

# **SVN gyorstalpaló**

Készítette:  
Fülöp Balázs

## Bevezetés

Jelen dokumentum a Subversion verziókövető rendszer felhasználásához nyújt segítséget. A leírás a megfelelő szerveroldali támogatás mellett a SmartSVN nevű kliensprogram meglétét feltételezi.

## A verziókövetésről röviden

A verziókövető rendszerek legfőbb célkitűzése egy olyan tároló biztosítása, amely megjegyzi a benne tárolt adatokon végzett változtatásokat. A verziókövetésre első megközelítésben tekinthetünk úgy, mint egy olyan eszközre, mellyel a módosítások korlátlan mélységben visszavonhatóak.

A legtöbb verziókövető rendszer valamilyen formában támogatja a csoportmunkát. A felhasználók egy közös tárolón dolgoznak, így gyorsan és egyszerűen tehetnek közzé, vagy érhetnek el friss adatokat. Emellett megtartják annak a lehetőségét, hogy bármikor visszatérjenek korábbi változatokhoz.

Ezeknek a rendszereknek elsősorban szoftverfejlesztők látják nagy hasznát. A fejlesztés alatt álló programok kódbázisának kezelésében komoly segítséget jelent a verziókövetés.

## A Subversion alapfogalmai

A Subversion (rövidebb nevén SVN) az egyik legnépszerűbb nyílt forráskódú, ingyenesen elérhető verziókövető megoldás (<http://subversion.tigris.org/>). Bár ez az útmutató kizárólag ennek a rendszernek a használatába ad betekintést, az itt ismertetett alapfogalmak más környezetekben is visszaköszönhetnek.

A Subversion központi tárolóját a felhasználók közvetlenül nem módosítják. Minden felhasználó saját munkapéldányon dolgozik. Mappákra és állományokra levetítve ez azt jelenti, hogy a tároló egy olyan fájlserver, amelyen nem lehet fájlműveleteket végezni. A felhasználóknak létre kell hozniuk a szerveren tárolt adatok (valamely változatának) egy helyi másolatát. Ezt a munkapéldányt szükség esetén frissítik, módosítják, illetve közzéteszik saját változtatásaikat.

Álljon itt egy szótár a Subversion fogalmaihoz:

repository	közös tároló
working copy	munkapéldány
checkout	munkapéldány létrehozása
update	munkapéldány frissítése
commit	munkapéldány változtatásainak közzététele

