Birte Kehr
Algorithmische Bioinformatik, Fachbereich für Mathematik und Informatik

Freie Universität Berlin

Versionskontrolle mit Subversion

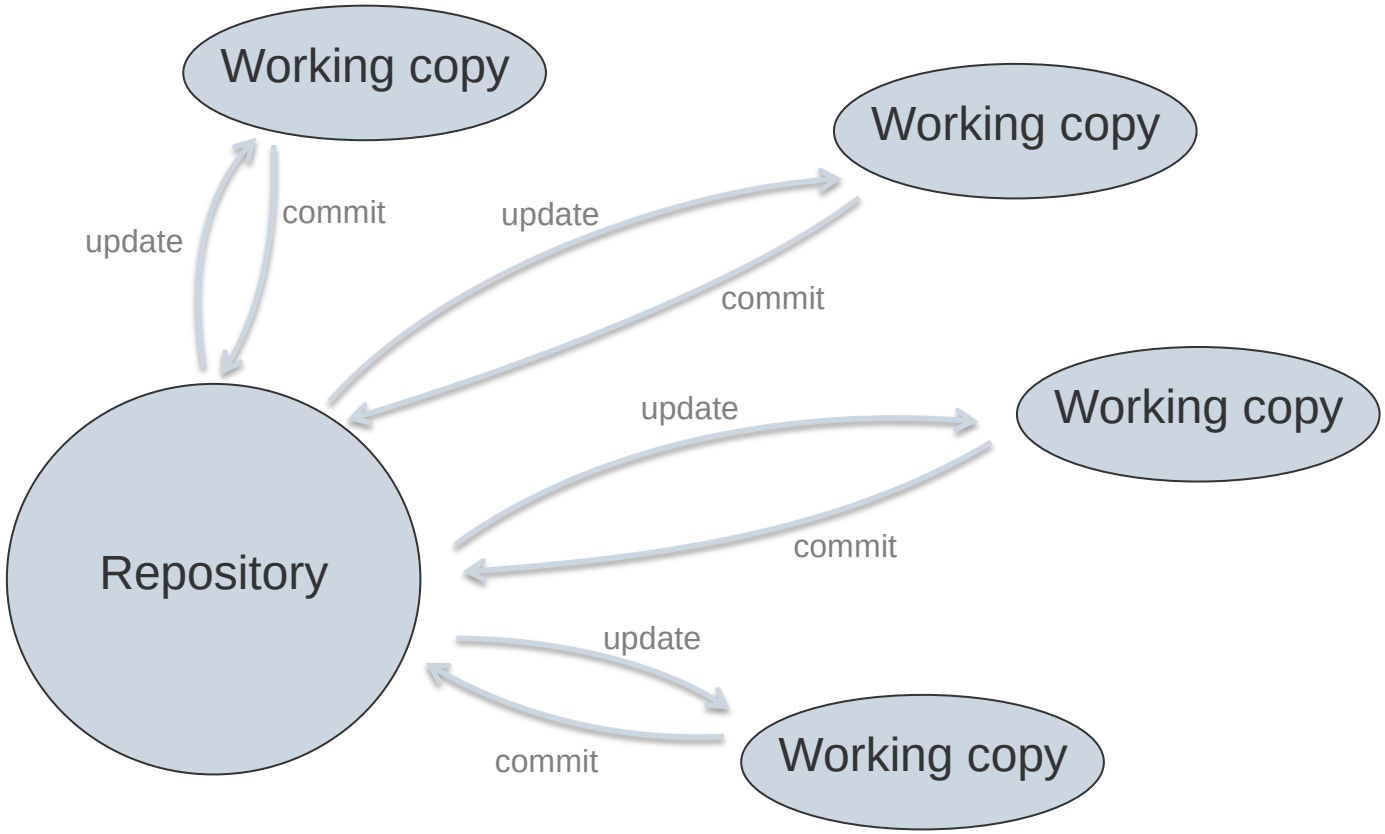
Eine kurze Einführung

Wozu Versionskontrolle?

- Zeitschiene aller Änderungen
  - Was?
  - Wann?
  - Wer?
- Möglichkeit zu einer alten Version zurück zu kehren (Backup)
- Koordinierung des Zugriffs durch mehrere Personen
  - Lesen
  - Schreiben

Was ist Subversion?

