VERZIÓKEZELÉS

SÁNDOR BALÁZS
BE4GEI

A FEJLESZTÉS KIHÍVÁSAI

- Egyre hosszabb, terjedelmesebb programkódok
- Nehezen követhető változások
- Nehéz a hibakeresés
- Több fejlesztő, még nagyobb kavarodás

VERZIÓKEZELÉS, MINT MEGOLDÁS

Mi is a verziókezelés?

- Verziókezelés alatt több verzióval rendelkező adatok kezelését értjük.

A verziókezelés lényege:

- Lehetőségünk van kódverziókat számon tartani, visszanézni
- Lehetőségünk van kódrészeket összeollózni több verzióból

Miért jó még?

- Segít megkülönböztetni az egyes verziókat
- A csapatmunkából adódó, egymást átfedő fejlesztések értelmezhetőségét segíti

FOGALMAK I.

- **Baseline** – egy fájl/dokumentum jóváhagyott verziója
- **Branch (ág)** – egy fájlnak több verziója is fejlődhet egyszerre
- **Check-out** – lokális másolat
- **Check-in, Commit vagy Submit** – lokális másolat feltöltése
- **Conflict** – két egymást „kiütő” módosítás
- **Change** – változtatás
- **Change list** – változások listája commit-on belül
- **Export** – check-out metaadatok nélkül
- **Head** – legutóbbi commit
- **Import** – lokális adatok felmásolása és verziókontroll alá helyezése
- **Mainline** – Egy branch (vagy a trunc) fő ága

FOGALMAK II.

- **Merge** – két változás összefűzése
- **Repository** – a verziók tárolóhelye
- **Reverse integration** – egyes ágak összefűzése
- **Revision** - verzió
- **Tag, label vagy címke**
- **Trunk –** a fejlesztés egy olyan vonala, ami nem branch
- **Resolve** – változási konfliktusok feloldásának művelete
- **Update** – a legutolsó szinkronizáció óta történt változások frissítése
- **Working copy** – munkamásolat, lokális gépen

VERZIÓKÖVETŐ RENDSZEREK

Több féle csoportosítás létezik:

- Központosított / elosztott
- Nyitott / zárt

KÖZPONTOSÍTOTT VERZIÓKEZELŐ RENDSZEREK

A hagyományos verziókezelők központosított modellel dolgoznak, ahol minden verziókezelési művelet egy közösen használt szerveren történik.

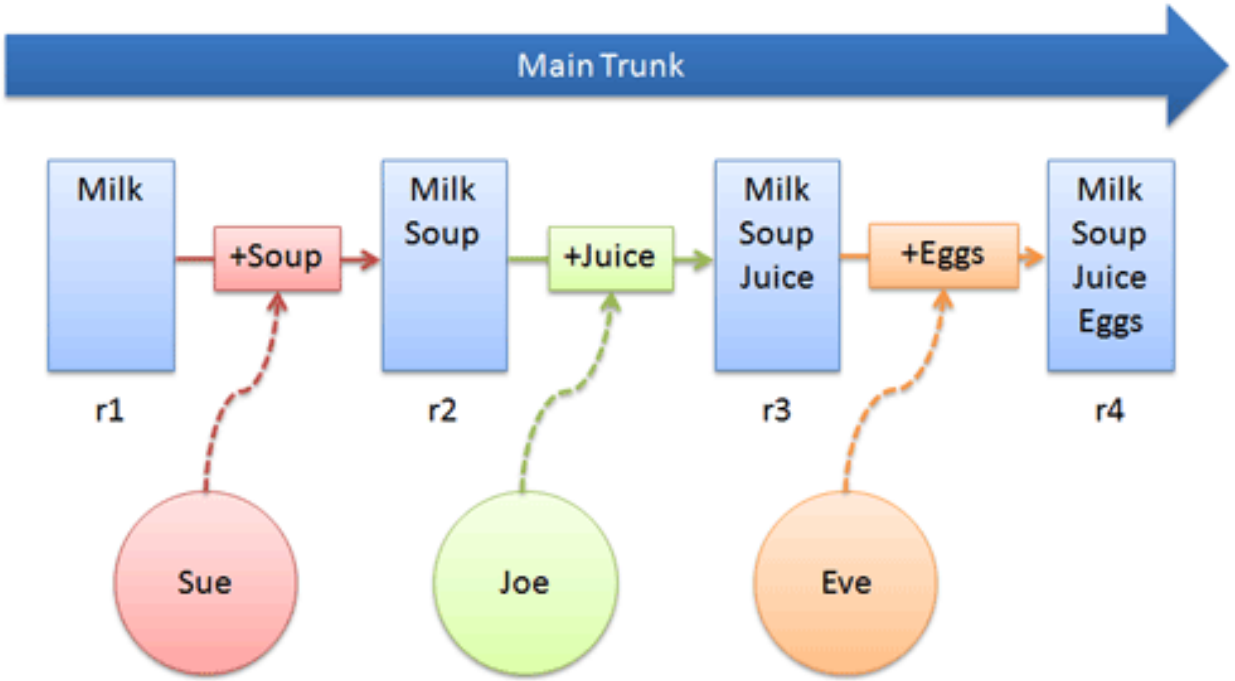
Konkurens használat kiküszöbölésének két módja:

- Zárolás (lock) - megtiltjuk a konkurens hozzáférést

- Összefésülés (merge) - a saját változtatását elsőként becsekkelő felhasználó mindenképpen sikerrel fog járni. Az összefésülés lehet automatikus vagy kézi.

Ilyenek például: SVN, CVS, Subversion

Centralized VCS



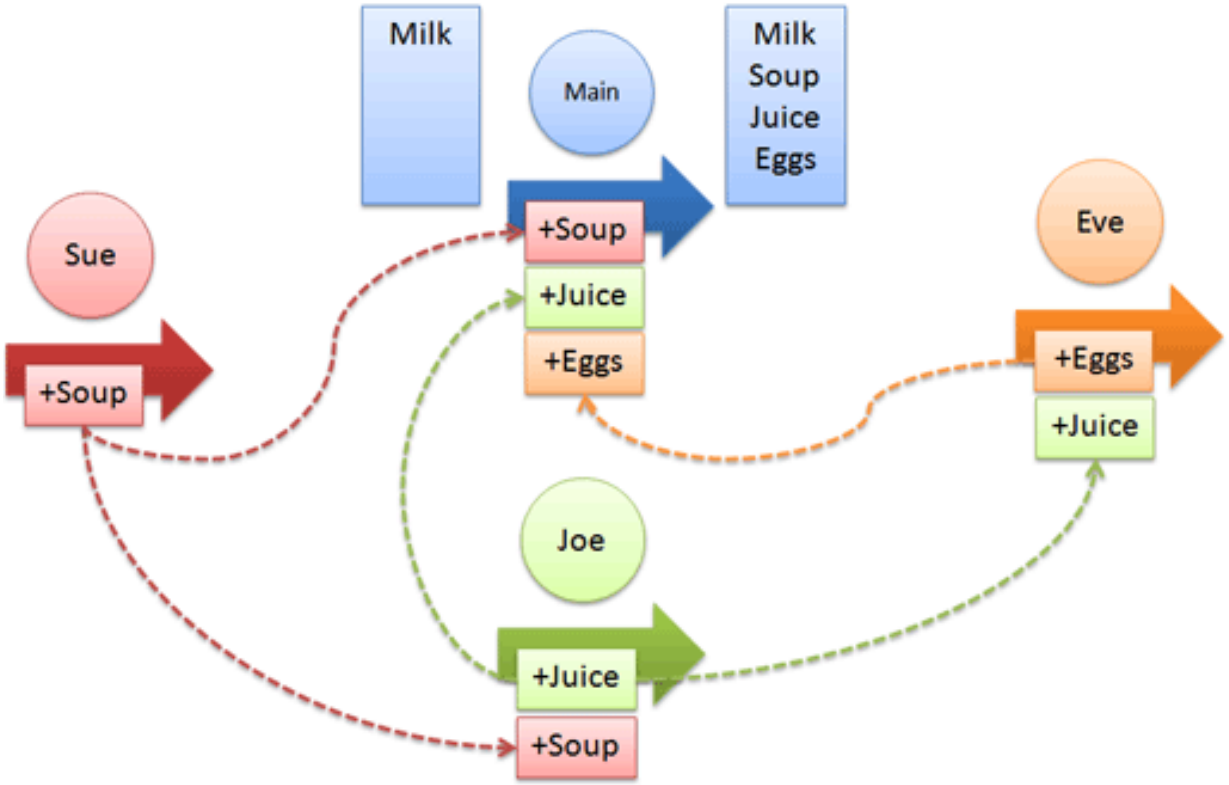
ELOSZTOTT VERZIÓKEZELŐ RENDSZEREK

Az elosztott verziókezelők decentralizált rendszerek. Itt egy központi tároló helyett minden felhasználó gépe egy-egy külön tárolóként jelenik meg. A szinkronizáció az egyes gépek között küldött patch-ek által valósul meg. Ez a megközelítés jelentős változásokat okoz:

- Nincs nagy központi adatbázis, csak munkamásolatok vannak.
- A gyakori műveletek, mint a becsekkelés, verziótörténet böngészés és a változtatások visszaállítása gyorsak, mert nem kell központi szerverrel kommunikálni.
- Minden munkamásolat egy-egy backup, ami természetes védelmet ad az adatvesztés ellen.

Ilyenek például: Aegis, Git, SVK

Distributed VCS



NYITOTT VERZIÓKEZELŐ RENDSZEREK

Két fajta elosztott verziókezelő létezik, a nyitott és a zárt. A nyitott rendszereket inkább nyílt forráskódú termékeknél használják, a zártakat inkább a nem nyilvános forráskodú termékeknél.

A nyitott, elosztott verziókezelők támogatják különböző ágak létezését, és erősen függenek a fent tárgyalt összefésülés (merge) művelettől. Általános jellemzőik a következők:

- Minden munkamásolat gyakorlatilag egy ág.
- Minden ág egy-egy munkamásolatként implementálódik. Az ágak összefésülés patch-ek küldözgetésével történik.
- Lehet válogatni az egyes változtatások között, nem kell feltétlenül minden változtatást letölteni.
- Új tagok bármikor csatlakozhatnak a rendszerhez, nincs szükség szerveroldali regisztrációra.

ZÁRT VERZIÓKEZELŐ RENDSZEREK

- A zárt, eloszott verziókezelők adatbázis replikáción alapulnak
- Csak egy baseline van
- Minden becheck-olt változás ebbe a baseline-ba kerül bele.

INTEGRÁCIÓ

A fejlettebb verziókezelők további lehetőségeket is kínálnak, melyek lehetővé teszik az integrációt más eszközökkel:

- Különböző IDE-khez (Eclipse és Visual Studio) gyakran letölthetőek különböző verziókezelői pluginok.

- A NetBeans IDE-t beépített verziókezelővel szállítják.