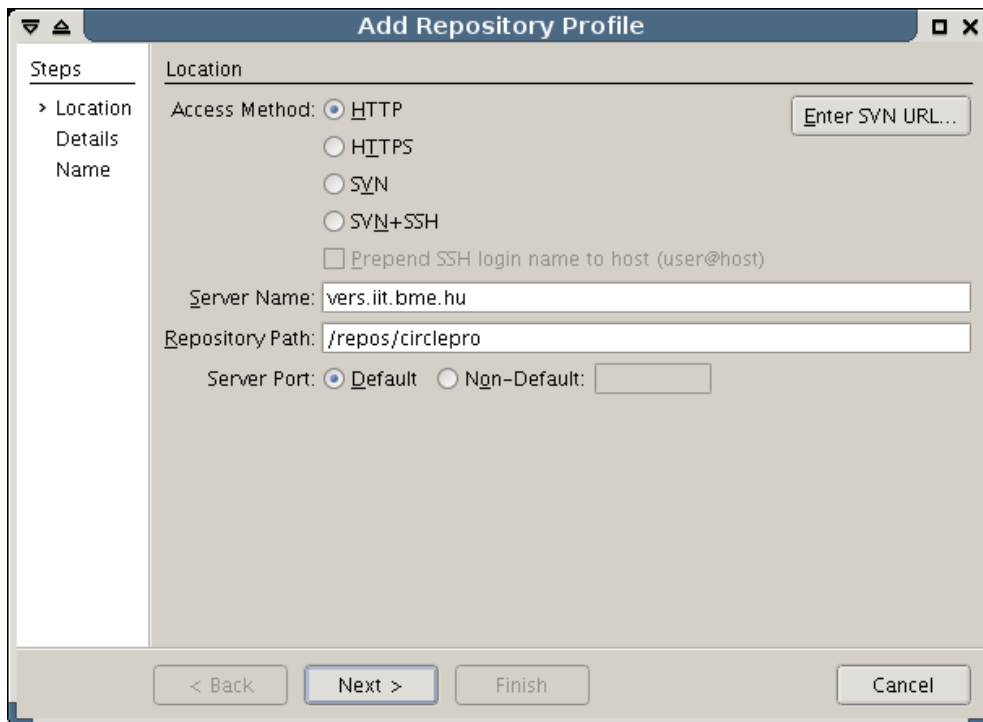
## Első lépések a SmartSVN-ben

A SmartSVN egy Java-ban írt (és így a legtöbb platformon használható), ingyenes változatban is elérhető grafikus kliensprogram (<http://www.syntevo.com/smartsvn/>).

A gyorstalpaló hátralévő részében egy C++ projekt SVN-ben végzett fejlesztését tekintjük át. A fejleszteni kívánt program a Circle Pro, amely tetszőleges kör területének meghatározására alkalmas. A projekt két résztvevője: Aladár és Béla. Aladár feladata a program belépési pontjának elkészítése, míg Béla az alkalmazás motorjáért felelős.

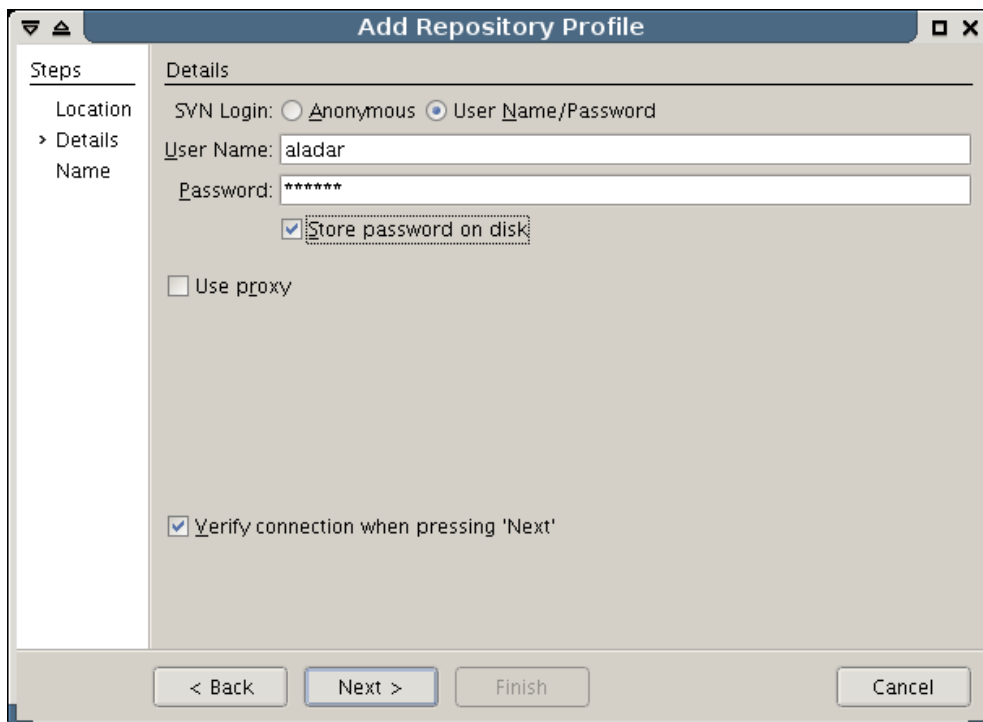
Mindenek előtt létre kell hozniuk saját munkapéldányukat. Lássuk, mit jelent ez Aladár esetében. Mivel a fejlesztés csak most indult, ez a kezdeti művelet nyilvánvalóan üres könyvtárat fog eredményezni. A SmartSVN Project/Check Out... menüpontját kiválasztva elénk tárul a Check Out

Project varázsló. Ennek első lépésként meg kell adni a tároló elérését, ehhez válasszuk a Manage... gombot. Az előugró ablakban kattintsunk az Add gombra. Az erre előugró Add Repository Profile varázslóban először a protokollt, a szerver címét és a tároló elérési útját kell megadnunk. Ezt értelemszerűen töltjük ki (lásd 1. kép).



