Verziókezelés alapjai

Bevezetés a Matlab programozásába

PPKE ITK MI-BSc képzés

20012. 02. 17.

Követelmények ismertetése

A követelmények a tárgy WIKI oldalán találhatjuk meg.
A wiki oldal címe:

http://wiki.itk.ppke.hu

Felíratkozás a matlab listára

Link a wiki-n

Verziókezelés alapjai

A verziókezelő feladatai:

- a kezelésébe adott fájlok, és könyvtárak időbeli változását tárolja.
- Mindezt visszakereshető és összevethető formában.

Célja, hogy több felhasználó dolgozhasson ugyanazon a projekten egyszerre, mindezt adatvesztés nélkül.
Már egy felhasználónál is megéri, hiszen a saját munkájának alakulását tudja követni, régebbi verziókat előhívni.

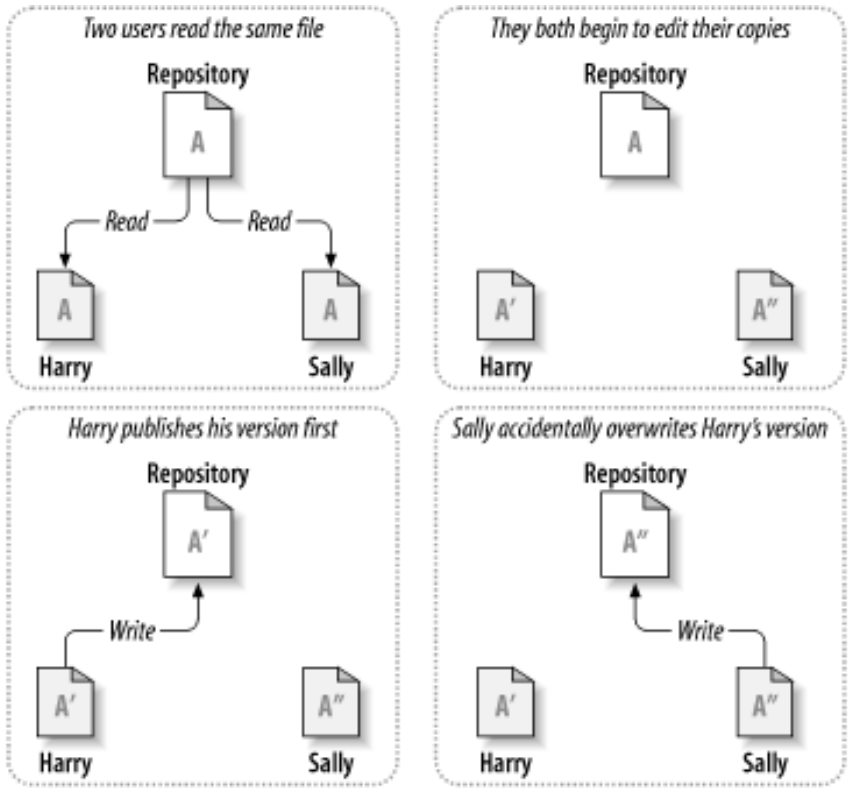
Felhasználási területek

Szoftverfejlesztési projektek elengedhetetlen kelléke, csoportmunka alapját jelenti. Ugyanakkor még jelentős területek:

- **Távoli adattárolás** Az egyes állományok, amit a munkám során keletkeznek, védett helyre kerülnek. Azokat bárhonnan elérem, régebbi változásokat nyomon tudom követni. (Ma már inkább a Dropbox és az Amazon Cloud veszik át ennek helyét)
- **Rendszerfelügyelet** Konfigurációs fájlok, domain adatok, linux alatt a etc könyvtár.
- **Webes tartalom közzététele** Weboldalak szöveges tartalmának kezelése, változtatás után lehetséges szkriptek segítségével a weboldalat azonnal újra generálni.
- **Dokumentumtárolás** Házi feladat projektek, szakdolgozatok, tudományos cikkek, fordítások és könyvek készítésénél.

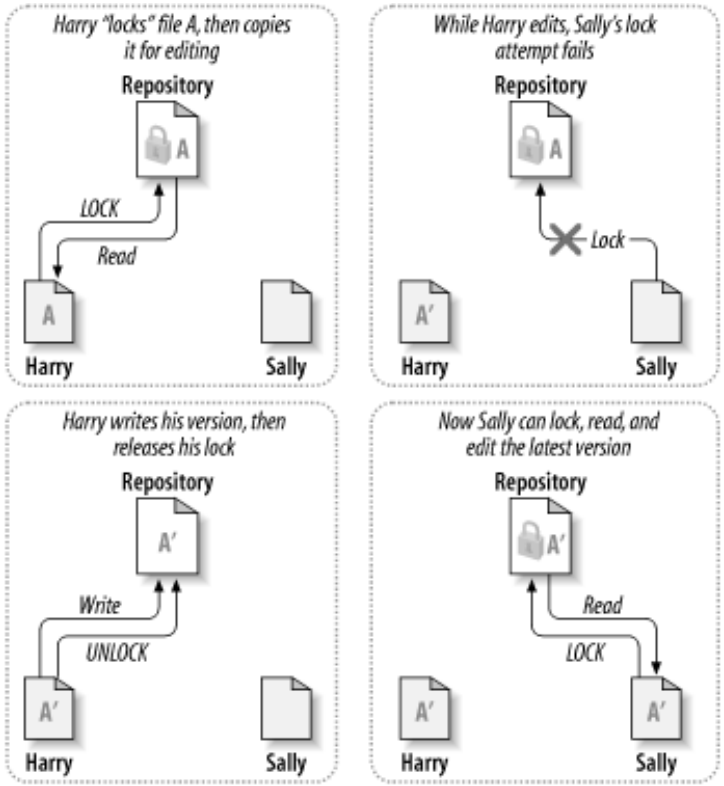
Verziókezelés modelljei

Minden verziókezelő rendszer egyik alapvető kérdése, hogy hogyan oldja meg a konzisztens adat kezelés problémáját. Azaz, hogyan akadályozza meg, hogy az egyik felhasználó által végzet módosítások tönkretegyék egy másik felhasználó munkáját. Hogyan fordulhat egy ilyen szituáció elő?



Lock-Modify-Unlock modell

- Egyszerre csak egy ember szerkesztheti az adott fájlt.
- A többiek csak olvashatják addig a fájlt, várniuk kell még a másik felhasználó feloldja a zárolást.