A SUBVERSION ÉS A TORTOISE SVN

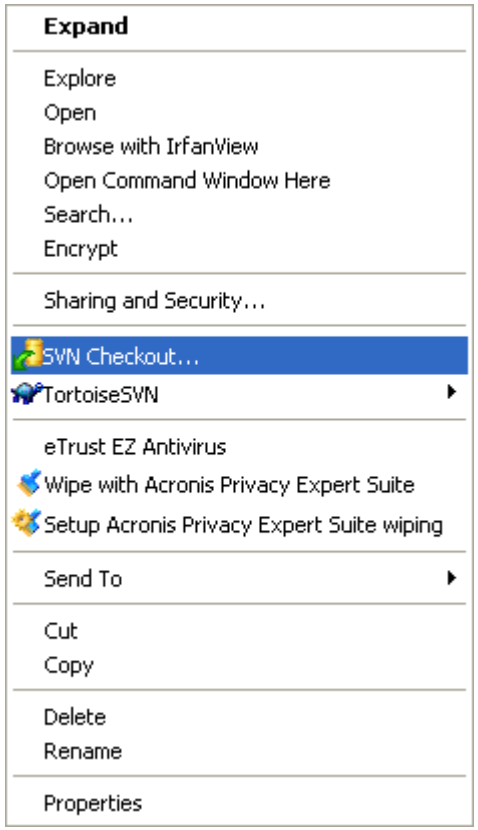
Subversion

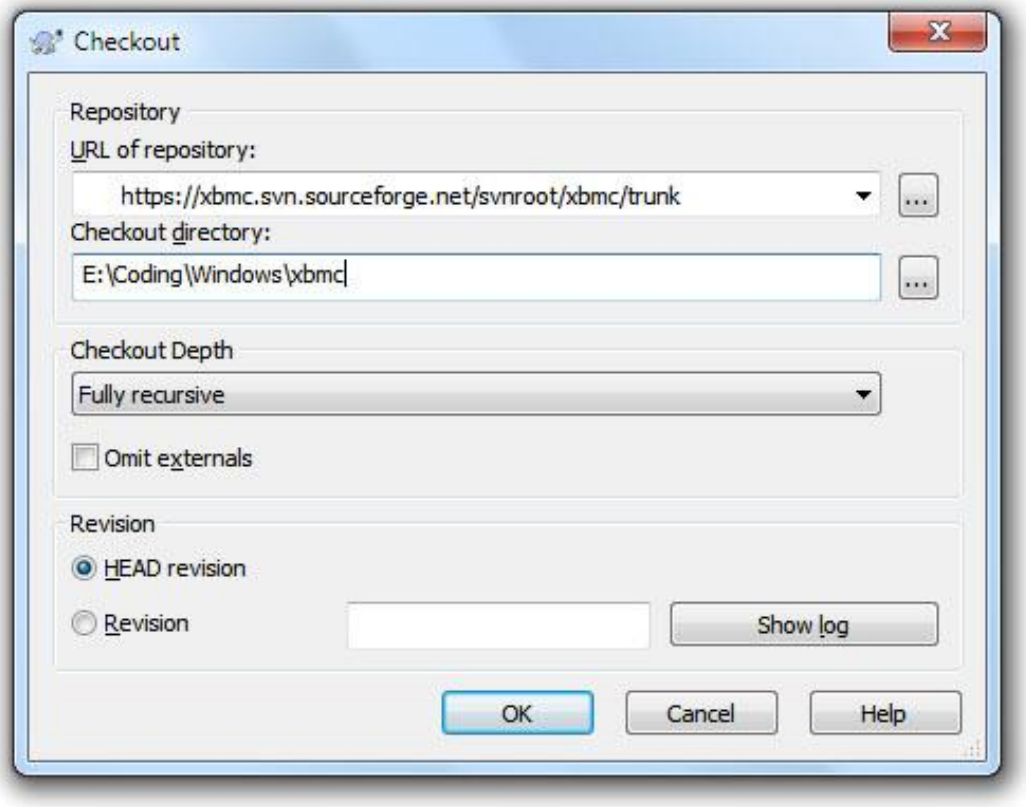
- Ingyenes, központosított
- Sok nyílt forrású projektben használják is, mint például: Apache Softwar Foundation,KDE, GNOME, FreeBSD
