Versionskontrolle mit Apache Subversion

Dr.-Ing. Mathias Magdowski

Lehrstuhl für Elektromagnetische Verträglichkeit
Institut für Medizintechnik
Otto-von-Guericke-Universität, Magdeburg

3. Juni 2015

Gliederung

Nachteile ohne Versionskontrolle

Vorteile mit Versionskontrolle

Konzepte

Software

Demo

Zusammenfassung

Nachteile ohne Versionskontrolle

Szenarios ohne Versionskontrolle

Schreiben einer Abschlussarbeit:

- über 6 Monate
- 60 bis 80 Seiten
- Entwurf $\rightarrow$ Ausarbeitung $\rightarrow$ Abgabe $\rightarrow$ Begutachtung
- unzählige Änderungen, Korrekturen, Überarbeitungen

Szenarios ohne Versionskontrolle

Schreiben einer Abschlussarbeit:

- über 6 Monate
- 60 bis 80 Seiten
- Entwurf $\rightarrow$ Ausarbeitung $\rightarrow$ Abgabe $\rightarrow$ Begutachtung
- unzählige Änderungen, Korrekturen, Überarbeitungen

Probleme:

- Welche ist die letzte Version?

Szenarios ohne Versionskontrolle

Schreiben einer Abschlussarbeit:

- über 6 Monate
- 60 bis 80 Seiten
- Entwurf $\rightarrow$ Ausarbeitung $\rightarrow$ Abgabe $\rightarrow$ Begutachtung
- unzählige Änderungen, Korrekturen, Überarbeitungen

Probleme:

- Welche ist die letzte Version?
- Was habe ich im März 2015 geändert?

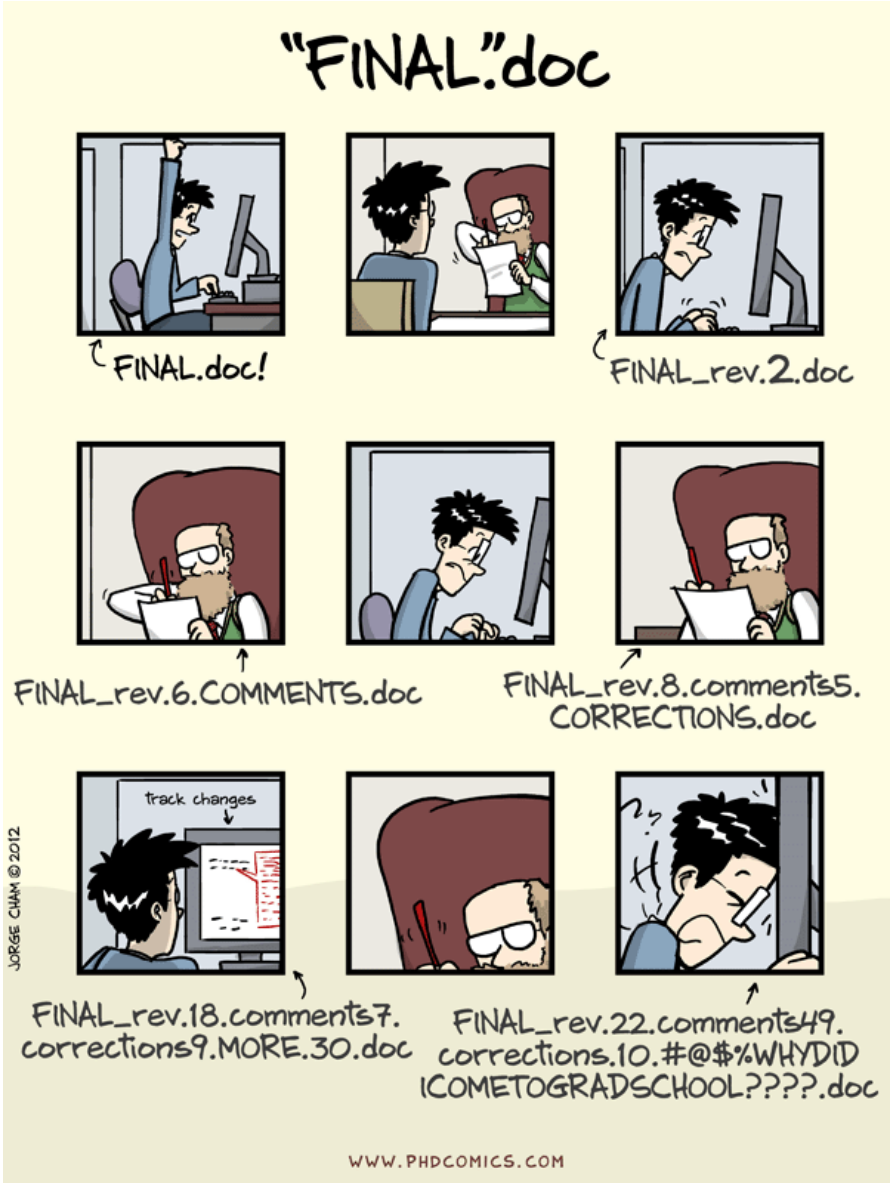
Szenarios ohne Versionskontrolle

Schreiben einer Abschlussarbeit:

- über 6 Monate
- 60 bis 80 Seiten
- Entwurf $\rightarrow$ Ausarbeitung $\rightarrow$ Abgabe $\rightarrow$ Begutachtung
- unzählige Änderungen, Korrekturen, Überarbeitungen

Probleme:

- Welche ist die letzte Version?
- Was habe ich im März 2015 geändert?
- Wann habe ich den Abschnitt auf Seite 3 überarbeitet?


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Szenarios ohne Versionskontrolle

Umsetzung eines Softwareprojekts:

- über mehrere Jahre
- tausende Zeilen Quelltext, viele abhängige Funktionen
- mehrere Entwickler
- unzählige Programmerweiterungen und Fehlerbehebungen

Szenarios ohne Versionskontrolle

Umsetzung eines Softwareprojekts:

- über mehrere Jahre
- tausende Zeilen Quelltext, viele abhängige Funktionen
- mehrere Entwickler
- unzählige Programmerweiterungen und Fehlerbehebungen

Probleme:

- Welche ist die letzte Version? Wer hat sie?

Szenarios ohne Versionskontrolle

Umsetzung eines Softwareprojekts:

- über mehrere Jahre
- tausende Zeilen Quelltext, viele abhängige Funktionen
- mehrere Entwickler
- unzählige Programmerweiterungen und Fehlerbehebungen

Probleme:

- Welche ist die letzte Version? Wer hat sie?
- Wer hat die Funktion x geschrieben?

