Versionsverwaltung mit Mercurial

by

Dr. Günter Kolousek

Begriff und Aufgaben

- Versionsverwaltung (revision control, version control, source control)
- Verwaltung der Änderungen an Dokumenten (Dateien wie Source Code,…)
  - Änderung hat
    - eindeutige Bezeichner (Ziffern, Buchstaben)
    - einen Zeitstempel
  - Aufgaben
    - Protokollieren der Änderungen: wer?, wann?
    - Wiederherstellen eines alten Zustandes: Änderungen rückgängig machen
    - Archivieren: auf alte Versionen zugreifen können
    - Koordinieren der Zugriffe: was passiert wenn mehrere gleichzeitig ändern wollen?
    - Verwalten mehrerer Entwicklungszweige (branch)

Semantic Versioning: semver.org

- `major.minor.patch` ... natürliche Zahlen
  - Pre-release: ”-” + ..., z.B. 1.0.4-alpha.3
  - Build-release: ”+” + ..., z.B. 2.3.0+build.20.d0ee82e726b2
- Regeln:
  - jeweils um eins erhöhen: 1.9.0 $\rightarrow$ 1.10.0 $\rightarrow$ 1.11.0
  - major++ $\rightarrow$ minor = 0, minor++ $\rightarrow$ patch = 0: 1.9.3 $\rightarrow$ 2.0.0
  - Projektanfang: major = 0, beliebige Änderungen
  - Änderung von patch: nur bug fixes (MUST)
  - Änderung von minor
    - rückwärtskompatible Änderungen am public API (MUST)
    - Teile des API als ”deprecated” markiert (MUST)
    - Substantielle Änderungen am privaten Teil (MAY)
    - patch-level Änderungen (MAY)
  - Änderung von major
    - rückwärtsinkompatible Änderungen (MUST)
    - minor- und patch-level Änderungen (MAY)

Versionsverwaltungssysteme

- Begriffe: RCS oder VCS
- 3 Arten
  - lokale Versionsverwaltung
  - zentrale Versionsverwaltung
  - verteilte Versionsverwaltung

Lokale Versionsverwaltung

- nur auf einem Computer
- verwaltet nur Dateien
- z.B.: RCS, SCCS
- daher: rein historische Bedeutung

Zentrale Versionsverwaltung

- zentrales Repository
  - Client/Server
  - zentrale Rechteverwaltung
- CVS
  - verwaltet keine Verzeichnisse
  - keine atomaren Aktionen
  - wird heute fast nicht mehr verwendet
- Subversion
  - Probleme beim Verschieben und Umbenennen von Dateien
  - Tool der Wahl, wenn Server/Client-Modell

Verteilte Versionsverwaltung

- Distributed Version Control System (DVCS)
- jeder hat eigenes Repository
  - jeder hat daher gesamte Änderungsgeschichte
  - einchecken ohne Verbindung zum Server
  - Änderungen müssen zusammengeführt werden (merge)
- z.B.: GIT, Mercurial

Vergleich von GIT & Mercurial

- Gemeinsamkeiten
  - Win, Unix, Mac OSX
  - GPL
  - Änderungen: gesamter Baum (nicht nur Datei)
- GIT
  - ursprünglich von Linus Torvalds
  - C, Shell Skripte, Perl
  - Linux Kernel, GNOME, Qt, Eclipse,…
- Mercurial
  - ursprünglich von Selenic Consulting
  - Python, C
  - Facebook, Mozilla, Netbeans, OpenJDK, nginx,…

Vergleich von GIT vs. Mercurial – 2

- Git
  - komplizierter in Bedienung
    - viele Einzelkommandos (>130)
  - (berühmt wegen) github
    - nur kostenlose *öffentliche* Repositories
  - gitlab
    - auch kostenlose private Repositories!
- Mercurial
  - einfacher in Bedienung
  - Anbindung an Git-Repository mittels Hg-Git Plugin
  - bitbucket
    - kostenlose private Repos bis 5 Benutzer
- Vergleich und Gegenüberstellung: GitConcepts
- Für (eingefleischte) Git-Benutzer: githelp Extension!
  - gibt an, wie hg Kommando zu angegebenen `git` Kommando lautet

Links zu Mercurial

- Mercurial
  - dort zu finden:
    - A Tutorial on Using Mercurial (auch in Deutsch)
    - Understanding Mercurial
- Hg Init
- Mercurial: the definitive guide
- Version Control with Mercurial

… und gleichzeitig auch die Quellen für den Mercurial-Teil dieser Folien!

