第 1 回 Subversion ゼミ

ゼミ担当者 : 平尾 洋樹，大西 祥代，上田 祐一郎，石田 裕幸，鍵谷 武宏，雨宮 明日香，菅原 麻衣子
指導院生 : 柴田 優
開催日 : 2006 年 9 月 28 日

1 はじめに

ソフトウェアやシステムの開発は，ソースコードなどの修正を繰り返して改良していくことが活動の基本である．ただし，開発現場ではソースコードやドキュメントに修正を加えるのが一人だけとは限らず，複数の開発者が並行して開発を進める場合も存在する．このため,「いつ」,「どこを」,「どう」修正したのかを管理することが重要となってくる．また，個人で開発する場合も，改良の履歴を管理することは非常に有用である [1]．一般に，変更の履歴などをバージョンとして管理するが，本ゼミではバージョン管理システム「Subversion」を用いたバージョン管理の方法について学ぶ．

2 Subversion

2.1 Subversion とは

Subversion とは，ファイルのバージョン管理をするアプリケーションソフトである．バージョン (version) とはファイルに対して変更を加え，その変更を確定するごとに上がっていく管理番号であり，リビジョン (revision) とも言う．Subversion は主にプログラムの開発現場などで使用されるが，プログラムのソースコードをはじめ，どんなファイルでも管理できる．また，複数人が同時に同じファイルを編集することができるが，この際手元で行った修正と他のユーザが行った修正が同じ箇所である場合，変更の衝突が発生する．これをコンフリクト (conflict) と呼ぶ．コンフリクトが発生した場合，ユーザ同士が話し合ってコンフリクトを解消する必要がある．また，コンフリクトが発生しない場合は両方の変更を自動的に統合することができる．

Subversion の概念図を Fig. 1 に示す．Subversion ではファイル群は一箇所にまとめられて管理される．このファイルをまとめたものをリポジトリ (repository) といい，この中にファイルの変更履歴も記録される．また，リポジトリにあるファイル群の編集は，リポジトリの内容をローカルにコピーし，そこで行う．これをワーキングコピー (working copy) という．

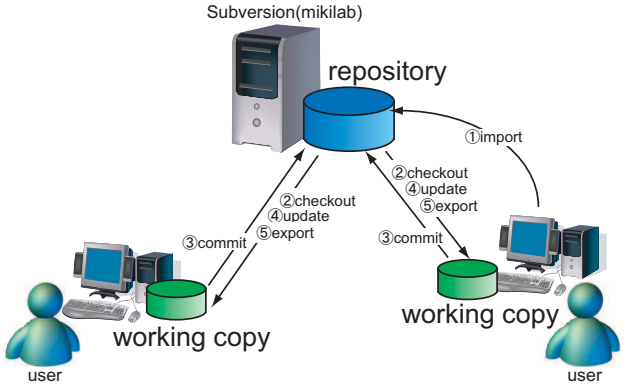

Fig. 1 Subversion の概念図 (出典：参考文献 [2])

ここで，Subversion で用いられる基本操作について，簡単に説明する．

1. インポート (import)
   最初にリポジトリにファイル群を登録すること．
2. チェックアウト (checkout)
   リポジトリからワーキングコピーを取り出すこと．
3. コミット (commit)
   ワーキングコピーでの更新結果を確定し，リポジトリに変更を反映させること．
4. アップデート (update)
   リポジトリの更新状況をワーキングコピーに反映させること．複数の開発者で開発を行っている場合，自分のワーキングコピーでは編集してなくても，他の開発者によりリポジトリが更新されている可能性がある．
5. エクスポート (export)
   編集するつもりはなく，管理ファイルを除いて対象ファイルだけを取り出す操作のこと．ソフトウェアのリリースなどに用いる．

2.2 リポジトリの構成

システム開発において，システム A を主に進めている場合を例に説明する．開発を進めた結果，システム A が一旦完成したとする．まだシステム A にはバグが存在している可能性があるが，新しい機能の追加を平行して行う．その際は，今後バグ修正を進めていくシステム A を開発の主流として残しておき，機能追加や構成の変更などを行うための開発の流れをシステム A+α として作る．また，システム A をベースとした別のシステム B を開発する際に，システム A の完成時のプログラムを利用する．この場合，システム A の完成時のリビジョンを目印として記録しておくと，後で便利である．