Szenarios ohne Versionskontrolle

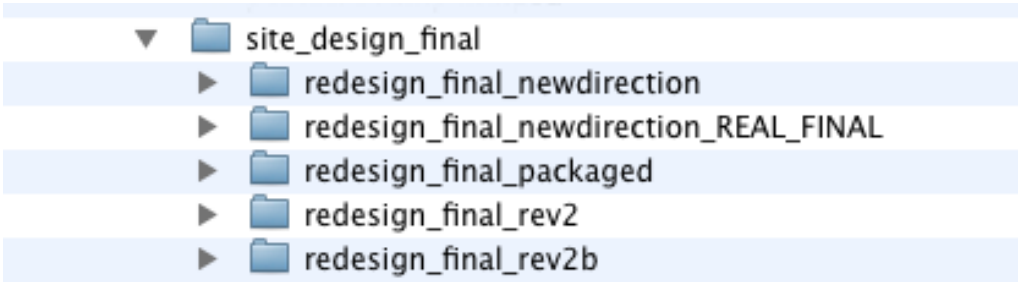
Umsetzung eines Softwareprojekts:

- über mehrere Jahre
- tausende Zeilen Quelltext, viele abhängige Funktionen
- mehrere Entwickler
- unzählige Programmerweiterungen und Fehlerbehebungen

Probleme:

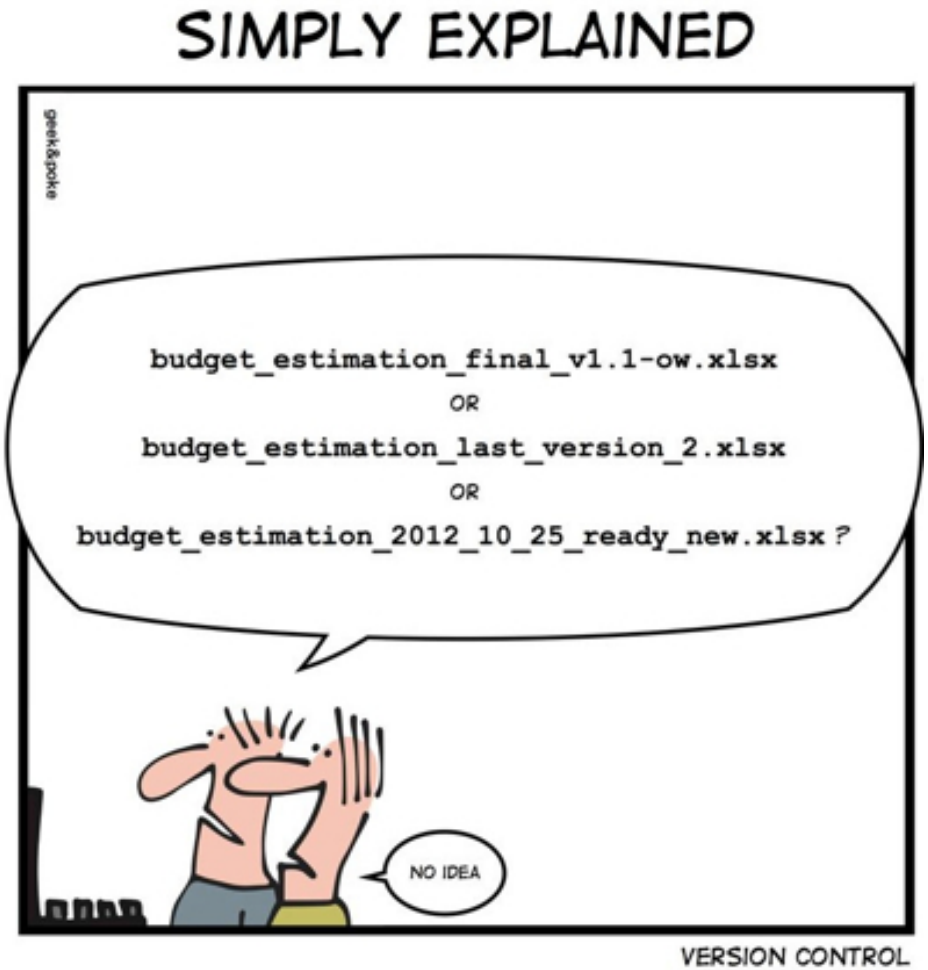
- Welche ist die letzte Version? Wer hat sie?
- Wer hat die Funktion x geschrieben?
- Wann wurde das Format für y in Programm z geändert?

Welche ist denn nun die letzte Version?



http://www.webdesignerdepot.com/2009/03/intro-to-git-for-web-designers/

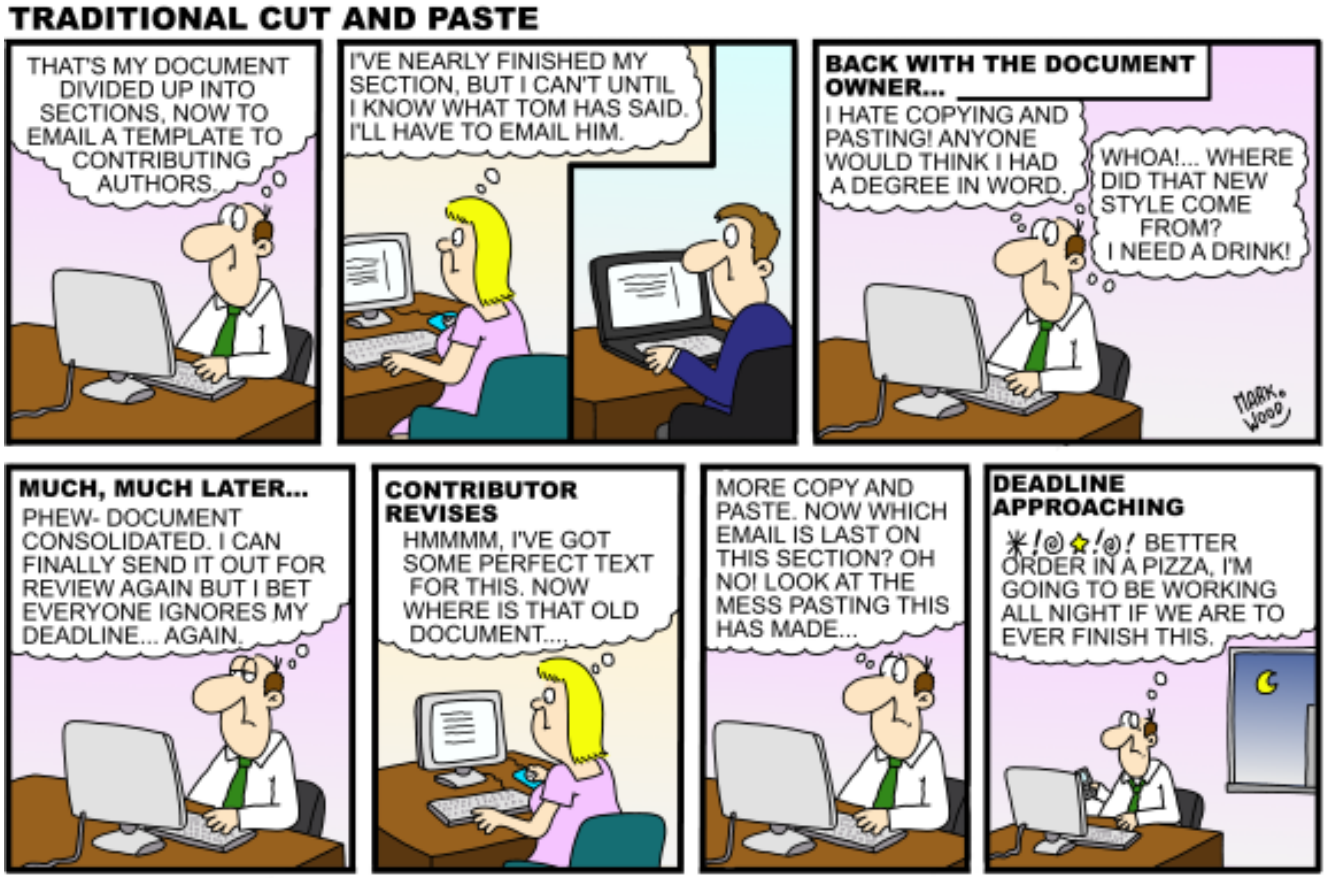
Welche genau jetzt?