1. kép

A Next gombra történő kattintást követően megadhatjuk az eléréshez szükséges felhasználónevet és jelszót (lásd 2. kép).

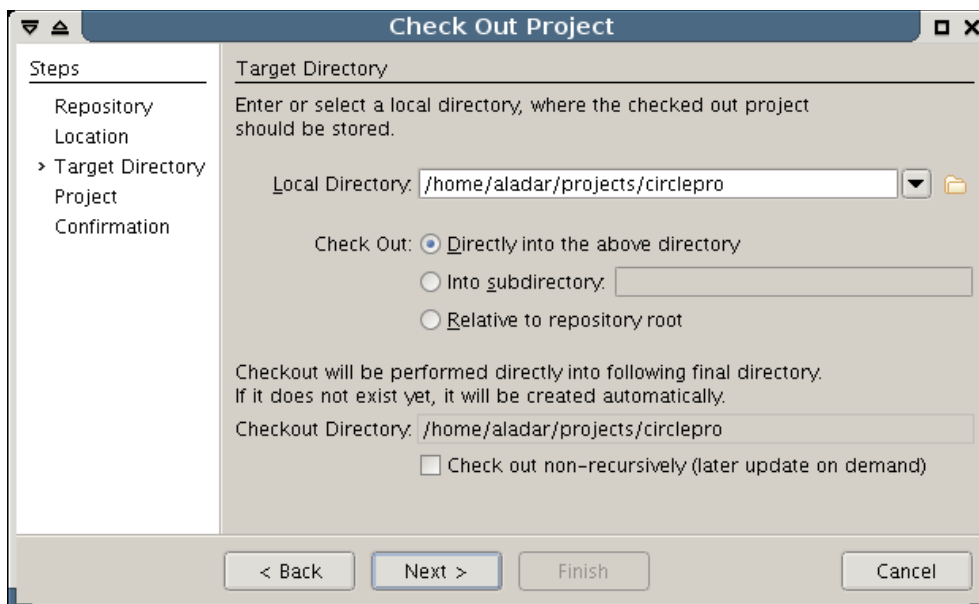


2. kép

A Next gombra történő kattintás után (ha az előző panelen másként nem rendelkezünk), a kliens rögtön megkísérli a csatlakozást, és probléma esetén szól. Ha minden rendben ment, az utolsó panelen beállíthatunk egy egyedi nevet az elérésnek.

A Repository Profiles ablakban kattintsunk az Ok gombra, majd folytassuk a Check Out Project varázslót a Next gombra való kattintással. Ezen a panelen kiválaszthatjuk, hogy mely fájlból, vagy könyvtárból szeretnénk munkapéldányt létrehozni. Mivel a tároló egyelőre üres, itt nincs semmi dolgunk, kattintsunk a Next gombra.

Ezen a panelen határozhatjuk meg, hogy melyik helyi könyvtárban jöjjön létre a munkapéldány (lásd 3. kép). Ezt válasszuk ki, majd kattintsunk a Next gombra.



3. kép

A következő panelen eldönthetjük, hogy verziókövetés alatt lévő munkapéldányt szeretnénk, vagy sem. Utóbbi esetet csak olvasható üzemmódként képzelhetjük el. Természetesen a munkapéldányt fogjuk tudni módosítani, de nincs lehetőségünk a változtatásokat a tárolóba feltölteni. Hagyjuk meg az alapértelmezett első üzemmódot, és kattintsunk a Next gombra.

