Subversion in der Lehre

Organisation von Vorlesung und Übungen

Stefan Klinger

Datenbanken und Informationssysteme
Universität Konstanz

Winter 2015

0.1 Wer, Wann, Wo

Personal Stefan Klinger

Mail stefan.klinger@uni-konstanz.de

Büro PZ804

Zeit 26. Februar, 13:30

Ort Raum M 901

Online https://svn.uni-konstanz.de/dbis/svn_15w/pub/

1
Was die Studenten wissen

- Dieses Kapitel wurde bisher immer[1] am Anfang der 2. Woche des ersten Semesters besprochen.
- Die zum Verständnis nötigen Shell-Kenntnisse sind dann bereits seit einer Woche bekannt, und die Teilnehmer haben sich in Zweiergruppen angemeldet.

[1] "immer" = seit Sommer 2013 in der *Schlüsselqualifikation für Informatiker*. Foliensatz z.B. unter https://svn.uni-konstanz.de/dbis/sq_15w/pub/

1.1 Subversion... as in coup d'état?

- Subversion (SVN) is a **version control system**. As such it provides means
  - to **keep track of changes** between versions of a project,
  - to allow people to **concurrently edit** the project,
  - to **resolve conflicting edits**, and
  - to **revert to earlier versions** of a project.

  $\Rightarrow$ SVN keeps a log of all changes!

- We consider **this lecture** a project. I do the slides, you solve the exercises, and the tutors revise them.

- SVN offers sufficient **access control** to isolate parts of the project.
  - Everybody can read the **public directory**. That's where the lecture slides and assignments are published.
  - Each **group of students** has read & write access to *their* subdirectory.
  - **The tutors** have read & write access to all group's directories, and probably some other rights.
  - My boss and I have **global access** to read & write.

Basic SVN work cycle

1. **Check out** a working copy from the central repository.
2. **Edit** as you like, maybe add further files to the project.
   - If you screwed up, revert to the version you have checked out.
3. **Update** your working copy to reflect changes others have committed in the meantime.
   - SVN tries to merge new changes into your working copy.
   - You need to resolve conflicts where SVN fails to guess right.
4. **Commit** your changes to the central repository.
5. `goto 2` (no need to check out again)

Note SVN can only merge edits in plain text data. It cannot trace changes in binaries (images, PDF or "office" documents, compiled programs)

- So *do not add binaries* to the repository if not absolutely necessary. (Although SVN does have space-efficient binary-diff storage)
- It is better to add the source that generates the binaries.
  ⇒ C sources instead of compiled programs**!**

Important advice

- Do not add files unrelated to **this** course.
- Only add the files asked for in the assignment.
- Do not add generated or downloaded data[2].

It is extremely difficult to remove data from the repository history, once it has been added.

[2] Better add information about how to generate, or where to find the data.

SVN's command line interface

SVN's understanding of arguments differs from traditional Unix tools:

- SVN can perform a bunch of different tasks.
- Instead of providing a separate program for each of them, there's only one `svn` program. (well, on the client side)
- Its first argument decides what task to perform.

`svn subcommand [argument...]` ▸ Execute one of svn's many subcommands with the appropriate arguments.

`svn help [subcommand]` ▸ Display a list of subcommands, or help for the one specified.

1.2 Checkout — svn co

`svn co url` ▸ Get a working copy from the repository at url.[3]

To check out the "public"[4] part of the lecture's repository:

- Due to magic, it is the same URL as on page 2.
- I'd suggest doing all this in a dedicated subdir, *e.g.*, `~/svn_15w`.

```
svn_15w $ svn co --username your.name https://svn.uni-konstanz.de/dbis/svn_15w/pub/
Authentication realm: <https://svn.uni-konstanz.de:443> Uni Konstanz Subversion
Repository
Password for 'your.name': ****              # the password for your.name@uni-konstanz.de
# ... a warning message about storing passwords unencrypted ...
Store password unencrypted (yes/no)? yes                                  # your choice
# ... list of what's being checked out, may be empty ...
Checked out revision 73.                                                  # number may differ
svn_15w $ ls
pub
svn_15w $ ls pub/
# ... you should see lecture slides here
```

[3] http://svnbook.red-bean.com/en/1.7/svn.tour.initial.html
[4] due to the current setup, authentification is required

- To check out your group's (say `foobar`) **private** directory[5]:

```
svn_15w $ svn co --username your.name \
> https://svn.uni-konstanz.de/dbis/svn_15w/group/foobar
# You will not be asked for credentials, if you have stored them in the previous step
# ... list of what's being checked out, may be empty ...
$ ls
foobar  pub
```

- You now have working copies of two different **sub**directories of the repository `https://svn.uni-konstanz.de/dbis/svn_15w`:

  `pub` is a copy of `^/pub/`

  `foobar` is a copy of `^/group/foobar/`

  (Where `^` is an abbreviation for the repository's location.)

[5] You'll learn your group's name from assignment 1, *cf.* page 2.

