Freie Universität Berlin


# Subversion als Werkzeug in der Software-Entwicklung

## Eine Einführung

Tobias G. Pfeiffer
Freie Universität Berlin

Seminar DG-Verfahren, 9. Juni 2009

# Voraussetzungen/Ziele des Vortrags

**Situation**
Der Zuhörer will in Zusammenarbeit mit anderen Personen einmalig eine „kleines“ Software-Projekt durchführen.

**Voraussetzungen**

- Zuhörer kann mit einem Computer mit Linux- oder Windows-Betriebssystem umgehen

**Ziel**
Der Zuhörer:

- kennt Grundlagen, Sinn und Zweck der Versionsverwaltung
- kann mit Subversion eigenen Code verwalten
- kann die Funktionen von Subversion nutzen, um die Entwicklungen anderer zu verfolgen

# Quellen und weiterführende Informationen

Freie Universität Berlin

- Subversion-Buch: `http://svnbook.red-bean.com`
- Wikipedia:
  - Versionsverwaltung
  - Subversion (Software)
  - ...
- eigene (teilweise schmerzvolle) Erfahrungen

Teile der Grafiken von `http://www.opensecurityarchitecture.org`
(Creative-Commons Share-alike-Lizenz)

# Überblick

Einführung in Versionsverwaltung

Existierende Daten in ein Repository importieren

An Daten arbeiten: Der Update-Commit-Zyklus

Häufige Probleme, Tipps

# Überblick

Einführung in Versionsverwaltung

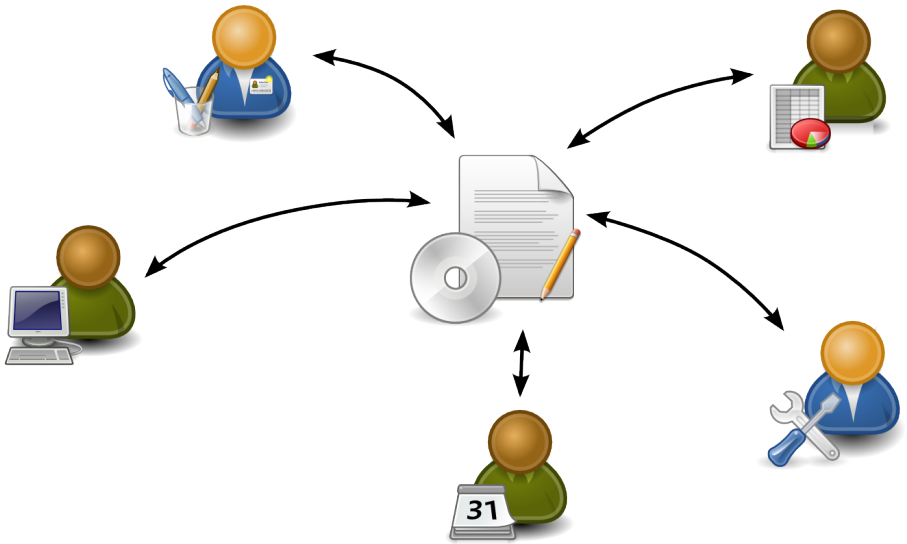
Existierende Daten in ein Repository importieren

An Daten arbeiten: Der Update-Commit-Zyklus

Häufige Probleme, Tipps

# Ziele der Versionsverwaltung

Hauptziel:




Dabei:

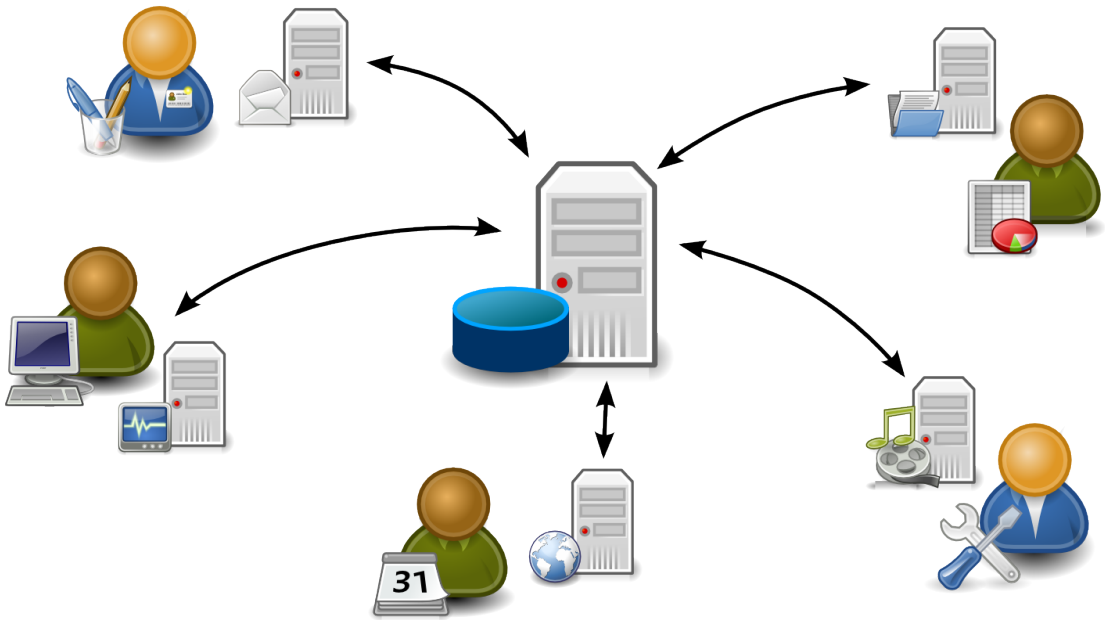
- Protokollieren
- Wiederherstellen
- Archivieren
- Koordinieren
- hier irrelevant: Releasemanagement, Entwicklungszweige, . . .

# Konzepte der Versionsverwaltung

- „Problem“: Dateien auf eigenem Rechner – wie verwalten?

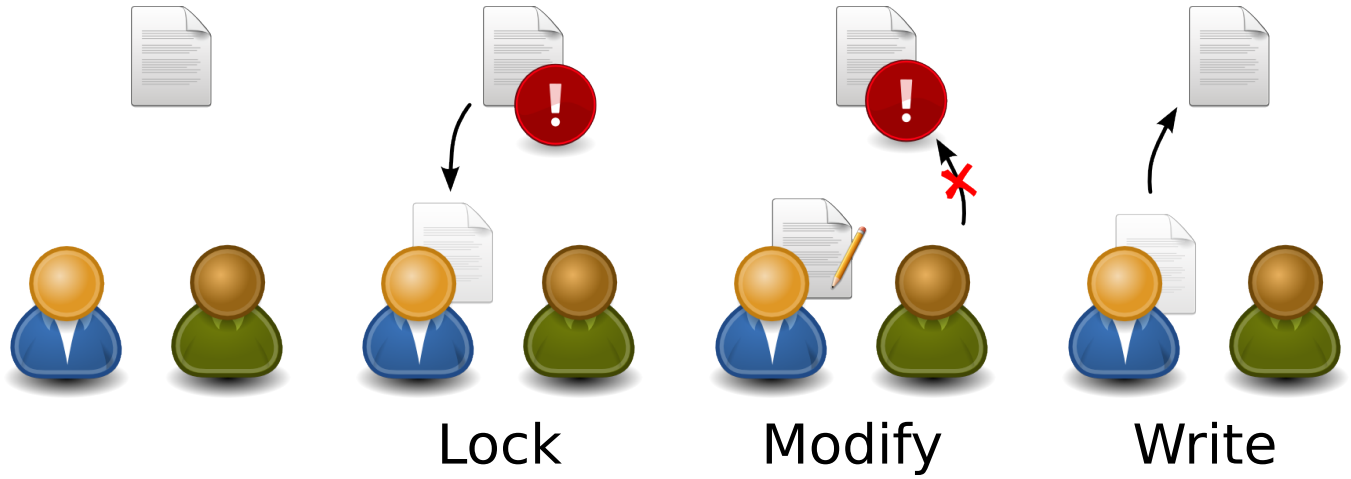
# Konzepte der Versionsverwaltung

- „Problem“: Dateien auf eigenem Rechner – wie verwalten?
- zentraler Server, lokale Arbeitskopien:

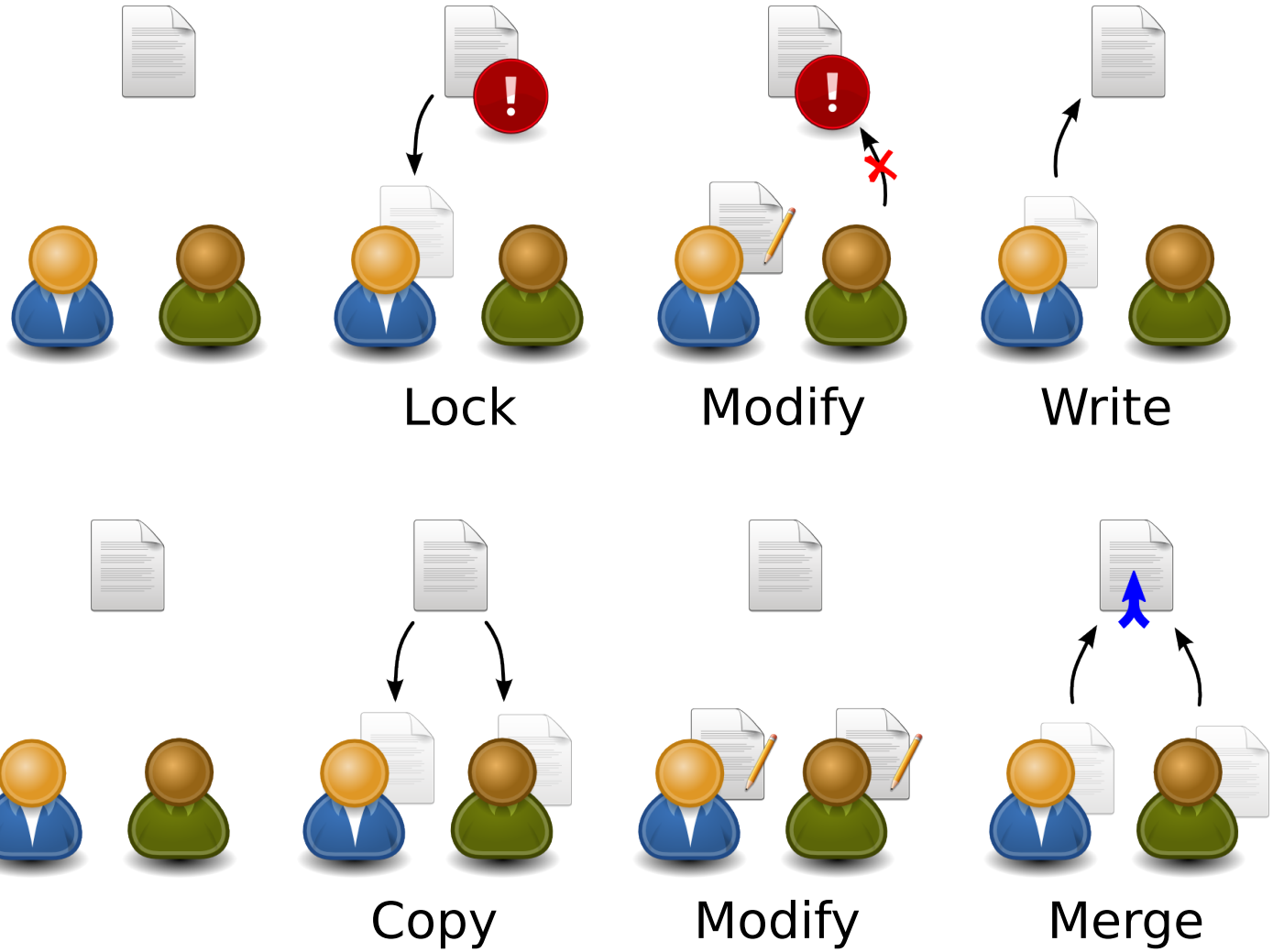



- gemeinsames Arbeiten:
  Lock-Modify-Write ⟷ Copy-Modify-Merge

# Konzepte der Versionsverwaltung II



# Konzepte der Versionsverwaltung II



# Bemerkungen zu Subversion

- Subversion (syn.: SVN) **ist ein** Versionskontrollsystem
  ⇒ zwei Teile:

Server-Software
(verwaltet „Repository“)
⇝ vom Admin bereitgestellt

Client-Software
(verwaltet „Working Copy“)
⇝ unsere Baustelle

- eigentlich: Kommandozeilenanwendung
- inzwischen auch viele GUIs und Plugins: TortoiseSVN (`http://tortoisesvn.tigris.org/`), Subclipse, ...

# Überblick

Einführung in Versionsverwaltung

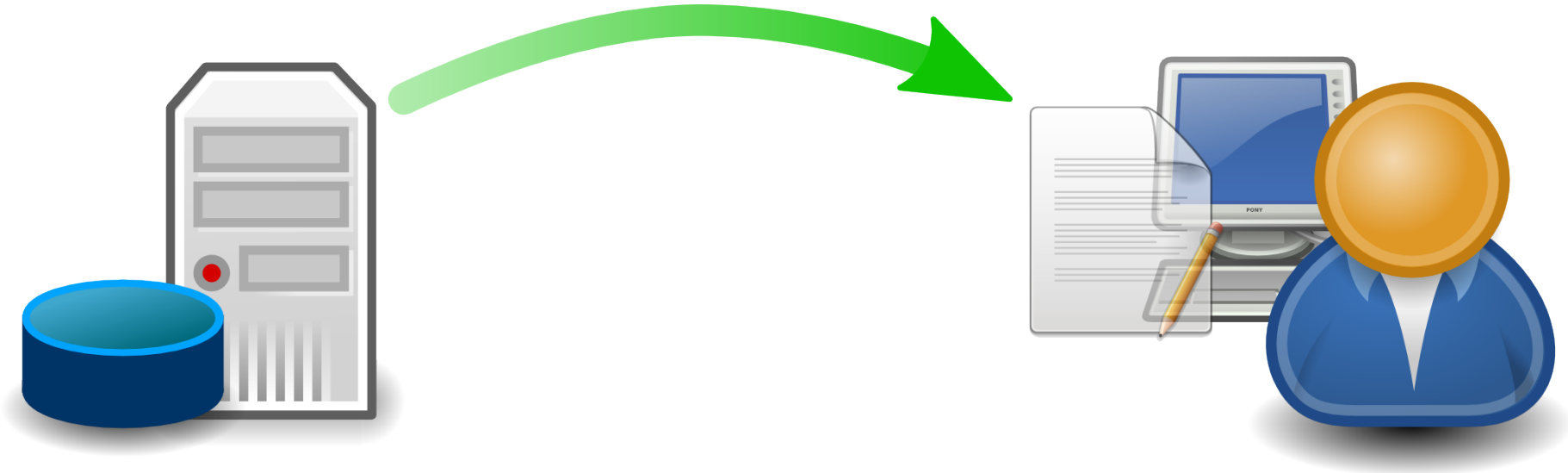
Existierende Daten in ein Repository importieren

An Daten arbeiten: Der Update-Commit-Zyklus

Häufige Probleme, Tipps

# Lokale Arbeitskopie erstellen

Situation: Repository existiert ⇝ Arbeitskopie erstellen (einmaliger Vorgang!)