Az utolsó, megerősítő panelen kattintsunk a Finish gombra.

Ha most megnyitjuk a munkapéldány könyvtárát egy fájlkezelőben, láthatjuk, hogy valójában nem üres. Tartalmaz egy .svn nevű könyvtárat, amelyet az SVN kliens az ezen példánnyal kapcsolatos adminisztráció céljából tart fenn. Annak érdekében, hogy ebben a könyvtárban ne legyen kezelhetetlenül sok bejegyzés, a projekt összes verziókövetés alatt lévő alkönyvtára tartalmaz ilyen könyvtárat, és mindegyik csak a saját fájljairól tartalmaz információt. Ezeket a könyvtárakat tilos letörölni, vagy kézzel módosítani, különben a munkapéldány használhatatlanná válhat.

## Munka a SmartSVN-nel

Aladár kedvenc szövegszerkesztőjével elkészíti a munkapéldány könyvtárában az alkalmazás Makefile-ját:

```
PROJECT = circlepro
OBJECTS = Main.o Circle.o
CC = g++

build: $(OBJECTS)
    $(CC) -o $(PROJECT) $(OBJECTS)

Main.o: Main.cpp Circle.h
    $(CC) -c -o Main.o Main.cpp

Circle.o: Circle.cpp Circle.h
    $(CC) -c -o Circle.o Circle.cpp
```

```
clean:
    rm -f *~ *.o $(PROJECT)
```

Továbbá megírja a belépési pontot jelentő Main.cpp fájlt:

```
#include "Circle.h"
#include <iostream>

using namespace std;

int main(int argc, char *argv[]) {

    double r;
    cout << "r = "; cin >> r;

    Circle circle(r);
    cout << "A = " << circle.GetArea() << endl;

    return 0;

}
```

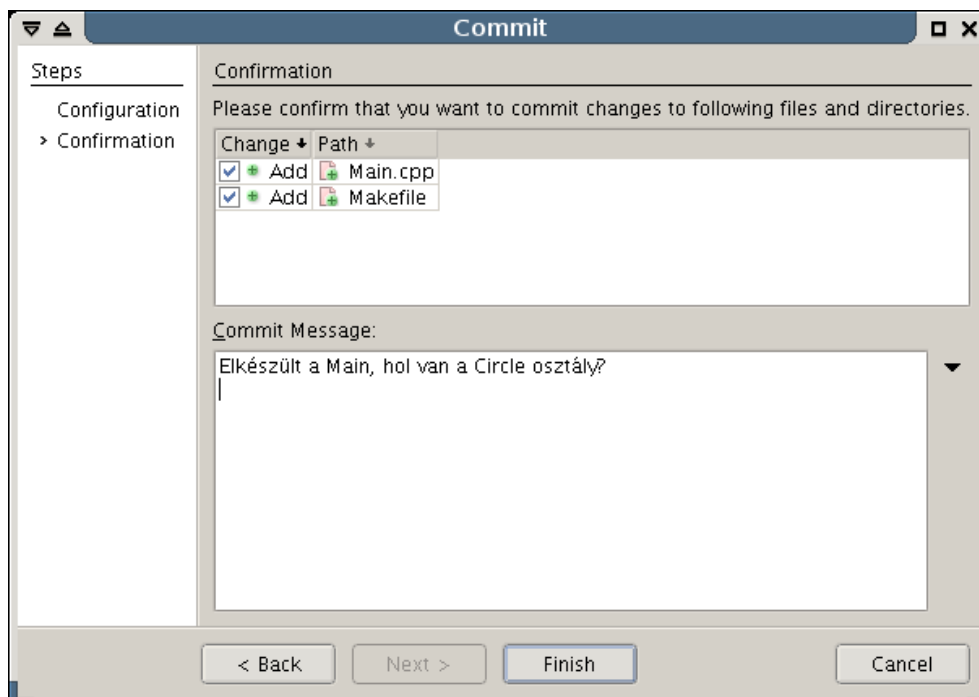
Ha ezek a fájlok nem jelennek meg automatikusan a SmartSVN-ben, válasszuk a View/Refresh menüpontot. A Local State oszlop alapján látható, hogy a fájlok nincsenek verziókövetés alatt.

