Verzió kontroll

- Version Control, Revision Control, Source Control
  - primitív
  - központosított
  - elosztott
- "time machine & concurrency manager"

Verzió kontroll: alapprobléma

- Dokumentumok, forráskód, stb. fejlesztése
  - eredetileg csak szöveg alapú file-ok
- Undo: korábbi verziók megtartása
  - változások követhetőek (history)
  - visszaállítható
- Többszörös hozzáférés
  - többen szerkeszthetik egyszerre

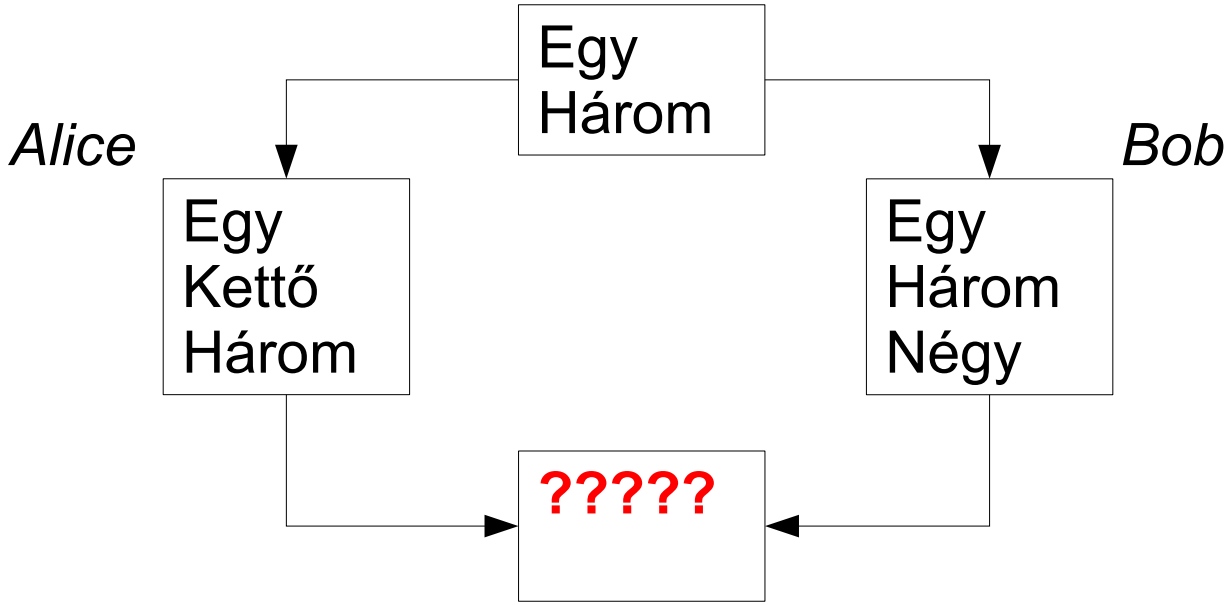
Primitív verzió kontroll

- Otthon már mindenki feltalálta
  - „ProjectReport_2008_07_13.doc”, „stack.c.bak”
  - verzió a fájl nevében
- Szabály: módosítás előtt mentés új néven

```
$ cat foo.txt
Egy
Három
$ cp foo.txt foo.2009_07_23.txt
$ edit foo.txt
$ cat foo.txt
Egy
Kettő
Három
```

Primitív verzió kontroll: több fejlesztő

- Megosztott könyvtárban: ssh, NFS, SMB, stb.
- Ha többen módosítanak egyszerre: ütközés



- Megoldás: zárolás
- Microsoft Visual SourceSafe (korai verziók)

Primitív verzió kontroll: problémák

- A szabályok betartása nem kényszeríthető ki
  - a módosítás során a fejlesztőnek az alábbi műveleteket kell sorrendben végrehajtani:

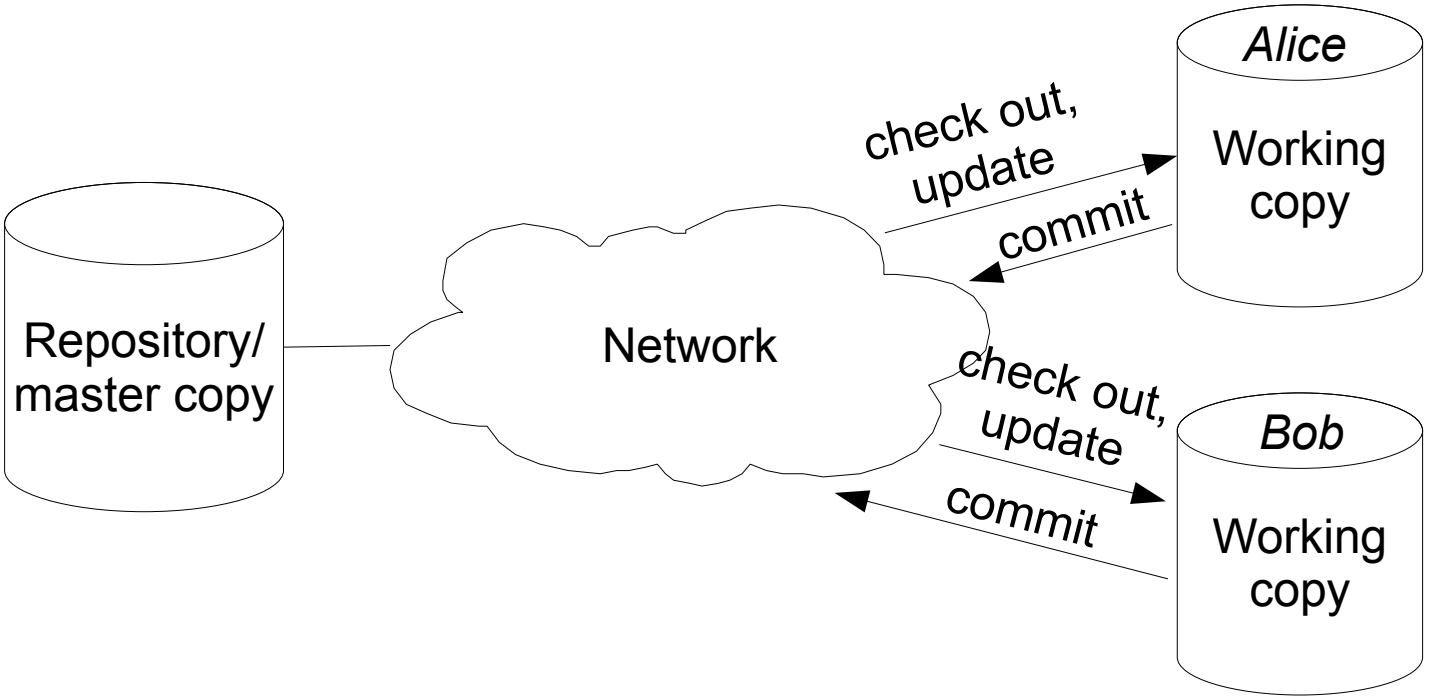
    `lock→archive→edit→save→unlock`

  - mi van, ha az egyik művelet kimarad?
- Coarse-grain multitasking
  - egyszerre csak egy felhasználó szerkeszt
  - rossz hatékonyság
- Nem skálázódik 2-3 fejlesztőnél többre
- Automatizálni kell a fejlesztés folyamatát!

Verzió kontroll: alapfogalmak

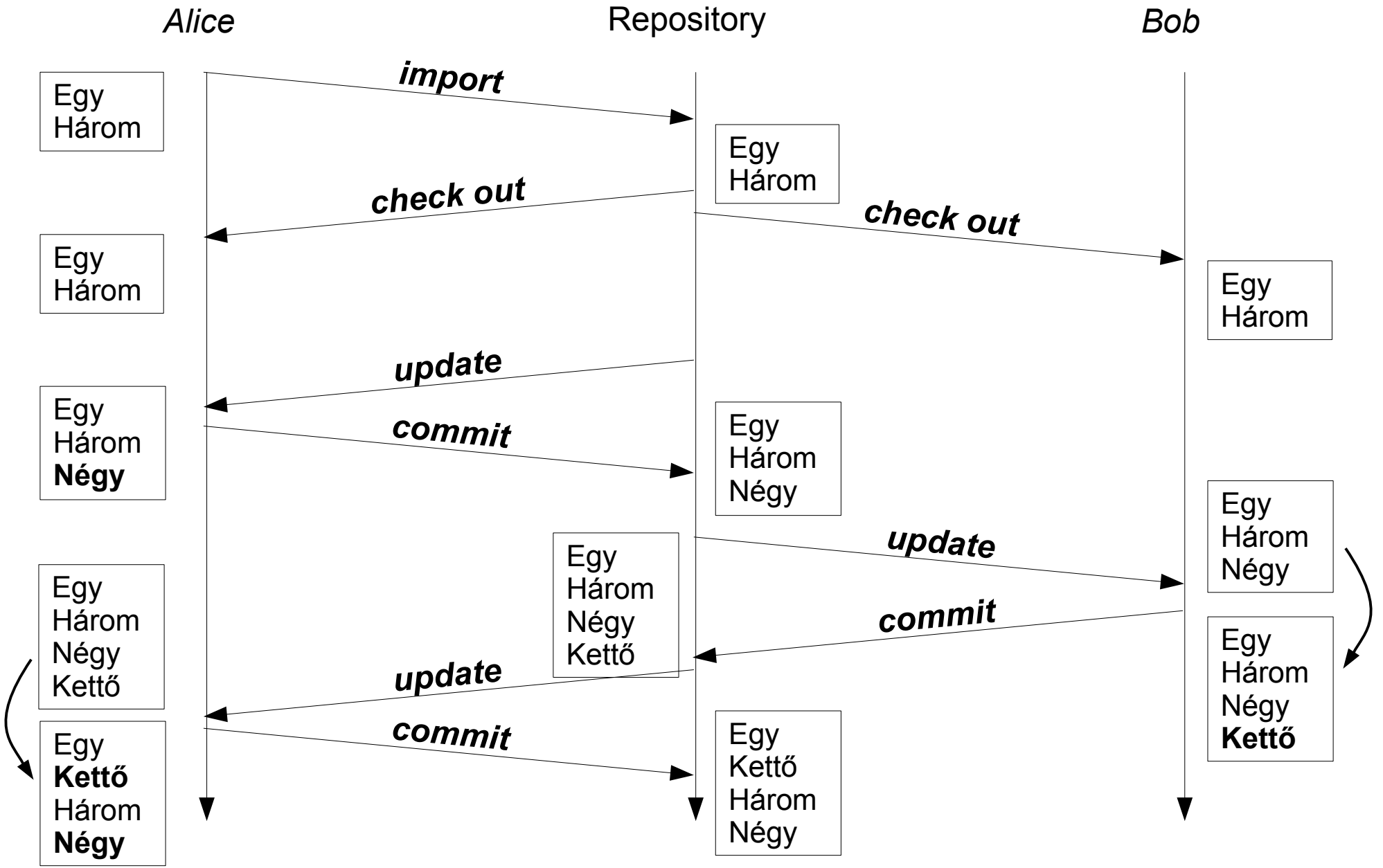
- ***Repository:*** központi szerveren őrzött mesterpéldány (master copy)
- ***Revision:*** verzió(szám)
  - vagy egyes file-okra saját verziószámmal
  - vagy az egész projektre egységes revision
- ***Head:*** legfrissebb változat a repo-ban
- ***Working copy:*** a fejlesztők lokális másolata
  - az egész projekt vagy csak egyes részei
  - tetszés szerint szerkeszthető