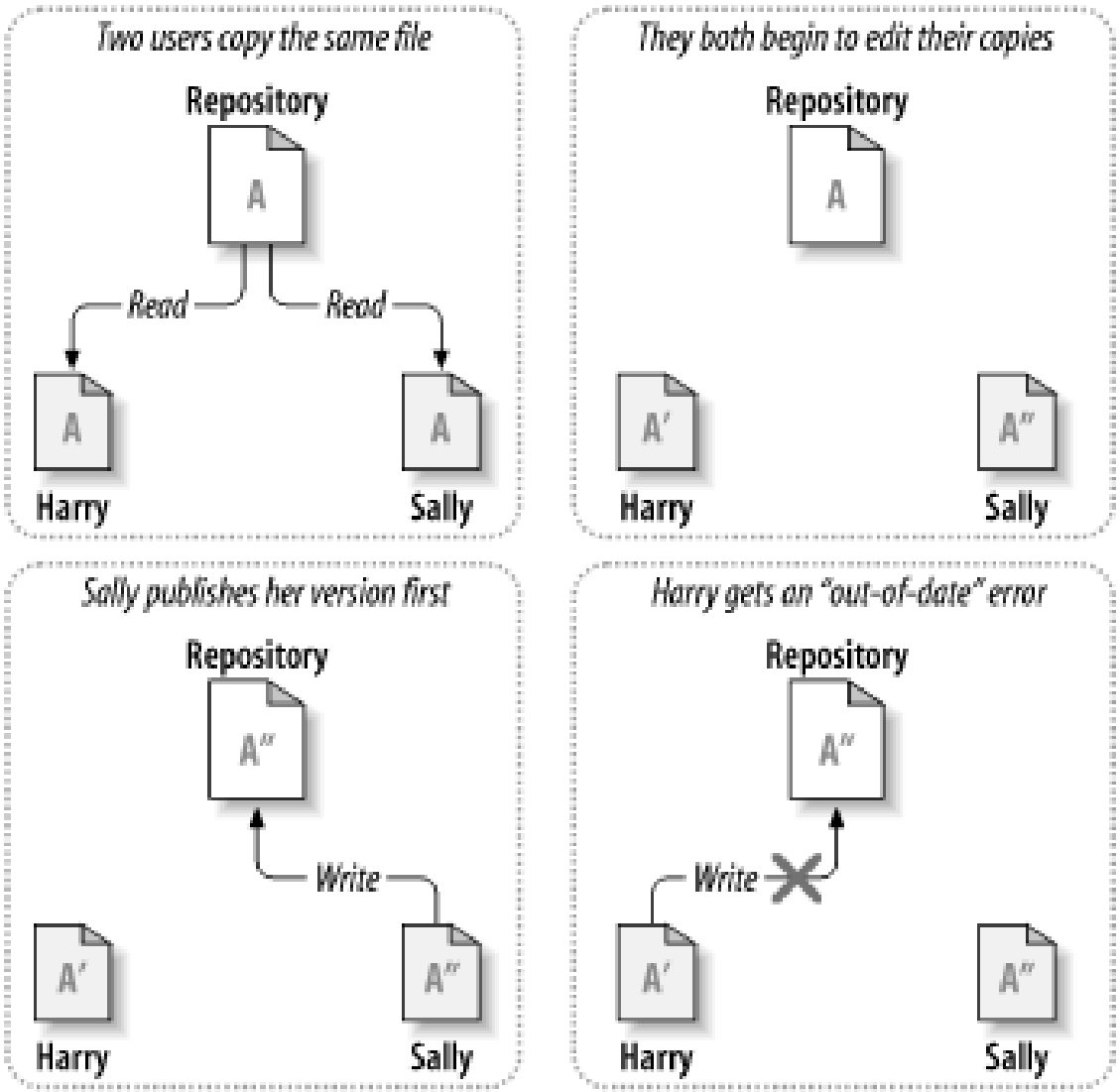
Lock-Modify-Unlock modell - hátrányok

- Adminisztrációs problémák. Mi történik, ha Harry zárolja a fájlt, és elfelejti feloldani a munka végeztével? Sally addig vár és nem tud dolgozni. Mi van, ha Harry egyszerűen szabira megy és elfelejti feloldani a fájlt?
- Előfordulhat, hogy Harry a fájl elején szeretne dolgozni, még Sally a fájl végén szeretne módosítások végezni. Ekkor csak egymás után tudják a módosításokat elvégezni.
- Az adat integritás hibás feltételezése. Ha bár egy fájlt zárolok és ezzel megőrzőm annak a fájlnak a konzisztenciáját, lehetséges, hogy ez a fájl egy másikkal együtt tud csak működni és azt épp más szerkeszti. A végeredmény az lehet, hogy a két fájl együttesen már nem add működőképes egységet.