Fontos megjegyezni, hogy egy munkapéldány nem feltétlenül csak azokat a fájlokat tartalmazza, amelyek a tárolóban is előfordulnak. Fejlesztés során a szövegszerkesztő létrehozhat backup, vagy átmeneti fájlokat, a fordító object fájlokat, és végül, de nem utolsó sorban a kész futtatható állomány is létrejöhet itt. Ezeket kifejezetten nem javasolt a tárolóban elhelyezni, mert csak feleslegesen foglalják a helyet.

A fentiek okán az újonnan létrehozott fájlok alapértelmezésben nincsenek verziókövetés alatt. Erről nekünk kell explicit módon gondoskodnunk a Modify/Add... menüpont segítségével.

Miután a fájlokat a munkapéldányban verziókövetés alá helyeztük, itt az ideje, hogy közzétegyük. Válasszuk ki a Modify/Commit... menüpontot.

A megjelenő varázslóban az első, konfigurációs panelt egy Next-tel átugorhatjuk. A következő panelen ellenőrizhetjük, hogy mely fájlok, illetve könyvtárak kerülnek feltöltésre. Továbbá megadhatunk egy rövidebb megjegyzést a feltöltéssel kapcsolatban (lásd 4. kép).



4. kép

Kattintsunk a Finish gombra.

Ha Béla ezután hozza létre munkapéldányát, vagy végez egy frissítést a Modify/Update... menüpont kiválasztásával, a fájlok megjelennek nála. Ezt követően ő is hasonló módszerrel töltheti fel az alkalmazás hiányzó részét.

Circle.h:

```
#ifndef CIRCLE_H
#define CIRCLE_H

class Circle {

private:
    double r;

public:
    Circle(double r);
    double GetArea();

};

#endif
```

Circle.cpp:

```
#include "Circle.h"
#include <cmath>

Circle::Circle(double r) {

    this->r = r;

}

double Circle::GetArea() {

    return r*r*M_PI;

}
```

A SmartSVN főablakában továbbra is szépen látszanak a munkakönyvtár fájljai, az utolsó módosítás szerzőjével és dátumával együtt (lásd 5. kép). Viszont feltűnhet egy érdekesség a Last Revision oszlopban: Béla fájljai a 2-es verziószámot viselik, míg Aladáréi az 1-es. Ez annak köszönhető, hogy a Subversion külön tartja nyilván az egyes fájlok verziószámait, és nem egy közös verziószámot tárol (ahogy pl. a CVS teszi). Ezen kívül egy új fájl verziószáma nem 1-től indul, hanem a tároló legfiatalabb fájljának verziószámánál lesz 1-el nagyobb. Így a verziószámok valójában azt jelentik, hogy a tároló hányadik verziójában történt az adott fájllal az utolsó változtatás.

Name	Revision	Local State	Lock	Last Rev.	Last Changed	Text Updated	Props Updated	Last Author
Circle.cpp	2	Unchanged		2	2007-02-23 01:31:05	2007-02-23 01:31:05		bela
Circle.h	2	Unchanged		2	2007-02-23 01:31:05	2007-02-23 01:31:05		bela
Main.cpp	2	Unchanged		1	2007-02-23 01:20:04	2007-02-23 01:20:04		aladar
Makefile	2	Unchanged		1	2007-02-23 01:20:04	2007-02-23 01:20:04		aladar