Subversion ist ein Versionskontrollsystem.



Erstellen einer Working Copy

1. Installation einer Client-Software:

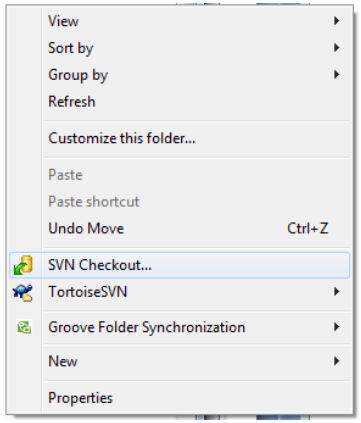
   Linux: `$ apt-get install subversion`

   Windows: z.B.  TortoiseSVN (http://tortoisesvn.tigris.org/)

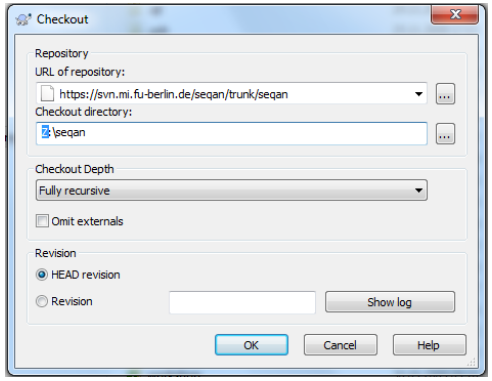
2. Repository auschecken (svn checkout / svn co)

   Linux: `$ svn checkout http://svn.mi.fu-berlin.de/seqan/.../swp11 swp11`

   Windows:
   1. Kontext-Menü im Windows-Explorer



   2. Checkout-Dialog ausfüllen

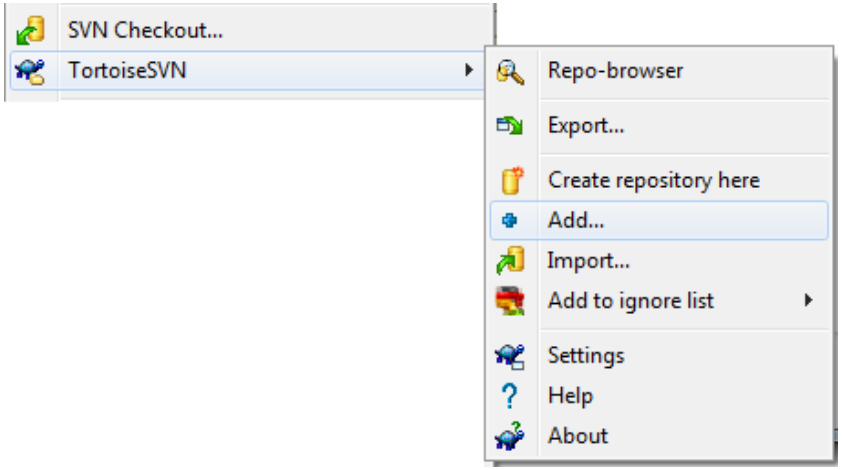


Dateien dem Repository hinzufügen

1. Hinzufügen vormerken (svn add):

   Linux: `$ svn add abschlussbericht.tex`

   Windows: Datei im Windows-Explorer mir rechter Maustaste auswählen



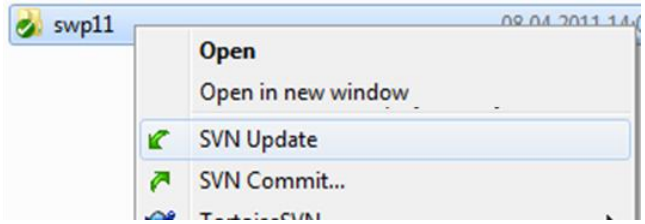
2. Änderung einchecken (svn commit / svn ci) *… siehe nächste Folie*

Aktualisieren und Änderungen einchecken

1. Die eigene Working Copy auf den aktuellen Stand bringen (svn update / svn up):

   Linux: `$ svn update`

   Windows:

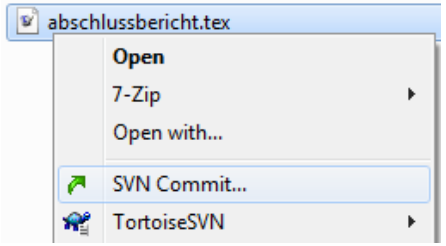


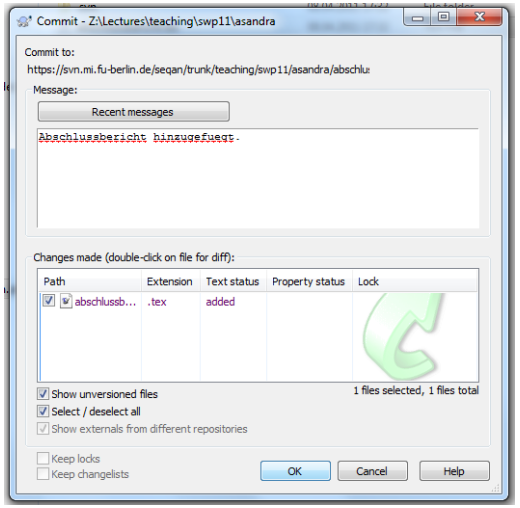
2. Änderungen einchecken (svn commit / svn ci):

   Linux:

```
$ svn commit abschlussbericht.tex \
        –m “Abschlussbericht hinzugefuegt.“
```

   Windows:



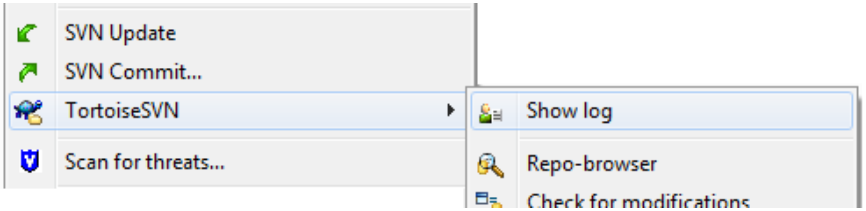


Änderungen verfolgen (1)

1. Zeitschiene anzeigen lassen (svn log):

   Linux: `$ svn log abschlussbericht.tex` `$ svn log –r 836:842`

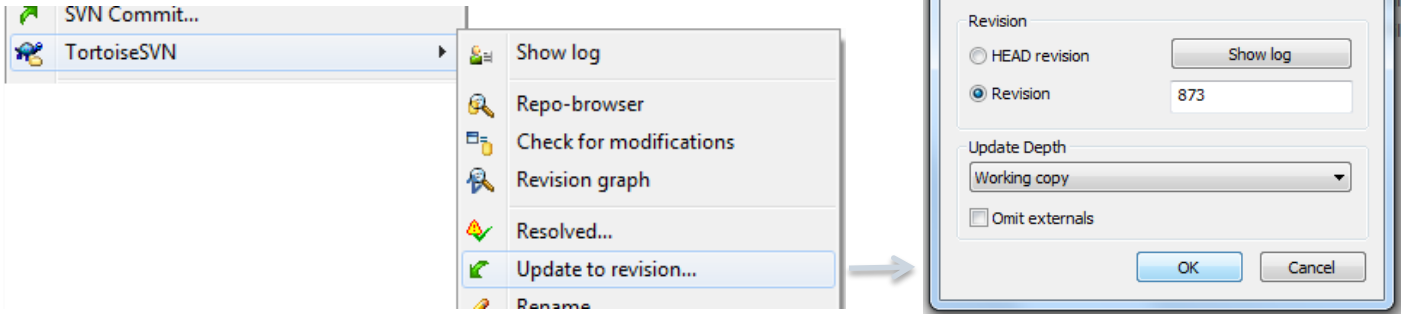
   Windows:



2. Alte Version anschauen (svn update / svn up):

   Linux: `$ svn update abschlussbericht.tex –r 873`

   Windows:

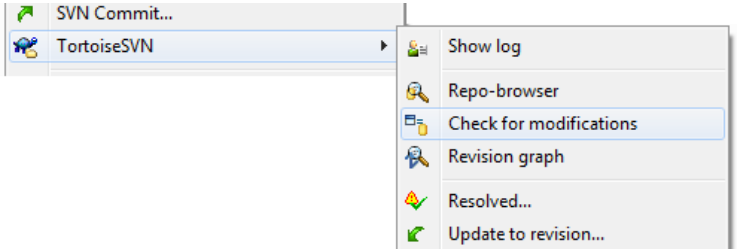


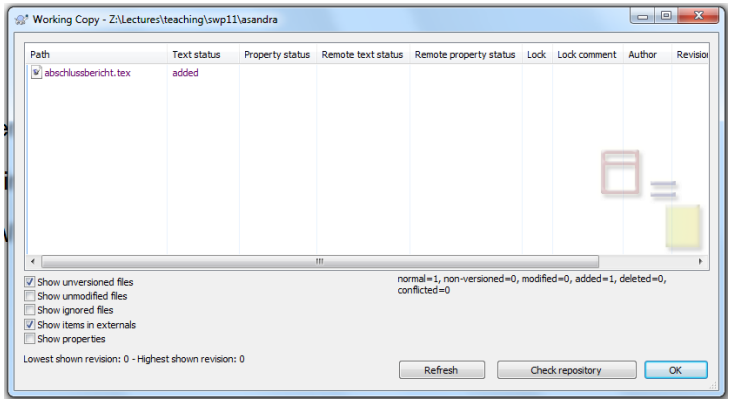
Änderungen verfolgen (2)

1. Status der eigenen Working Copy abfragen (svn stat):

   Linux: `$ svn stat` → M – Modified, A – Added, C – Conflicted, …

   Windows:

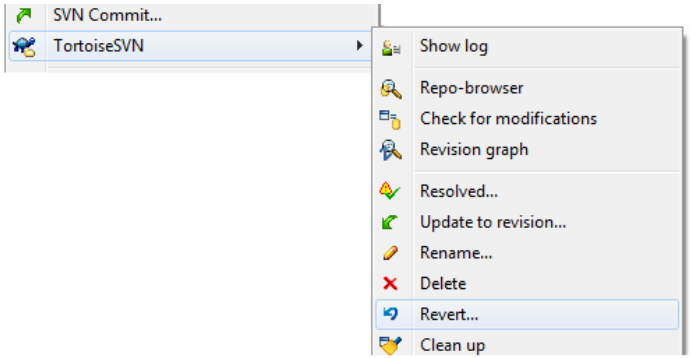




2. Lokale Änderungen verwerfen (svn revert):

   Linux: `$ svn revert abschlussbericht.tex`

   Windows:



Konflikte am Beispiel

Beispiel-Datei hello.cpp:

```
#include <iostream>

int main(int, char **) {
    std::cout << "Hello World";
    return 1;
}
```

```
$ svn add hello.cpp
$ svn commit hello.cpp –m “Adding hello.cpp.“
```

Konflikte am Beispiel

Person A:

$ svn update

```
#include <iostream>

int main(int, char **) {
    std::cout << "Hello World!";
    return 1;
}
```

$ svn commit hello.cpp \
-m „Ausrufungszeichen“

Person B:

$ svn update

```
#include <iostream>

int main(int, char **) {
    std::cout << "Hello World\n";
    return 1;
}
```

↳ Commit nicht möglich

$ svn update → Konflikt

Konflikte am Beispiel

1. Datei editieren:

```
#include <iostream>

int main(int, char **) {
<<<<<<< .mine
    std::cout << "Hello World\n";
=======
    std::cout << "Hello World!";
>>>>>>> .r9303
    return 1;
}
```

→ `std::cout << "Hello World!\n";`

2. Konflikt als gelöst markieren:

```
$ svn resolve hello.cpp
```

3. Änderungen einchecken:

```
$ svn commit hello.cpp –m „Zeilenumbruch“
```

Subversion im Softwarepraktikum

1. Softwarebibliothek auschecken
   - SeqAn: https://svn.mi.fu-berlin.de/seqan/trunk/seqan
   - OpenMS: https://open-ms.svn.sourceforge.net/svnroot/open-ms/OpenMS

2. Projektdateien (Berichte und Programme) verwalten
   - SeqAn: https://svn.mi.fu-berlin.de/seqan/trunk/teaching/swp11
     https://svn.mi.fu-berlin.de/seqan/trunk/seqan/sandbox/swp11
   - OpenMS: … Git

3. Abgabe der Programme und Abschlussberichte
   - Zugriffsrechte werden zum Abgabetermin
     von read/write auf read only gesetzt