1.3 Query status / Add files — svn st / svn add

`svn st` ▸ List files with **changes**, and **new files** not yet under SVN's control. No output ⇒ nothing to report.[6]

`svn add file...` ▸ Schedule files for **addition** to the repository.[7]

- SVN does not automatically take care of **files you create** in a working copy. You need to tell SVN which files to add to the repository.

```
20 svn_15w $ cd foobar                          # in your group's directory
21 svn_15w/foobar $ nano newfile                # create a new file
22 svn_15w/foobar $ svn st
23 ?       newfile              # huh? — SVN does not yet handle this file
24 svn_15w/foobar $ svn add newfile
25 A         newfile                  # scheduling this file for Addition
26 svn_15w/foobar $ svn st
27 A       newfile       # SVN is planning to Add this file to the repository
```

[6] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.examine
[7] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.edit

1.4 Commit changes — svn ci

`svn ci -m 'message' [path...]` Commit (*aka.*, **check in**) all changes, providing a log message. The commit can be limited to certain paths.[8]

- When satisfied with your changes (*e.g.*, adding a file), you need to **commit** them to the repository.
- You *have to* provide a **log message**.

  Without `-m`, an editor (probably `nano`) will be launched where you can enter a message.

```
28 svn_15w/foobar $ svn st
29 A       newfile       # SVN is planning to Add this file to the repository
30 svn_15w/foobar $ svn ci -m 'blah blah blah'   # use concise messages
31 Adding         newfile                        # actually adding this file
32 Transmitting file data .
33 Committed revision 14.
34 svn_15w/foobar $ svn st                       # no output
```

[8] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.commit

1.5 Updating your working copy — svn up

- You should regularly update your working copies to receive any changes committed by your fellow students, tutors, or lecturers.

`svn up [path...]` **update** the working copies in each path, or the current working directory if no path is given.[9]

```
35 svn_15w $ svn up pub
36 At revision 16.                          # no new lecture slides in pub.
37 svn_15w $ svn up foobar
38 A    foobar/greeting                     # greeting has been Added.
39 Updated to revision 16.
40 svn_15w $ ls foobar
41 greeting newfile
```

- How did `greeting` get there? Probably someone in your group was a bit faster with the homework.

[9] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.update

1.6 Show modifications — svn di

`svn di [file...]` Show the **differences** in the given files, or all files.[10]

- When editing, `svn st` tells you which files are new, or have changes, `svn di` lists the **differences**. In a format known as *unified diff*.

```
42 svn_15w/foobar $ nano greeting  # change the file your fellow has checked in
43 svn_15w/foobar $ svn st
44 M       greeting                # this file has local Modifications
45 svn_15w/foobar $ svn di greeting
46 Index: greeting
47 ===================================================================
48 --- greeting    (revision 95)    # working copy was last updated to rev 95
49 +++ greeting    (working copy)
50 @@ -1 +1 @@
51 -hello                          # this line has been removed
52 +hello world                    # this line has been added
```

- You may commit modifications with `svn ci`, *cf.* page 11.

[10] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.examine

1.7 Undo your changes — svn revert

`svn revert file...` Revert changes on file made since the last update or checkout.[11]

- If you are unhappy with your modifications, ou may **revert** to the version of your last update.

```
53 svn_15w/foobar $ svn revert greeting
54 Reverted 'greeting'
55 svn_15w/foobar $ svn st                                    # no output
```

- The `svn merge` command[12] even allows to "undo" previously committed revisions. See the manual, this is rather advanced!

[11] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.revert
[12] http://svnbook.red-bean.com/en/1.7/svn.branchmerge.basicmerging.html

1.8 When things go awry

- There are many possibilities where **conflicts** can arise.
  - Two people edit the same region of a file in different ways, so that SVN cannot merge the edits.
  - You try to add a file that someone has added after your last checkout.
  - ...
- If your working copy of modified files is **out of date**, then `svn ci` will fail. You need to update first.
- The update may fail, leaving you enough information to sort out the mess.

Note Resolution of tree conflicts (*cf.* page 19) has changed significantly since Subversion 1.5. A rather complex example[13] can be found in the manual.

[13] http://svnbook.red-bean.com/en/1.7/svn.tour.treeconflicts.html

A text conflict

Text conflicts arise, when SVN cannot merge concurrent edits that happened to the same region of a file.

- Assume you have edited a file `data` which is already part of the repository:

```
1 svn_15w/foobar $ svn st
2 M       data                  # There are local Modifications, waiting to be committed
```

- But when you try to commit, something strange may happen:

```
3 svn_15w/foobar $ svn ci -m 'better now'
4 Sending        data
5 svn: E155011: Commit failed (details follow):
6 svn: E155011: File '/home/sk/svn_15w/foobar/data' is out of date    # local path
7 svn: E160028: File '/group/foobar/data' is out of date              # path in repos
```

  - The local working copy of `data` is **out of date**, *i.e.*, someone else has **committed** changes of `data` after your last update.
  - SVN has failed to automatically merge your edits with those already committed.

- So you first have to **update** your working copy.