Texteditor

Viele Editoren bieten eine Unterstützung für Mercurial an. Die besten/beliebtesten Editoren:

- Emacs, vi
- Sublime Text, Atom
- Visual Studio Code
- Netbeans, Eclipse, QtCreator, Visual Studio

Feel free to choose whatever you want…

Aber bedenke:

> *You are free to choose, but you are not free to alter the consequences of your decisions. – Esra Taft Benson*

Installation von Mercurial

Linux Am einfachsten ist es für den Linux-Benutzer, denn der kann einfach sein Lieblingspaketverwaltungstool zum Einsatz bringen. Exemplarisch für ArchLinux bzw. Manjaro:

```
sudo pacman -S mercurial
```

Windows Der geneigte Windows-User hat zwei Möglichkeiten. Entweder man lädt sich das Programm von Mercurial herunter oder (die bevorzugte Variante) man installiert sich das Programm TortoiseHg (siehe folgenden Abschnitt), das für Windows auch Mercurial beinhaltet!

Ausprobieren: `hg version`

Diff- und Merge-Tool

meld Ein graphisches Tool, das sowohl für Windows als auch für Unix-Varianten zur Verfügung steht. Für Windows kann man die aktuelle Version direkt von Meld herunterladen, während Linux-User wieder das Paketverwaltungstool ihrer Wahl bemühen können. Hier wieder das Kommando für ArchLinux bzw. Manjaro:

```
sudo pacman -S meld
```

TortoiseHg Es handelt sich um eine visuelle Schnittstelle zu Mercurial, das mit Unterstützung externer Tools (in der Windows-Version integriert) auch zum Mergen verwendet werden kann.
Sowohl für Windows als auch für Linux kann es direkt von TortoiseHg heruntergeladen werden.

Initiale Konfiguration

- Anlegen und Befüllen der Datei `~/.hgrc`
  - oder unter Windows: `mercurial.ini`!
  - → `hg config --edit`
    - erzeugt eine initiale `.hgrc` Datei bzw.
    - öffnet bestehende `.hgrc` Datei mit Editor
- Zugehörige Referenzdoku:
  - `man hg` … allgemeine Dokumentation für hg
    - enthält auch einen Abschnit zu `config`
  - `man hgrc` … Dokumentation zu den Konfigurationsdateien
  - `hg help` … zeigt Hilfe zu hg an
    - `hg help config` … zeigt speziell Hilfe zu `config` an
    - `hg help glossary` … zeigt Hilfe zu Begriffen

Konfigurieren: ~/.hgrc – 2

```
[ui]
username = FNAME LNAME <...@student.htlwrn.ac.at>
# the next configuration specifies your
# preferred editor. You can omit this if you are
# happy with the default editor of your system.
editor = emacsclient
# the next line specifies the merge tool used
# ...we will install it a bit later
merge = meld
# rollback is deprecated -> deactivate it!
rollback = false
# verbose output, e.g. with ”hg log”
verbose = true
color = yes
```

Konfigurieren: ~/.hgrc – 3

```
[extensions]
# command to delete untracked files from workdir
#   RTFM: hg help purge
purge =
# display progress bar when appropriate
progress =
# make it possible to use exernal diff programs
extdiff =
# configuration section for extdiff
[extdiff]
# new command to use meld instead of diff
cmd.meld = meld
[pager]
# config for linux pager
pager = less -FRXd
```

Konfigurieren: `.hgignore`

- je Repository, d.h. im Wurzelverzeichnis des Projektes
- soll ebenfalls versioniert werden
- → `man hgignore` oder `hg help ignore`
- Syntax:

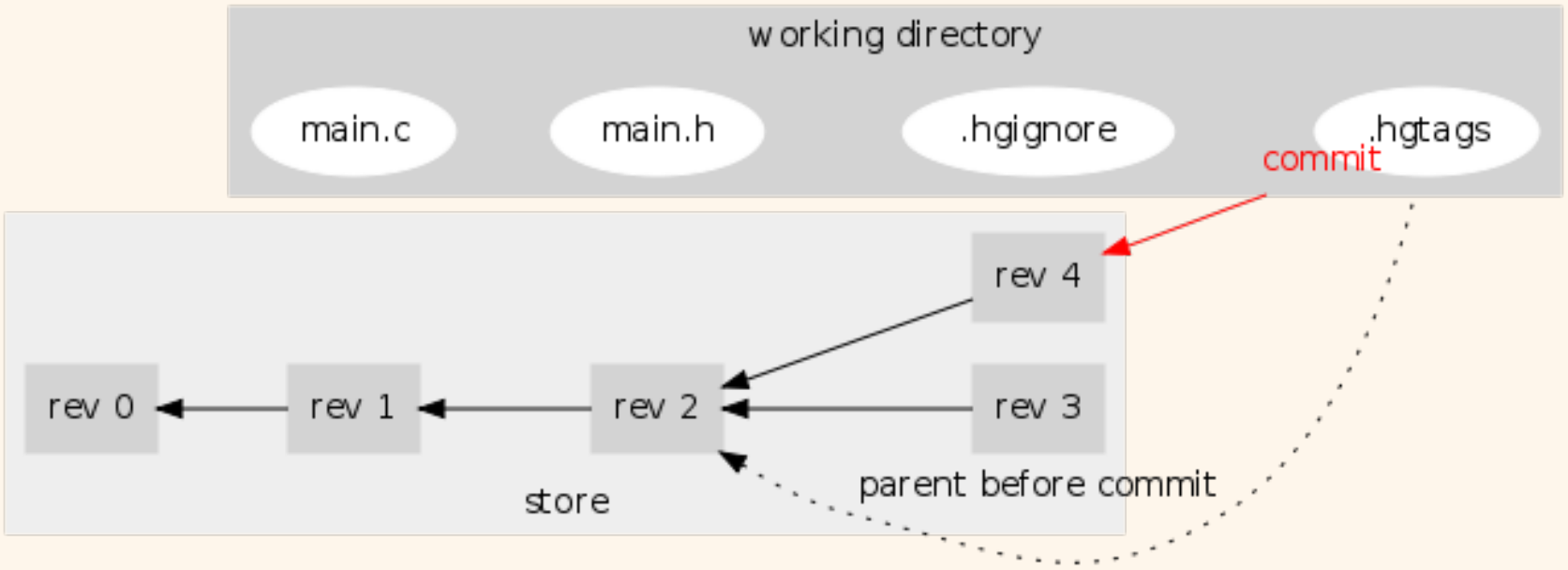
```
syntax: glob

CMakeCache.txt
*.exe
build
_minted*
generated/**.cpp
```

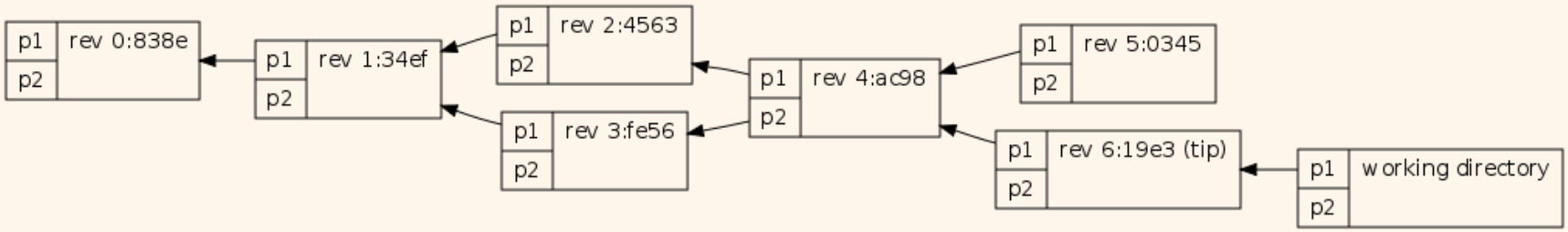
Begriffe und Konzepte

- Repository (kurz: repo)
  - working directory (kurz: workdir)
    - Dateien zu einem gewissen Zeitpunkt
  - store (auch: repo): komplette Historie (Verzeichnis `.hg`)
- changeset
  - fasst Änderungen *atomar* zusammen $\rightarrow$ Revision
  - bildet gerichteten azyklischen Graphen (DAG)
    - Mehrere Wurzeln (root) möglich
    - Jedes changeset hat 0 ($\rightarrow$ root), 1 oder 2 ”parents”
    - Zweige (branch)
    - Zweig ohne ”child changesets” $\rightarrow$ head
    - Head mit höchster revision nummer $\rightarrow$ tip
- Revision
  - revision number: sequenzielle Nummer (lokal eindeutig)
  - changeset ID: Hashwert (global eindeutig)

