlook` zur Verfügung. Mit diesem lässt sich u.a. abfragen:

- der Commit-Kommentar:
  `svnlook log -t "$TXN" "$REPOS"`
- eine Liste der übergebenen Dateien:
  `svnlook changed -t "$TXN" "$REPOS"`
  Format dieser Liste → CFP1.7[9]
- Inhalt einer übergebenen Datei:
  `svnlook cat -t "$TXN" "$REPOS" "$FILEPATH"`

`$TXN` und `$REPOS` sind dabei die beiden Parameter, mit denen das Skript aufgerufen wird.

Zum Testen lässt sich `svnlook` für bereits erfolgte Transaktionen mit dem Parameter `-r $REVISION` aufrufen.

[8] http://svnbook.red-bean.com/de/1.7/svn.serverconfig.pathbasedauthz.html (5.9.2019)
[9] http://svnbook.red-bean.com/de/1.7/svn.ref.svnlook.c.changed.html (13.9.2019)

Aufgabe:

- Legen Sie ein `post-commit` Hook-Skript an, das für jeden Commit den Commit-Kommentar in eine Logdatei protokolliert.
- Legen Sie ein `pre-commit` Hook-Skript an, das einen Commit zurückweist, wenn sich die Buchstabenfolge `NOCOMMIT` in der Datei befindet.

# 3 Grundlegende Benutzung[10]

### Empfohlene Aufteilung des Projektarchivs

Im Gegensatz zu anderen KM-Werkzeugen legt Subversion seine Daten „zweidimensional" ab. Alle Zweige und Marken werden in einem virtuellen Dateisystem abgebildet. Der Wechsel in einen anderen Entwicklungszweig bedeutet dann den Wechsel in einen anderen Verzeichniszweig. Daher ist es notwendig gleich zu Beginn Verzweigungsmöglichkeiten zu schaffen. Ein typisches SVN-Archiv hat daher folgende Grundstruktur:

```
/path/to/repos
  /branches/
  /tags/
  /trunk/
```

### Diverse Befehle auf der Arbeitskopie

- Datei/Verzeichnis zum Löschen vormerken: `$ svn delete FILE`
- Datei/Verzeichnis kopieren: `$ svn copy FOO BAR`
- Datei/Verzeichnis verschieben: `$ svn move FOO BAR`
- Verzeichnis anlegen und im Projektarchiv hinzufügen: `$ svn mkdir DIR`
- Überblick über Änderungen: `$ svn status` → CFP1.7[11]
- Änderungsstatus unter Berücksichtigung des Projektarchivs: `svn status -u`
- Aufzeigen der Änderungen im Einzelnen: `$ svn diff [FILE]`

  Anmerkung: Das Ergebnis dieses Kommandos hat das Format, mit dem der `patch`-Befehl versorgt werden kann. Es kann also zur Erstellung eines Hotfixes verwendet werden.
- Rückgängigmachen lokaler Änderungen: `$ svn revert FILE`

[10] CFP1.7: Kap. 2 (21.6.2018)
[11] http://svnbook.red-bean.com/de/1.7/svn.tour.cycle.html#svn.tour.cycle.examine.status (21.6.2018)

- Arbeitskopie mit dem Projektarchiv synchronisieren: `$ svn update`
  Dabei können Konflikte zu Tage treten. Zum Auflösen von Konflikten → CFP1.7[12]

  Tipp: Wenn Konflikte auftreten, wenden diese erst mal zurückgestellt (*postpone*. Dann werden die Datei untersucht und die Konflikte beseitigt. Wenn man damit fertig ist, muss dies dem System mit dem Befehl `syn resolved DATEI` mitgeteilt werden.
- Soll ein Befehl nur testweise ausgeführt werden, also ohne, dass Archiv oder Arbeitskopie geändert werden, so kann die Option `--dry-run` verwendet werden.

### Diverse Befehle im Projektarchiv

- Verzeichnisstruktur eines Projektarchivs anzuschaun: `$ svn list -R URL`
- Inhalt einer Datei ausgeben: `svn cat URL@REV`
- Datei/Verzeichnis löschen: `$ svn delete URL`
- Datei/Verzeichnis kopieren: `$ svn copy URL1 URL2`
- Datei/Verzeichnis verschieben: `$ svn move URL1 URL2`
- Verzeichnis direkt im Projektarchiv anlegen: `$ svn mkdir URL/DIR`

Aufgabe:

- Fügen Sie dem Projektarchiv die Standardverzeichnisse hinzu.
- Verschieben Sie die bereits existierenden Elemente unter `trunk`.
- Überprüfen Sie das Ergebnis mit `$ svn list -R`
- Erstellen Sie eine Arbeitskopie für den `trunk`.
- Fügen Sie Dateien hinzu und wenden Sie alle obigen Befehle an.