- Let's do so:

```
svn_15w/foobar $ svn up
Updating '.':
C    data                                    # this file contains the conflict
Updated to revision 3.
Conflict discovered in file 'data'.
Select: (p) postpone, (df) show diff, (e) edit file, (m) merge,
        (mc) my side of conflict, (tc) their side of conflict,
        (s) show all options: p              # see the manual for the other options
Summary of conflicts:
  Text conflicts: 1                          # there is one text conflict
```

- The update creates a bunch of new files:

```
svn_15w/foobar $ svn st
C       data        # merged version of data, contains conflict markers
?       data.mine   # backup of your version of data
?       data.r2     # copy of revision 2 of data, i.e., before your edits
?       data.r3     # copy of revision 3 of data, i.e., the one up to date
Summary of conflicts:
  Text conflicts: 1
```

- Use an editor to fix data, using the other files for reference:

```
$ nano data
```

Resolving conflicts — `svn resolve`

`svn resolve --accept=working file...` Use the current version of files to **resolve** conflicts. See the manual for other values than `working` to use with `--accept`.[14]

- You cannot commit the working copy until the conflict is resolved:

```
1 svn_15w/foobar $ svn ci -m 'better now'
2 svn: E155015: Commit failed (details follow):
3 svn: E155015: Aborting commit: 'svn_15w/foobar/data' remains in conflict
```

- You need to tell SVN that you are **done with resolving** the conflict:

```
4 svn_15w/foobar $ svn resolve --accept=working data   # tell SVN: conflict is resolved
5 Resolved conflicted state of 'data'
6 svn_15w/foobar$ svn st
7 M       data                          # no conflict any more, but uncommitted modifications
8 svn_15w/foobar$ svn ci -m 'merged my edits'    # try to commit again, may fail again
9 Sending        data
10 Transmitting file data .
11 Committed revision 4.
```

[14] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.resolve

A tree conflict

Tree conflicts arise, when SVN cannot merge concurrent changes in the directory structure.

- Assume you have created `file1`, but not (yet?) scheduled it for addition to the repository.

```
1 svn_15w/foobar $ svn st
2 ?       file1                                  # SVN does not care about this file
```

- But when you try to update, something strange may happen again:

```
1 svn_15w/foobar $ svn up
2 Updating '.':
3    C file1
4 At revision 4.
5 Tree conflict on 'file1'
6    > local file unversioned, incoming file add upon update
7 Select: (r) mark resolved, (p) postpone, (q) quit resolution, (h) help: p
8 Summary of conflicts:                          # again, we have typed p here
9   Tree conflicts: 1
```

  - Someone else has **added** a `file1` after your last update.

Here we can choose one of two options:

- **Option 1:** `revert file1` will toss your version of the file, and replace it with the one already committed.
  - You revert to the version where `file1` did not exist, then it's added by the update.

```
svn_15w/foobar $ svn revert file1
svn_15w/foobar $ svn st # No output, conflict disappeared
svn_15w/foobar $ ls
file1 # this is the file that came from the repository
```

- **Option 2**: Resolve by accepting the state of the working copy. This will schedule `file1` for deletion.
  - Note, that this is consistent: In your working copy, `file1` is *not* part of the repository.

```
svn_15w/foobar $ svn resolve --accept=working file1
Resolved conflicted state of 'file1'
svn_15w/foobar $ svn st
D       file1                                   # Note the D: scheduled for deletion.
svn_15w/foobar $ svn ci -m'file1 should not be checked in'
Deleting       file1
Committed revision 9.
```

1.9 Tree changes — svn rm/mv/cp

You must not **(re)move** files under SVN's control using `rm` or `mv`.

- SVN will report moved files as missing, since it cannot trace the OS's own `mv` command.
- Use SVN's subcommands to (re)move or copy files[15]:

`svn rm file...`	Mark files for **removal** from the repository.
`svn mv src dst`	**Rename/move** files under svn's control.
`svn cp src dst`	**Copy** files under svn's control.

- Conflicting operations on the directory structure may also lead to conflicts on update.

[15] http://svnbook.red-bean.com/en/1.7/svn.tour.cycle.html#svn.tour.cycle.edit

Disclaimer

- Das wird **kein SVN-Kurs** — ich biete keinen SVN-Support.
- Das ist ein **Erfahrungsbericht** — keine SVN-Mission.
- Ich organisiere **nur meine** Vorlesungen.

Zielgruppe

- Personen:
  - Vorlesungsorganisierer, Gruppeneinteiler, Autoren von Übungsblättern,
  - Tutoren, Dozenten.
- Vorlesungen:
  - Studenten bearbeiten **Übungen in Gruppen**.
  - Übungen enthalten **viel Quellcode** oder zumindest Plain Text (und sind zu 100% elektronisch abgebbar).

3
Aus Sicht der Studenten

3.1 Anmelden und Einrichten

Vorlesungsmaterial besorgen

- Auf der Webseite zum SVN-Repository sind alle Materialien zugänglich:

  https://svn.uni-konstanz.de/dbis/svn_15w/pub

  - Für die ganze Welt, auch ohne SVN.
  - Dieser Link steht im LSF.

- Folien oder 1. Übungsblatt enthält Informationen zur Anmeldung.

Partnersuche & Anmeldung

Die Studenten sollen die Übungen in 2er-Gruppen bearbeiten.

- 1. Aufgabe: **Partner suchen**.
- Jede Gruppe schreibt **eine Mail**. Wenn alle Gruppenmitglieder als Absender oder im Cc genannt werden, dann
  - fallen **Fehler** (Typos) gleich auf, und
  - alle Beteiligten sind informiert.

```
1 From: micha.kraemer@uni-konstanz.de
2 To: stefan.klinger@uni-konstanz.de
3 Cc: andre.krueger@uni-konstanz.de
4 Subject: svn_15w Übungsgruppe
5
6 Kein weiterer Inhalt?  Evtl. Wunschtermine für Tutorien?
```

Wichtig Wir verwenden die **@uni-...Mail-Adressen**, weil die Authentifizierung vom Rechenzentrum darüber läuft.