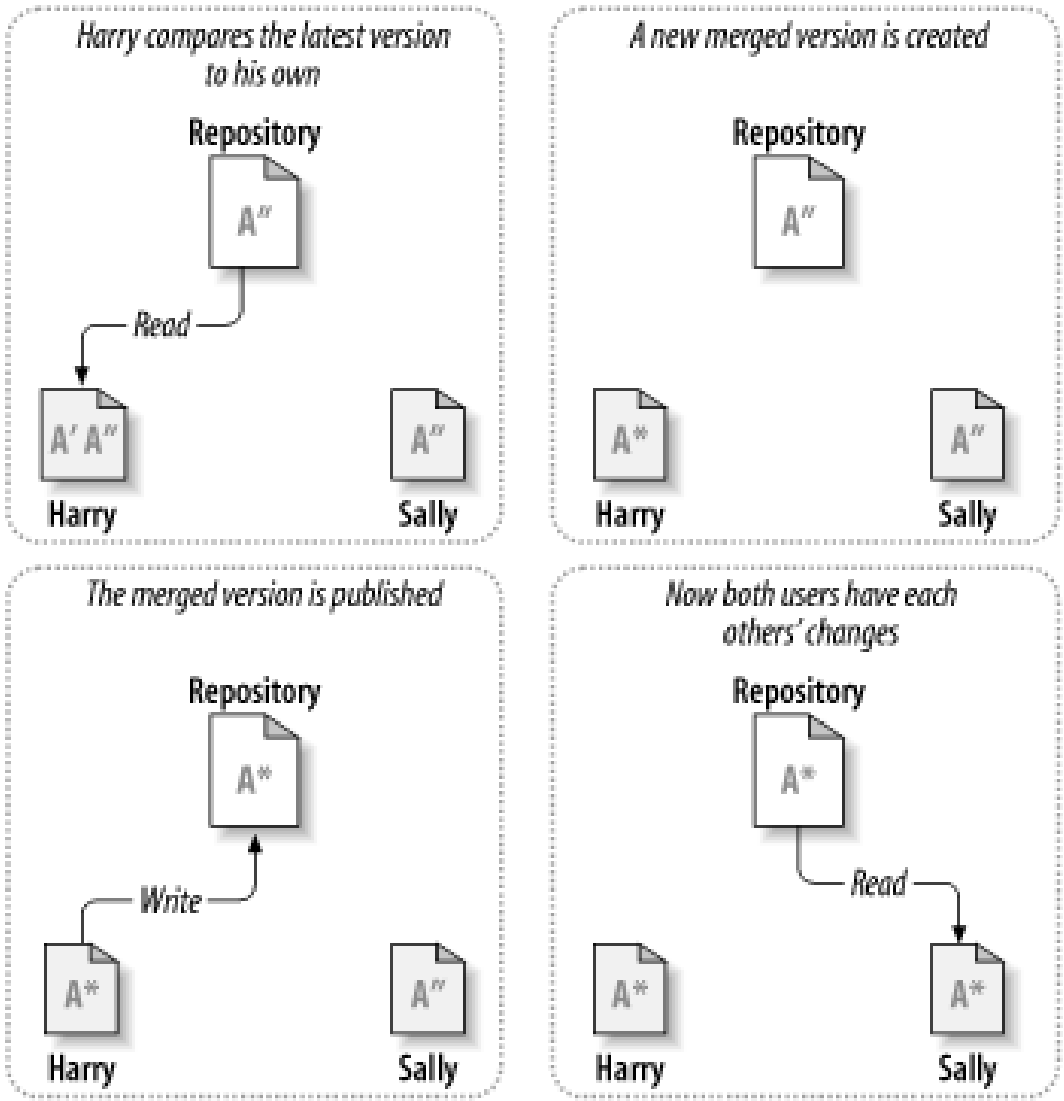
Copy-Modify-Merge modell

- Minden felhasználó saját **munkapéldánnyal** rendelkezik, ami a repo lokális másolata.
- Minden felhasználó a saját példányán végzi el a módosításokat. A munka végeztével ezeket a módosítások beküldi a repoba, ahol a változások **rögzítésre** kerülnek.
- A többi felhasználó a saját munkapéldányának **frissítése** után értesül a változásokról.
- Átfedő munkák esetén, ha ugyanazt a fájlt ugyanott szerkesztettük, **konfliktus** fordul elő. A konfliktusokat a rendszer nem tudja feloldani, hiszen nem ismeri a kezelt tartalom jelentését. Ezért a felhasználónak kell értelmezni a problémát és feloldani a konfliktust.

Copy-Modify-Merge modell folyt.



Copy-Modify-Merge modell folyt.



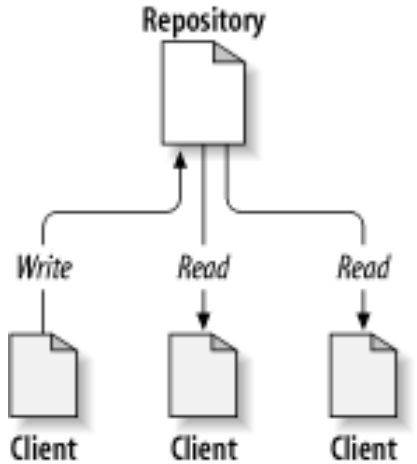
Copy-Modify-Merge modell előnyök

- Az egyes felhasználók nem zavarják egymás munkáját.
- Egy azon fájl, ugyanazon részét ritkán szerkesztik, a konfliktusok ritkák.
- Van értelme még a zárolásnak? Igen, van! A fentieket alapvetően szöveges állományokra érvényes. Bináris állományok esetén a konfliktust nem lehet kézzel megoldani, vagy nagyon időigényes. Ezért például hangfájloknál és képeknél vagy más bináris állományoknál zárolással biztosak lehetünk abban, hogy nem fogják közben felülírni a munkánkat.

Repok és munkapéldányok

Repository

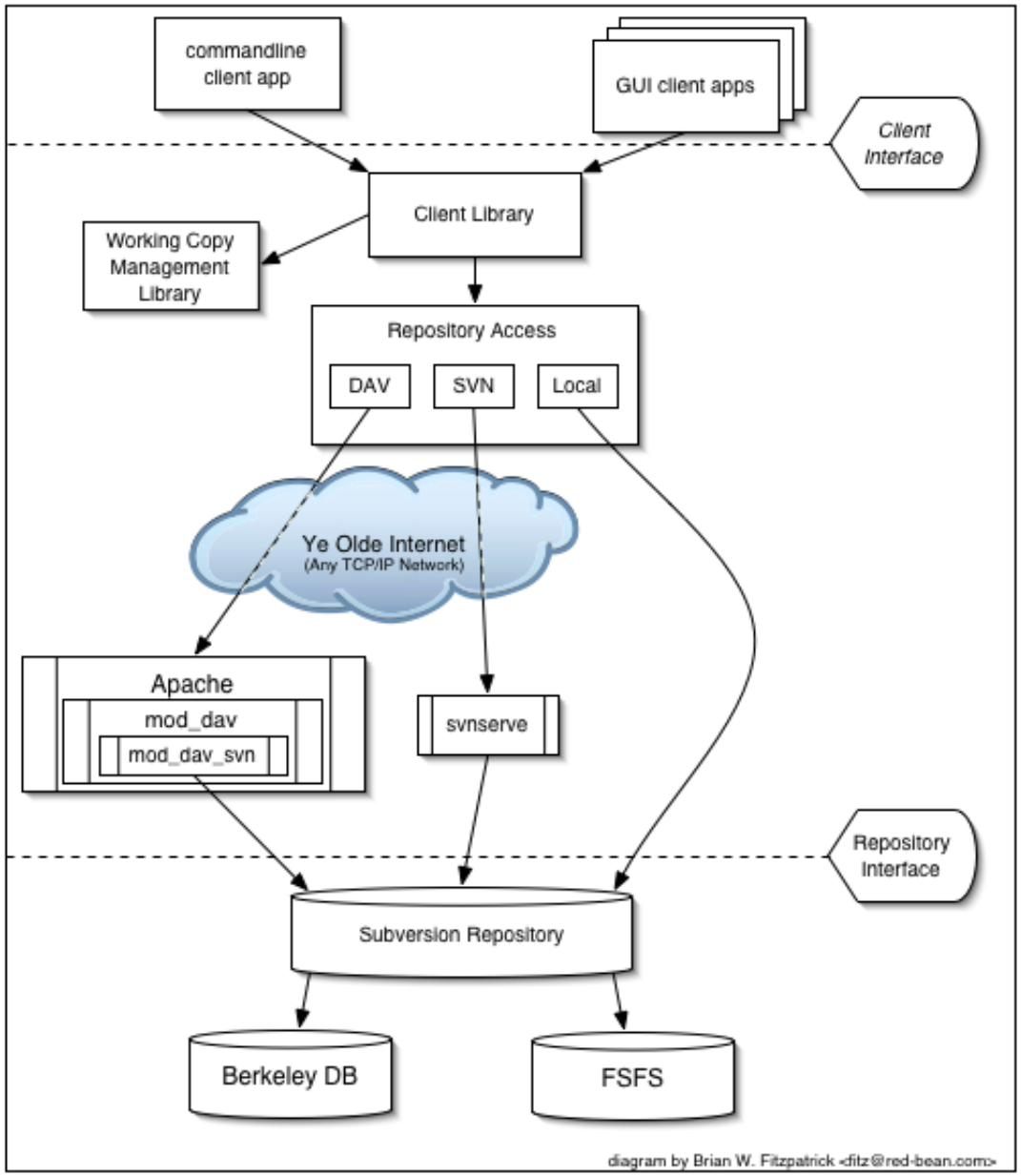
A rendszer központi részét a "repository" képzi, röviden csak repo. Célja, hogy az rendszerbe beküldött fájlokat és verziókat tárolja, és a kliensek számára elérhetővé tegye. A kliensek által beküldött változásokat integrálja repository tartalmával.



Munkapéldány

A munkapéldány a repository másolata, amellyel minden felhasználó külön-külön rendelkezik. A felhasználók nem láthatják egymás munkapéldányát, viszont a változásokat igen. A repo korlátlan számú klienssel tud kapcsolatot tartani kérésre információt szolgáltatni.