Operation: checkout

# svn checkout

- Ziel: Arbeitskopie erstellen
- alle Beispiele zuerst auf Kommandozeilenebene

```
user @ ~ $
```

# svn checkout

- Ziel: Arbeitskopie erstellen
- alle Beispiele zuerst auf Kommandozeilenebene

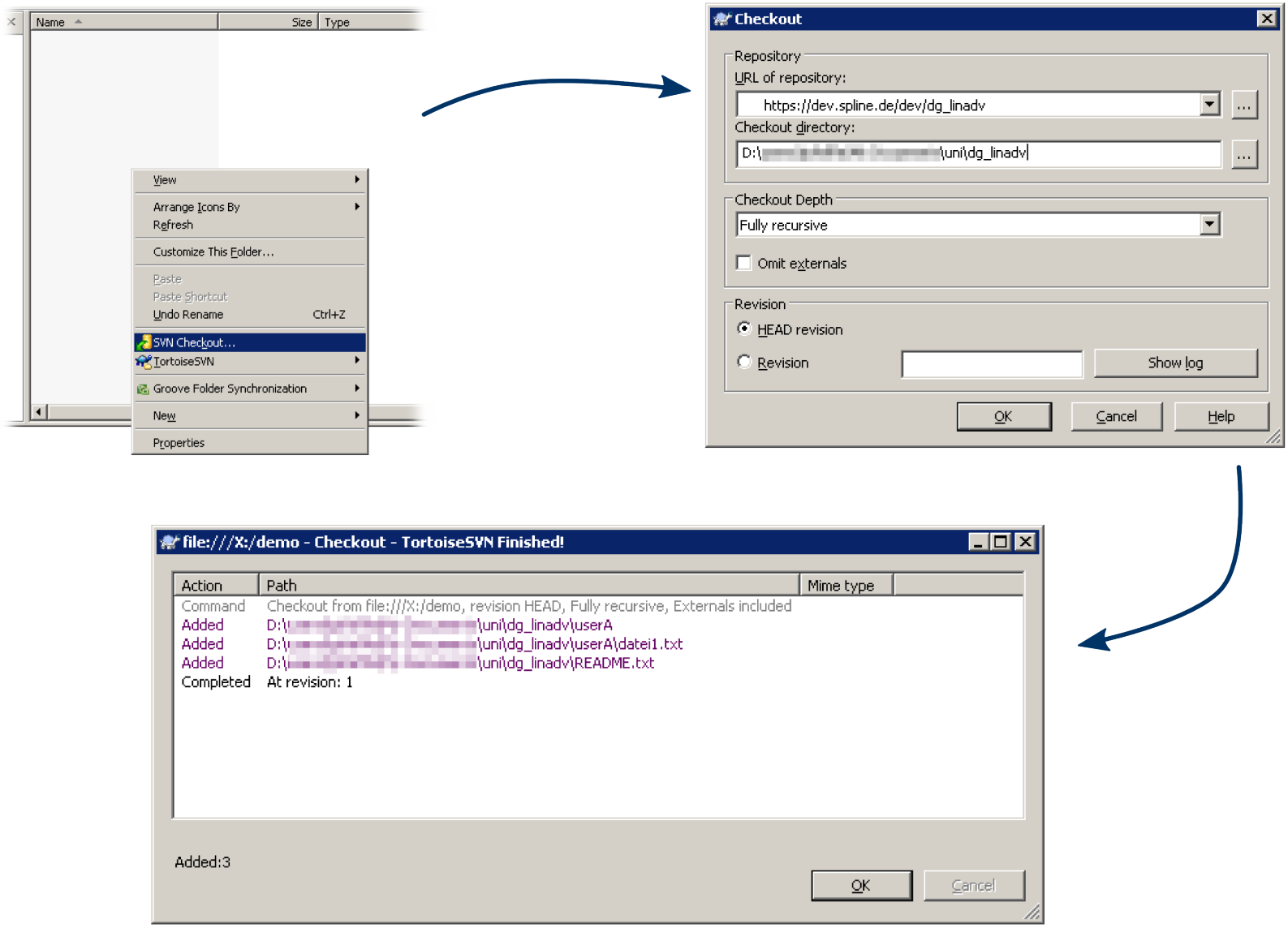
```
user @ ~ $ svn help checkout
checkout (co): Checkt eine Arbeitskopie aus einem Projektarchiv aus.
Aufruf: checkout URL[@REV]... [PFAD]
[...]
user @ ~ $
```

# svn checkout

- Ziel: Arbeitskopie erstellen
- alle Beispiele zuerst auf Kommandozeilenebene

```
user @ ~ $ svn help checkout
checkout (co): Checkt eine Arbeitskopie aus einem Projektarchiv aus.
Aufruf: checkout URL[@REV]... [PFAD]
[...]
user @ ~ $ svn checkout  https://dev.spline.de/svn/dg_linadv  ~/uni/dg_linadv
A    userA
A    userA/datei1.txt
A    README.txt
Ausgecheckt, Revision 1.
user @ ~ $
```

# TortoiseSVN: checkout



# Existierende Daten hinzufügen

Freie Universität Berlin

- eigene Dateien in die frisch angelegte Arbeitskopie kopieren
- neue Dateien noch nicht unter Versionskontrolle!

Operation: add

```
user @ ~/uni/dg_linadv $ svn help add
add: Stellt Dateien und Verzeichnisse unter Versionskontrolle und
plant sie zur Übertragung ins Projektarchiv ein.
Das tatsächliche Hinzufügen findet erst beim nächsten Übertragen statt.
Aufruf: add PFAD...
[...]
user @ ~/uni/dg_linadv $
```