- Als Antwort bekommt jeder Student eine Mail.
  - Kann ein paar Tage dauern.

```
From: Stefan Klinger <stefan.klinger@uni-konstanz.de>
To: micha.kraemer@uni-konstanz.de, andre.krueger@uni-konstanz.de
Subject: svn_15w Gruppe

Willkommen Gruppe kraekrue!

Die aktuellen Vorlesungsmaterialien können Sie mit

  svn co --username user https://svn.uni-konstanz.de/dbis/svn_15w/pub

auschecken, und dann wie gewohnt mit ‘svn up‘ aktualisieren.

Für Sie wurde außerdem ein privates Verzeichnis in unserem
SVN-Repository angelegt, in dem Sie Ihre wöchentlichen Übungen
abgeben:

  svn co --username user https://svn.uni-konstanz.de/dbis/svn_15w/group/kraekrue

Ersetzen Sie dabei user mit Ihrem Email-Alias @uni-konstanz.de.
```

3.2 Neue Folien & Übungsblätter runterladen

- Mit dem Webbrowser auf die Seite vom SVN-Repo gucken.
  - Funktioniert immer ⇒ **Fallback**.

- **Einmalig**, am Anfang des Semesters.

```
~ $ mkdir ~/studium/svn_15w/ # Lokales Verzeichnis für diese Vorlesung anlegen
~ $ cd ~/studium/svn_15w/
~/studium/svn_15w $ svn co --username andre.krueger \
                                https://svn.uni-konstanz.de/dbis/svn_15w/pub
# Beim allerersten Mal muss man das RZ-Passwort eingeben
A    pub/lecture01.pdf
Checked out revision 12.
```

- **Aktualisieren**. Wieder, und immer wieder...

```
~/studium/svn_15w $ svn up pub
Updating 'pub':
A    pub/assignment01.pdf    # schön: ein neues Übungsblatt
Updated to revision 12.
```

3.3 Das Grupenverzeichnis

- **Einmalig**, am Anfang des Semesters.

```
~ $ cd ~/studium/svn_15w/ # Lokales Verzeichnis für diese Vorlesung
~/studium/svn_15w $ svn co --username andre.krueger \
                        https://svn.uni-konstanz.de/dbis/svn_15w/group/kraekrue
Checked out revision 12.
~/studium/svn_15w $ ls -l
total 0
drwx------ 1 sk users  8 Feb 18 18:00 kraekrue  # Für meine Gruppe, Schreibrechte.
drwx------ 1 sk users 34 Feb 18 17:47 pub       # Hatten wir schon: public, nur lesbar.
```

- **Aktualisieren**.

```
~/studium/svn_15w $ svn up kraekrue
Updating 'kraekrue':
A    kraekrue/01/korrektur.txt     # schön: Korrektur vom Tutor
Updated to revision 14.
```

  - Hier sehe ich auch wenn mein Partner die Lösung verbessert hat.

Alles auf einmal

```
1 ~ $ svn up studium/svn_15w/*
```

3.4 Übungen abgeben / einchecken

```
~/studium/svn_15w/kraekrue $ mkdir 01                    # Verzeichnis fürs 1. Blatt
~/studium/svn_15w/kraekrue $ emacs 01/loesung.txt
~/studium/svn_15w/kraekrue $ svn add 01
A         01
A         01/loesung.txt
~/studium/svn_15w/kraekrue $ svn ci -m 'Ideen für Aufgabe 1'
Adding         01
Adding         01/loesung.txt
Transmitting file data .done
Committing transaction...
Committed revision 13.
```

- Ab jetzt ist diese Lösung auf dem SVN Server.
  - Der **Übungspartner**/Tutor/Dozent kann sie abrufen: `svn up`
  - Wenn die **Deadline** abläuft gilt der letzte Commit davor.

Struktur Wir schreiben den Studenten vor, die Lösungen in entsprechend benannten Unterverzeichnissen abzulegen (Blatt 1 → 01).
⇒ Wenig Such- oder Korrekturaufwand beim Tutor.

Zusammenarbeit mit dem Partner

- Die Lösung kann **weiter bearbeitet** werden:

```
~/studium/svn_15w/kraekrue/01 $ vi loesung.txt
~/studium/svn_15w/kraekrue/01 $ svn st
M       loesung.txt
~/studium/svn_15w/kraekrue/01 $ svn ci -m'bugfix'
Sending        loesung.txt
Transmitting file data .done
Committing transaction...
Committed revision 15.
```

- Von allen **Gruppenmitgliedern**:

```
andererStudi/svn_15w/kraekrue/01 $ svn up
Updating '.':
U    01/loesung.txt     # Aha, mein Partner hat was getan
Updated to revision 16.
```

- SVN speichert die **Historie**. (Billiges Backup; Verteidigung bei Plagiat?)

Kollaboration In der SQ-Vorlesung haben wir das ein Semester lang geübt. In der Einführung habe ich auch gezeigt **wie man Konflikte auflöst**.