Begriffe und Konzepte – 2



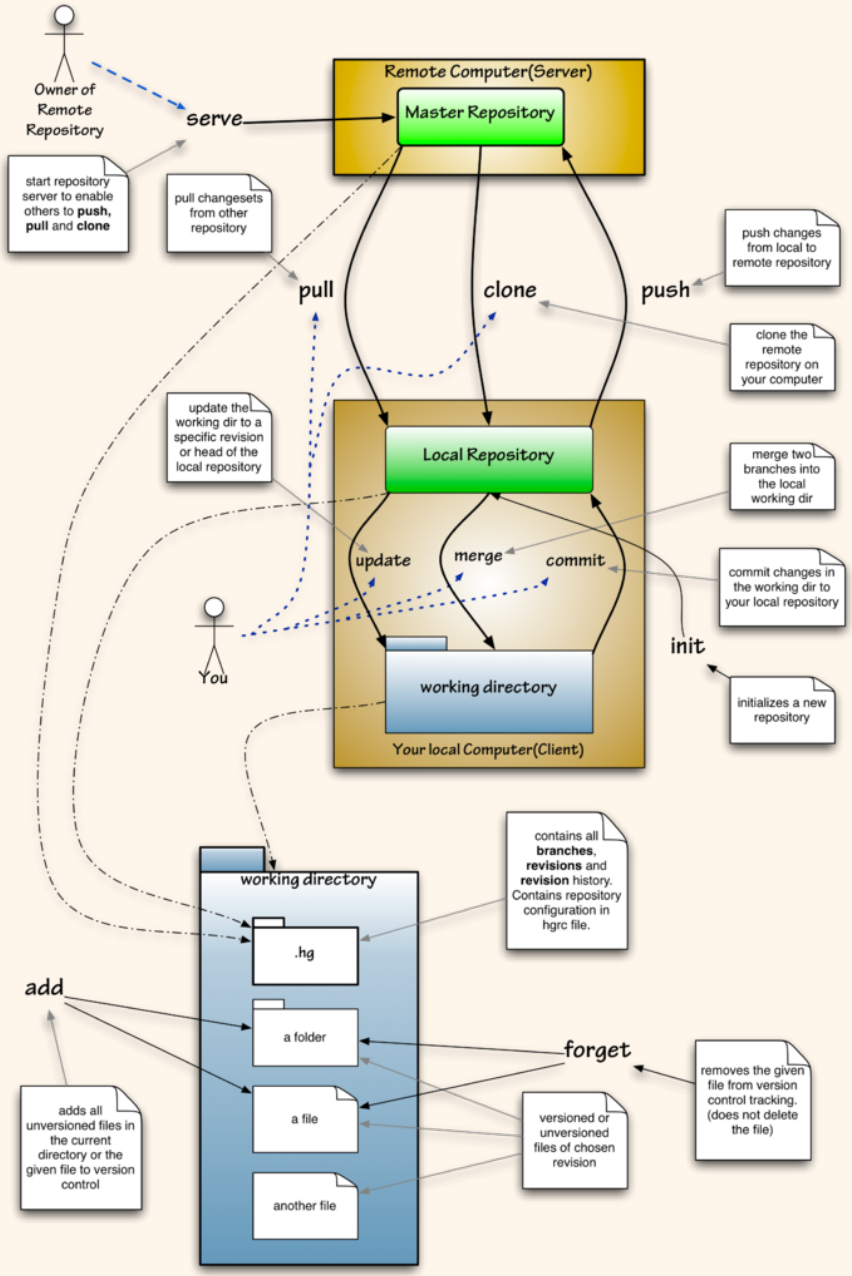
Begriffe und Konzepte – 3



example repository

- parents
  - von Revision 4: Revisionen 2 und 3
    - `hg parents -r 4`
  - vom working dir: Revision 6
    - `hg parents`
- heads: Revisionen 5 und 6
  - `hg heads`
- tip: Revision 6
  - `hg tip`
- merge changeset: Revision 4

Mercurial Kommandos im Überblick



Projekt und Repository anlegen

```
$ hg help
...
$ hg init tudu
$ cd tudu
$ ls -a
.  ..  .hg
$ touch tudu.py
$ hg status
? tudu.py
$ hg help status
...
```

hg status

- `M` Datei verändert (modified)
- `A` Datei hinzugefügt (added)
- `R` Datei entfernt (removed)
- `C` Dateien, die ”clean” sind, ignoriert werden oder die Source einer copy oder move Operation sind, werden nicht angezeigt, außer wenn Option `-c`, `-i`, `-C` oder `-A` angegeben.
- `!` Datei nicht vorhanden (missing)
- `?` Datei ”not tracked”
- `I` ignored