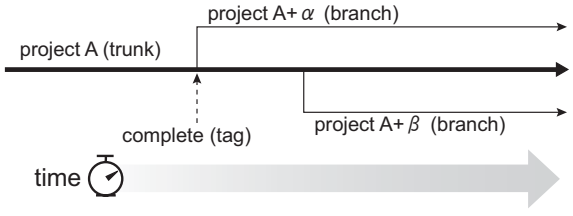

Fig. 2　開発の流れ (出典：参考文献 [3])

システム開発では上記のような場合が考えられるが，Subversion では，ディレクトリの構成を工夫することでこれを実現している．具体的には，リポジトリの構成において標準化されたものがあり，以下に示す 3 つのディレクトリをあらかじめ作成することが推奨される．

/trunk
/branches
/tags

システム A のような開発の主流となるデータは trunk ディレクトリで管理する．システム A+α のような主流の開発から分岐したものをブランチといい，branches の下にブランチを作成する．プログラムが完成したとき，またブランチを作ったときなどのリビジョンの目印をタグといい，タグを管理するためのディレクトリが tags である．

ブランチやタグの作成方法の詳細は後ほど説明するが，ブランチもタグもディレクトリをコピーすることで作成する．すなわち，ブランチとタグに違いはなく，システム開発者がコピーしたディレクトリをブランチとして，もしくはタグとして扱うというポリシーである．

また，リポジトリが複数プロジェクトを含む場合は，Fig.3 のようにプロジェクトごとに構成をインデックス化することが望ましい．

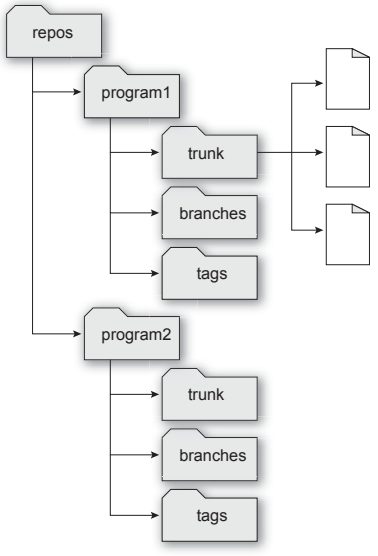

Fig. 3　リポジトリの構成 (出典：参考文献 [3])

3 Subversion コマンド

Subversion を用いてバージョン管理する際に必要となるコマンドについて紹介する．

3.1 リポジトリへの情報の入出力

リポジトリへ情報を入出力にする際に用いるコマンドについて説明する．

- svnadmin create
  カレントディレクトリにリポジトリを作成して初期化する．

```
svnadmin create [リポジトリ名]
```

  Subversion を利用する際，このコマンドで，あらかじめリポジトリを作成しておく必要がある．

- import
  バージョン管理するファイル群をリポジトリに登録する．

```
svn import -m "コメント" URL[インポート先]
```

  カレントディレクトリにあるファイルが全て，リポジトリに登録される．
  Subversion ではリポジトリの場所は URL により表現される．リポジトリにアクセスするための URL は以下のようなものがある．
  - file:///　mikilab 内でリポジトリへ直接アクセスする場合に用いる．
    （例）file:///home/svn/personal/yueda/repos
  - svn+ssh://　SSH を利用して，リモートマシンからリポジトリへアクセスする場合に用いる．
    （例）svn+ssh://onishi@mikilab.doshisha.ac.jp/home/svn/personal/yueda/repos
  - svn://　Subvesion サーバに対する独自 TCP/IP プロトコル経由でアクセスする場合に用いる．
    （例）svn://onishi@mikilab.doshisha.ac.jp/home/svn/personal/yueda/repos
  - http://　Subversion を考慮した Apache サーバへの WebDAV プロトコル経由でアクセスする場合に用いる．
    （例）http://svn.example.com/repos/
  - https://　http://と同じだが，SSL による暗号化を行う場合に用いる．
    （例）https://svn.example.com/repos/

- checkout
  リポジトリからワーキングコピーを取り出す．

```
svn checkout URL[チェックアウト元] [ディレクトリ名]
```

  リポジトリから指定した名前のディレクトリの中に，編集用にファイル群を取り出す．ディレクトリ名を指定しなければ，リポジトリと同じ名前のディレクトリが作成される．なお,「checkout」は「co」と略されたコマンドでも同様の動作となる．

- commit
  ワーキングコピーでの編集をリポジトリへ反映させる．

```
svn commit -m "コメント"
```

- export
  リポジトリからファイルを取り出す．

```
svn export URL[エクスポート先]
```

  チェックアウトと違いエクスポートしたデータは，編集用ではないのでバージョン管理用のファイルを含まない．

3.2 ワーキングコピー内でのファイル操作

ワーキングコピー内でファイル操作を行う場合に用いるコマンドについて説明する.