4
Aus Sicht der Tutoren

4.1 Struktur des SVN-Repositories

- Das Repository hat eine ganze Menge **Unterverzeichnisse**:

`/pub/` **Öffentlich:** Folien, Übungsblätter, Musterlösungen...

`/lect/` Vorbereitung der **Folien**.

`/ass/` Vorbereitung künftiger **Übungsblätter**.

`/tut/` **Musterlösungen** für die Tutoren.

`/group/` Die Verzeichnisse der einzelnen **Gruppen**, darin z.B.:

- `groupname/01/...` Abgabe zum 1. Blatt.
- ...

`/misc/` **Verwaltung:** Gruppen, Bewertungen, Noten, ...

- `groups` Einteilung der Studis in **Gruppen und Tutorien**.
- `tutorname/` Ein Verzeichnis je Tutor, darin z.B.:
  - `punkte` **Bewertung** der korrigierten Gruppen.
  - ... Persönlicher Bedarf.

Hervorgehoben: Schreibrechte für alle Tutoren. Sonst nur Leserechte.

4.2 Arbeitszyklus für Tutoren

- **Einmalig**, am Anfang des Semesters.

```
1 ~ $ cd ~/tutorials/ # Lokales Verzeichnis für meine Tutorien
2 ~/tutorials $ svn co https://svn.uni-konstanz.de/dbis/svn_15w
```

  - Tutoren Leserechte auf der **Wurzel** des Repos.

- **Regelmäßig**:
  - Checkout zur Deadline[16] — hier am 26. Februar 2016, um 13:15.

```
1 ~/tutorials/svn_15w $ svn up -r'{2016-2-26 13:15}'
2 ~/tutorials/svn_15w $ less misc/groups    # Welche Gruppen muss ich korrigieren?
```

  - Korrektur. Punkte eintragen. Commit:

```
1 ~/tutorials/svn_15w $ svn ci -m'Korrektur Tutorium A fertig'
2 Sending        group/foobar/01/korrektur.txt
3 # ... viele weitere Gruppen die ich korrigiert habe...
4 Sending        misc/tutorname/punkte          # Punktetabelle
5 Transmitting file data ..done
6 Committing transaction...
7 Committed revision 22.
```

[16] http://svnbook.red-bean.com/en/1.7/svn.tour.revs.specifiers.html#idp9002784

Mögliches Problem: Abgabe nach Deadline

- So sieht's aus:

```
~/tutorials/svn_15w $ svn up -r'{2016-2-22 17:30}'   # Checkout zur Deadline
# ... *vormichhinkorrigier* ...
~/tutorials/svn_15w $ svn ci -m'Korrektur Blatt 2 fertig'
Sending        group/foobar/02/hello_world
svn: E155011: Commit failed (details follow):
svn: E155011: File 'svn_15w/group/foobar/02/hello_world' is out of date
svn: E160024: resource out of date; try updating
```

- **Überschreibe** spätere Änderungen.

```
~/tutorials/svn_15w $ svn up --accept mine-full group/foobar/02
Updating 'group/foobar/02':
C    group/foobar/02/hello_world
Updated to revision 24.
Resolved conflicted state of 'group/foobar/02/hello_world'
~/tutorials/svn_15w $ svn ci -m'Korrektur Blatt 2 fertig'
#...
Committed revision 25.
```

  - SVN merkt sich was da überschrieben wurde. Alles ist **wiederherstellbar**.

Obacht Nicht die Abgaben für das **kommende Blatt** löschen!

4.3 Punkte verwalten

- Die Tutoren müssen den **Punktestand** der Gruppen/Studenten notieren.
- Gruppenverzeichnisse sind **nicht der richtige Ort**, weil die Studenten dort Schreibrechte haben. (wäre Nachvollziehbar, macht Arbeit)

Wo aufschreiben?

- **Zentral:** `/misc/punktestand`
  - Es gibt eine Datei in die jeder Tutor die Punkte einträgt.
  - Tendentiell wollen alle gleichzeitig Punkte eintragen.
- **Dezentral:** `/misc/tutorname/punkte`
  - Jeder Tutor pflegt eine eigene Punktetabelle.
  - Die Daten müssen zusammengeführt werden.
  - Script mit **Konsistenzchecks**.
    (nicht mehr als erreichbare Punkte; nur eine Bewertung je Blatt und Student; ...)
- **Feedback:** Den Studenten regelmäßig ihren Punktestand mitteilen.
  - z.B. eine regelmäßig aktualisierte Datei `/group/foobar/punkte`.

Was aufschreiben?

- Punkte für jeden Studenten einzeln aufschreiben.
- Punkte nur für jede Gruppe aufschreiben.
  - ☼ Vermeidet Redundanz und damit Fehlerquellen.
  - ☼ Leichter nachvollziehbar was passiert ist.
  - ☔ Schwieriger auszurechnen, insbesondere bei Gruppenänderungen.
    ⇒ Scripten.

Dateiformat Prinzipiell egal, aber:

- Subversion verfolgt Änderungen in **Plain Text**.
- CSV lässt sich leicht scripten und in **Tabellenkalkulationen** importieren.
- Prinzipiell können Punktetabellen auch in einer R- oder Python-Datei liegen...