### Umsehen im Projektarchiv

- `svn diff` mit Versionsangabe: CFP1.7[13]
- Anzeigen der Änderungsgeschichte: `$ svn log [FILE]` → dto.
- `svn log -v .` zeigt die Änderungshistorie im aktuellen Verzeichnis.
- Anzeigen von Dateiinhalten: `$ svn cat -r 34 FILE`
- Bereitstellung älterer Projektarchiv-Schnappschüsse: `$ svn update -r 22`

[12] http://svnbook.red-bean.com/de/1.7/svn.tour.cycle.html#svn.tour.cycle.resolve (21.6.2018)
[13] http://svnbook.red-bean.com/de/1.7/svn.tour.history.html (21.6.2018)

Aufgabe:

Wenden Sie auch diese Kommandos auf ihr Projektarchiv an.

### Wiederherstellung gelöschter Dateien

Für die Wiederherstellung gelöschter Dateien gibt es verschiedene Vorgehensweisen:

- Arbeitskopie auf alten Stand zurücksetzen
- Gelöschte Datei „irgendwohin" kopieren
- Arbeitskopie aktualisieren
- Datei wieder hinzufügen

Dieses Vorgehen ist naheliegend, hat aber einen Nachteil: Die Historie bis zum Löschen der Datei geht verloren. Es wird mit dem neuen Hinzufügen eine neue Historie angelegt. Soll die alte Historie mit wiederhergestellt werden, empfiehlt sich folgendes Vorgehen:

- Stand ermitteln, bei dem die Datei noch vorhanden war: N
- `svn copy <svn-url der Datei>@N .`
- Datei einchecken

Aufgabe:

- Löschen Sie eine Datei in einer Arbeitskopie und checken Sie das Löschen ein.
- Machen Sie eine weitere Änderung in der Arbeitskopie und checken Sie diese ein.
- Ermitteln Sie mit `svn log -v` bei welcher Revision die Datei noch vorhanden war. Ggf. muss der Baum zuvor aktualisiert werden.
- Holen Sie die Datei zurück.
- Betrachten Sie sich die Historie der Datei.

# 4 Fortgeschrittene Themen

### Eigenschaften

Dateien und Verzeichnisse können in Subversion mit Eigenschaften belegt werden. Dafür stehen folgende Befehle zur Verfügung:

- `$ svn propset NAME WERT DATEI_ODER_VERZEINCHNIS`: Setzen von Eigenschaften
- `$ svn propget NAME DATEI_ODER_VERZEINCHNIS`: Anzeigen einer Eigenschaft

- `$ svn proplist DATEI_ODER_VERZEINCHNIS`: Anzeigen aller Eigenschaften
- `$ svn propdel NAME DATEI_ODER_VERZEINCHNIS`: Löschen einer Eigenschaft
- Mit `svn propset NAME WERT -R ZIEL` wird die Eigenschaft für `ZIEL` und rekursiv für alle darunter liegenden Elemente gesetzt.

Nützliche Eigenschaften sind:

- `svn:executable 1`: Datei wird als audführbar markiert.
- `svn:ignore PATTERN`: Wird diese Eigenschaft für ein Verzeichnis gesetzt, werden alle Dateien, die dem Suchmuster gehorchen, von Subversion ignoriert.

### Ersetzung von Schlüsselworten

Subversion kann beim Einchecken Schlüsselwörter ersetzen → CFP1.7.[14]

Um den Mechanismus zu aktivieren muss die Eigenschaft `svn:keywords` mit den zu ersetzenden Schlüsselwörtern belegt werden.

Aufgabe:

- Führen Sie das „Wetterbericht"-Beispiel aus *CFP1.7: Kap 3* durch.
- Legen Sie eine neue Datei in Ihrer Arbeitskopie an.
  - Prüfen Sie mit `$ svn status`, dass die Datei noch nicht im Projektarchiv ist.
  - Setzen Sie im Verzeichnis ein `svn:ignore`-Muster.
  - Überprüfen Sie den Erfolg mit `$ svn status`

### Sperren

Subversion geht davon aus, dass es sich im Normalfall um Textdateien handelt, deren Änderungen nur einzelne Zeilen betreffen. Im Falle eines Konflikts versucht Subversion die unabhängig geänderten Dateien zu harmonisieren (optimistisches Sperren). Bei Binärdateien macht dieses Vorgehen keinen Sinn. Hier bietet sich ein pessimistisches Sperren an. Dazu muss die `svn:needs-lock 1`-Eigenschaft gesetzt werden:

- `$ svn lock DATEI`: Holt exklusive Sperre für `DATEI`.
- Mit einem `checkin` wird die Sperre wieder zurückgegeben.
- Falls die Datei nicht verändert wurde, kann mit `$ svn unlock DATEI` die Datei wieder freigegeben werden.