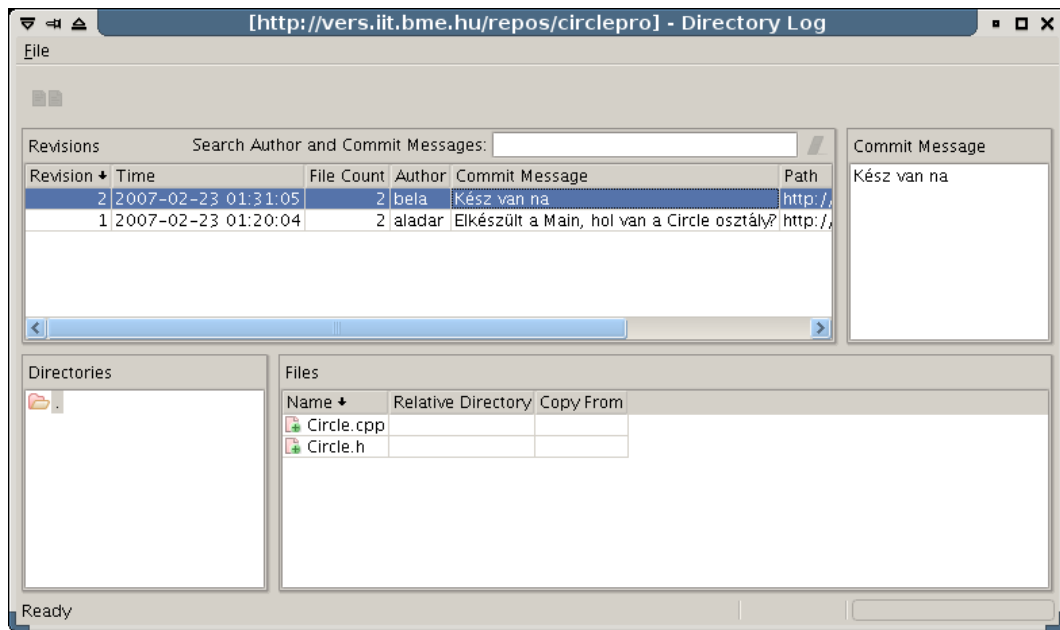
5. kép

## További ujjgyakorlatok

Tegyük fel, hogy Aladár véletlenül módosítja a Circle.cpp fájlt, és már nem tudja helyreállítani a helyesen működő változatot. Válasszuk ki a fájlt a főablakban, majd használjuk a Modify/Revert... menüpontot.

Elképzelhető az is, hogy Aladár letörli a saját munkapéldányából a kérdéses fájlt. Ezen egy Modify/Update... művelet segít. Fontos megjegyezni, hogy ha frissítés esetén nem a legfrissebb változatot (HEAD) választjuk ki, lehetőség van a munkapéldányt egy korábbi állapotba visszaállítani.

A felhasználók a feltöltések megjegyzéseit is böngészhetik a Query/Log... menüpont segítségével (lásd 6. kép).



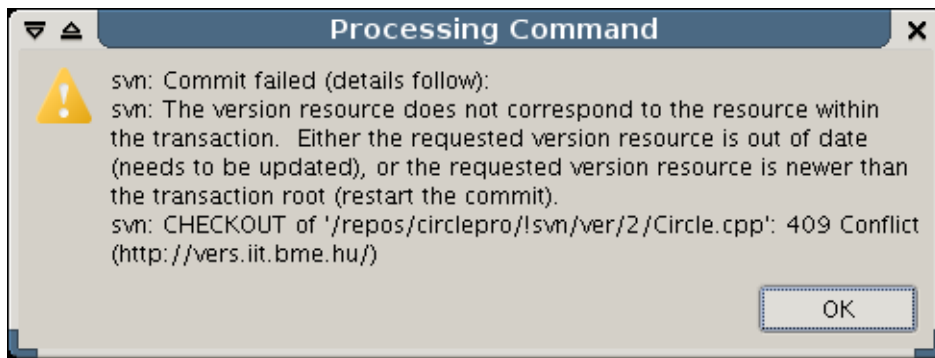
6. kép

## Konfliktus-kezelés

Egy központi kérdése a verziókövetésnek az, hogy miként lehetne megoldani a felhasználók között fellépő versenyhelyzetet. Tegyük fel, hogy Aladár lekérdező, Béla beállító metódust ír a Circle osztály r tagváltozójához. Párhuzamosan módosítják a saját munkapéldányukat, majd Béla közzéteszi a változtatásait. Ezután Aladár úgy kísérli meg a közzétételt, hogy a saját változata nem tartalmazza Béla változtatásait. Ha a műveletet a rendszer szó nélkül elvégezné, Béla munkája elveszne, ráadásul erről egyikük sem értesülne.