- add
  Subversion の管理下にファイルを追加する.

```
svn add [ファイル名]
```

  コミットするまでリポジトリには反映されない.

- delete
  ファイルをワーキングコピー内から削除する.

```
svn delete [ファイル (ディレクトリ) 名]
```

  コミットするまでリポジトリには反映されない.

- move
  「move」には 2 通り使い方があり，リポジトリに対して UNIX のコマンドの「mv」と同じ操作が可能である.
  ファイル (ディレクトリ) 名の変更を行う.

```
svn move [古いファイル名]  [新しいファイル名]
```

  ファイル (ディレクトリ) の移動

```
svn move [ファイル名]  [移動先]
```

  コミットするまでリポジトリには反映されない.

- mkdir
  Subversion の管理下にディレクトリを作成する.

```
svn mkdir [ディレクトリ名]
```

  コミットするまでリポジトリには反映されない.

- copy
  作業コピーまたはリポジトリ中のファイルやディレクトリをコピーする．なお，コピー元 (SRC) とコピー先 (DST) は，作業コピー (WC) 中のパスでも URL でも構わない．ただし，ファイルは 1 つのリポジトリの内部でのみコピー可能であり，Subversion はリポジトリ間コピーをサポートしていない.

```
svn copy SRC[コピー元] DST[コピー先]
```

  なお,「copy」は「cp」と略されたコマンドでも同様の操作となる.
  - WC → WC　追加用にファイルをコピーし，追加予告する.
  - WC → URL　WC のコピーを直接 URL にコミットする.
  - URL → WC　URL を WC にチェックアウトし，追加予告する.
  - URL → URL　完全なサーバ上のみでのコピーであり，普通ブランチ (branches) やタグ (tags) に用いられる.

ここでは，例としてブランチの作成について 2 通りの方法を説明する.「/home/svn/personal/yueda/svnsemi/trunk」の内容を「/home/svn/personal/yueda/svnsemi/branches」内の「branch1」にコピーする.

まず，いったんローカルマシンにチェックアウトする方法を説明する．

```
$ svn checkout svn+ssh://hhirao@mikilab.doshisha.ac.jp/home/svn/personal/yueda/svnsemi test
Enter passphrase for key '/home/Hiroki Hirao/.ssh/id_dsa':
A test/trunk
A test/trunk/sample.txt
A test/branches
Checked out revision 2.
```

チェックアウトして新たにできたワーキングコピー内のディレクトリに移動しコピーする．

```
$ cd test
$ svn copy trunk branches/branch1
A branches/branch1
```

最後に変更をコミットするとコピーの完了である．この際ネットワーク越しに作業コピーデータの全体を再送信しているわけではなく，ワーキングコピーをコピーすることでリポジトリに新たなディレクトリを作成している．

```
$ svn commit -m "create new branch1"
Enter passphrase for key '/home/Hiroki Hirao/.ssh/id_dsa':
Adding branches/branch1
Committed revision 3.
```

次に，チェックアウトせずに引数として直接 URL を 2 つとる方法を説明する．

・mikilab 内からアクセスする場合

```
mikilab$ svn copy file:///home/svn/personal/yueda/svnsemi/trunk ¥
file:///home/svn/personal/yueda/svnsemi/branches/branch1 -m "trunk to branch1"
```

・mikilab 外からアクセスする場合

```
$ svn copy svn+ssh://yueda@mikilab.doshisha.ac.jp/home/svn/personal/yueda/svnsemi/trunk/ ¥
svn+ssh://yueda@mikilab.doshisha.ac.jp/home/svn/personal/yueda/svnsemi/branches/branch1 ¥
-m "trunk to branch1"
Enter passphrase for key '/home/Hiroki Hirao/.ssh/id_dsa':
```