Dateien dem Repo bekanntmachen

```
$ hg add tudu.py
$ hg st
A tudu.py
# kein ”add” von Verzeichnissen, aber...
$ mkdir src
$ touch src/.hidden
$ hg add
adding src/.hidden
$ hg st
A src/.hidden
A tudu.py
$ hg forget tudu.py  # -> hg help forget
$ hg st
A src/.hidden
? tudu.py
$ hg add
A tudu.py
```

Dateien dem Repo bekanntm. – 2

```
$ touch ups.txt
$ hg add
adding ups.txt
$ hg forget ups.txt
$ hg st
A src/.hidden
A tudu.py
? ups.txt
$ hg addremove
adding ups.txt
$ rm ups.txt
$ hg addremove  # auch wenn schon committed!
removing ups.txt
$ ls ups.txt
ls: cannot access 'ups.txt': No such file or direct
```

Dateien dem Repo hinzufügen

```
$ hg commit -m "Add tudu.py and src/.hidden"
$ hg st
$ hg log
changeset:   0:4c681c55fb91
tag:         tip
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Thu Jun 29 12:19:27 2017 +0200
summary:     Add tudu.py and src/.hidden
```

- lokale eindeutige Revisionnummer!
- global eindeutige Changeset-ID
- Revision 0 hat außerdem eine Marke (tag): "tip"

Log mit Dateien und *gesamter* Beschreibung: `hg log -v`

Änderung vornehmen

```
$ emacs tudu.py # Lieblingseditor verwenden
# und eine Zeile mit ”aaa\n” hinzufuegen
Waiting for Emacs...
$ hg st
M tudu.py
? tudu.py~
$ hg diff tudu.py
diff -r 4c681c55fb91 tudu.py
--- a/tudu.py   Mon Jul 03 20:02:18 2017 +0200
+++ b/tudu.py   Mon Jul 03 20:03:50 2017 +0200
@@ -0,0 +1,1 @@
+aaa
$ # zeigt Aenderungen im unified diff-Format an!
$ hg ci -m ”Append aaa to tudu.py”
```

`diff` und das ”unified diff” Format

- `diff` zeigt Unterschiede zwischen 2 Dateien an
- `hg diff` ... defaultmäßig:
  - Unterschied workdir zur letzten Version im repo
  - im ”unified diff” (im git Format: Option `-g`)
- unified diff
  - `---` erste Datei
  - `+++` zweite Datei
  - `@@ -r,s +x,y @@` ... startet Block von Unterschieden
    - `-r,s` erste Datei: Zeilennummer `,` Anzahl der Zeilen
    - `+x,y` zweite Datei: Zeilennummer `,` Anzahl der Zeilen
  - Einzelne Zeile eines Blockes
    - keine Markierung $\rightarrow$ kein Unterschied
    - `-` als Markierung: Zeile gelöscht
    - `+` als Markierung: Zeile hinzugefügt

Änderung vornehmen – 2

```
$ hg st
? tudu.py~
$ # Backup-Datei von Emacs
$ #   -> soll nicht versionsverwaltet werden!
$ cat > .hgignore << EOF
heredoc> syntax: glob
heredoc> *~
heredoc> EOF
$ hg st
? .hgignore
$ hg add  # alle Dateien (inkl. Unterverz.!)
adding .hgignore
$ hg ci -m "Add '*~' to .hgignore"
$ hg st
$
```

Änderungen zurücknehmen

```
$ echo bbb >> tudu.py
$ cat tudu.py
aaa
bbb
$ # working dir zurueck auf letztes commit:
$ hg revert tudu.py
$ # d.h. wenn kein commit!
$ # hg revert --all fuer alle Dateien
& #   (auch geloeschte)
$ hg st
? tudu.py.orig
$ cat tudu.py
aaa
$ cat tudu.py.orig
aaa
bbb
$ rm tudu.py.orig
```

Änderung, die rückgängig macht

- Neues changeset, das Änderungen bis zur angegebenen Revision rückgängig macht!
- z.B. letztes Commit (aka tip):

```
$ echo ccc >> tudu.py
$ hg ci -m "Append ccc to tudu.py"
$ hg log
...
$ hg backout tip
  ... konfigurierter Editor öffnet sich!
  ...   Backed out changeset 1091cf71c2af
reverting tudu.py
Waiting for Emacs...
changeset 4:40506d2a6da0 backs out changeset 3:7
```