A rendszer felépítése



Repository

- A verziókezelő rendszerbe való írás esetén információt adunk át a többi részvevőnek, olvasás esetén információt gyűjtünk a többi részvevő munkájáról.
- Az eddig elmondottak tökéletesen leírják a fájlszerver működését. Berakok egy fájlt, a többiek azt kiveszik. A verziókezelés annyival tud többet, hogy **"emlékszik"** minden egyes változtatásra, amit a rá bízott könyvtárakon és fájlokon végeztünk.
- Az olvasás ebben az esetben azt jelenti, hogy a repo tetejét, azaz a legutolsó verziótját látjuk a kezelésre bízott fájl és könyvtárszerkezetnek. Természetesen lehetőség van a korábbi verziók megtekintésére is, valamint az egyes verziók összehasonlítására is.

Ezzel lehetőség nyílik a következő kérdések megválaszolására:

- Mi volt a tárolt adatok állapota 2 nappal ezelőtt?
- Hát ezt meg ki csinálta ide ?! :)

Repository hozzáférési protokollok

Hogyan tudunk hozzáférni a helyi, esetleg távoli reponkhoz? Az svn számos lehetőséget ismer, az egyetemen az svn+ssh használható távoli eléréshez.

Schema	Access method
`file:///`	Direct repository access (on local disk)
`http://`	Access via WebDAV protocol to Subversion-aware Apache server
`https://`	Same as `http://`, but with SSL encryption.
`svn://`	Access via custom protocol to an `svnserve` server
`svn+ssh://`	Same as `svn://`, but through an SSH tunnel.

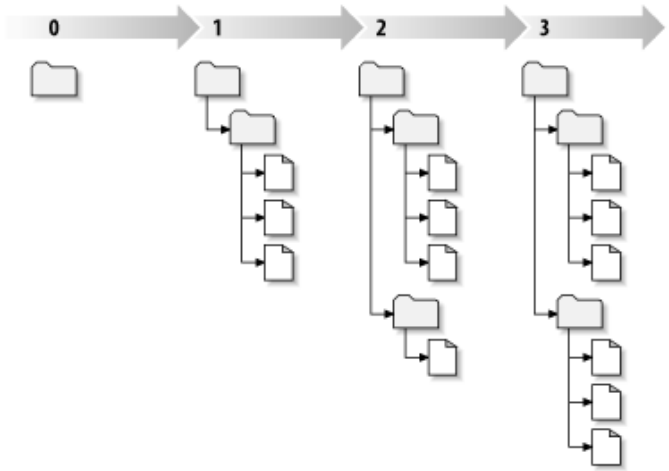
A repo lehet távoló gépen és a lokálisgépen egyaránt.

Munkapéldány

- Megszokott faszerkezetű fájlrendszer, szerkeszthető. Ha forráskód, akkor programmá fordítható.
- Teljesen privát munkaterület, a verziókezelő rendszer más felhasználói nem láthatják az én munkám, amíg azt nem "publikálom a rendszerbe"
- Ha frissítek akkor jelennek meg mások változtatásai az én munkapéldányomban.
- Egy adminisztratív könyvtár is létrejön **.svn**, ennek segítségével állapítja meg a rendszer, hogy mi változott a repohoz képest.

Verziók (Revisions)

A továbbiakban a SubVersioN (SVN) nevű programról fogunk beszélni, mert jelenleg ez a legelterjedtebb és az "átlagos" felhasználásra is kiválóan alkalmas.
Az SVN globális verziószámozást használ, azaz nem fájloknak van egyedi verziója, hanem az egész reponak egy közös. Miután a repo elfogad egy commitot, azután létrehozza a fájlszerkezet új állapotát, amit revíziónak hívunk.
Tehát minden revíziószám generálódásnál létrejön egy "fénykép" a reporól, ami az N-dik commit után keletkezik. A revíziószám természetes egész szám, csak növekedni tud.



A Munkapéldány szerkezetének megváltoztatása

Miután a munkapéldányunk egy másolata a repository-nak, ezért szabadon végezhetünk rajta műveleteket az operációs rendszer szolgáltatásain keresztül. Tehát másolhatjuk, törölhetjük, áthelyezhetjük az egyes állományokat.
DE a verziókezelő ezeket a műveleteket nem látja, és nem fogja tudni követni a módosításokat. Az esetleges módosításokat (másolás, áthelyezés, törlés) mindig a verziókezelő rendszer segítségével a kliens oldalon **kell** elvégeznünk!

Az SVN rendszer szolgáltatásai strukturális átalakításra

- svn copy, svn move, svn delete

Használatukra később láthatunk példát.

Alapvető munkaciklus

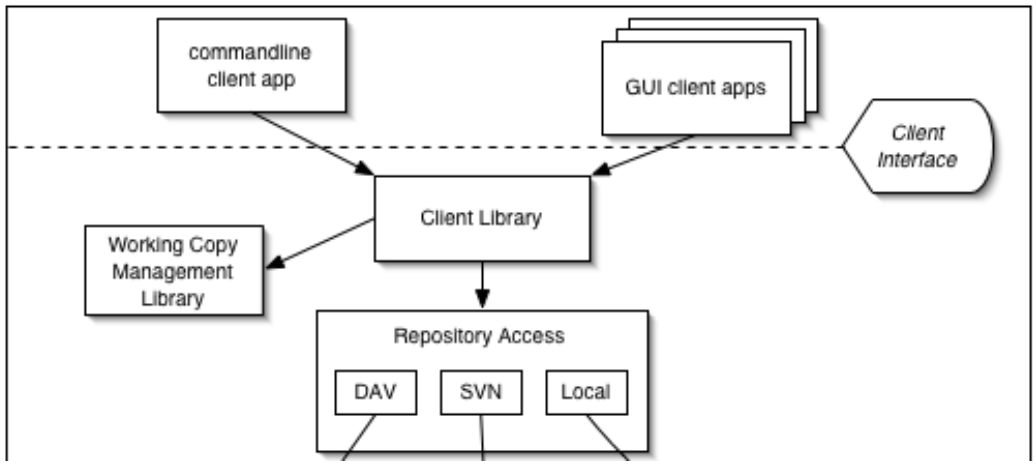
Ezzel a munkaciklussal minimálisra csökkenthetjük annak az esélyét, hogy konfliktusok alakuljanak ki, így kevesebb problémát okozhatunk magunknak és a projekten velünk együtt dolgozóknak.

- Módosítás előtt mindig frissítsük a helyi változatunk. Így elérhetjük, hogy mindig a legfrissebb változat legyen nálunk. **svn update**
- Végezzük el a változtatásainkat **svn add, svn copy, svn move, svn delete**, valamint a tényleges fájlmódosításokat.
- Vizsgájuk meg a változtatásainkat **svn status, svn diff**
- Frissítsük a helyi változatunk, hogy tudjuk, nem történt-e azóta frissítés a repository-ban és lássuk, ha valamivel ütköznek a módosításaink. **svn update**
- Ha szükséges, vonjuk vissza a változtatásaink **svn revert**
- Szüntessük meg a konfliktusokat (olvasszuk be mások változtatásait) **svn update, svn resolved**
- Commitoljuk a változtatásainkat **svn commit**, Fűzzünk megjegyzést a változáshoz, hogy később könnyebben értelmezhessük!

Kliens programok

- command line: Windows cmd, Linux bash
- Tortoise SVN: Windows
- RapidSVN: Platform független
- Eclipse plugin Subversive: Platform független

TortoiseSVN fogjuk használni, ingyenesen letölthető, használható. Beépül a Windows intézőbe, ezáltal kényelmes a használata.