Verzió kontroll: modell



- ***import:*** első verzió feltöltése a repo-ba
- ***check out*** („kiszedni”): working copy létrehozása
- ***update*** („frissíteni”): legújabb verzió letöltése
- ***commit*** („komitolni”): a változások feltöltése

A verzió kontroll folyamata



Subversion (svn)

- Kezdetben vala az SCCS, RCS és a CVS
- Subversion: továbbfejlesztett CVS
  - time-machine & többszörös hozzáférés
  - központosított kliens/szerver architektúra
  - autentikáció
  - hálózati transzparencia: http, https, WebDAV, svn protokoll, stb.
  - tranzakciók (atomic commits)
  - meta-adatok verzió kontroll alatt

Subversion: alapok

- Repository-k létrehozása

```
svnadmin create <PATH>
server.org$ svnadmin create /svn
```

  - a helyi file-rendszerben, PATH alatt

- Projekt importálása

```
svn import <PATH> <URL/PROJECT>
```

  - például

```
$ svn import /home/user/MyProject \
    https://server.org/svn/MyProject \
    -m 'MyProject importálása'
```

Mi kerül verzió kontroll alá?

- Milyen fájlokat adjunk a repóhoz?
- Alapszabály: mindent, ami a szoftver fordításához kell
  - források, tesztek, build fájlok, licenc
  - dokumentáció forrása, példák, stb.
- De semmilyen „generált” fájl!
  - object fájlok, végrehajthatók
  - dokumentáció pdf-ben (elég a forrás, amiből létrehoztuk)
  - szemét (tmp fájlok, binárisok, stb.)

Subversion: alapok

- ***Check out***
  - working copy létrehozása <PATH> alatt

```
svn checkout <URL/PROJECT> <PATH>
```

  - az N. revision kiszedése a repo-ból

```
svn checkout -r N <URL/PROJECT> <PATH>
```

  - például

```
$ svn co -r 21345 \
    https://server.org/svn/MyProject \
    /home/user/devel
```

Subversion: alapok

- ***Update***
  - a <PATH> alatt levő projekt frissítése

```
svn update <PATH>
```

  - a repo címét nem kell megadni, a check out során tárolódik
  - a working copy-ból kiadva elég az `svn up`

```
$ cd /home/user/devel
$ svn update MyProject
```

```
$ cd /home/user/devel/MyProject
$ svn update
```