4.4 Korrekturen

- Wir sind hart bei der Deadline.
  - Relevant ist die Uhrzeit auf dem SVN-Server.
  - Erfahrung: Jede Deadline wird ausgetestet.
- Ich stelle den Tutoren frei wie sie die Korrektur einfügen,
  - als **separate Datei** mit Anmerkungen, oder
  - durch **Ändern der Abgabe**.
    - Sinnvoll z.B. bei Fixes in Sourcecode.
    - Dabei markante Kommentare verwenden.
    - SVN merkt sich die Geschichte und zeigt Änderungen an!
- Änderungen an den Gruppen.
  - Es ist wichtig den Tutoren mitzuteilen welche Gruppen sie korrigieren sollen.
  - Aktualisierung von `/misc/groups` ⇒ Mail an Tutoren.

5
Aus Sicht des Dozenten

5.1 Wie bekomme ich ein SVN-Repository?

- Der SVN-Dienst wird vom Rechenzentrum angeboten[17].

  Für jede **Arbeitsgruppe** *ist mindestens ein* **delegierter Administrator** *erforderlich. Dieser verwaltet die Benutzer und Repostories für die Gruppe.*

  —*RZ Webseite*

  - Der hat einen **SSH-Zugang** auf `svn.uni-konstanz.de`.
  - Wir werden **oft Berechtigungen ändern** müssen (neue Studi-Gruppe).
    ⇒ Am besten man macht's selber.

- Ansprechpartner im Rechenzentrum ist

  Rainer Rutka, Raum V511

  - Support, neue SSH-Zugänge, *etc.*...

[17] http://svn.uni-konstanz.de

Repository anlegen

- Dokumentation vom RZ lesen!
  - Eine `README` die man nach dem SSH-login findet.
  - Ein Beispiel findet sich dort auch...
- Repository anlegen.

```
1 ~ $ ssh stefan.klinger@svn.uni-konstanz.de          # Dein Name natürlich
2 Last login: Wed Feb 24 15:10:07 2016 from phobos90.inf.uni-konstanz.de
3 groups: cannot find name for group ID 200009951
4 ~ $ cd /home/svn/ag                  # statt ag das Kürzel Deiner Arbeitsgruppe
5 /home/svn/ag $ svnadmin create repos/repo          # Name des neuen Repos
```

- Das Repository ist da, aber bis jetzt kann niemand darauf zugreifen.
  $\Rightarrow$ Wir müssen die Zugriffsrechte klären.

5.2 Zugriffsrechte

Die Zugriffsrechte für **alle Repositories der Arbeitsgruppe** sind in einer **gemeinsamen** Datei `/home/svn/ag/access` festgelegt.

⇒ Obacht, das ist wenig fehlertolerant.

Struktur Windows `ini`-style:

- Ein Abschnitt `[groups]` in dem mehrere User zusammengefasst werden können: (hier nenne ich die **SVN-Gruppe**)

```
1 gruppenname = vorname.nachname, ...
```

  - Ich benutze das *nicht* um Übungsgruppen abzubilden.

- Ein Abschnitt je **Verzeichnis**, gefolgt von einer Liste von Benutzern oder SVN-Gruppen und deren Rechte:

```
2 [repo:/path/inside/repo]
3 vorname.nachname = r    # Dieser User darf lesen (checkout)
4 @gruppenname = rw       # Die dürfen alle lesen und schreiben (commit)
5 ...
```

Good Practice

- Regelmäßig aufräumen: Einträge für **alte Repos löschen**.
- Bei mehreren Admins: Auf **konkurrierende Edits** achten!
- Jeder SVN-Gruppenname beginnt mit dem Namen des entsprechenden Repos.
- Bei vielen Repos[18] kann `access` **sehr unübersichtlich** werden.
  - Eine Rechte-Datei pro Repository verwalten.
  - `/home/svn/ag/access` von einem Skript[19] erzeugen lassen.

Obacht 

- Bei Fehlern in der `access`-Datei sieht man **keine Fehlermeldung**.
- Verwendung undefinierter SVN-Gruppe ⇒ **kein Repository** mehr zugreifbar!

[18] DBIS verwendet 10–20 Repositories gleichzeitig.
[19] Beispiel: https://svn.uni-konstanz.de/dbis/svntools/mkaccess/ —kein Support!

Beispiel

```
[groups]
svn_15w_staff = marc.scholl, stefan.klinger
svn_15w_hiwi = elizabeth.parker, chestnut.barrow
# ... Gruppen für andere Repositories ...
```

```
[svn_15w:/]
@svn_15w_staff = rw
@svn_15w_hiwi = r

[svn_15w:/pub]
* = r                    # Öffentlich lesbar
@svn_15w_staff = rw      # Dozenten dürfen schreiben

[svn_15w:/group]         # Die Abgeben der Studis
@svn_15w_hiwi = rw

[svn_15w:/tut]           # Musterlösungen
@svn_15w_hiwi = rw

[svn_15w:/exam]          # Klausurvorbereitung
@svn_15w_hiwi =          # HiWis dürfen nicht lesen

# ... Rechte anderer Repositories ...
```

```
# Ein Abschnitt je Übungsgruppe:

[svn_15w:/group/kraekrue]
micha.kraemer = rw
andre.krueger = rw

[svn_15w:/group/becklang]
alfonso.becker = rw
walther.lang = rw

[svn_15w:/group/vogbramey]
angelo.vogel = rw
gunnar.braun = rw
volker.meyer = rw

# Die können nicht mehr abgeben:
[svn_15w:/group/meiewurs]
sven.meier = r
niklas.wurst = r

# ... ... ...
```

Anmerkungen

- Rechte werden auf Unterverzeichnisse **vererbt**[20]...

```
31 [svn_15w:/group]
32 @svn_15w_hiwi = rw          # Tutoren können alle Gruppenverzeichnisse lesen & schreiben
```

  - ...und können für Unterverzeichnisse **erweitert oder eingeschränkt** werden.
  - **Nicht lesbare** Unterverzeichnisse werden beim Checkout übersprungen.