# Existierende Daten hinzufügen

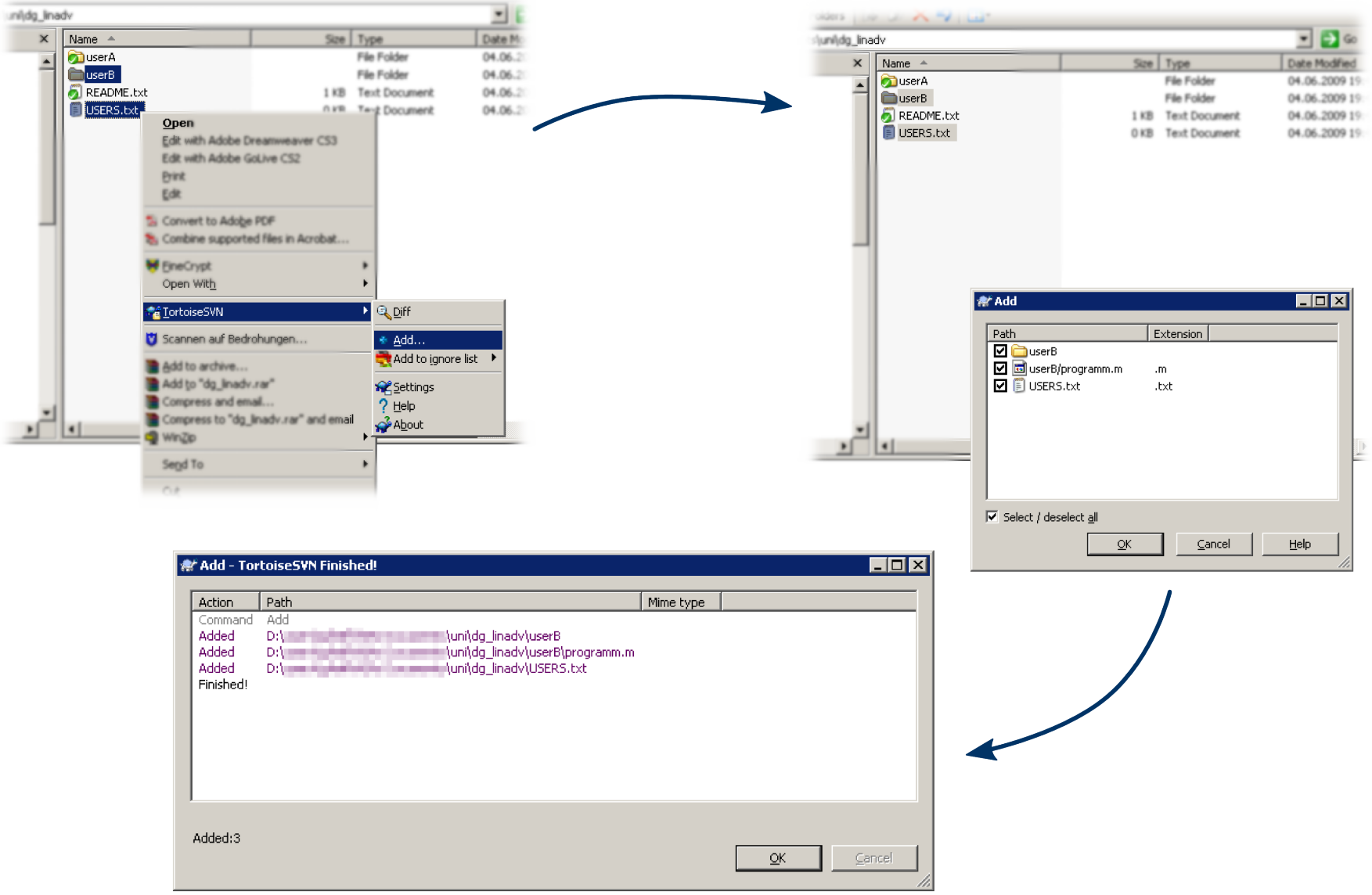
Freie Universität Berlin

- eigene Dateien in die frisch angelegte Arbeitskopie kopieren
- neue Dateien noch nicht unter Versionskontrolle!

Operation: add

```
user @ ~/uni/dg_linadv $ svn help add
add: Stellt Dateien und Verzeichnisse unter Versionskontrolle und
plant sie zur Übertragung ins Projektarchiv ein.
Das tatsächliche Hinzufügen findet erst beim nächsten Übertragen statt.
Aufruf: add PFAD...
[...]
user @ ~/uni/dg_linadv $ svn add userB/ USERS.txt
A         userB
A         userB/programm.m
A         USERS.txt
user @ ~/uni/dg_linadv $
```

# TortoiseSVN: add



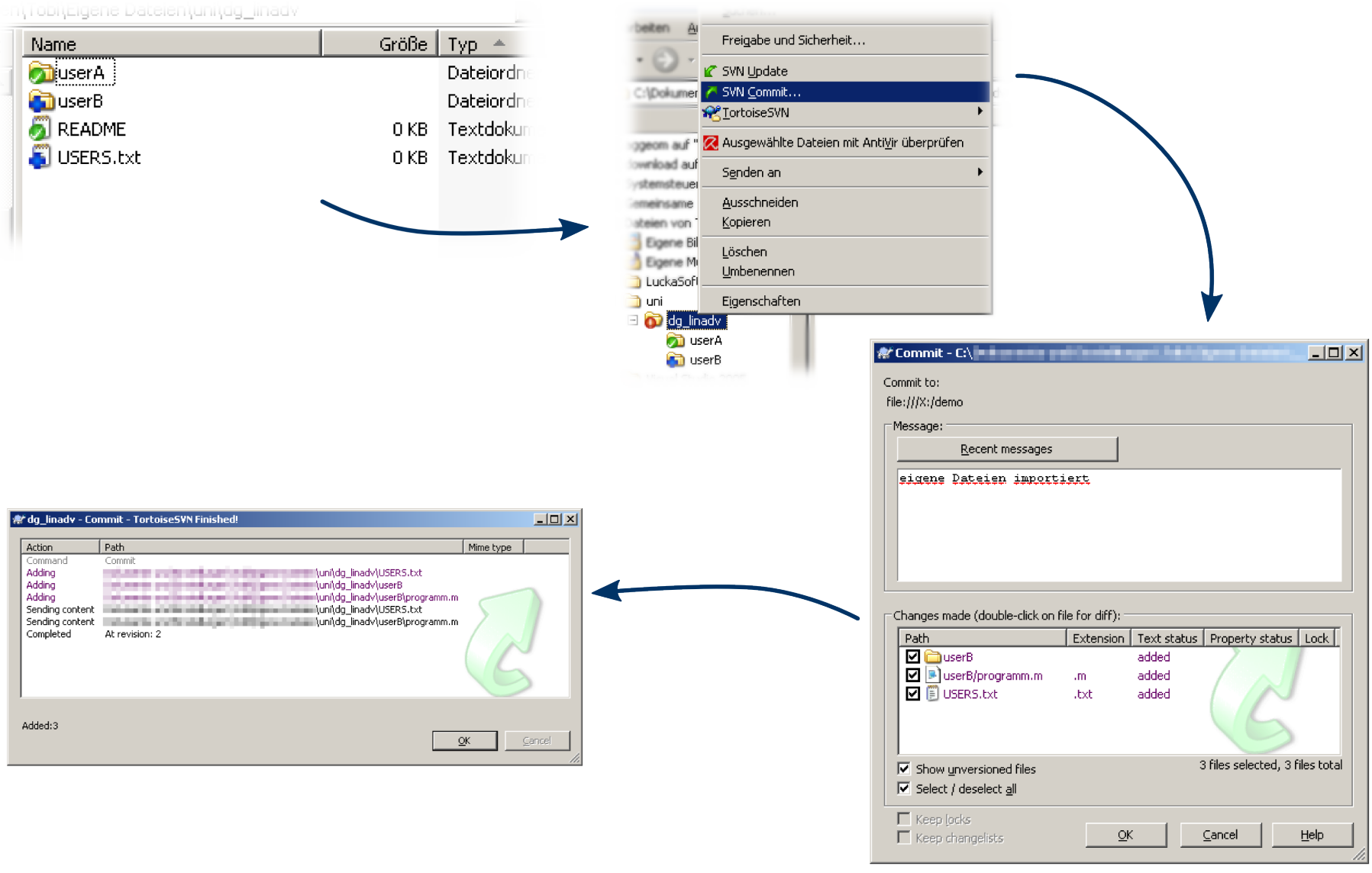
# Neue Daten übertragen

Ziel: neue Daten ins Repository übertragen

Operation: commit

```
user @ ~/uni/dg_linadv $ svn help commit
commit (ci): Überträgt Änderungen Ihrer Arbeitskopie ins Projektarchiv.
Aufruf: commit [PFAD...]
[...]
user @ ~/uni/dg_linadv $ svn commit -m "eigene Daten importiert"
Hinzufügen     USERS.txt
Hinzufügen     userB
Hinzufügen     userB/programm.m
Übertrage Daten ..
Revision 2 übertragen.
user @ ~/uni/dg_linadv $
```