  - adott verzióra

```
$ svn up -r 21345
```

Subversion: alapok

- ***Commit:*** working copy-ban ejtett változások feltöltése a repo-ba

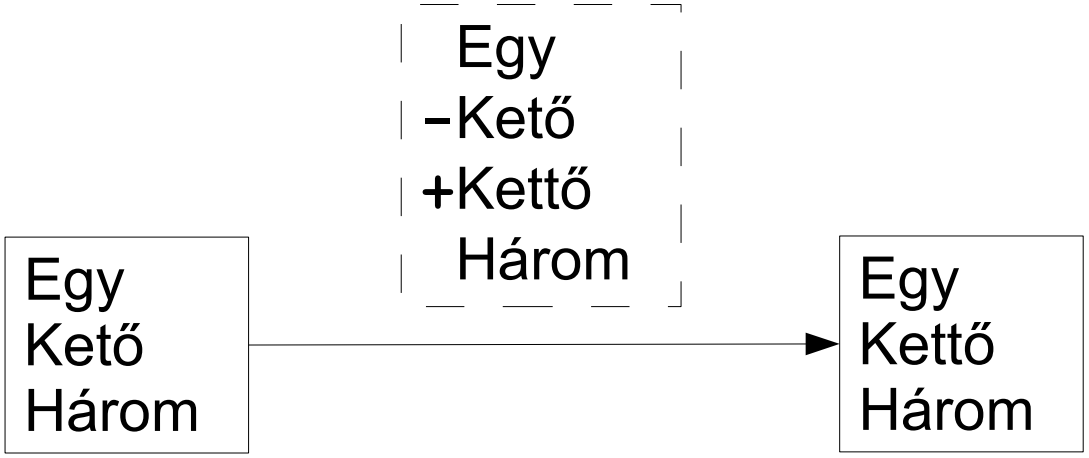
```
svn commit -m 'commit üzenet...' <PATH>
```

- Commit-ot leíró informatív üzenet: kötelező!
  - kommunikáció a többi fejlesztővel
- Érdemes kis, független változásokat commit-olni
  - pl.: egy forrás fájl és egy nem kapcsolódó doc fájl módosítása: külön commit, egyedi log

```
$ svn ci -m 'a „Kettő” hozzáadva a listához' \
    /home/user/devel/MyProject
```

Változások követése

- ***Diff:*** különbség egy fájl két verziója közt
- ***Patch:*** két revision összes file-ja közötti diff-ek összessége
- ***History:*** a projekt élettörténete = patch-ek + commit log-ok



Subversion: változások követése

- ***Diff***
  - lokális változtatások megtekintése

```
svn diff <PATH>
```

  - két revision különbsége

```
svn diff -r N1:N2 <PATH>
```

- ***Log***
  - projekt history commit log-ok sorozataként

```
svn log <PATH>
```

  - adott file története

```
svn log <PATH/FILE>
```

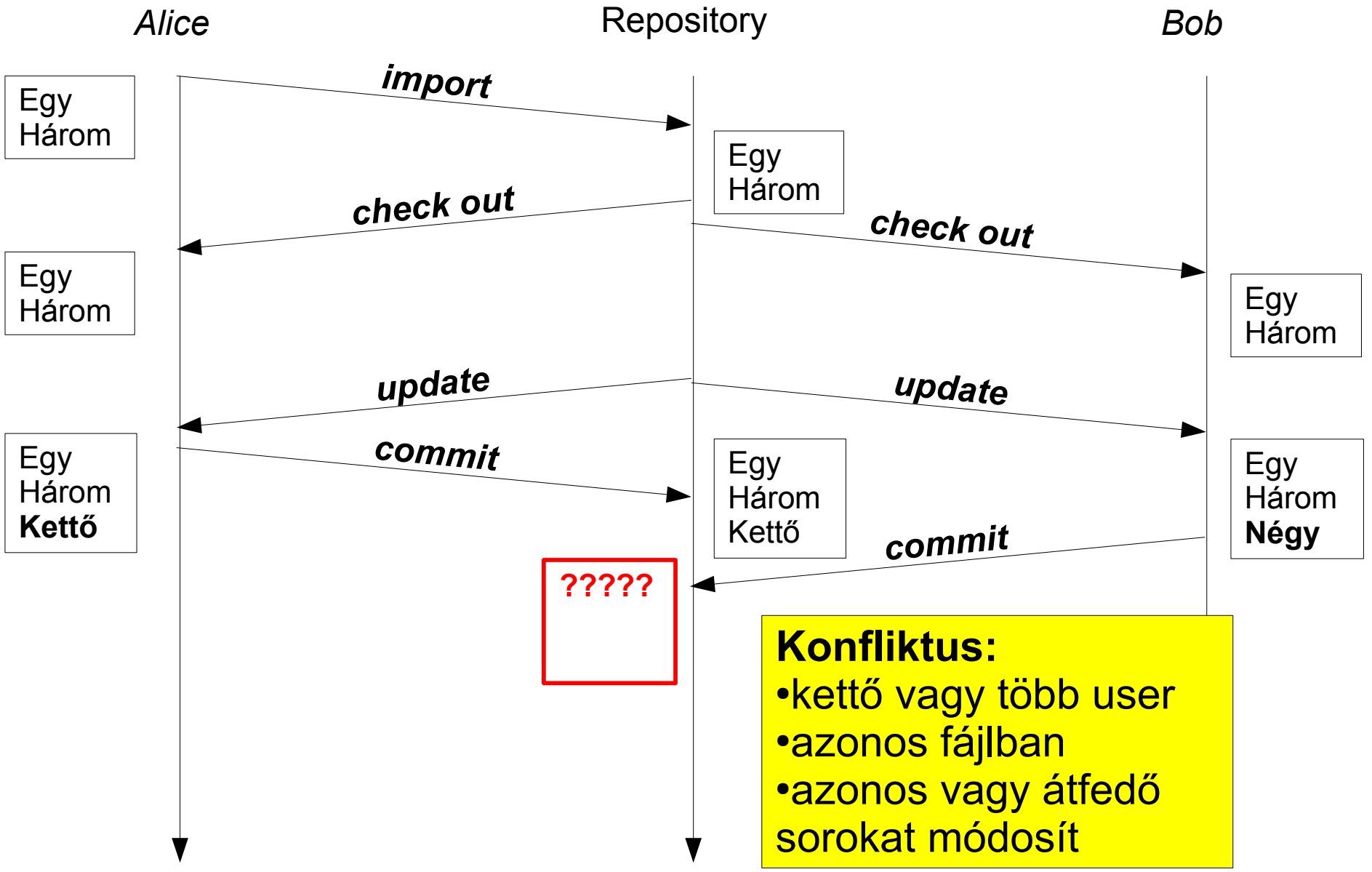
Subversion: változások követése

```
$ svn log MyProject/foo.txt
------------------------------------------------
r3 | user2 | 2008-XX-XX  | 2 lines
++ Kettő helyesen
------------------------------------------------
r2 | user1 | 2008-XX-XX  | 2 lines
++ Kettő hozzáadva
------------------------------------------------
r1 | user1 | 2008-XX-XX  | 1 line
initial import
$ svn diff -r 2:3 MyProject
Index: foo.txt
===================================================
--- foo.txt     (revision 2)
+++ foo.txt     (revision 3)
@@ -1,3 +1,3 @@
 Egy
-Kető
+Kettő
 Három
```