なお，これらの方法には何の違いもないが，後者の方がリポジトリの大きなコピーをチェックアウトしたり，作業コピーそのものを用意する必要がないため，容易であるといえる．

3.3 状態の確認

リポジトリの状態やワーキングコピー内の状態を確認する場合に用いるコマンドについて説明する．

- status
  ワーキングコピーにあるファイルの更新状態を表示させる．

```
svn status
```

  行のはじめに表示される大文字英字がファイルの状態を示す (Table 1)．ワーキングコピーが変更されていない場合，何も表示されない．

  「-v」オプションをつけることでより詳細を表示することができる．

```
svn status -v
```

Table 1　記号の表す状態

記号	ファイルの状態
?	Subversion の管理下にない
A	add してまだコミットされていない
M	修正部分がマージされた
D	ファイルが削除されている
C	コンフリクトしている

左から順に，ファイルの状態，ワーキングコピーのバージョン名，最後にコミットしたバージョン，コミットしたユーザ名，ファイル名が表示される．

- ls
  リポジトリ内のディレクトリエントリを一覧表示する．

```
svn ls URL
```

オプションで-v を用いると，UNIX の「ls -l」のように，詳細な情報を表示することができる．

```
svn ls -v URL
```

左から順に，最後のコミットのリビジョン番号，最後のコミットをした人，データサイズ，最後のコミットの日時が表示される．なお,「ls」は「list」というコマンドでも同様の動作となる．

- diff
  ワーキングコピー内の変更された点を参照する．

```
svn diff
```

ファイルの編集前後の削除および追加部分を知ることができる．

- log
  指定したファイルの作業履歴を調べる．

```
svn log [ファイル名]
```

変更のあった際のリビジョン番号，変更したアカウント名，変更時刻，変更操作が表示される．

- cat
  指定したファイルまたは URL の内容をチェックアウトすることなしに表示する．なお，ディレクトリの表示については，前節で示した「svn list」を用いる．

```
svn cat URL[ディレクトリ名]
```

例として,「/home/svn/personal/yueda/svnsemi」内にある「sample.txt」の内容 (ここでは「Subversion seminar」という文字列) を表示したいときの方法を以下に示す.

・mikilab 内からアクセスする場合

```
mikilab$ svn cat file:///home/svn/personal/yueda/svnsemi/sample.txt
Subversion seminar
```

・mikilab 外からアクセスする場合

```
$ svn cat svn+ssh://hhirao@mikilab.doshisha.ac.jp/home/svn/personal/yueda/svnsemi/
sample.txt
Enter passphrase for key '/home/Hiroki Hirao/.ssh/id_dsa':
Subversion seminar
```

また，以前のバージョンのファイルが見たいが，2 つのファイルの違いを見る必要はない場合には，このコマンドが有効である．外部の diff プログラムによって 2 つのリビジョンのファイル間の差分を見たい場合，古いバージョンのコピーを取得する必要がある．このとき，その内容をファイルに出力したものと，作業コピー中の両方を外部の diff プログラムに渡さなければならない．そのため，他のリビジョンとの間の差分をとるよりも，その古いバージョンのファイル全体を見る方が簡単なことがある．

3.4 マージ

- merge
  指定した 2 つのソースの差をワーキングコピーの指定したパスに反映させる

```
svn merge URL1@N URL2@M ファイル（ディレクトリ）
（N，M：リビジョン番号）
```

「URL1@N」は「N 番目のリビジョンにある URL1 の場所のファイル，またはディレクトリ」を指定する場合に用いる.「svn merge」は 1 番目の引数である URL1@N と 2 番目の引数である URL2@M との差（URL2@M − URL1@N）を 3 番目の引数であるファイル，またはディレクトリに反映させる.

trunk と branch に別々の修正を行って（branch は以前に trunk をコピーしたもの），branch の修正を trunk に反映させたい場合などに用いる.「svn merge」を用いる場合の具体例を，以下に説明する.