Repo Browser

A TortoiseSVN Repository Browser-ének segítségével, megnézhetjük, hogy jelenleg a verziókövető rendszer milyen adatokat tart számon a repoban. Mindezt a nélkül tudjuk megtenni, hogy munkapéldányt hozzánk létre, vagy a már meglévő munkapéldányunkat frissítenénk.
Nézzük meg először, hogy is néz ki egy igazi projekt SVNben. Tekintsük meg a Gnome projekt gnome-panel modulját.
Az asztalon jobb egér gomb, Tortoise SVN, Repo-browser.

A címsorba írjuk a következőt

http://svn.gnome.org/svn/gnome-panel

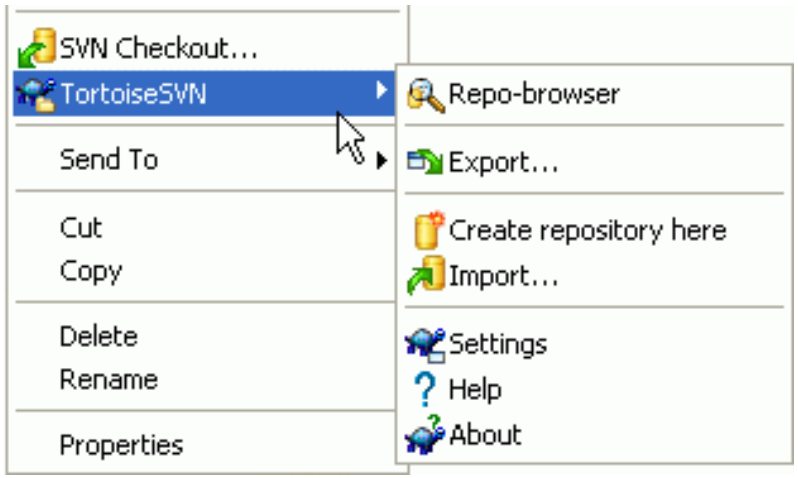
Három fő könyvtárt fogunk találni, ezt a szerkezetet az SVN ajánlja.

- **branches** Elágazások kezelése. Példa később lesz rá.
- **tags** Kiadások tárolása. Munkánk során eljöhetnek nevezetes pontok, amiket szeretnénk megjelölni és később azokat előhívni. Például egy projektmunka első fázisának vége, ezt taggel megjelöljük és bármikor előhívhatjuk. Másik példa Windows 7 kiadása.
- **trunk** Ez a fő fejlesztési vonal, ide rakjuk a napi munkánkat.

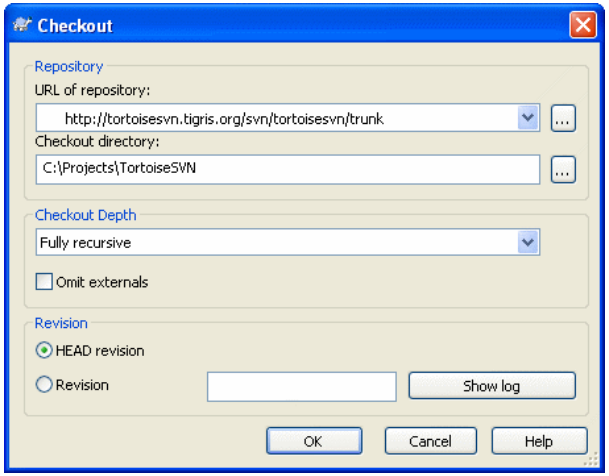
Kezdjük el!

Most, hogy láttuk, hogyan csinálják a nagyok, nézzük meg, hogyan tudjuk mi is használni a rendszert.

- Hozzunk létre egy "working copy" nevű könyvtárat a **turdoson**.
- Lépjünk bele a könyvtárba, és jobb egér gomb, **SVN checkout...**



Munkapéldány létrehozása



Repo cím

Minden csoport rendelkezik egy saját könyvtárral az alábbi repoban, melynek címe:

https://wsn.itk.ppke.hu/bev_szamtech_repo/trunk/x_csop

- A username helyére saját digitusos azonosítónkat írjuk
- Az x_csop helyére írd a saját csoport számod
- Amikor jelszavadat kéri, akkor az egyetemi jelszavadat add meg!

A .svn könyvtár tárolja a verziókövetéshez szükséges helyi

Hogyan tovább?

- Tegyük fel, hogy Forma 1-es autó fejlesztési adatait tároló rendszert kell létrehoznunk.
- Minden a fejlesztés közben létrejövő változás itt lesz tárolva, a mérnökök ide vezetik fel az újításokat.
- Páronként fogunk dolgozni. Az **A csapat** hozzon létre egy könyvtárat a **trunk** könyvtáron belül, a párok vezetéknevét felhasználva. Pl: Nagy Géza és Kiss Béla esetén: nagy_kiss
- Az alábbi webcímről töltsük le a csapatunknak megfelelő tömörített állományokat.

http://digitus.itk.ppke.hu/ tuzzoan/f1_car/

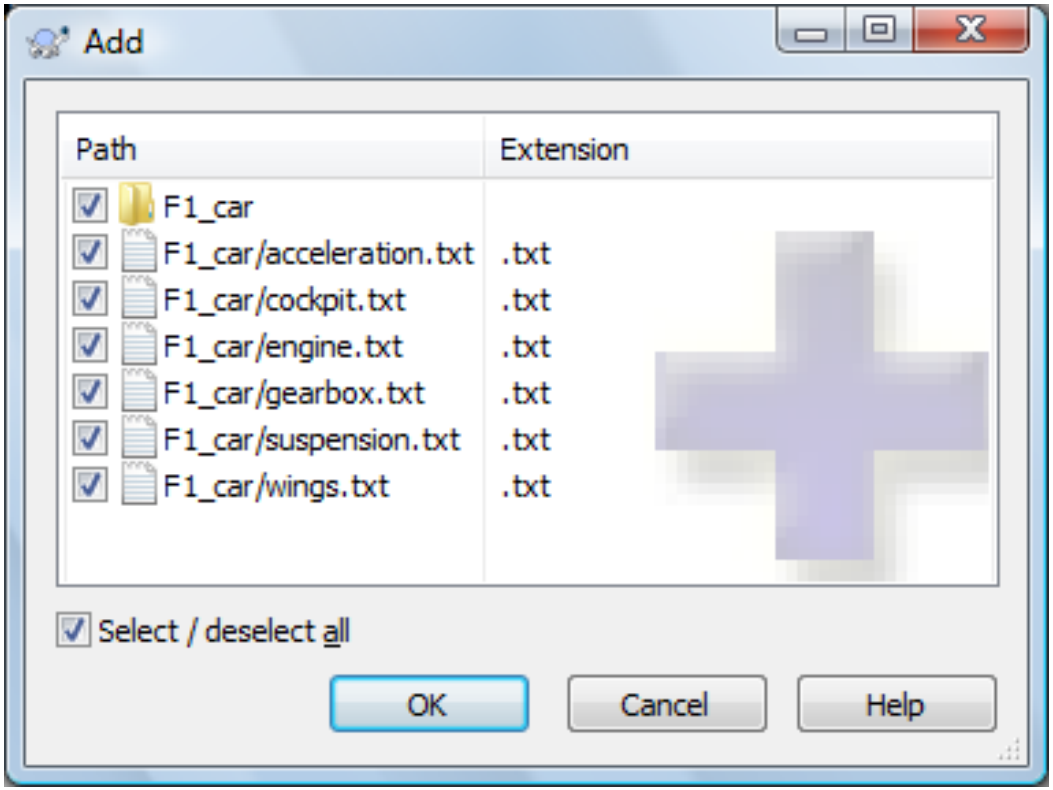
Milyen szereplők jelennek meg a rendszerfelhasználói között?