[14] http://svnbook.red-bean.com/de/1.7/svn.advanced.props.special.keywords.html (21.6.2018)

Aufgabe:

- Setzen Sie für eine Datei die `svn:needs-lock 1`-Eigenschaft.
- Checken Sie diese Änderung ein.
- Synchronisieren Sie Ihre beiden Arbeitskopien (`$ svn update`).
- Überprüfen Sie in beiden Arbeitskopien die Schreibrechte (`$ ls -l`)
- Holen Sie sich für eine Arbeitskopie die Sperre. und ändern Sie die Datei.
- Versuchen Sie dasselbe in der anderen Arbeitskopie.
- Checken Sie die geänderte Datei ein.
- Versuchen Sie erneut in der anderen Arbeitskopie eine Sperre zu erhalten.

**Weitere Themen**

- Administrativer Umgang mit Sperren → CFP1.7[15]
- Einbinden externer Projektarchive → CFP1.7[16]

# 5 Verzweigen und Zusammenführen

## 5.1 Erzeugen von Zweigen

Subversion erzeugt Zweige dadurch, dass von einem Sourcestand eine Kopie angelegt wird.

- Projektarchiv vorher → CFP1.7[17]
- Kopierbefehl

```
$ svn copy http://svn.example.com/repos/calc/trunk \
           http://svn.example.com/repos/calc/branches/my-calc-branch \
      -m "Privaten Zweig von /calc/trunk angelegt."

Revision 341 "ubertragen.
```

- Projektarchiv nachher → CFP1.7[18]

[15] http://svnbook.red-bean.com/de/1.7/svn.advanced.locking.html (21.6.2018)
[16] http://svnbook.red-bean.com/de/1.7/svn.advanced.externals.html (21.6.2018)
[17] http://svnbook.red-bean.com/de/1.7/svn.branchmerge.using.html#svn.branchmerge.using.dia-1 (28.6.2018)
[18] http://svnbook.red-bean.com/de/1.7/svn.branchmerge.using.html#svn.branchmerge.using.create.dia-1 (28.6.2018)

- Anlegen einer Arbeitskopie:

```
$ svn checkout http://svn.example.com/repos/calc/branches/my-calc-branch
A  my-calc-branch/Makefile
A  my-calc-branch/integer.c
A  my-calc-branch/button.c
Ausgecheckt, Revision 341.
```

Anmerkung:

> Sie sollten sich zwei Lektionen aus diesem Abschnitt merken. Erstens besitzt Subversion kein internes Konzept für einen Zweig – es weiß lediglich, wie Kopien angelegt werden. Wenn Sie ein Verzeichnis kopieren, ist das entstehende Verzeichnis bloß ein „Zweig“, weil Sie ihm diese Bedeutung geben. Sie mögen über das Verzeichnis anders denken oder es anders behandeln, doch für Subversion ist es einfach ein gewöhnliches Verzeichnis, das nebenbei mit einigen zusätzlichen historischen Informationen ausgestattet ist.
>
> Zweitens bestehen die Zweige von Subversion, bedingt durch den Kopiermechanismus, als normale Dateisystemverzeichnisse im Projektarchiv. Das ist ein Unterschied zu anderen Versionskontrollsystemen, bei denen Zweige typischerweise definiert werden, indem auf einer eigenen Ebene den Dateisammlungen „Etiketten“ hinzugefügt werden. Der Ort Ihres Zweig-Verzeichnisses spielt für Subversion keine Rolle. Die meisten Teams folgen der Konvention, alle Zweige in einem Verzeichnis namens /branches abzulegen, jedoch steht es Ihnen frei, eine Vorgehensweise nach Ihren Wünschen zu erfinden.[19]

## 5.2 Zusammenführen von Zweigen

Wenn Entwicklungsstänge auseinanderlaufen, ist es wichtig, diese auch wieder zusammenzuführen. Möglicherweise in beide Richungen:

- Änderungen, die im Hauptzweig gemacht werden, werden in den Seitenzweig übernommen.
- Wenn der Seitenzweig eine „Vorausentwicklung” war, soll diese später in den Hauptzweig übernommen werden.