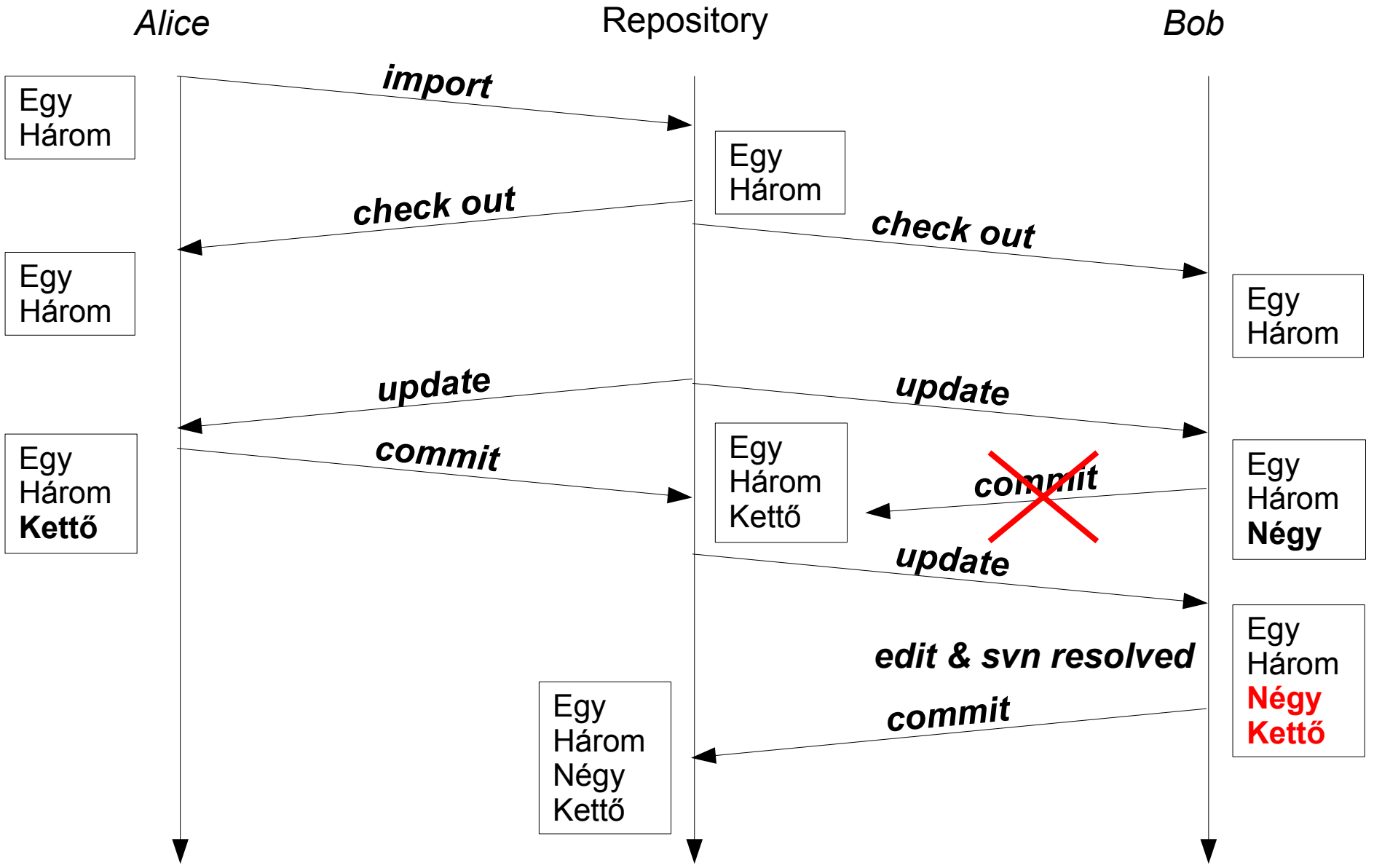
Subversion: hasznos parancsok

- ***svn status:*** a projekt file-jainak állapota a working copy-ban (módosított/változatlan, stb.)
- ***svn revert:*** lokális változtatások eldobása
- ***svn add:*** új file-ok hozzáadása
- ***svn move:*** file-ok mozgatása a projekt-en belül
- ***svn delete:*** file-ok, könyvtárak törlése
- ***svn help***

Conflict (ütközés)



Conflict resolution



Subversion: conflict resolution

```
bob$ cat MyProject/foo.txt
Egy
Három
Négy
bob$ svn commit -m 'Négy hozzáadva' MyProject
Sending        MyProject/foo.txt
svn: Commit failed (details follow):
svn: Out of date: '/MyProject/foo.txt' in
transaction '13-1'
bob$ svn update --non-interactive MyProject
C    MyProject/foo.txt
U    MyProject/README
Updated to revision 13.
```

- ***U*** (Updated) vagy ***G*** (Merged): frissítés OK
- ***C*** (conflict): ütközés, emberi beavatkozás kell
- ***A*** (Added), ***D*** (Deleted), stb.

Subversion: conflict resolution

```
$ cat MyProject/foo.txt
Egy
Három
<<<<<<< .mine
Négy
=======
Kettő
>>>>>>> .r13
$ edit MyProject/foo.txt
$ svn resolved MyProject/foo.txt
Resolved conflicted state of 'MyProject/foo.txt'
$ svn commit -m 'Négy hozzáadva' MyProject
Sending        MyProject/foo.txt
Transmitting file data .
Committed revision 14.
```

- ***svn resolved:*** ütközés kézzel feloldva
  - minden conflict-ra svn resolve

Tagging

- Hivatalos kiadások menedzselése (release engineering)
- ***Tag:*** adott revision jelölése (pl. RELEASE_2.1)
- ***Trunk:*** a fejlesztés fő ága
- A repo szervezése

```
$ tree MyProject
MyProject
|-- branches
|-- tags
`-- trunk
    |-- README
    `-- foo.txt
```

Subversion: tagging

- A trunk másolása a tags könyvtárba új néven
  - a tags könyvtárba nem commit-olunk!
- ***svn copy***
  - adott revision elmentése <TAG> címkével

```
svn copy <URL/PATH/trunk> <URL/PATH/tags/TAG>
```