# TortoiseSVN: commit



# Überblick

Einführung in Versionsverwaltung

Existierende Daten in ein Repository importieren

An Daten arbeiten: Der Update-Commit-Zyklus

Häufige Probleme, Tipps

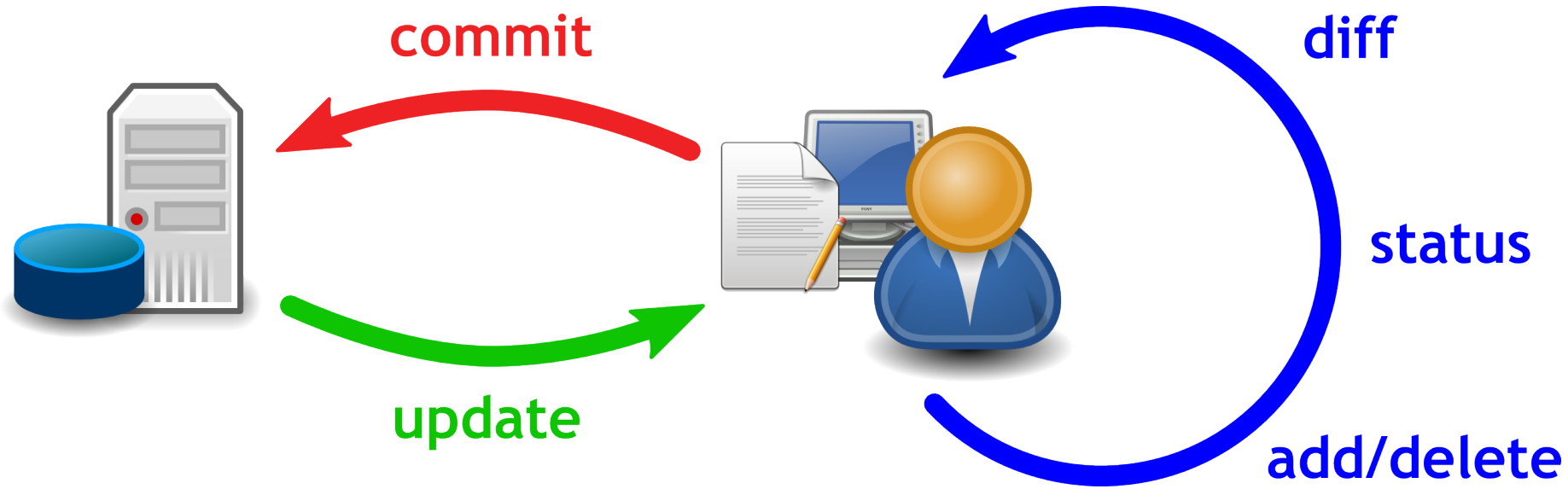
# Überblick: Arbeiten mit Subversion

Freie Universität Berlin

- eigene Dateien sind jetzt unter Versionskontrolle
  ⇝ regulär mit Dateien weiter arbeiten
- Wie fügt sich Subversion jetzt in den Arbeitsablauf ein?

# Überblick: Arbeiten mit Subversion

- eigene Dateien sind jetzt unter Versionskontrolle
  ⇝ regulär mit Dateien weiter arbeiten
- Wie fügt sich Subversion jetzt in den Arbeitsablauf ein?
- „Update-Commit-Zyklus“:



# Neueste Versionen abholen

- „kollaborativ“ = „zusammen arbeiten“
- Ich habe gerade an meiner Datei programm.m geschrieben – was haben meine Mitarbeiter gemacht?

Operation: update & log

```
user @ ~/uni/dg_linadv $ svn update
A    userA/datei2.txt
G    userB/programm.m
U    README.txt
Aktualisiert zu Revision 3.
user @ ~/uni/dg_linadv $
```

# Neueste Versionen abholen

- „kollaborativ“ = „zusammen arbeiten“
- Ich habe gerade an meiner Datei programm.m geschrieben – was haben meine Mitarbeiter gemacht?

Operation: update & log

```
user @ ~/uni/dg_linadv $ svn update
A    userA/datei2.txt
G    userB/programm.m
U    README.txt
Aktualisiert zu Revision 3.
user @ ~/uni/dg_linadv $ svn log -r 3
------------------------------------------------------------------------
r3 | userA | 2009-06-05 14:36:56 +0200 (Fr, 05. Jun 2009) | 2 lines

Readme erweitert, neue Daten
------------------------------------------------------------------------
user @ ~/uni/dg_linadv $
```

# Neueste Versionen abholen

Freie Universität Berlin

- „kollaborativ“ = „zusammen arbeiten“
- Ich habe gerade an meiner Datei programm.m geschrieben – was haben meine Mitarbeiter gemacht?

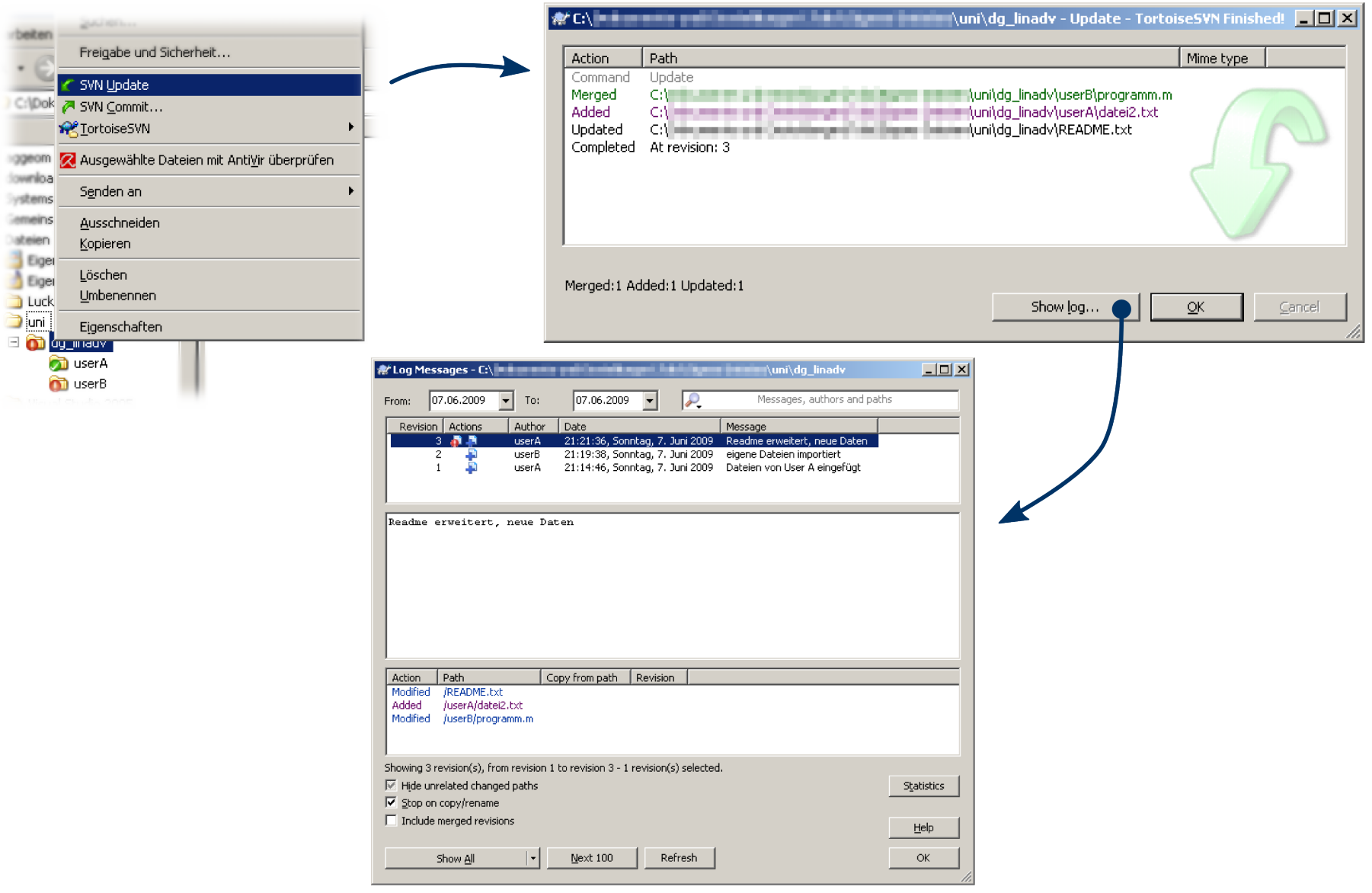
Operation: update & log

```
user @ ~/uni/dg_linadv $ svn update
A    userA/datei2.txt
G    userB/programm.m
U    README.txt
Aktualisiert zu Revision 3.
user @ ~/uni/dg_linadv $ svn log -r 3
------------------------------------------------------------------------
r3 | userA | 2009-06-05 14:36:56 +0200 (Fr, 05. Jun 2009) | 2 lines

Readme erweitert, neue Daten
------------------------------------------------------------------------
user @ ~/uni/dg_linadv $
```

- hier findet der Merge statt! ⇝ evtl. Konflikte

# TortoiseSVN: update & log



# Eigene Änderungen übertragen

Eigener Code soll ins Repository ⇝ ggf. Dateien hinzufügen oder löschen, Änderungen prüfen, kommentieren, hochladen

Operation: add/delete, status, diff & commit

```
user @ ~/uni/dg_linadv/userB $ svn status
?      programm2.m
?      daten.txt
M      programm.m
user @ ~/uni/dg_linadv/userB $
```