Wie tauscht man die Quelltexte aus?

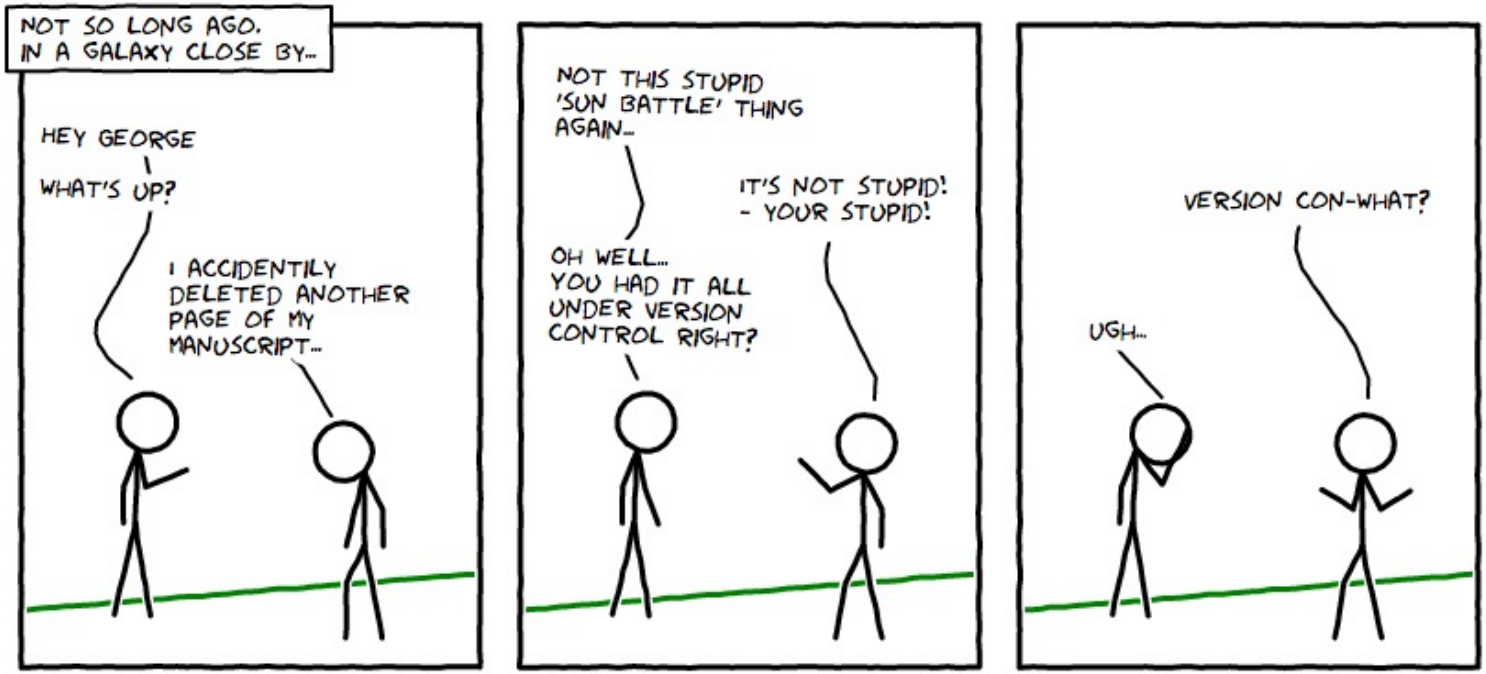
Wo speichert man die Quelltexte?

Wie fügt man am Ende alles zusammen?



http://www.pleasetech.com/cartoons.aspx

Etwas aus Versehen gelöscht?



http://www.sitepoint.com/8-essential-skills-developers-can-learn-in-a-weekend/

Vorteile mit Versionskontrolle

Versionskontrolle

Man kann:

- auf jede Version (von der ursprünglichen bis zur jetzigen) zugreifen
- sehen, wer, was, wann geändert hat
- zusammen an einer Datei bzw. einem Projekt arbeiten
- Änderungen oder Versionen kommentieren
- Änderungen rückgängig machen

Versionskontrolle

Man kann:

- auf jede Version (von der ursprünglichen bis zur jetzigen) zugreifen
- sehen, wer, was, wann geändert hat
- zusammen an einer Datei bzw. einem Projekt arbeiten
- Änderungen oder Versionen kommentieren
- Änderungen rückgängig machen

https://motherearthseries.files.wordpress.com/2013/04/undo_key.jpg

Studenten benutzen es nicht . . .

http://www.loc.gov/exhibits/treasures/images/tlc0090.jpg

. . . weil die Betreuer es nicht benutzen

Sinnvoller Einsatz für Abschlussarbeiten/Artikel

Vorteile:

- gegenseitiger Austausch ohne eMails
- automatische Sicherung aller Versionen
- Änderungen zwischen Versionen nachvollziehbar
- Einbindung von Vorlagen möglich

Sinnvoller Einsatz für Abschlussarbeiten/Artikel

Vorteile:

- gegenseitiger Austausch ohne eMails
- automatische Sicherung aller Versionen
- Änderungen zwischen Versionen nachvollziehbar
- Einbindung von Vorlagen möglich

Vorteile für den Betreuer:

- Fortschritt des Studenten ist nachvollziehbar
- Einfügen von Kommentaren möglich
- kein unnötiger Austausch von Emails, keine unnötigen Treffen

Versionskontrolle

Geeignet für:

- Quelltexte, Texte, Dokumente
- Rohdaten von Messungen oder Diagrammen

Versionskontrolle

Geeignet für:

- Quelltexte, Texte, Dokumente
- Rohdaten von Messungen oder Diagrammen

Bedingt geeignet für:

- Bilder und Grafiken

Versionskontrolle

Geeignet für:

- Quelltexte, Texte, Dokumente
- Rohdaten von Messungen oder Diagrammen

Bedingt geeignet für:

- Bilder und Grafiken

Nicht geeignet für:

- große Binärdateien
- Videos, Animationen

Konzepte

Begriffe

Repository:

- Speicherort für die Daten
- alle Versionen in Form einer Datenbank
- lokal oder auf einem Webserver

Begriffe

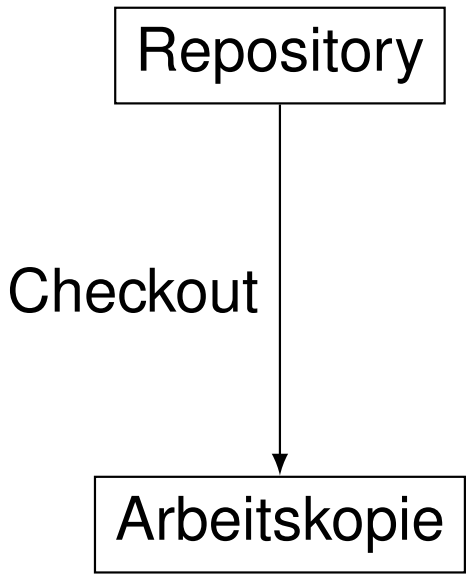
Repository:

- Speicherort für die Daten
- alle Versionen in Form einer Datenbank
- lokal oder auf einem Webserver

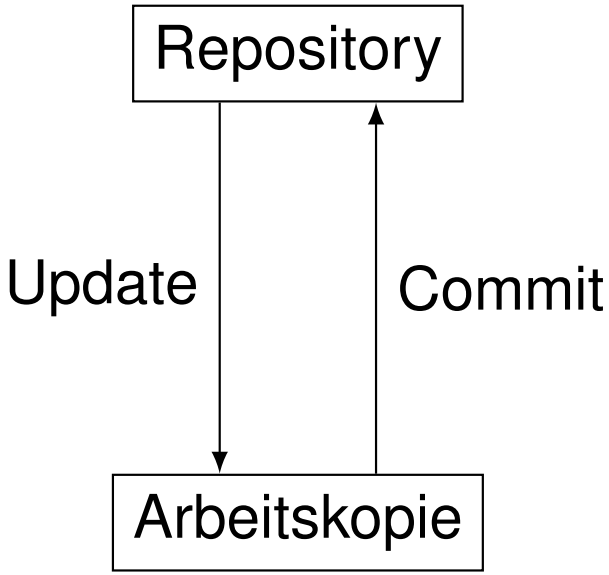
Arbeitskopie:

- lokaler Speicherort für die entsprechende Version
- nur aktueller (oder älterer) Stand für die Bearbeitung
- kann jederzeit aus dem Repository erzeugt werden

Erstellen einer lokalen Arbeitskopie



Datenaustausch



Arten

Lokale Versionsverwaltung:

- Repository ist lokal gespeichert
- oft wird nur eine Datei versioniert (z. B. Büroanwendungen)

Arten

Lokale Versionsverwaltung:

- Repository ist lokal gespeichert
- oft wird nur eine Datei versioniert (z. B. Büroanwendungen)

Zentrale Versionsverwaltung:

- Client-Server-System, Netzwerkzugriff aufs Repository
- Rechteverwaltung

Arten

Lokale Versionsverwaltung:

- Repository ist lokal gespeichert
- oft wird nur eine Datei versioniert (z. B. Büroanwendungen)

Zentrale Versionsverwaltung:

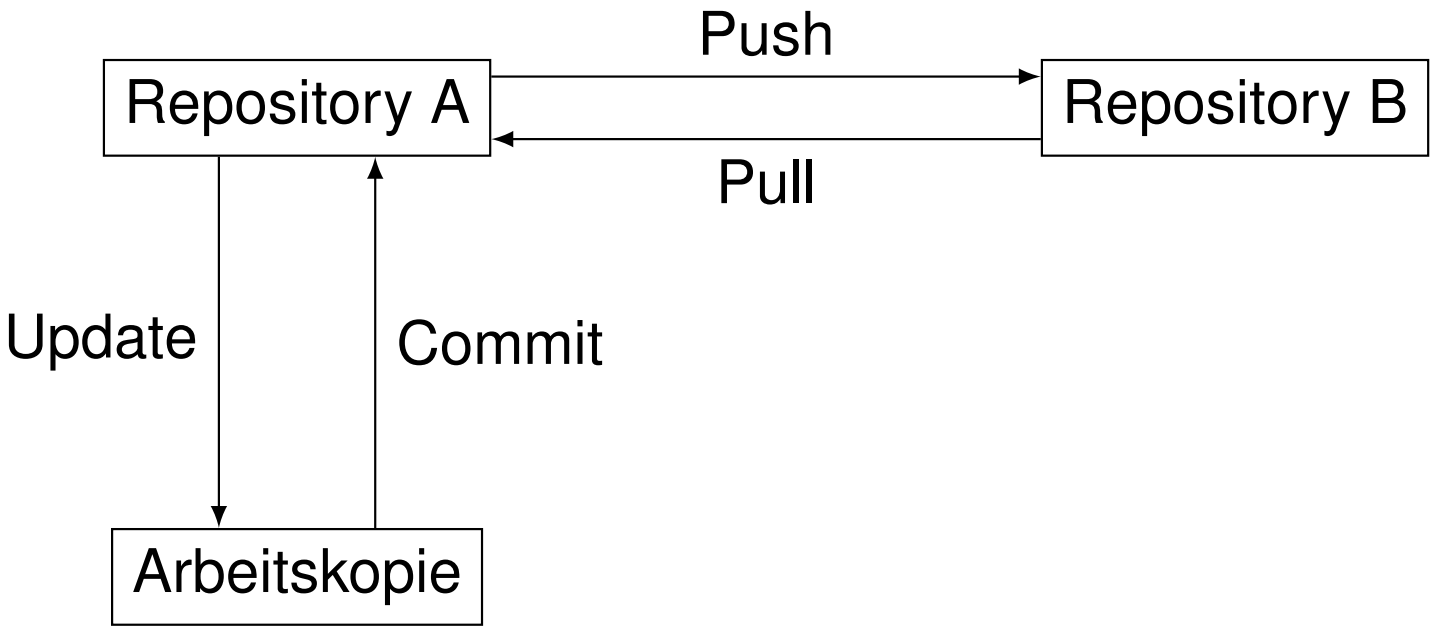
- Client-Server-System, Netzwerkzugriff aufs Repository
- Rechteverwaltung

Verteilte Versionsverwaltung:

- jeder hat sein eigenes Repository, gegenseitiger Abgleich

Erstellen eines lokalen Repositories

Datenaustausch in der verteilten Versionsverwaltung



Änderungskonzepte

Lock Modify Write:

- Sperrung einer Datei vor einer Änderung
- Freigabe nach Abschluss der Änderung
- kein Zusammenführen von Versionen nötig

Änderungskonzepte

Lock Modify Write:

- Sperrung einer Datei vor einer Änderung
- Freigabe nach Abschluss der Änderung
- kein Zusammenführen von Versionen nötig

Copy Modify Merge:

- gleichzeitige Änderungen durch mehrere Benutzer an einer Datei sind möglich
- Änderungen werden automatisch oder manuell zusammengeführt (Merge)

Software

Concurrent Versions System (CVS)

Eigenschaften:

- *das* klassische Versionskontrollsystem
- wird nicht mehr aktiv weiterentwickelt
- lokales Repository oder über Webserver
- ursprünglich ein reines Kommandozeilen-Programm

Concurrent Versions System (CVS)

Eigenschaften:

- *das* klassische Versionskontrollsystem
- wird nicht mehr aktiv weiterentwickelt
- lokales Repository oder über Webserver
- ursprünglich ein reines Kommandozeilen-Programm

GUI für Windows:

TortoiseCVS von `http://www.tortoisecvs.org`

Apache Subversion (SVN)

Eigenschaften:

- Quasi-Nachfolger von CVS
- zentrales Projektarchiv mit einfacher Revisionszählung
- es werden nur die Unterschiede übertragen
- erlaubt das Verschieben und Umbenennen von Dateien

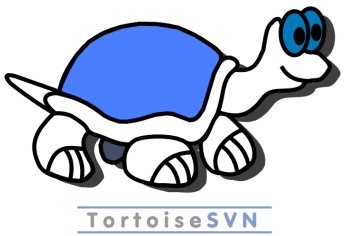
Apache Subversion (SVN)

Eigenschaften:

- Quasi-Nachfolger von CVS
- zentrales Projektarchiv mit einfacher Revisionszählung
- es werden nur die Unterschiede übertragen
- erlaubt das Verschieben und Umbenennen von Dateien

GUI für Windows:

TortoiseSVN von `http://tortoisesvn.net`



`http://blogs.wandisco.com/tag/tortoisesvn/`

Mercurial (hg)

Eigenschaften:

- verteiltes Versionskontrollsystem
- fast vollständig in Python entwickelt
- alle Kommandos beginnen mit `hg`
- effizient, skalierbar und robust

Mercurial (hg)

Eigenschaften:

- verteiltes Versionskontrollsystem
- fast vollständig in Python entwickelt
- alle Kommandos beginnen mit `hg`
- effizient, skalierbar und robust

GUI für Windows:

TortoiseHg von `http://tortoisehg.bitbucket.org`

TortoiseHg

Git

Eigenschaften:

- verteiltes Versionskontrollsystem
- für den Linux-Kernel entwickelt
- kryptographische Sicherheit
- webbasiertes Hosting im *GitHub*

Git

Eigenschaften:

- verteiltes Versionskontrollsystem
- für den Linux-Kernel entwickelt
- kryptographische Sicherheit
- webbasiertes Hosting im *GitHub*

GUI für Windows:

TortoiseGit von `https://tortoisegit.org/`

`http://www.kodteyner.com/2014/01/17/`

Namensgebung

Linus Torvalds auf `git.wiki.kernel.org`:

I'm an egotistical bastard, and I name all my projects after myself. First „Linux", now „Git".

Namensgebung

Linus Torvalds auf `git.wiki.kernel.org`:

> *I'm an egotistical bastard, and I name all my projects after myself. First „Linux", now „Git".*

Linus Torvalds in *Lord of the Files: How GitHub Tamed Free Software (And More)*:

> *The joke 'I name all my projects for myself, first Linux, then git' was just too good to pass up. But it is also short, easy-to-say, and type on a standard keyboard. And reasonably unique and not any standard command, which is unusual.*

Verwendung

CVS:

viele Open-Source-Projekte (früher)

SVN:

Free Pascal, FreeBSD, GCC, Mono

hg:

Facebook, Mozilla (Firefox, Thunderbird), SourceForge, Google Inc. (Google Chrome, Google Code), Atlassian (Bitbucket)

Git:

Android, Debian, Fedora, Git selbst, KDE, LibreOffice, Linux-Kernel, Perl, PHP, Ruby, Samba, VLC media player, Wine

Demo

Repository erstellen

1. leeren Ordner erstellen (z. B. auf Desktop)
2. umbenennen in „Repository“
3. Rechtsklick → „TortoiseSVN“ → „Create repository here“ → „Ok“
4. Ordner bekommt ein spezielles Icon und den entsprechenden Inhalt für die Datenbank

Arbeitskopie erstellen

1. leeren Ordner erstellen (z. B. auf Desktop)
2. umbenennen in „Arbeitskopie“
3. Rechtsklick → „SVN Checkout“
4. URL of repository:
   `file:///C:/Users/xxx/Desktop/Repository` oder den Ordner mittels „...“ auswählen
5. „Ok“
6. „Checkout Finished!“ mit „Ok“ bestätigen
7. Ordner bekommt ein spezielles, grünes Icon und einen Unterordner `.svn`

LaTeX-Quelltext hinzufügen

1. Datei „test.tex“ anlegen
2. Inhalt:

```
\documentclass{scrartcl}

\begin{document}

Hallo Welt!

\end{document}
```

3. Abspeichern und mit `pdflatex` kompilieren

Commit in das Repository

1. Rechtsklick auf den Ordner „Arbeitskopie“ → „SVN Commit“
2. bei „Message:“ einen sinnvollen Kommentar eingeben
3. nur die Datei `test.txt` anklicken
4. bei allen anderen Dateien: Rechtsklick → „Add to ignore list“ → *.xxx (mit xxx als Dateiendung)
5. „Ok“
6. „Commit Finished!“ mit „Ok“ bestätigen
7. Ordner sollte wieder ein grünes Icon haben

LaTeX-Quelltext ändern

1. Datei „test.tex“ öffnen
2. irgendwas am Inhalt ändern, z. B. eine Zeile hinzufügen

```
Das ist eine neue Zeile.
```

3. Abspeichern und mit `pdflatex` kompilieren
4. Ordner bekommt ein rotes Icon

Commit der Änderungen in das Repository

1. Rechtsklick auf den Ordner „Arbeitskopie“ → „SVN Commit“
2. bei „Message:“ einen sinnvollen Kommentar eingeben
3. Rechtsklick auf die Datei `text.tex` → „Compare with base“
4. Prüfen den Änderungen und Schließen des „TortoiseMerge“
5. „Ok“
6. „Commit Finished!“ mit „Ok“ bestätigen
7. Ordner sollte wieder ein grünes Icon bekommen

Zweite Arbeitskopie erstellen

1. leeren Ordner erstellen (z. B. auf Desktop)
2. umbenennen in „Arbeitskopie2“
3. Rechtsklick → „SVN Checkout“
4. URL of repository:
   `file:///C:/Users/xxx/Desktop/Repository` oder den Ordner mittels „...“ auswählen
5. „Ok“
6. „Checkout Finished!“ mit „Ok“ bestätigen
7. Ordner bekommt ein spezielles, grünes Icon und einen Unterordner `.svn`