Änderung, die rückgängig macht – 2

```
$ cat tudu.py
aaa
$ hg log
changeset:   4:40506d2a6da0
tag:         tip
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Sun Jun 30 15:22:19 2019 +0200
summary:     Backed out changeset 751b7cb500db
...
```

Ändern des letzten Commits

```
$ echo ddd >> tudu.py
$ hg ci -m "Append ddd to tudu.py"
$ hg log
changeset:   5:24a9f2f59f37
tag:         tip
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Sun Jun 30 15:24:59 2019 +0200
summary:     Append ddd to tudu.py
...
$ # ups! sollte eee sein (anstatt ddd)
```

Ändern des letzten Commits – 2

```
$ emacs tudu.py # ddd durch eee ersetzen
$ cat tudu.py
aaa
eee
$ # jetzt Commit berichtigen:
$ hg ci -m "Append eee to tudu.py" --amend
saved backup bundle to...
$ hg log
changeset:   5:4fb18f5b99cf
tag:         tip
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Sun Jun 30 15:24:59 2019 +0200
summary:     Append eee to tudu.py
...
```

Commits

- **Ein Commit sollte eine logische Einheit sein!**
  - etwas für 1 Patch oder 1 Revert als Ganzes
  - wenn etwas aus Einzelteile, dann mehrere Commits **!!!!!**
- Commit - Message

```
Schreibe klar für *was* Commit ist

Beschreibe das *Problem*, das dieser
Commit loest oder
den *Use-Case* für dieses Feature.
Begründe *warum* du diese Lösung
gewählt hast.
```

Commits – 2

- Regeln
  - in aktiver Schreibweise (Gegenwart)
    - `füge hinzu` anstatt `hinzugefügt`
    - `Fix: #4711 (IE8 not supported)`
  - Erste Zeile kurz und bündig (ca. 50 Zeichen, max. 72)
    - beginnt mit Großbuchstabe und ohne Punkt am Ende.
    - Das ist die ”summary” in `hg log`!
  - Danach ein oder mehrere Absätze (max. 72 Zeichen/Zeile).
  - Wenn schwer zu beschreiben, dann mehrere Commits!?

Commits – 3

”The difference between a tolerable programmer and a great programmer is not how many programming languages they know, and it’s not whether they prefer Python or Java. It’s whether they can communicate their ideas… By writing clear comments and technical specs, they let other programmers understand their code, which means other programmers can use and work with their code instead of rewriting it. Absent this, their code is worthless. By writing clear technical documentation for end users, they allow people to figure out what their code is supposed to do, which is the only way those users can see the value in their code.” – Joel Spolsky (`http://www.joelonsoftware.com/`)

Überblick: update/commit/pull/push

- `hg update` … aktualisiert Arbeitsverzeichnis aus lokalem store
  - `hg merge` … führt andere Revision in das working directory zusammen
- `hg commit` … Änderungen des Arbeitsverzeichnisses in den lokalen store
- `hg clone` … kopiert anderes Repository in lokales Verzeichnis
- `hg pull` … kopiert Changesets von anderem Repository in lokalen store
- `hg push` … kopiert Changesets aus lokalem store in anderes Repository

Clonen und pullen

```
$ cd ..; hg clone tudu tudu2
updating to branch default
3 files updated, 0 files merged, 0 files removed, 0
$ cat tudu2/tudu.py
aaa
eee
$ cd tudu
$ echo fff >> tudu.py
$ hg ci -m "Append fff to tudu.py"
```

- klonen auch vom Internet, z.B.:

```
hg clone http://www.selenic.com/repo/hello myhello
```

Clonen und pullen – 2

```
$ cd ../tudu2;
$ hg incoming
comparing with .../tudu
searching for changes
changeset:   6:dcef16c80491
tag:         tip
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Mon Jul 03 20:15:45 2017 +0200
summary:     Append fff to tudu.py
$ hg pull
pulling from .../tudu
...
added 1 changesets with 1 changes to 1 files
new changesets dcef16c80491
(run 'hg update' to get a working copy)
```

Clonen und pullen – 3

```
$ cat tudu.py
aaa
eee
$ hg up
1 files updated, 0 files merged, 0 files removed, 0
$ cat tudu.py
aaa
eee
fff
$ # pullen und updaten in einem Schritt:
$ #  hg pull -u
```