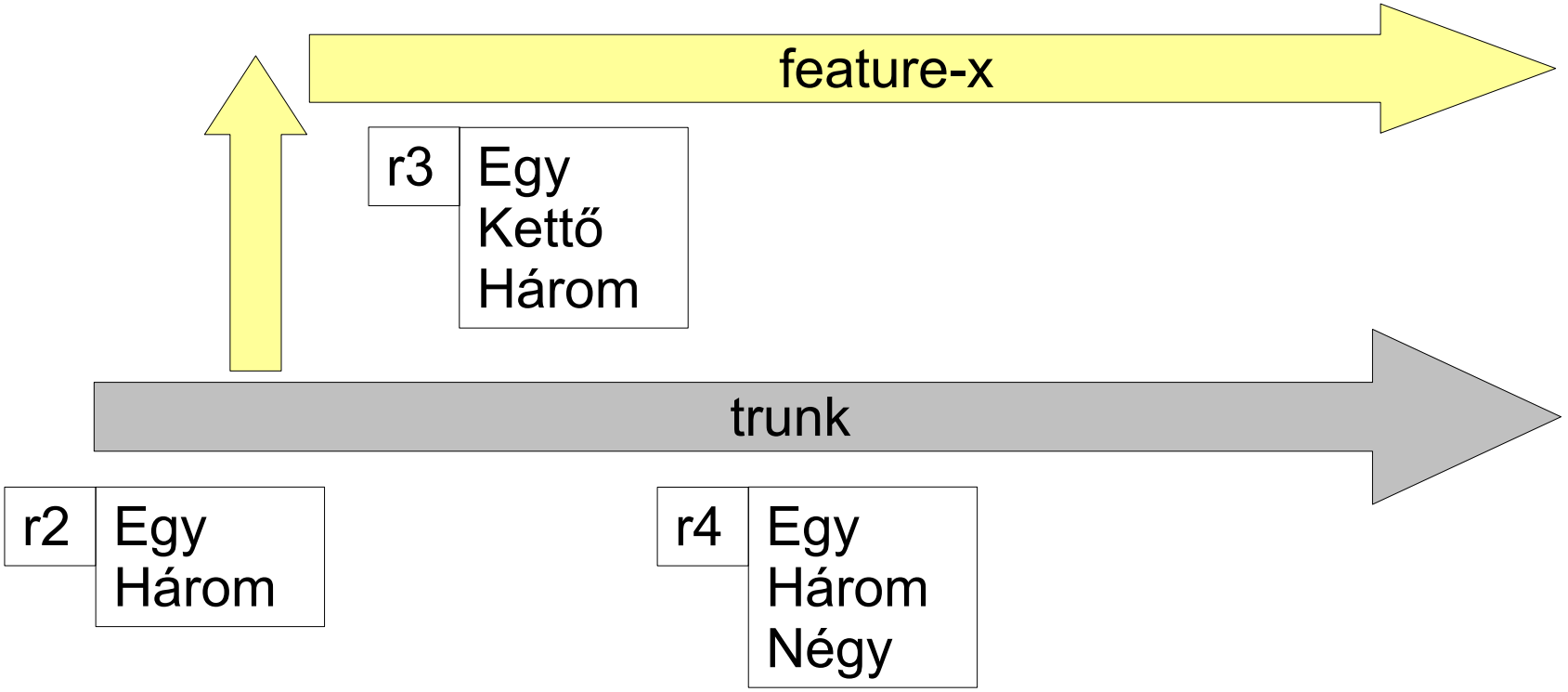
  - például

```
$ svn copy \
    https://server.org/svn/MyProject/trunk \
    https://server.org/svn/MyProject/tags/2_1\
    -m 'a 2.1-es verzió kiadása'
```

Verzió kontroll: elágaztatás

- A trunk-be csak fordítható, tesztelt kódot commit-olunk
- Nagyobb léptékű változtatások?
  - hosszabb idő: refactoring, reimplementation
  - ha folyamatosan commit-olunk: jó eséllyel hosszabb időre lefordíthatatlan a projekt
  - ha csak a tesztelt kódot commit-oljuk: elvész a verzió kontroll lényege
- ***Branch:*** alternatív fejlesztési ág
- ***Merging:*** a branch összefésülése a trunk-kel

Verzió kontroll: összefésülés



- Ha `r3` és `r4` különbségét alkalmazzuk, elvész az `r2` -> `r4` módosítás („Négy”)
- Csak az új ágban történt módosítást kell visszaírni: `r3` és `r2` különbsége

Subversion: elágaztatás

- Branching
  - a projekt elágaztatása BRANCH néven

```
svn copy <URL/PATH/trunk> \
   <URL/PATH/branches/BRANCH>
```

  - például

```
$ svn copy -m 'új feature-x ág' \
 https://server.org/svn/MyProject/trunk \
 https://server.org/svn/MyProject/branches/f-x
```

  - a working copy-t át kell állítani az új ágra

```
svn switch <URL/PATH/branches/BRANCH> <PATH>
```

```
$ svn switch \
 https://server.org/svn/MyProject/branches/f-x
```

Subversion: összefésülés

- Merging
  - elágazás rN-revíziónál, rM-nél összefésülés

```
svn merge -rN:M <URL/PATH/branches/BRANCH> \
    <PATH_OF_WORKING_COPY_OF_TRUNK>
```

  - például

```
$ cd trunk
$ svn merge -r2:3 \
 https://server.org/svn/MyProject/branches/f-x
$ svn commit -m 'merge f-x' .
```

- Probléma: nehéz többször összefésülni
  - egyes diff-ek esetleg többszörösen alkalmazódnak
  - informatív commit üzenet!