- Tutoren nicht auf ihre auf ihre Gruppen einschränken:
  - ☼ Tutor bekommt nur die Gruppen die er auch korrigieren soll.
  - 🌧 Wenig flexibel (Änderungen der Gruppenstruktur; Vertretung; ...)
  - 🌧 Im Zweifel steht im Log wer was geändert hat.

[20] http://svnbook.red-bean.com/en/1.7/svn.serverconfig.pathbasedauthz.html

- Sollte `svn_15w:/` für alle lesbar sein?
  - man kann die Rechte für alle **Unterverzeichnisse einzeln** einschränken.

Studenten brauchen nur einen Checkout an der Wurzel.

```
~/studium/svn_15w $ svn co --username andre.krueger \
                                    https://svn.uni-konstanz.de/dbis/svn_15w
A    svn_15w/pub
A    svn_15w/group/kraekrue
Checked out revision 12.
```

Obacht beim Anlegen neuer Verzeichnisse.

- Klausurvorbereitung lesbar für die Welt?

Student hat eine unnütze Hierarchieebene mehr (`group`).

5.3 Gruppen verwalten

- Subversion kann das nicht, ist ja keine Gruppenverwaltung.
- Tu was Du willst, meinetwegen nimm Excel.
- Ich mag Plain Text, das **lässt sich gut scripten**!

```
1 svn_15w $ less misc/groups
2 # Format: <gruppe> <tutor> <teilnehmer>+
3 kraekrue B  micha.kraemer andre.krueger
4 becklang A  alfonso.becker walther.lang
5 meiewurs ~  sven.meier niklas.wurst       # → buscmeie
6 brauschm ~  gunnar.braun jan.schmid       # jan.schmid hört auf
7 #...
```

Bewährt hat sich für mich dieses Format:

- Name der Gruppe
- Zuteilung zu Tutor(ium), ~ steht für geschlossene Gruppen.
- Aliase der Mitglieder.

Und Kommentare der #-eol Form. **Obacht**, kein Quoting!

Was ist eine Übungsgruppe?

Übungen werden von Studenten in *Gruppen* bearbeitet.

Tutoren korrigieren die Abgaben der *Gruppen*.

Jeder Student hat so viele Punkte wie seine Gruppe?

- Brauchen **Zuordnung**: Student → Gruppe, und Gruppe → Tutor.
- Was ist wenn ein Student die **Gruppe wechselt**?

  (meist studiert der Partner doch lieber "was ohne Mathe")
  - Bewährt hat sich **Gruppen nicht zu ändern**.
  - Wenn sich Studenten **neu zusammentun** bekommen sie eine **neue Gruppe**. Die alte Gruppe wird gesperrt (keine weitere Abgabe).

⇒ Ein Student hat so viele **Punkte wie alle seine Gruppen**.

- Jeder Student ist in max. einer **aktiven** Gruppe.

Von Hand, zu Fuß, oder auf allen vieren?

- Die Studenten melden ihre Gruppen per Mail an (*cf.* page 25).

```
From: micha.kraemer@uni-konstanz.de
To: stefan.klinger@uni-konstanz.de
Cc: andre.krueger@uni-konstanz.de
Subject: svn_15w Übungsgruppe
```

- Man kann die Gruppen **von Hand** einteilen ihnen **je eine Mail** schreiben, die einzelnen **Rechte** in die access-Datei tippen, **Gruppenverzeichnisse** erstellen...
- Oder: **Ein Script** wird mit den Aliasen gefüttert, und erledigt all diese Aufgaben.

Aufwand Immens beim ersten Mal, minimal in den Jahren danach.

Anregung: Script newgroup[21]

```
svn_15w/misc/sk $ run/newgroup
========================================================== Add email
aliases of group members, empty line ends.
svn_15w alias: aaron.aal
svn_15w alias: babette.brett
svn_15w alias:                                    # Keine Eingabe: Gruppe vollzählig

svn_15w group name: aalbret
svn_15w tutorial: b
------------------------------------------------------------
  group: aalbret
members: aaron.aal babette.brett
  tutor: B
continue? [yN]yes

mail: ok, svn: ok, permissions: ok.
============================================================
Add email aliases of group members, empty line ends.          # nächste Gruppe
...
```

- Selber scripten! (Sorry — ich passe das jedes Semester ein wenig an)

[21] https://svn.uni-konstanz.de/dbis/svn_15w/pub/scripts/newgroup —kein Support!