Das Zusammenführen geschieht mit dem Befehl `$ svn merge`:

```
$ svn merge ^/calc/trunk
-- Zusammenf"uhren von r345 bis r356 in >>.<<:
U    button.c
U    integer.c
-- Aufzeichnung der Informationen f"ur Zusammenf"uhrung von r345 in >>.<<:
 U   .
```

Aufgabe:

- Erstellen Sie einen Zweig Ihres Projekts.

[19] http://svnbook.red-bean.com/de/1.7/svn.branchmerge.using.html#svn.branchmerge.using.concepts (28.6.2018)

- Erstellen Sie eine Arbeitskopie für Ihren Zweig, der diesselbe Verzeichnisstruktur wie Ihr Projekt selbst aufweist.
- Nehmen Sie Änderungen sowohl im Hauptzweig, wie im Nebenzweig vor und checken Sie diese ein.
- Führen Sie die Zweige zusammen.

### Sondersituationen

- Es sollen nur Änderungen eines bestimmten Versionsbereichs zusammengeführt werden:
  `$ svn merge ^/trunk -r399:HEAD`
- Es soll ein bestimmter Stand eingepflegt werden:
  `$ svn merge ^/trunk@REV`
- Es soll eine bestimmte Änderung übernommen werden:
  `$ svn merge ^/trunk -cREV`
- Finale Rückintegration in die Hauptlinie:
  `$ svn merge --reintegrate ^/calc/branches/my-calc-branch`

  Dieser Befehl holt alle noch fehlenden Änderungen. Anschließend sollte der Zweig gelöscht werden.

### Baumkonflikte

Besonders unangenehm in der Handhabung sind *Baumkonflikte*. Es gibt (mindestens) zwei Varianten, wie diese entstehen:

- Eine Datei/ein Verzeichnis ist bereits vorhanden, wenn Subversion versucht, dieses aus dem Projektarchiv als neu der Arbbeitskopie hinzuzufügen:
  - In Arbeitskopie 1 werden neue Verzeichnisse/Dateien angelegt und eingecheckt.
  - In Arbeitskopie 2 werden dieselben Verzeichnisse/Dateien angelegt.
  - Der Benutzer von Arbeitskopie 2 will einchecken und stellt fest, dass seine Arbeitskopie nicht mehr aktuell ist.
  - Der Benutzer von Arbeitskopie 2 versucht einen Update und erhält Baumkonflikte.

  SVN (Kommandozeile) stellt leider nicht fest, dass es dieselben Verzeichnisse/Dateien sind.
- Eine Datei/ein Verzeichnis wird sowohl im Hauptzweig, als auch in einem Seitenzweig angelegt und eingecheckt. Anschließend wird versucht, die Zweige zu mergen.

Um einen Baumkonflikt aufzulösen gibt es folgende Möglichkeiten:

- Der Benutzer von Arbeitskopie 2 stellt die Konflikte zurück und macht einen Revert (`svn revert -R`), danach erneut einen Update.

- Der Benutzer entfernt die Dateien/Verzeichnisse in einem der Zweige und versucht erneut einen Merge.

Aufgabe:

Erzeugen Sie Baumkonflikte und lösen Sie diese auf.

# 6 Weitere Clients

Neben dem Kommandozeilenprogramm `svn` gibt auch graphische Clients:

- TortoiseSVN[20]
- Eclipse-Plugin

Aufgabe:

- Starten Sie Eclipse.
- Checken Sie Ihr Projekt aus.
- Nehmen Sie Änderungen von.
- Checken Sie die Änderungen wieder ein.

# 7 Quellen


CFP2004 Ben Collin-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, Version Control with Subversion, O'Reilly 2004

CFP1.7 Ben Collin-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, Versionskontrolle mit Subversion, Online-Version unter `http://svnbook.red-bean.com/de/1.7/index.html`


[20] https://tortoisesvn.net (30.6.2018)

## Vielen Dank für Ihre Aufmerksamkeit

**Ihr Referent und das Team von**
**IT-Schulungen.com**

Fon +49 (0) 911 650 08 - 30
Fax +49 (0) 911 650 08 - 399
Mail info@it-schulungen.com
Web www.it-schulungen.com

**Education Center der New Elements GmbH**
Thurn-und-Taxis-Straße 10
90411 Nürnberg

www.newelements.de