# Eigene Änderungen übertragen

Eigener Code soll ins Repository ⇝ ggf. Dateien hinzufügen oder löschen, Änderungen prüfen, kommentieren, hochladen

Operation: add/delete, status, diff & commit

```
user @ ~/uni/dg_linadv/userB $ svn status
?      programm2.m
?      daten.txt
M      programm.m
user @ ~/uni/dg_linadv/userB $ svn diff
Index: programm.m
===================================================================
--- programm.m  (Revision 3)
+++ programm.m  (Arbeitskopie)
@@ -1 +1,3 @@
 % was machst du denn hier??
+% hier passieren tolle Sachen
user @ ~/uni/dg_linadv/userB $
```

# Eigene Änderungen übertragen

Eigener Code soll ins Repository ⇝ ggf. Dateien hinzufügen oder löschen, Änderungen prüfen, kommentieren, hochladen

Operation: add/delete, status, diff & commit

```
user @ ~/uni/dg_linadv/userB $ svn status
?      programm2.m
?      daten.txt
M      programm.m
user @ ~/uni/dg_linadv/userB $ svn diff
Index: programm.m
===================================================================
--- programm.m  (Revision 3)
+++ programm.m  (Arbeitskopie)
@@ -1 +1,3 @@
 % was machst du denn hier??
+% hier passieren tolle Sachen
user @ ~/uni/dg_linadv/userB $ svn add programm2.m
A         programm2.m
user @ ~/uni/dg_linadv/userB $ svn status
?      daten.txt
M      programm.m
A      programm2.m
user @ ~/uni/dg_linadv/userB $
```

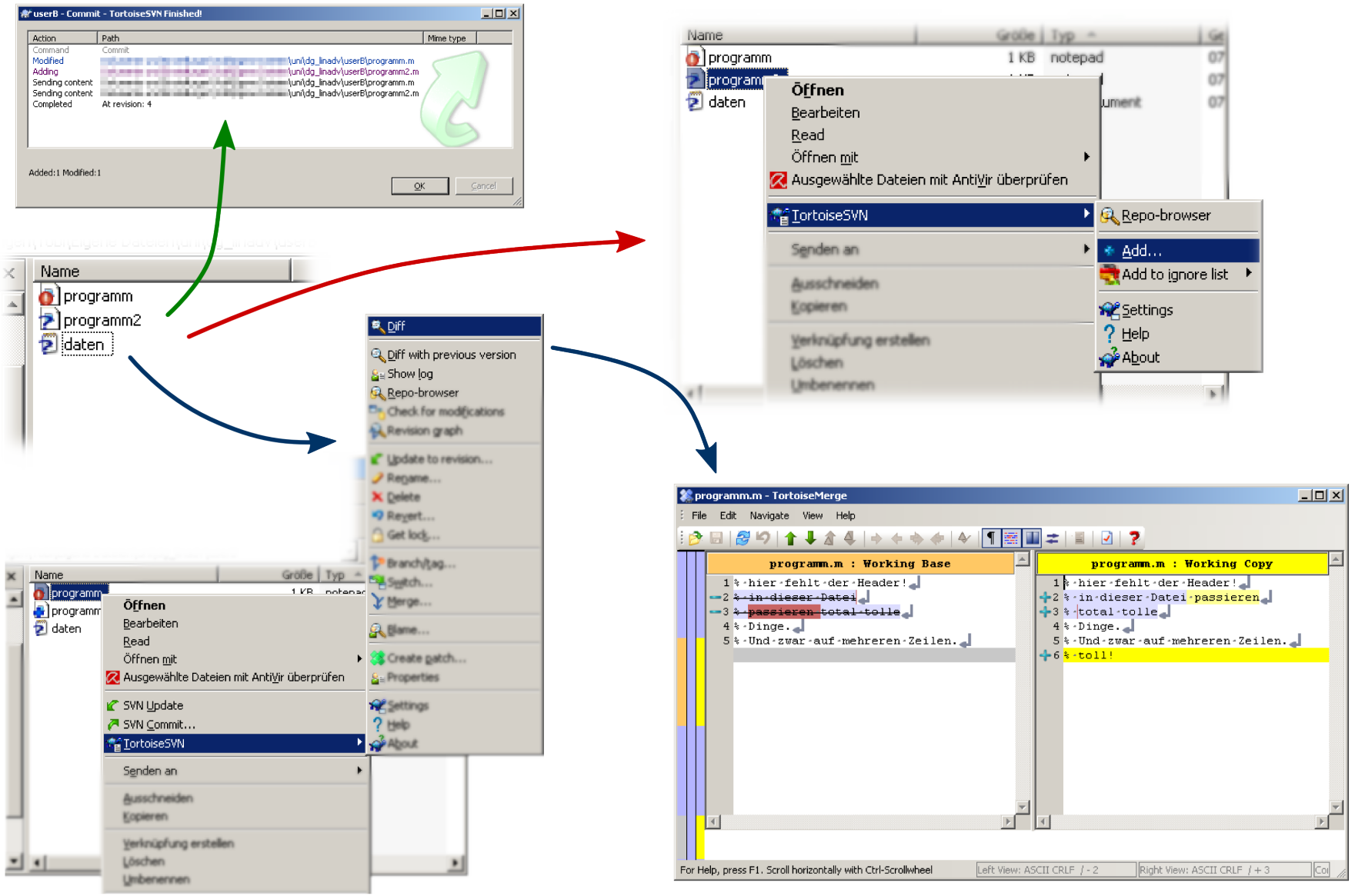
# Eigene Änderungen übertragen

Eigener Code soll ins Repository ⇝ ggf. Dateien hinzufügen oder löschen, Änderungen prüfen, kommentieren, hochladen

Operation: add/delete, status, diff & commit

```
user @ ~/uni/dg_linadv/userB $ svn status
?      programm2.m
?      daten.txt
M      programm.m
user @ ~/uni/dg_linadv/userB $ svn diff
Index: programm.m
===================================================================
--- programm.m  (Revision 3)
+++ programm.m  (Arbeitskopie)
@@ -1 +1,3 @@
 % was machst du denn hier??
+% hier passieren tolle Sachen
user @ ~/uni/dg_linadv/userB $ svn add programm2.m
A         programm2.m
user @ ~/uni/dg_linadv/userB $ svn status
?      daten.txt
M      programm.m
A      programm2.m
user @ ~/uni/dg_linadv/userB $ svn commit -m "zweites Programm erstellt"
Sende          userB/programm.m
Hinzufügen     userB/programm2.m
Übertrage Daten ..
Revision 4 übertragen.
```

# TortoiseSVN: add, diff & commit



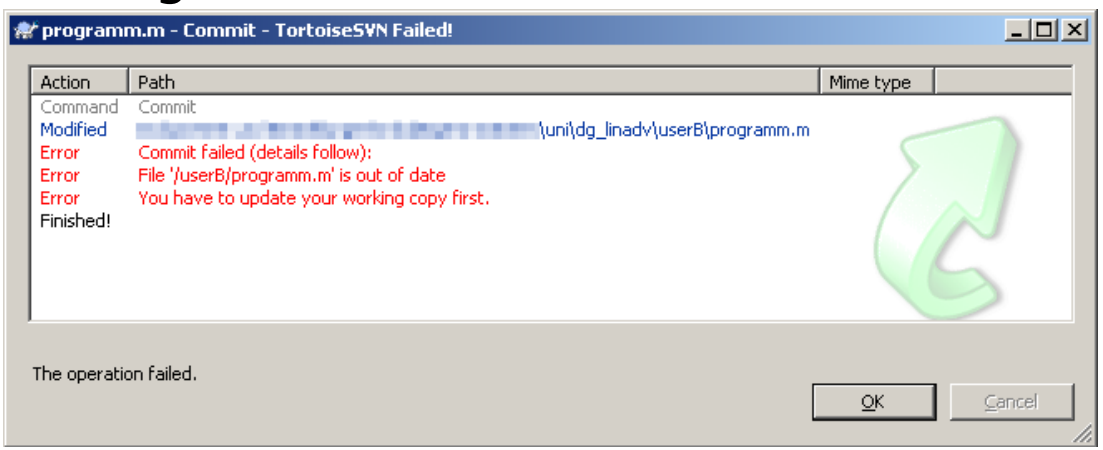
# Nachbereitung

- nochmal: Merge passiert beim Update – nicht beim Commit!
- Folgerung: Änderungen von Mitarbeitern in eigenen Dateien führen beim Commit (ohne vorheriges Update) zum Fehler!

```
user @ ~/uni/dg_linadv $ svn commit
Sende          userB/programm.m
svn: Übertragen schlug fehl (Details folgen):
svn: Datei »/userB/programm.m« ist veraltet
```

  ⇝ zum Beheben: Update

- analog in TortoiseSVN:




- regelmäßig updaten, aber immer vor einem Commit!