Anregung: Script updateSvnAccess[22]

```
12 key='svn_15w';              # Name des Repositories
13 groupfile='../groups';      # Gruppendefinitionen
14
15 {
16     echo -e '# File autogenerated by ‘updateSvnAccess‘.\n';
17
18     cat svn.access.head;    # Statische Rechte: Tutoren, /pub, ...
19
20     sed -r 's/#.*$//; /^\s*$/d' "${groupfile}" | while read g a ms; do
21
22         test "$a" = '~' && rights=r || rights=rw;
23
24         echo -e "\n[${key}:/group/${g}]";
25         for m in ${ms}; do echo "${m} = ${rights}"; done;
26
27     done;
28 } | ssh svn "tee dbis/access.d/${key} >/dev/null; cd dbis; ./mkaccess";
```

- **Obacht** — das ist eine vereinfachte Version.

[22] https://svn.uni-konstanz.de/dbis/svn_15w/pub/scripts/updateSvnAccess —kein Support!

5.4 Stolpersteine

- Potentiell hat jeder Teilnehmer eine **andere Plattform**.
- Unterschiede in **Dateisystemen** und **Text Encoding**, ...

Dateisystem

```
svn_15w/group/foobar $ ls -l
total 21k
-rw------- 1 sk users   6 Feb 25 10:46 abgabe_2.txt
-rw------- 1 sk users   6 Feb 25 10:46 Abgabe_2.txt
-rw------- 1 sk users 223 Feb 25 11:01 README
-rw------- 1 sk users  20 Feb 25 10:44 Übung_1.txt
-rw------- 1 sk users  13 Feb 25 10:45 Übung_1.txt
```

- Unter Linux kann man das auschecken.
- Auf Apple nicht.
- Windows?

- Kommt **extrem selten** vor, passiert eigentlich nur aus Versehen.
- Die Studenten kennen das Problem und Lösungen.
  - Wenn's passiert ist: Checkout unter Linux, eine Datei löschen.

Zeilenende und Encoding

Für Textdateien sollte man...

- ...die **EOL-Konvention** auf "`native`" setzen. Subversion konvertiert dann automatisch in die **jeweils lokale** Konvention.

```
1 $ svn ps svn:eol-style native filename
```

- ...ein **gemeinsames Encoding** vorschreiben: UTF-8.

```
1 $ svn ps svn:mime-type 'text/plain;charset=UTF-8' filename
```

MIME-Type

Der Apache muss den Typ der Dateien in `/pub` wissen:

```
1 $ svn ps svn:mime-type 'application/pdf' folie.pdf
2 $ svn ps svn:mime-type 'image/jpeg'  foto.jpeg
3 $ svn ps svn:mime-type 'text/plain;charset=UTF-8' sourcecode.java
```

- **Subversion Properties**[23] sind im Handbuch beschrieben.
- Automagisch[24] kann man das in `~/.subversion/config` festlegen, z.B.:

```
1 [miscellany]
2 enable-auto-props = yes
3
4 [auto-props]
5 *.lhs = svn:mime-type=text/plain;;charset=UTF-8;svn:eol-style=native
6 *.pdf = svn:mime-type=application/pdf
```

- 1. Übung in *Informatik 2*: Konfigurieren Sie SVN entsprechend.

[23] http://svnbook.red-bean.com/en/1.7/svn.ref.properties.html
[24] http://svnbook.red-bean.com/en/1.7/svn.advanced.confarea.html

5.5 Sonstiges

Verteilung der Gruppen auf Tutorien

- LSF kann nur Studenten auf Tutorien verteilen — **pain in the back**!
  - Zuteilung ist zufällig.
  - Hat kein Konzept von Gruppen.
  - Viel Handarbeit nötig.
- Ich lasse die Studis ein **Ranking ihrer Wunschtermine** abgeben.
  - Vorrecht bei nachgewiesener Kollision mit **Pflichtvorlesung**, ...
  - oder für Studis **mit Kindern** (Nachweis: StEP).
- Ein Skript findet schmerzarme Verteilung auf die Tutorien.
  - Stichtag vor dem ersten Tutorium.
  - Info über weitere Mail.

Änderung der Gruppen

- Alte Gruppe schließen, neue Gruppe erzeugen.
  - Zusammensetzung einzelner Gruppen nie ändern.
- Schreibrechte auf alte Gruppe entfernen.
  - Keine versehentlich Abgabe dort.
  - Im Auge behalten: Wann ist die nächste Deadline? Ggf. Übergangszeit einplanen.
- Tutoren informieren.

Backup und Archivierung

- Nach der Nachklausur (weil ich auch die Klausur und Notenvergabe im Repo halte).
- Mindestens[25] SVN **Dump** (für die Historie der Abgaben), und **letzte Version**.
- Alte Repos auf `svn.uni-konstanz.de` nicht löschbar.
  - Verschieben in einen Ordener `DELETE_ME`, und hin und wieder einen SVN-Admin im RZ fragen den zu löschen.

Missbrauch der Repos z.B. für andere Veranstaltungen.

- Das können **sehr viele Daten** werden die nicht in's Archiv gehören.
- Evtl. kann man große Commits mit **Server-seitigen Hooks** blocken — habe ich nie probiert.
- Klare Ansage an die Studis war bisher ausreichend.

[25] Beispiel: https://svn.uni-konstanz.de/dbis/svntools/freezeRepos/ —kein Support!