Erre a kérdésre egy lehetséges válasz a zárolás. Ez sajnos több problémát vet fel, mint amennyit megold. Mi történik, ha egy kliens zárol egy fájlt, majd elfelejti felszabadítani? Kinek a felelőssége a bent ragadt zárat feloldani? Ha egy nagy fájlban az egyik kliens csak az elejét, egy másik csak a végét szeretné módosítani, miért szükségeszerű, hogy egyikük kizárja a másikat?

A Subversion válasza az úgynevezett copy-modify-merge modell. Ebben a felhasználók természetes módon, zárok nélkül dolgoznak. Ha konfliktus alakul ki, a rendszer kísérletet tesz arra, hogy a később érkezett felhasználó munkáját beolvassa a friss változatba. Ez sok esetben sikerül is, de például a fenti esetben meghiúsul (mindketten a fájl végéhez fűznek adatot). Ilyenkor hibaüzenettel jelzi, hogy a közzététel nem sikerült (lásd 7. kép).



7. kép

Ez esetben Aladár tudja, hogy frissítenie kell a saját munkapéldányát a közzététel előtt. Ha ezt megteszi, a 8. kép látványa fogadja.

Name	Revision	Local State	Lock	Last Rev.	Last Changed	Text Updated	Props Updated	Last Author
Circle.cpp	3	Conflict		3	2007-02-23 ...	2007-02-23 ...		bela
Circle.cpp.mine		Unversioned						
Circle.cpp.r2		Unversioned						
Circle.cpp.r3		Unversioned						
Circle.h	3	Unchanged		2	2007-02-23 ...	2007-02-23 ...		bela
Main.cpp	3	Unchanged		1	2007-02-23 ...	2007-02-22 ...		aladar
Makefile	3	Unchanged		1	2007-02-23 ...	2007-02-23 ...		aladar

8. kép

Látható, hogy a Subversion nem volt képes a konfliktus feloldására, ezért a felhasználóra bízta ennek a kezelését. Létrehozott egy .r2 kiterjesztésű változatot a fájlból, amely a munkapéldány régi változata. Az .r3 kiterjesztésű változat az, amelyik jelenleg a tárolóban van. A .mine az, amelyiknek a közzététele az imént nem sikerült. Végül a kiterjesztés nélküli változat, amelynek állapota Conflict, az alábbi tartalmazza:

```
#include "Circle.h"
#include <cmath>

Circle::Circle(double r) {
    this->r = r;
}

double Circle::GetArea() {
    return r*r*M_PI;
}
<<<<<<< .mine

double Circle::getRadius() {
    return r;
}
=====

void Circle::setRadius(double r) {
    this->r = r;
}
>>>>>>> .r3
```

A felhasználó döntése, hogy melyik fájlból kiindulva, miként oldja fel a konfliktust. Aladár jelen esetben tudhatja, hogy egyszerűen a <<<<<<<, =====, >>>>>>> jelölőket kell eltávolítani a fájlból, és a probléma megoldást nyer. Ezt megteheti a SmartSVN-en belül is, ha az Edit/Open menüpontot választja. Miután a konfliktust feloldotta, ezt jeleznie kell az SVN-nek a Modify/Mark Resolved... menüponttal. Ezt követően már sikeresen közzéteheti a mindkettejük változtatásait tartalmazó, 4. verziót.

A copy-modify-merge modellből is jól látszik, hogy a verziókövető rendszer nem gondolkozik a fejlesztő helyett, és nem helyettesíti az emberi kommunikációt. Mindemellett elengedhetetlen segédeszköz a fejlesztés során.