- CVS újradolgozása, hibáinak javítása

TortoiseSVN

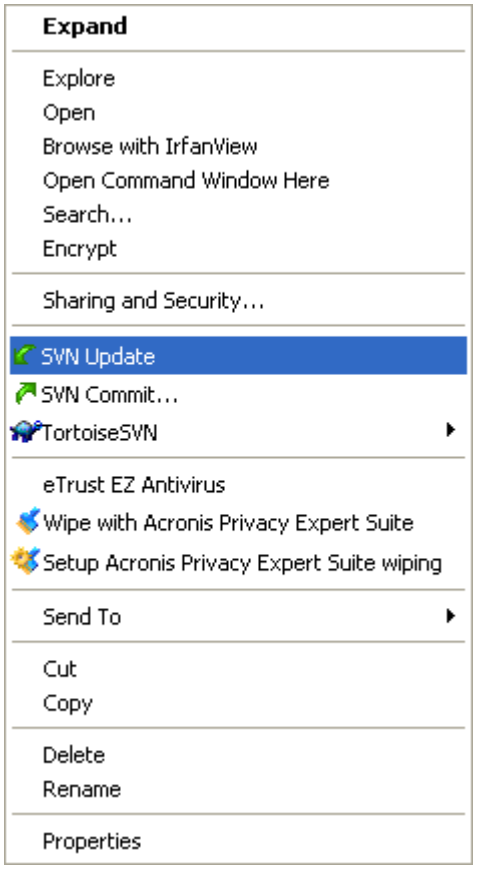
- Windows Explorer/Shell SVN integrációs eszköz
- Magyarul: a Windows directory-rendszerébe épül