- Fejlesztőmérnökök.
- Elemzők
- Marketingesek
- Vezetőség

Minden szereplőnek lehet írás és olvasási joga, lehetősége van a fejlesztést követni, valamint egyes jellemzők alakulását egymással összevetni. Mindezt természetesen anélkül, hogy adatvesztés lenne, vagy más felhasználókat hátráltatnánk munkájukban.

Megjelölés verziókövetésre

Ahhoz, hogy az egyes fájlok verziókezelés alá kerüljenek, meg kell jelölnünk az **add** parancs segítségével. A következő commit során a rendszer észreveszi az új fájlokat és beküldi a repoba tarolásra.



A csapat

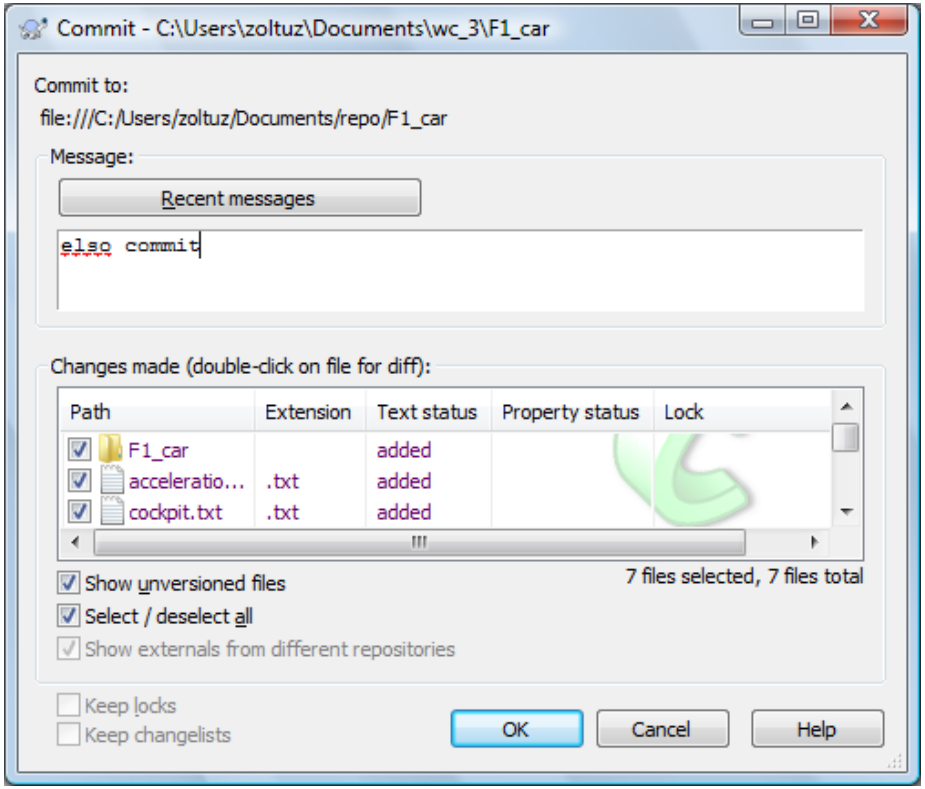
- Belemásolja letöltött zip tartalmát a létrehozott közös könyvtárba (pl. nagy_kiss)
- Megjelőli verziókövetésre a könyvtárat. TortoiseSVN menü, **add...** parancs.

B csapat

- Figyel a csapattársára.

Hozzáadás a verziókezelő rendszerhez

Végezzük el a kezdeti commit-ot. Jobb egér gomb, "SVN Commit..."



Mindig fűzzünk kommentárt a commitokhoz, hogy később könnyebben emlékezhessünk arra, hogy mit is csináltunk, valamint mások is tudják értelmezni a munkánkat.

A csapat

- Commitoljuk az eddigi munkánkat.
- Ne feledjük a kommentet!

B csapat

- Ha az A csapat végezett, az **SVN update...** segítségével töltsük le a másik csapat munkáját.
- Keressük meg azt a könyvtárat, ahová közösen fogunk dolgozni. (pl. nagy_kiss könyvtár)

A csapat

- Ha a B csapat végzett, akkor az SVN update segítségével töltsük le a változásokat.

B csapat

- Másoljuk bele a munkapéldányunkban lévő közös könyvtárba azokat a fájlokat, amitket korábban töltöttünk le.
- Jelőljük meg verziókezelésre. (SVN add...)
- Commitoljuk. (SVN commit...)

Update

Most a következő dolgokat figyelhetjük meg:

- Megjelenik a többi pár munkája a mi munkapéldányunkban
- A párunkkal közös könyvtárban egyesülnek a projekt fájlok, megkapom azt felet is, amit nem én tettem be a rendszerbe.
- A teljes csoport tud egymás mellett dolgozni, úgy, hogy nem zavarjuk egymás munkáját.
- Emlékezzük az "Alapvető munkaciklusra"! Minden alkalommal frissítsük a munkapéldányukat azelőtt, hogy dolgozni kezdenénk vele.

Az első fejlesztésünk (Az első commit)

Biztonsági okokból a cockpit anyagát és a biztonsági öv típusát lecserélték.

A csapat

- Nyissuk meg a **cockpit.txt** fájlt és felhasznált anyagot nevezzük át *Szénszál és Kevlár*-ra.

B csapat

- Nyissuk meg a **cockpit.txt** fájlt és a biztonságiöv típusát változtassuk meg *5 pontosra*.

A változásokat commitoljuk a rendszerbe. A napló üzenet: cockpit anyagának fejlesztése. Mind a két csapat a saját fejlesztését írja bele az kommentbe.

Update

- Ha most frissítjük a munkapéldányunkat, megérkezik a másik csapat változása a mi munkapéldányunkba is.
- A Copy-Modify-Merge modellnek megfelelően tudtuk egyszerre szerkeszteni ugyanazt a fájlt konfliktus nélkül.
- Nyissuk meg a cockpit.txt és győződjünk meg róla, hogy minden változást látszik a fájlon.

Több fejlesztést is tudunk egyszerre rögzíteni

Tegyük fel, hogy a mérnököknek sikerült új motort és váltót készíteni az autóhoz. Ezáltal megváltozott a kocsi több paramétere is. Vezessük fel ezeket a változtatásokat.

Csapat A

- A hengerűrtartalom megnőtt 1.8 litertől 2.2 literre.
- A kocsi teljesítménye 560 hp-ről 700 hp-re változott.

Csapat B

- Ennek megfelelően 0.2 mp-el kevesebb idő alatt gyorsul az autó, 100-ra, 200-ra, stb.
- És az első légterelő szárnyakat 2 cm-rel lejjebb vitték.