# Überblick

Einführung in Versionsverwaltung

Existierende Daten in ein Repository importieren

An Daten arbeiten: Der Update-Commit-Zyklus

Häufige Probleme, Tipps

# Konflikte

- Problem: eigene und fremde Änderungen an der gleichen Stelle der gleichen Datei ⇝ wie mergen?

```
user @ ~/uni/dg_linadv $ svn update
C    USERS.txt
Aktualisiert zu Revision 6.
```

# Konflikte

- Problem: eigene und fremde Änderungen an der gleichen Stelle der gleichen Datei ⇝ wie mergen?

```
user @ ~/uni/dg_linadv $ svn update
C    USERS.txt
Aktualisiert zu Revision 6.
```

- Datei USERS.txt enthält Markierung, die Konflikt anzeigt:

```
<<<<<<< .mine
userB
=======
userA
>>>>>>> .r6
```

- außerdem jetzt vorhanden:
  - USERS.txt.mine: eigene Datei vor dem Update
  - USERS.txt.r4: Original-Datei vor eigener Bearbeitung
  - USERS.txt.r6: Datei, wie im Repository vorhanden
- jetzt: Datei reparieren, Reparatur mitteilen, hochladen

Operation: resolved

# Konflikte

- Problem: eigene und fremde Änderungen an der gleichen Stelle der gleichen Datei ⇝ wie mergen?

```
user @ ~/uni/dg_linadv $ svn update
C    USERS.txt
Aktualisiert zu Revision 6.
```

- Datei USERS.txt enthält Markierung, die Konflikt anzeigt:

```
<<<<<<< .mine
userB
=======
userA
>>>>>>> .r6
```

- außerdem jetzt vorhanden:
  - USERS.txt.mine: eigene Datei vor dem Update
  - USERS.txt.r4: Original-Datei vor eigener Bearbeitung
  - USERS.txt.r6: Datei, wie im Repository vorhanden
- jetzt: Datei reparieren, Reparatur mitteilen, hochladen

Operation: resolved

```
user @ ~/uni/dg_linadv $ svn resolved USERS.txt
Konflikt von »USERS.txt« aufgelöst
user @ ~/uni/dg_linadv $ svn commit -m "User-Datei repariert"
[...]
```

# Konflikte

- Problem: eigene und fremde Änderungen an der gleichen Stelle der gleichen Datei ⇝ wie mergen?

```
user @ ~/uni/dg_linadv $ svn update
C    USERS.txt
Aktualisiert zu Revision 6.
```

- Datei USERS.txt enthält Markierung, die Konflikt anzeigt:

```
<<<<<<< .mine
userB
=======
userA
>>>>>>> .r6
```

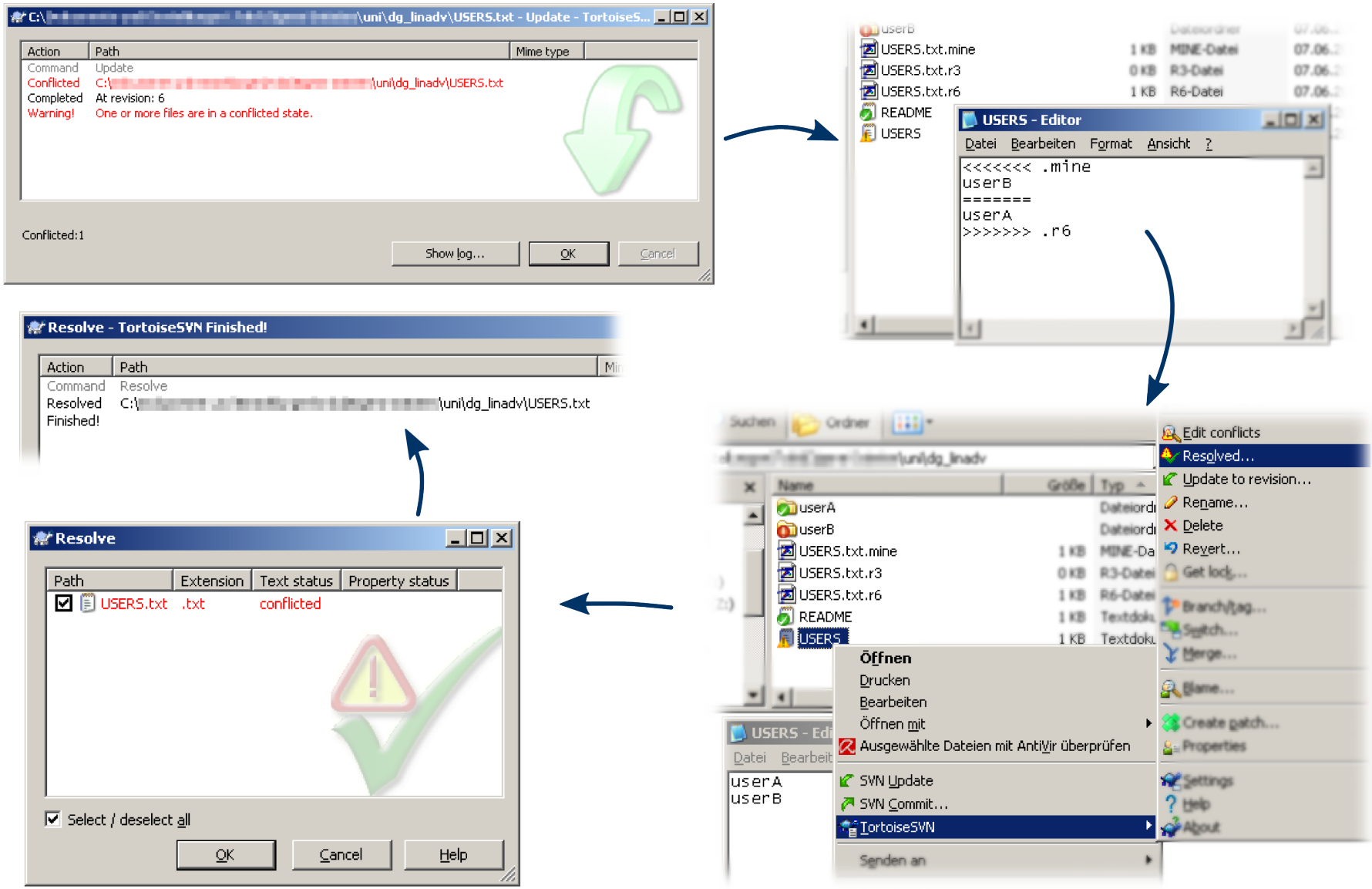
- außerdem jetzt vorhanden:
  - USERS.txt.mine: eigene Datei vor dem Update
  - USERS.txt.r4: Original-Datei vor eigener Bearbeitung
  - USERS.txt.r6: Datei, wie im Repository vorhanden
- jetzt: Datei reparieren, Reparatur mitteilen, hochladen

Operation: resolved

```
user @ ~/uni/dg_linadv $ svn resolved USERS.txt
Konflikt von »USERS.txt« aufgelöst
user @ ~/uni/dg_linadv $ svn commit -m "User-Datei repariert"
[...]
```

- beste Methode gegen Konflikte: Kommunikation!