CHECKOUT

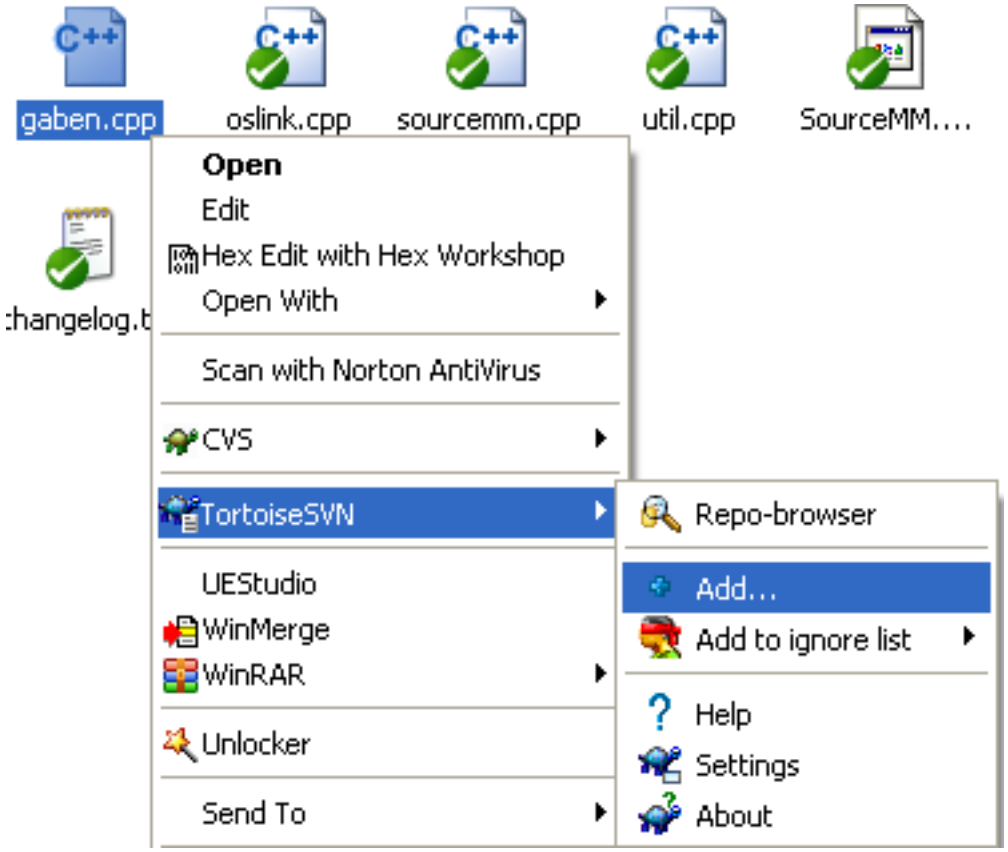

Expand
Explore
Open
Browse with IrfanView
Open Command Window Here
Search...
Encrypt
Sharing and Security...
SVN Checkout...
TortoiseSVN
eTrust EZ Antivirus
Wipe with Acronis Privacy Expert Suite
Setup Acronis Privacy Expert Suite wiping
Send To
Cut
Copy
Delete
Rename
Properties


Checkout
Repository
URL of repository:
https://xbmc.svn.sourceforge.net/svnroot/xbmc/trunk
Checkout directory:
E:\Coding\Windows\xbmc
Checkout Depth
Fully recursive
Omit externals
Revision
HEAD revision
Revision
Show log
OK
Cancel
Help