LaTeX-Quelltext in den zweiten Arbeitskopie ändern

1. Datei „test.tex“ in der „Arbeitskopie2“ öffnen
2. irgendwas am Inhalt ändern, z. B. eine Zeile hinzufügen

   ```
   Das ist noch eine neue Zeile.
   ```

3. Abspeichern und mit `pdflatex` kompilieren
4. Ordner bekommt ein rotes Icon

Commit der Änderungen in das Repository

1. Rechtsklick auf den Ordner „Arbeitskopie2“ → „SVN Commit“
2. bei „Message:“ einen sinnvollen Kommentar eingeben
3. Rechtsklick auf die Datei `text.tex` → „Compare with base“
4. Prüfen den Änderungen und Schließen des „TortoiseMerge“
5. „Ok“
6. „Commit Finished!“ mit „Ok“ bestätigen
7. Ordner sollte wieder ein grünes Icon bekommen

Update der ersten Arbeitskopie

1. Rechtsklick auf den Ordner „Arbeitskopie“ → „SVN Update“
2. Nachverfolgung der Änderungen mit „Show log . . . “
3. Rechtsklick auf die Datei `text.tex` → „Show changes“ in jeder Revision
4. „TortoiseMerge“ und „TortoiseSVN“ schließen
5. „Commit Finished!“ mit „Ok“ bestätigen
6. Ordner sollte wieder ein grünes Icon haben

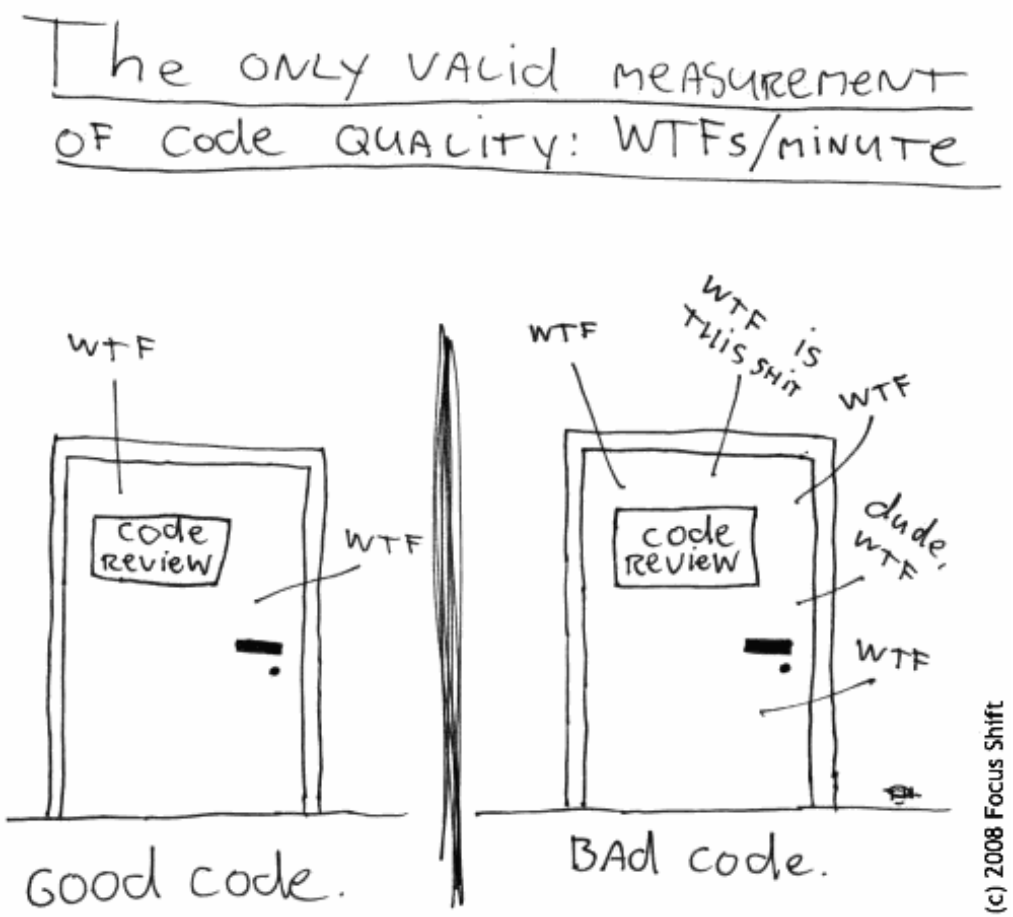
Weitere Dinge, die man ausprobieren sollte

- (falsche) Änderungen mit „Revert“ rückgängig machen
- Urheber von Änderungen mit „Blame“ sichtbar machen
- Änderungen und Commit in „Arbeitskopie“; Änderung, Update und Commit in „Arbeitskopie2“
  - Änderung unterschiedlicher Stellen $\rightarrow$ automatische Zusammenführung
  - Änderung der gleichen Stelle $\rightarrow$ Konflikt
- Umbenennen einer versionierten Datei mittels „SVN Rename“
- Verschieben einer versionierten Datei mittels „SVN Move“
- Wiederherstellen eine alter Version mittels „Update to revision“