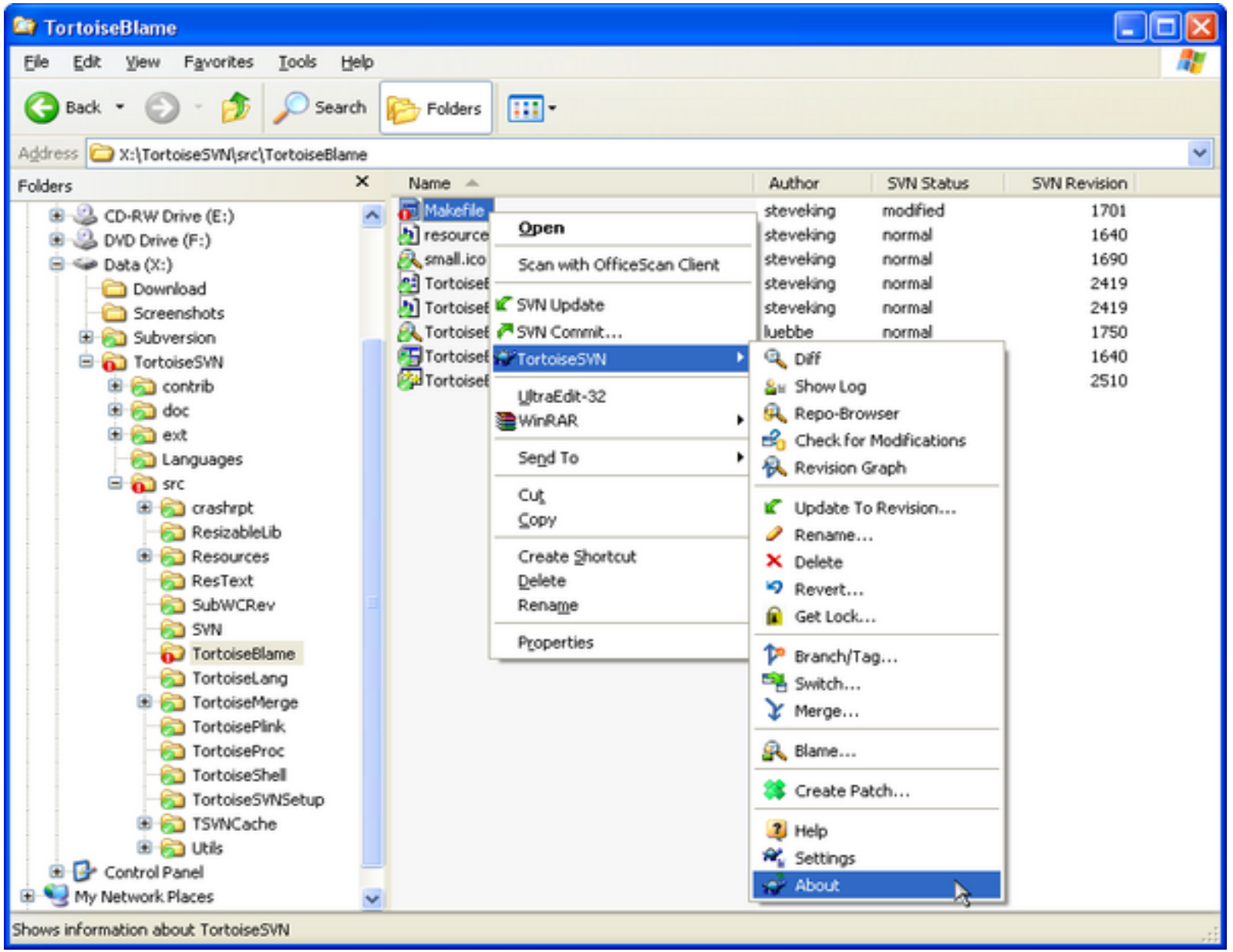
Verzió kontroll: autentikáció

- Hozzáférés
  - read-only vagy commit jog
  - korlátozható bizonyos könyvtárakra, file-okra
- Megadható, elvehető
- Publikus, read-only hozzáférés
  - anonymous svn
  - a projekt fejlődése nyilvános

Verzió kontroll: meta-adatok

- Bináris file-ok jelölése
- Sorvége konvenciók (UNIX/DOS/Mac)
- Mime-types
- Subversion:
  - meta-adatok verzió kontroll alatt
  - property: tetszőleges kulcs-érték páros
  - propget/propset/propdel
  - egyes property nevek foglaltak
  - automatikus meta-adat beállítás (bináris file-ok érzékelése, stb.)

TortoiseSVN



Központosított verzió kontroll: kritika

- 1970-es évek, big-iron (mainframe) világkép
  - kliens-szerver modell
- Linux kernel fejlesztés: hierarchikus diktatúra
  - egy ember integrál (Linus)
  - területenként karbantartók (VFS, VMS, ext3, stb.)
  - ha központosított verzió kontrollt használnának
    - az egyes területek saját branch-ben: merge???
    - a karbantartóknak kéne Linus repo-jába commit-olni
    - ehelyett Linus akarja a neki tetsző patch-eket átemelni
  - teljesen eltérő verzió kontroll modell kell