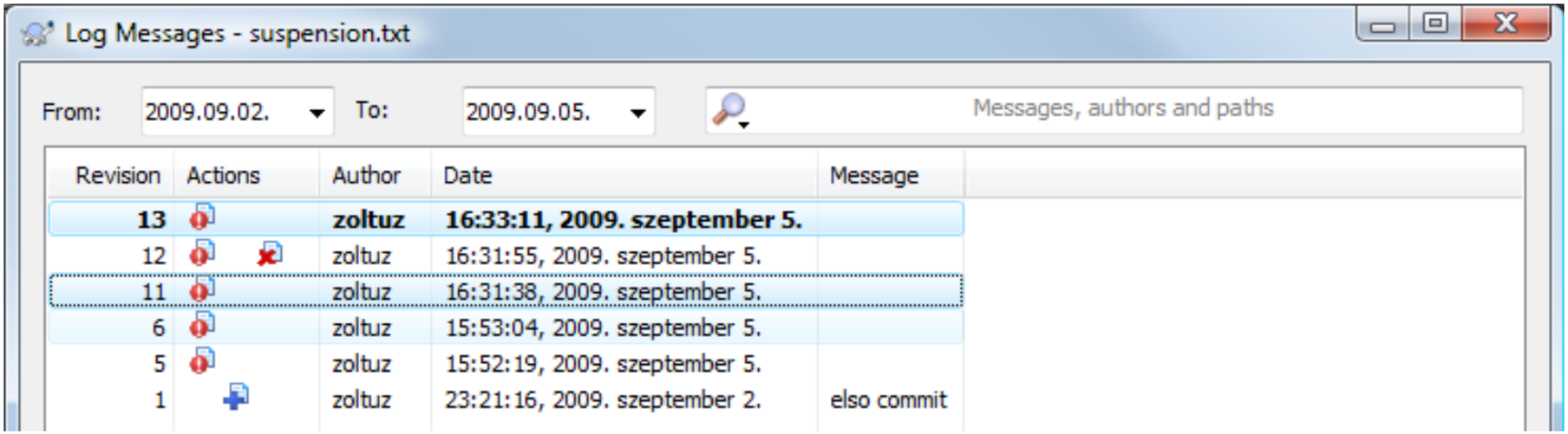
Rögzítsünk a változtatásokat. Ne felejtsünk el értelmes kommentet írni a commithoz!!!
Valamint SVN update-tel lekérni az esetleges változásokat.

SVN diff

Frissítés után mindenkinek a munkapéldányában megjelennek a változtatások. Ahhoz, hogy megnézzük mi változott az egyes fájlokban, válasszuk az TortoiseSVN menüből a *Diff with previous* version menüpontot. Színes-szagos kétablakos menüben tudjuk összehasonlítani a változásokat.

SVN diff 2.

Válasszunk ki egy fájlt. A TortoiseSVN menü **show log** segítségével jelenítsük meg a történetét. A fájlra vonatkozó összes információ elérhető, ki-mikor-melyik verzióban szerkesztette. Ha itt egy revízióra jobb gombbal kattintunk, ugyanazt láthatjuk, mint az előzőben. De ha Ctrl+kattintás segítségével ki jelölünk két tetszőleges verziót, akkor meg tudjuk nézni a két verzió közötti különbséget.



Konfliktusok feloldása

A konfliktusok akkor jönnek létre, amikor egyszerre több felhasználó ugyanannak a fájlnak ugyanazt a sorát szerkeszti. Hozzunk létre egy konfliktust!

A csapat

- Az engine.txt-ben teljesítményt írjuk át 800 hp-re. (1)
- A wings.txt a hátsó szárnyat tegyük 50 cm-re. (1)
- Csak az **engine.txt** commitoljuk! (2)
- Ha a B csoportban mindenki végzet, akkor mi is commitoljuk a wing.txt. "out of date" hiba, update-eljünk.

B csapat

- Az engine.txt-ben teljesítményt írjuk át 805 hp-re. (1)
- A wings.txt a hátsó szárnyat tegyük 45 cm-re. (1)
- Csak a wings.txt-t commitoljuk!
- Ha az A csoportban mindenki végzet, akkor mi is commitoljuk az engine.txt. "out of date" hiba, update-eljünk.

Most mindenki konfliktussal rendelkezik a rendszerben! Sajnos az SVN nem (annyira) okos, hogy ezt megoldja. Ezért nekünk kell megoldani.

Konfliktusok feloldása

- Kattints arra a fájlra, amivel a konfliktus van. TortoiseSVN menü Edit Conflicts.
- Megint egy színes, szagos szerkesztő. 3 fő ablak, sorrendben: a többiek verziója, az én verzióm, és az felső kettő összefésüléséből keletkező fájl.
- A felső két ablakban soronként kiválaszthatjuk, mi kerüljön bele az összefésültbe. Jobb gombbal kattintsunk a "+" jelre és válasszunk a felmerülő lehetőségek közül.
- ha végeztünk, akkor TortoiseSVN menü *Resolved* parancs segítségével jelezhetjük, hogy feloldottuk a konfliktust, most már tudjuk commitolni a változtatásunkat.

Szabad rablás, avagy ki a tettes?

Mindeki keressen magának egy szimpatikus párost, lehetőleg ne a sajátját. Mindenki szerkesszen bele **suspension.txt** fájlba, lehetőleg több sorban végezzünk változtatásokat, valamint adjunk hozzá sorokat. Commitoljuk!

Ki volt, aki bele piszkált a munkámban?

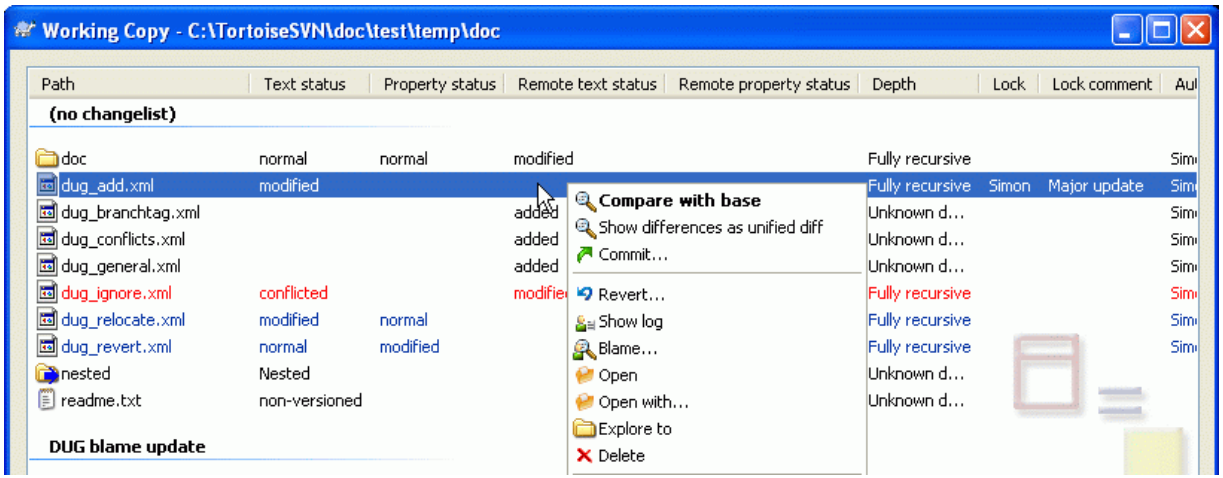
Update-eljünk, és nézzük meg ki mit csinált! Az TortoiseSVN Blame... parancs segítségével meg tudjuk nézni a fájlok történetét soronként, mellette módosítást végző személy azonosítójával.