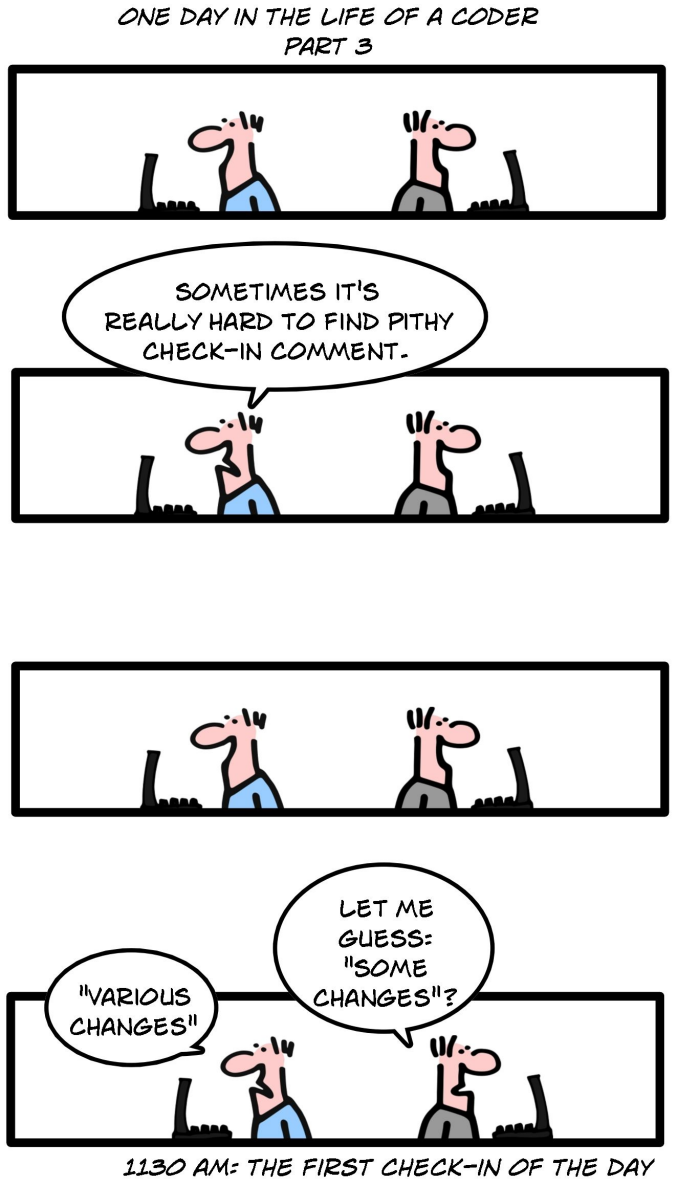
Zusammenfassung

Zusammenfassung

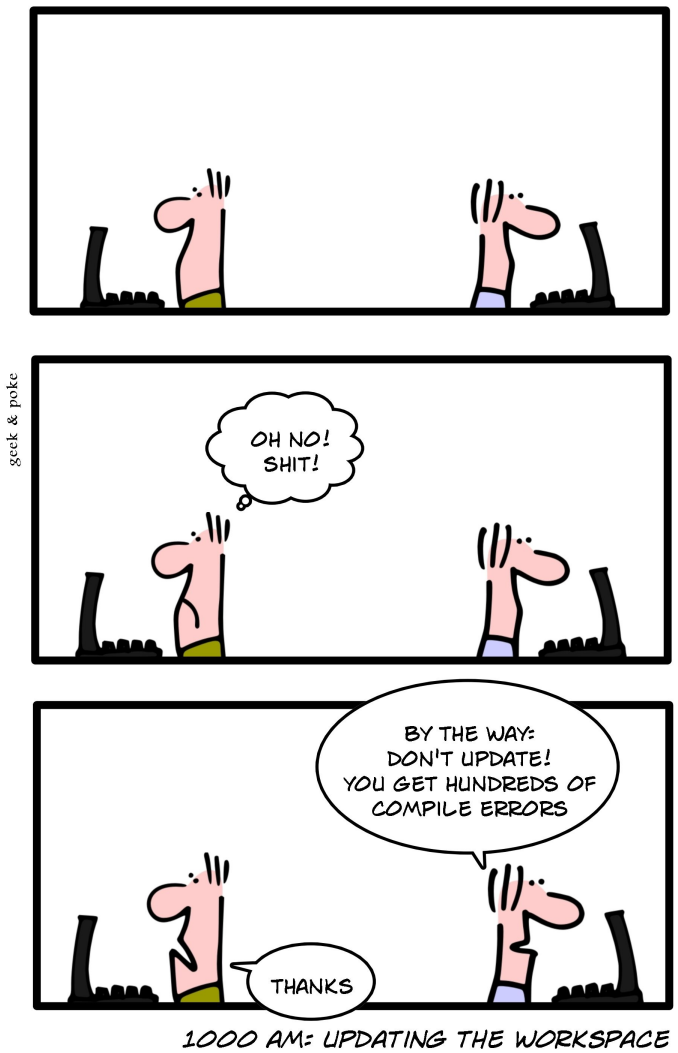
- Versionskontrolle ist ein extrem nützliches Werkzeug
- erfolgreicher Einsatz für:
  - MATLAB-Quelltexte
  - Python-Messprogramme
  - Vorlesungsskripte, Übungsaufgaben, Praktikumsanleitungen, GET-Buch, Jahres- und Institutsberichte (in LaTeX)
- aller Anfang ist schwer (am Ende verstehen es aber auch die Sekretärinnen)
- man muss sich auf ein System festlegen


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
Bad code.
(c) 2008 Focus Shift