# TortoiseSVN: Konfliktbehandlung

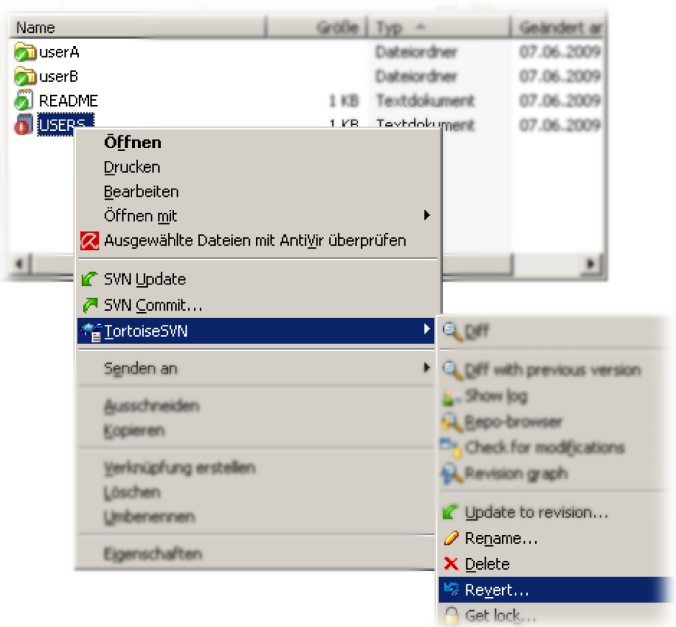


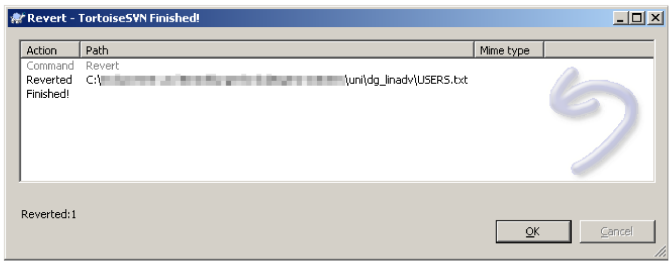
# Änderungen rückgängig machen

- eigene Änderungen vor Commit rückgängig machen: einfach

Operation: revert

```
user @ ~/uni/dg_linadv $ svn status
M       USERS.txt
user @ ~/uni/dg_linadv $ svn revert USERS.txt
Rückgängig gemacht: »USERS.txt«
user @ ~/uni/dg_linadv $
```



# Änderungen rückgängig machen II

Freie Universität Berlin

- ganze Commits rückgängig machen: nicht so einfach
- kein explizites „Undo“-Kommando
- Idee: aktuelle Dateien auf alten Stand patchen:

$$\text{ziel} = \text{aktuell} + (\text{ziel} - \text{aktuell})$$

Operation: merge

# Änderungen rückgängig machen II



- ganze Commits rückgängig machen: nicht so einfach
- kein explizites „Undo“-Kommando
- Idee: aktuelle Dateien auf alten Stand patchen:

$$\text{ziel} = \text{aktuell} + (\text{ziel} - \text{aktuell})$$

Operation: merge

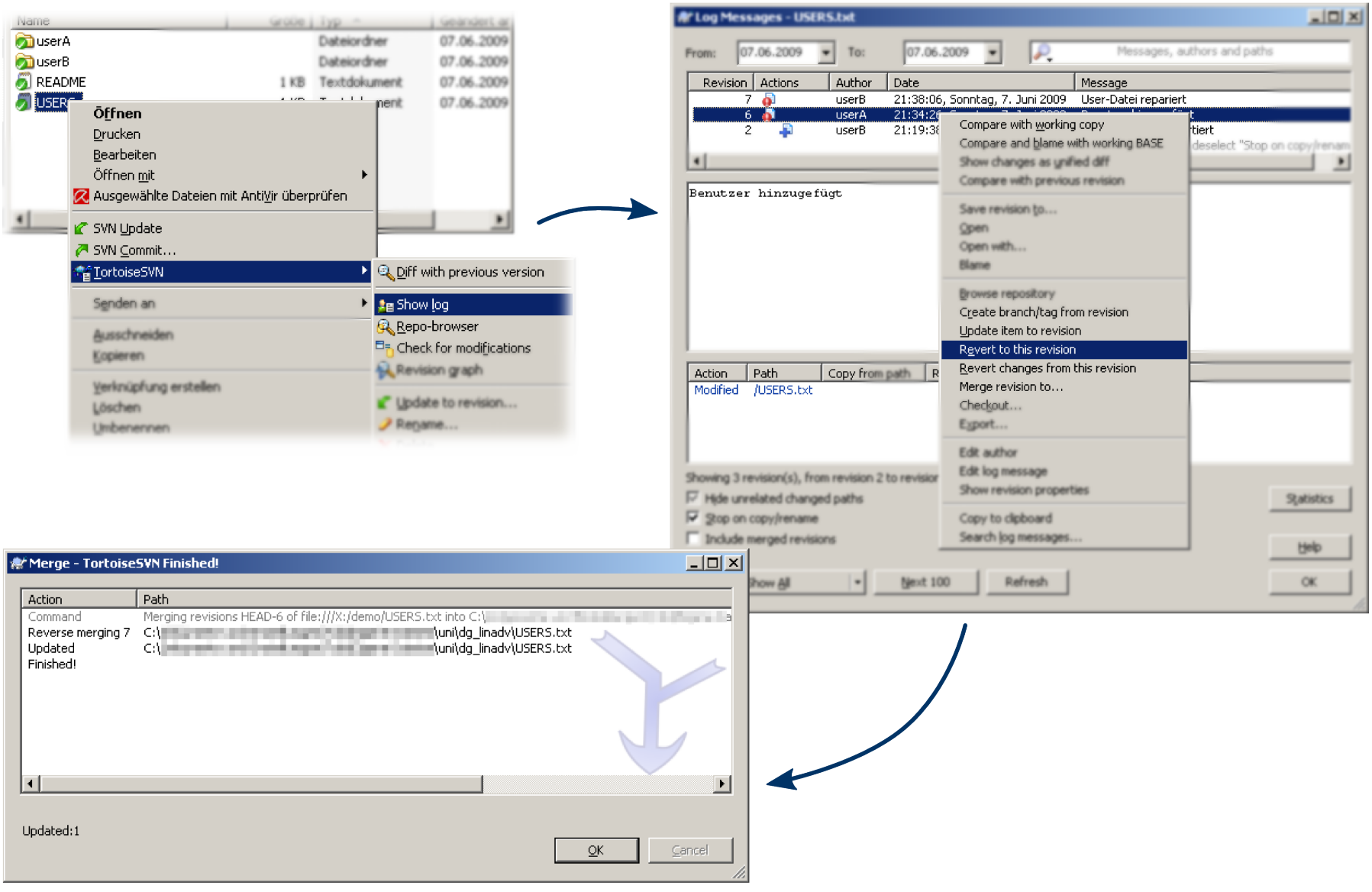
- Beispiel: aktuelle USERS.txt von rev7 zurück nach rev6:

$$6 = 7 + (6 - 7)$$

```
user @ ~/uni/dg_linadv $ svn merge -r 7:6 USERS.txt
-- Rückwärtiges Zusammenführen von r7 in »USERS.txt«:
U    USERS.txt
user @ ~/uni/dg_linadv $ svn commit -m "User B wieder entfernt"
Sende          USERS.txt
Übertrage Daten .
Revision 8 übertragen.
user @ ~/uni/dg_linadv $
```

- Vorsicht: nur einer von *n* Anwendungsfällen von merge
  ⇝ ganz schnell unerwarteter Effekt

# TortoiseSVN: merge



# Abschluss

- Subversion: tolles Werkzeug für die Kollaboration
- wie bei jedem Werkzeug: Umgang will gelernt sein
- TortoiseSVN erspart „Vokabeln lernen“ – Konzepte müssen trotzdem verstanden sein

Fragen?