SVN revert - hoppá, ezt elszúrtam!

"Egyszerűen nem működik! Szeretném visszavonni az egészet!", hangzik el a vágyakozás. Normál fájlokkal ezt nem lehet visszavonást csinálni, de verziókezeléssel vissza tudjuk vonni a repository és a munkapéldányuk között különbségét, ha nagyon elvesznénk!
Nyissunk meg pár fájlt és módosítsuk őket. De ne commitoljuk a változások, ha nem a rombolás végén válasszuk a TortoiseSVN menüből a **revert** parancsot.

Változások vizsgálata még Update előtt

Néhány esetben előfordulhat, hogy szeretnénk megnézni mik a különbséget az aktuális munkapéldány és a repo között anélkül, hogy azt integrálnánk a mi munkapéldányunkba.
Ehhez a TortoiseSVN menü **Check for modifications...** menüpontja nyújt segítséget.



Fájlok zárolása

Még az elején tisztáztuk, hogy a *Copy-Modify-Merge* modellben nincs szükség a fájlok zárolására a napi használat során. De előfordulhatnak olyan állományok, amit ha két felhasználó egyszerre szerkeszt, utána nem lehetséges azok összefésülése. Ilyenek például a bináris állományok, azaz képek videók, stb.
A fájlok zárolásának menete a következő:

- Kiválasztjuk a zárolandó fájlt, fájlokat. TortoiseSVN menü, **Get Lock...**
- A felugró ublakban megadunk egy kommentet, miért zároltuk az adott fájlt.
- Szerkesszük a fájlt, majd commitoljuk a változásokat.
- Engedjük fel a zárolást. TortoiseSVN menü, **Release Lock...**

Fájl zárolása - Feladat

A csapat

- Zárolja a wings.txt-t
- Próbálja módosítani és commitolni, az engine.txt-t.
- Oldja fel a zárolást.
- Próbálja újra commitolni, az engine.txt-t.

B csapat

- Zárolja az engine.txt-t
- Próbálja módosítani és commitolni, a wings.txt-t.
- Oldja fel a zárolást.
- Próbálja újra commitolni, a wings.txt-t.

Kiadások létrehozása

- A munka során elérkezhet olyan pont, amikor szeretnénk az eddig megoldott feladatokat megjelőlni. Például, amikor egy szoftver megjelenik a 6.5-ős verzióval, az azt jelenti, hogy a verziókezelő rendszerben létrehoztak egy címkét, ami egy "nevesített" fénykép a repositoryról, ez a 6.5 verzió. Ezt a verziót bármikor ki checkout-olhatjuk és felhasználhatjuk.
- A tag-elés esetén egy "fénykép" készül az repositoryról (vagy arról amit megjelőlünk), és ezt a fényképet a tags könyvtárba mentjük el.
- Taget létrehozni a TortoiseSVN menü **Branch/tag...** menüpontjával tudunk.
- A **To url**-el adjuk meg a tag elérését.

- Adjuk meg, hogy miből szeretnék csinálni a taget.
  - Head, azaz a repo aktuális állapotáról
  - Meghatározott revízióról
  - Az aktuális munkapéldányról
- Fűzzünk kommentet a munkánkhoz.

A csapat tag url-je

https://wsn.itk.ppke.hu/bev_szamtech_repo/tags/x_csop/**f1_2009a**

B csapat tag url-je

https://wsn.itk.ppke.hu/bev_szamtech_repo/tags/x_csop/**f1_2009a**

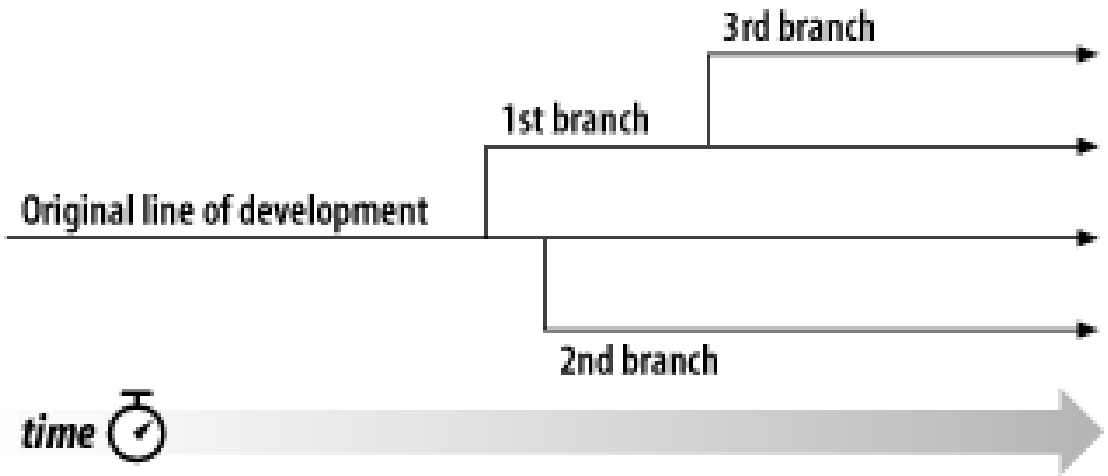
Váltás tag-ek és a főverzió között

Az TortoiseSVN Switch parancs segítségével tudunk váltani az egyes tag-ek, branch-ek, és a trunk között.

- A főfejlesztési vonal a trunk alatt található.
- A tags alatt vannak az eddig elkészült tag-ek
- Branch alatt pedig az elágaztatások (részletesen később)

Elágazások kezelése

Branchet akkor hozunk létre, ha valami olyasmivel szeretnénk kísérletezni, ami nem feltétlenül illik bele az aktuális fejlesztő ágba (trunk). Ilyen lehet például, ha mondjuk valamelyik programkönyvtárat le akarjuk cserélni egy másikra. SVN-ben ilyenkor csupán annyit teszünk, hogy a trunkot átmásoljuk a branches könyvtárba a TortoiseSVN Branch/Tags... segítségével.



Természetesen lehetséges az egyes elágazások beolvasztása a főfejlesztési vonalba (trunk), de ez nem képzi a mostani óra anyagát.

Összefoglalás

- Wiki, digitus, turdus, számtech levlista: most már ismerős fogalmak.
- Mai alkalom célja volt, hogy megismerkedjünk a verziókezelés alapjaival. Ehhez felhasználtuk az SVN programot.
- Az félév soron a házi feladatokat SVN keresztül kell majd beadni.
- Most már profi SVN felhasználó vagyok? Sajnos távolról sem, aki szeretne tovább fejlődni annak az egyik lehetőség: Az SVN Book olvasgatása, http://svnbook.red-bean.com.
- Valamint a gyakorlatvezetőktől is lehet kérdezni.