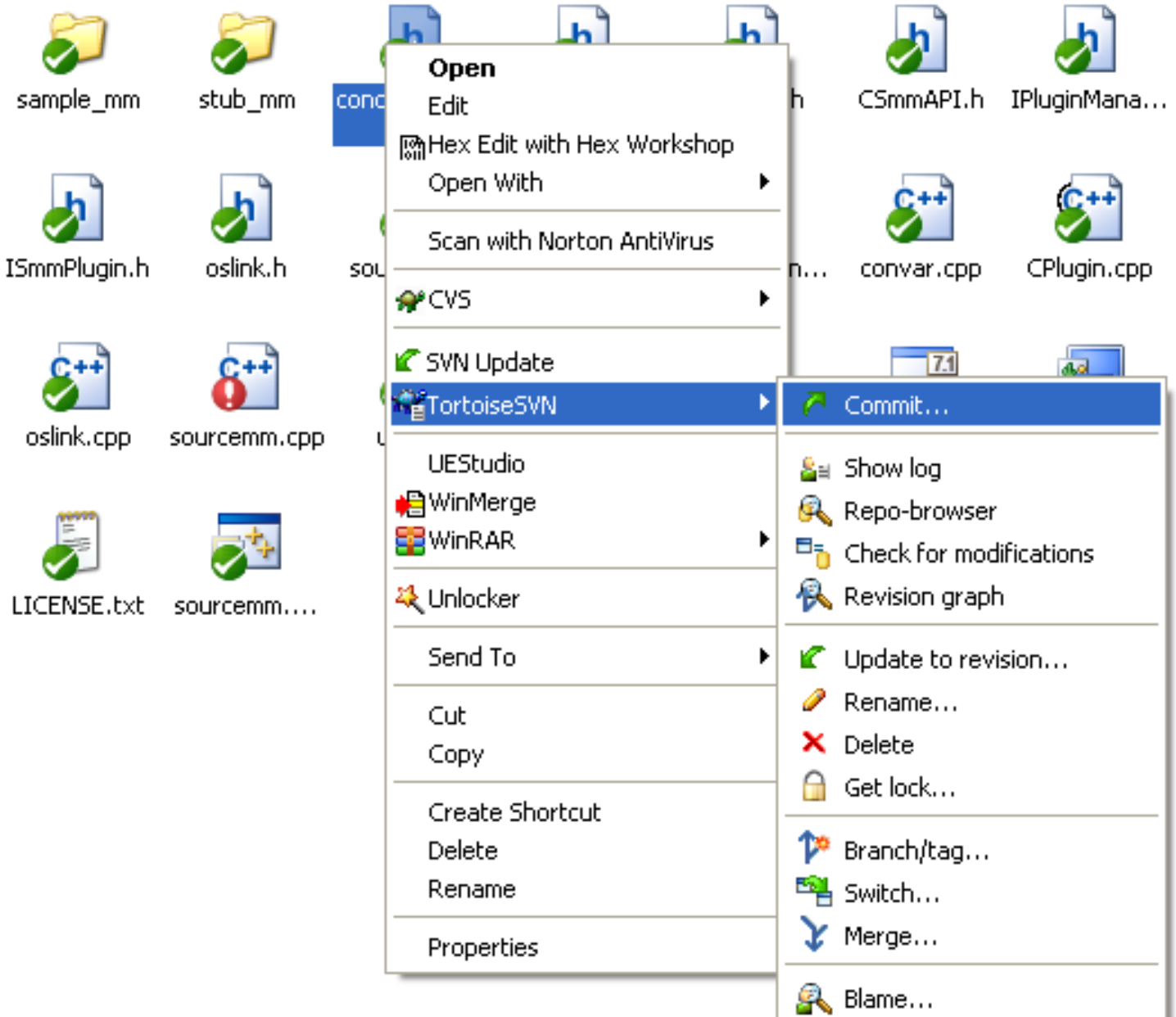
UPDATE



ÉS ADD



ÉS A TÖBBI...


sample_mm
stub_mm
CSmmAPI.h
IPluginMana...
ISmmPlugin.h
oslink.h
convar.cpp
CPlugin.cpp
oslink.cpp
sourcemm.cpp
LICENSE.txt
sourcemm....
Open
Edit
Hex Edit with Hex Workshop
Open With
Scan with Norton AntiVirus
CVS
SVN Update
TortoiseSVN
UEStudio
WinMerge
WinRAR
Unlocker
Send To
Cut
Copy
Create Shortcut
Delete
Rename
Properties
Commit...
Show log
Repo-browser
Check for modifications
Revision graph
Update to revision...
Rename...
Delete
Get lock...
Branch/tag...
Switch...
Merge...
Blame...

KÖSZÖNÖM A FIGYELMET!