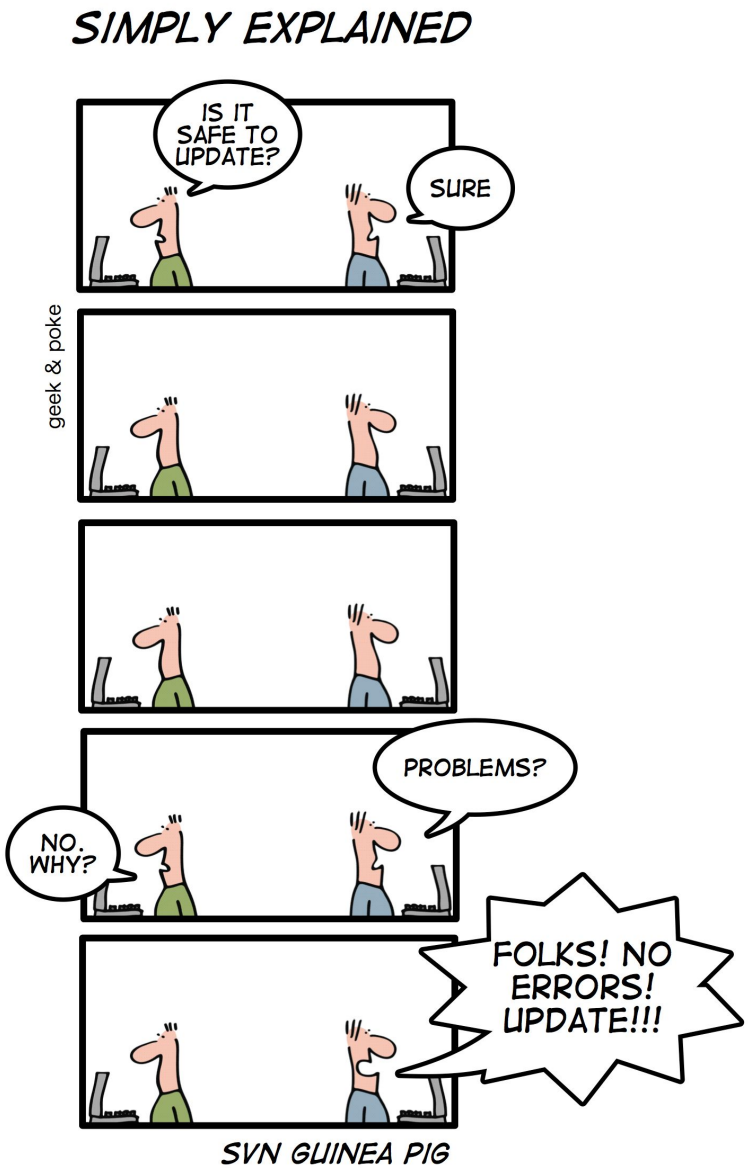
Verbleibende Probleme

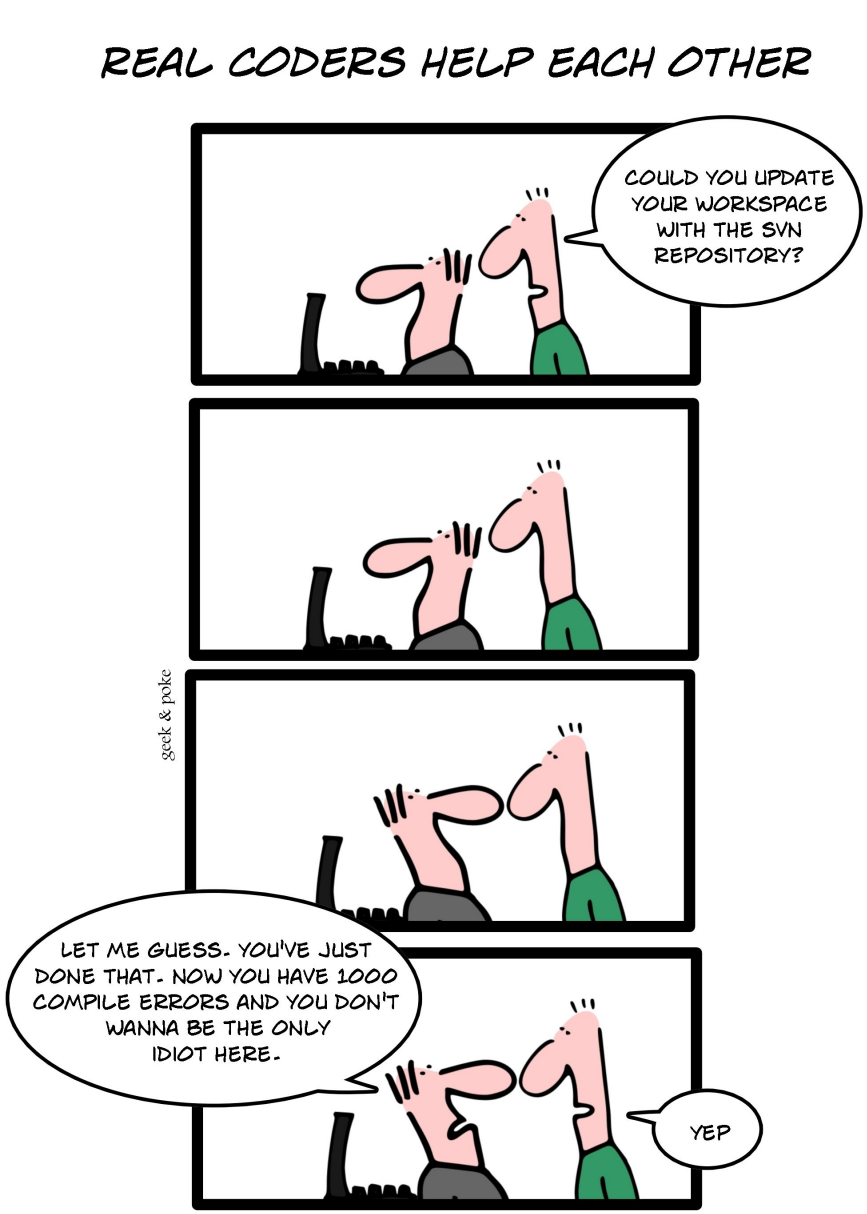

ONE DAY IN THE LIFE OF A CODER
PART 3
SOMETIMES IT'S REALLY HARD TO FIND PITHY CHECK-IN COMMENT.
"VARIOUS CHANGES"
LET ME GUESS: "SOME CHANGES"?
1130 AM: THE FIRST CHECK-IN OF THE DAY

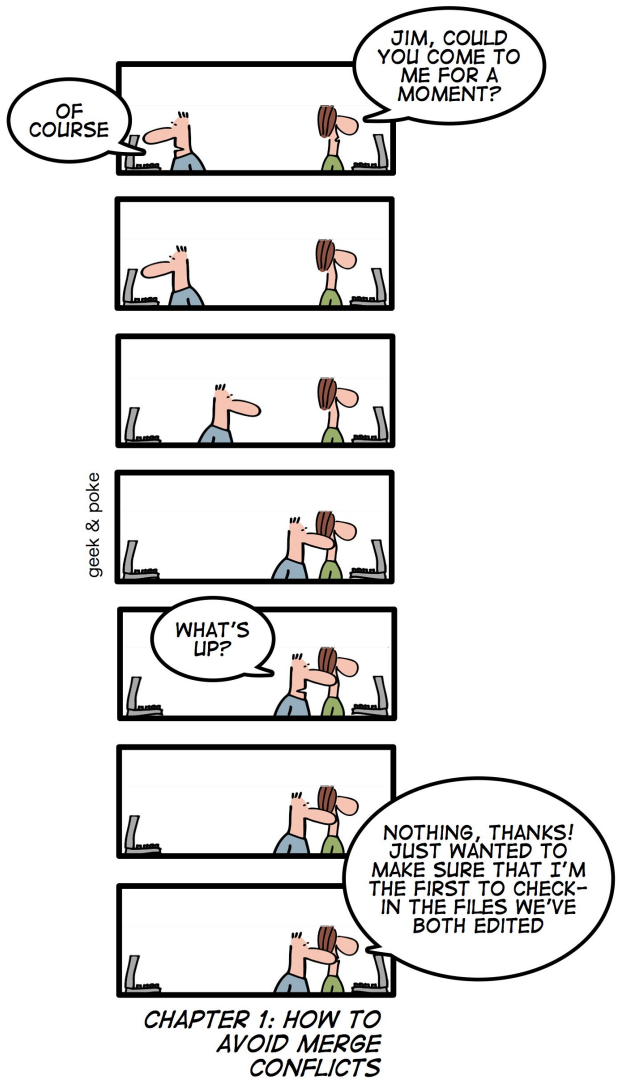
ONE DAY IN THE LIFE OF A CODER
PART 2



1000 AM: UPDATING THE WORKSPACE


SIMPLY EXPLAINED
IS IT SAFE TO UPDATE?
SURE
geek & poke
PROBLEMS?
NO. WHY?
FOLKS! NO ERRORS! UPDATE!!!
SVN GUINEA PIG


REAL CODERS HELP EACH OTHER
COULD YOU UPDATE YOUR WORKSPACE WITH THE SVN REPOSITORY?
geek & poke
LET ME GUESS. YOU'VE JUST DONE THAT. NOW YOU HAVE 1000 COMPILE ERRORS AND YOU DON'T WANNA BE THE ONLY IDIOT HERE.
YEP


BEING A CODER MADE EASY
JIM, COULD YOU COME TO ME FOR A MOMENT?
OF COURSE
geek & poke
WHAT'S UP?
NOTHING, THANKS! JUST WANTED TO MAKE SURE THAT I'M THE FIRST TO CHECK-IN THE FILES WE'VE BOTH EDITED
CHAPTER 1: HOW TO AVOID MERGE CONFLICTS

Vielen Dank für Eure Aufmerksamkeit!

Gibt es Fragen?