Pullen und mergen

```
$ cd ../tudu
$ # Zwischen aaa und eee: bbb hinzufuegen
$ emacs tudu.py
$ hg ci -m "Insert row bbb"
$ cd ../tudu2
$ # Zwischen aaa und eee: ccc hinzufuegen
$ emacs tudu.py
$ hg ci -m "Insert row ccc"
$ hg pull
pulling from /home/.../tudu
searching for changes
...
added 1 changesets with 1 changes to 1 files (+1 he
new changesets 91af51296a56
(run 'hg heads' to see heads, 'hg merge' to merge)
```

Pullen und mergen – 2

```
$ hg heads
changeset:   8:91af51296a56
tag:         tip
parent:      6:f6e9f83f8455
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Mon Jul 03 20:18:14 2017 +0200
summary:     Insert row bbb

changeset:   7:fed54dd16eea
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Mon Jul 03 20:19:04 2017 +0200
summary:     Insert row ccc
```

Pullen und mergen – 3

```
$ hg merge
... meld oeffnet sich: Reihenfolge bbb gefolgt von
... ccc zwischen aaa und eee herstellen
merging tudu.py
running merge tool meld for file tudu.py
0 files updated, 1 files merged, 0 files removed, 0
(branch merge, don't forget to commit)
$ hg ci -m "Merge with remote"
```

Pullen und mergen – 4

```
$ hg log -G
@    changeset:   9:39fedf2a34ac
|\   tag:         tip
| |  parent:      7:fed54dd16eea
| |  parent:      8:91af51296a56
| |  user:        Guenter Kolousek <ko@htlwrn.ac.at>
| |  date:        Mon Jul 03 20:20:24 2017 +0200
| |  summary:     Merge with remote
| |
| o  changeset:   8:91af51296a56
| |  parent:      6:dcef16c80491
| |  user:        Guenter Kolousek <ko@htlwrn.ac.at>
| |  date:        Mon Jul 03 20:18:14 2017 +0200
| |  summary:     Insert row bbb
| |
o |  changeset:   7:fed54dd16eea
...
```

Pushen und updaten

```
$ # aber bis jetzt nur in tudu2, nicht in tudu!
$ hg outgoing
comparing with .../tudu
searching for changes
changeset:   7:fed54dd16eea
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Mon Jul 03 20:19:04 2017 +0200
summary:     Insert row ccc

changeset:   9:39fedf2a34ac
tag:         tip
parent:      7:fed54dd16eea
parent:      8:91af51296a56
user:        Guenter Kolousek <ko@htlwrn.ac.at>
date:        Mon Jul 03 20:20:24 2017 +0200
summary:     Merge with remote
```

Pushen und updaten – 2

```
$ hg push
pushing to .../tudu
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 1 files
```

Pushen und updaten – 3

```
$ cd ../tudu
$ # aber noch nicht in workdir!
$ cat tudu.py
aaa
bbb
eee
fff
$ # hg update oder kurz:
$ hg up
1 files updated, 0 files merged, 0 files removed, 0
$ cat tudu.py
aaa
bbb
ccc
eee
fff
```

Löschen aus dem Repository

Datei wird nicht mehr gebraucht…

```
$ touch to_be_removed.txt
$ hg add
adding to_be_removed.txt
$ hg commit -m ”Add to_be_removed.txt”
$ hg remove to_be_removed.txt
$ hg st
R to_be_removed
$ ls
```

Löschen aus dem Repository – 2

```
$ hg commit -m ”Remove to_be_removed.txt”
$ hg log
changeset:   ...
tag:         tip
...
summary:     Remove to_be_removed.txt

changeset:   ...
...
summary:     Add to_be_removed.txt
```

Löschen aus dem workdir

d.h. zurücksetzen des workdir auf repo!

```
$ touch do_not_remove.txt
$ hg add
adding do_not_remove.txt
$ hg st
A do_not_remove.txt
$ hg commit -m ”Add do_not_remove.txt”
$ hg st
$ rm do_not_remove.txt
$ hg st
! do_not_remove.txt
$ hg revert do_not_remove.txt
$ hg st
$ ls do_not_remove.txt
do_not_remove.txt
```

Kopieren einer Datei

```
$ touch tree.py
$ hg add
adding tree.py
$ hg ci -m ”Add tree.py”
$ hg copy tree.py avl.py
$ hg st
A avl.py
$ hg ci -m ”Add avl.py (copy of tree.py)”
$ cp avl.py redblack_tree.py
$ hg st
? redblack_tree.py
$ hg cp avl.py redblack_tree.py --after
$ hg st
A redblack_tree.py
$ hg ci -m ”Add redblack_tree.py (copy of avl.py)”
```

Verschieben bzw. Umbenennen

```
$ hg mv avl.py avl_tree.py
$ hg st
A avl_tree.py
R avl.py
$ # ...bewirkt ein Kopieren und Löschen
$ # --after auch bei ”hg mv” und ”hg rm”!
$ # ”hg revert” macht rückgängig (auch bei ”hg cp”)
$ hg diff  # keine Ausgabe im unified diff!
$ hg diff -g  # git-Variante von diff (mit Header):
diff --git a/avl.py b/avl_tree.py
rename from avl.py
rename to avl_tree.py
$ hg ci -m ”Rename avl.py to avl_tree.py”
$ hg diff -g
$
```

Arbeiten mit Versionen

```
$ hg tag v1.0
$ hg tip
changeset:   ...
tag:         tip
...
summary:     Added tag v1.0 for changeset 274f5c3b0
$ hg tags
tip                                17:d14d89c3a273
v1.0                               16:274f5c3b0461
$ touch btree.py
$ hg add
adding btree.py
$ hg ci -m "Add btree.py"
$ ls btree.py
btree.py
```

Arbeiten mit Versionen – 2

```
$ hg up -r v1.0
0 files updated, 0 files merged, 2 files removed, 0
$ # Tags in Datei .hgtags!
$ ls
avl_tree.py  do_not_remove.txt  redblack_tree.py  s
$ echo xxx > bst.cpp
$ hg add
adding bst.cpp
$ hg ci -m "Add feature xxx"
created new head
```

Arbeiten mit Versionen – 3

```
$ hg log -G
@  changeset:   19:4147f503740e
|  tag:         tip
|  parent:      16:274f5c3b0461
|  user:        Guenter Kolousek <ko@htlwrn.ac.at>
|  date:        Mon Jul 03 22:53:27 2017 +0200
|  summary:     Add feature xxx
|
| o  changeset:   18:5c50e050b3a4
| |  user:        Guenter Kolousek <ko@htlwrn.ac.at>
| |  date:        Mon Jul 03 22:41:43 2017 +0200
| |  summary:     Add btree.py
| |
| o  changeset:   17:d14d89c3a273
|/   user:        Guenter Kolousek <ko@htlwrn.ac.at>
|    date:        Mon Jul 03 22:40:45 2017 +0200
|    summary:     Added tag v1.0 for changeset 1e447d690a08
|
o  changeset:   16:274f5c3b0461
|  tag:         v1.0
```

Arbeiten mit Versionen – 4

```
$ hg merge
2 files updated, 0 files merged, 0 files removed, 0 files unres
(branch merge, don't forget to commit)
$ hg ci -m ”Merge with tip”
$ hg log -G
@    changeset:   20:85a9e1487a62
|\   tag:         tip
| |  parent:      19:4147f503740e
| |  parent:      18:5c50e050b3a4
| |  user:        Guenter Kolousek <ko@htlwrn.ac.at>
| |  date:        Mon Jul 03 22:57:14 2017 +0200
| |  summary:     Merge with tip
| |
| o  changeset:   19:4147f503740e
| |  parent:      16:274f5c3b0461
| |  user:        Guenter Kolousek <ko@htlwrn.ac.at>
| |  date:        Mon Jul 03 22:53:27 2017 +0200
| |  summary:     Add feature xxx
| |
o |  changeset:   18:5c50e050b3a4
```

Anzeigen von Informationen

- `hg identify` (kurz: `hg id`)
  - zeigt die Version des working dir an
- `hg summary` (kurz: `hg sum`)
  - zeigt aktuellen Status des working dir an
- `hg annotate <file>` (”kurz”: `hg blame <file>`)
  - zeigt zeilenweise die Änderungen von `<file>`

Servermöglichkeiten

- Filehosting-Dienste
  - Bitbucket (git, hg)
  - GitHub (git)
  - GitLab (git)
  - RhodeCode (git, hg, Subversion): kommerziell
  - HgLab (hg, nur Windows): kommerziell
- Eigene(r) Server (self-hosted)
  - SCM-Manager (git, hg, Subversion)
  - Kallithea (git, hg)
  - RhodeCode (git, hg, Subversion)
  - GitLab (git)
  - Phabricator (git, hg, Subversion)
  - HgLab (hg, nur Windows)
  - hg serve,… (hg)