まず,「svn ls」によって trunk, branches, tags のディレクトリがワーキングコピー内に存在することを確認する．また，trunk ディレクトリには sample.txt があり，その内容は「cat」で出力した通りである.

```
$ svn ls
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
branches/
tags/
trunk/

$ svn ls trunk
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
sample.txt

$ cat trunk/sample.tex
int a,b,c;
a=3;
b==5;
c = a*b;
```

trunk の内容を braches 内の sampleBranch ディレクトリにコピーし，その変更をコミットする.

```
$ svn copy trunk branches/sampleBranch
A branches/sampleBranch

$ svn commit -m 'copy trunk to branches/sampleBranch'
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
Adding branches/sampleBranch
Adding branches/sampleBranch/sample.txt

Committed revision 40.
```

この変更は，コピーをコミットしただけなので，trunk と sampleBranch の内容は同じである.

sampleBranch 内の sample.txt を編集する (下線部は変更した箇所を示す).「svn status」を行うと，sampleBranch 内の sample.txt のファイルの状態が「M」であり，修正が加えられたことが確認できる．この修正をコミットする.

```
$ vi branches/sampleBranch/sample.txt
```

```
$ cat branches/sampleBranch/sample.txt
int a,b,c;
a=3;
b=5;
c = a*b;
$ svn status
M branches/sampleBranch/sample.txt

$ svn commit -m 'edit branches/sampleBranch/sample.txt'
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
Sending branches/sampleBranch/sample.txt
Transmitting file data .
Committed revision 41.
```

この処理により，リポジトリ内で，sampleBranch にある sample.txt に変更が加えられた．turnk にある sample.txt は変更されていない．次に，trunk 内のファイルを以下に変更する (下線部は変更した箇所を示す).「svn status」を行うと，trunk 内の sample.txt の状態が「M」であり，修正が加えられたことが確認できる．この修正をコミットする．

```
$ vi trunk/sample.txt

$ cat trunk/sample.txt
int a,b,c;
a=3;
b==5;
c = a*b;
printf(c);

$ svn status
M trunk/sample.txt

$ svn commit -m 'edit trunk/sample.txt'
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
Sending trunk/sample.txt
Transmitting file data .
Committed revision 42.
```

この処理により，リポジトリ内で，trunk にある sample.txt に変更が加えられた．sampleBranch にある sample.txt と，trunk にある sample.txt は，元は同じファイルであったが別々の変更が加えられている．ここで，sampleBranch 内の変更を trunk に反映させる事を考える．その時に用いるのが「svn merge」コマンドである.「svn merge」を行う.

```
$ svn merge branches/sampleBranch@40 branches/sampleBranch@41 trunk
Enter passphrase for key '/home/Hiroyuki Ishida/.ssh/id_dsa':
U trunk/sample.txt

$ cat trunk/sample.txt
int a,b,c;
a=3;
b=5;
c = a*b;
printf(c);
```

「merge branches/sampleBranch@40 branches/sampleBranch@41 trunk」では，リビジョン 40 の sampleBranch からリビジョン 41 への差分を計算し（”b==5”を削除し，”b=5”を追加），その差分をワーキングコピーの trunk に反映させる．これによって，ワーキングコピーの trunk 内の sample.txt は「cat」で表示されるように変更され，バク修正とプログラム拡張がうまくマージされているのがわかる．下線部は，マージされた箇所である．

- update
  リポジトリの変更をワーキングコピーに反映させる．

```
svn update [path]
```

ユーザ A の trunk に sample.txt を作成し，commit する．

```
userA$ vi sample.txt
userA$ cat sample.txt
      int a,b,c;
      a=3;
      b=5;
userA$ svn status
M sample.txt
userA$ svn commit -m "edit sample.txt"
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
Sending trunk/sample.txt
Transmitting file data .
Committed revision 12.
```

次にユーザ B が，先ほどユーザ A のコミットした trunk のワーキングコピーを取り出し，trunk 内の sample.txt に下線部分を付け加えコミットする．

```
userB$ svn checkout ¥
> svn+ssh://tkagitani@mikilab.doshisha.ac.jp/home/svn/personal/yueda/semirepos/trunk
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
A trunk/sample.txt
Checked out revision 12.
userB$ cd trunk/
userB$ ls
sample.txt
userB$ vi sample.txt
userB$ cat sample.txt
      int a,b,c;
      a=3;
      b=5;
      c=a*b;
userB$ svn status
M sample.txt
userB$ svn commit -m "edit sample.txt"
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
Sending sample.txt
Transmitting file data .
Committed revision 13.
```

ユーザ B がコミットした後，ユーザ A は下線部分のように sample.txt を変更し，コミットを行うとエラーが起こる．これはユーザ B の sample.txt への変更が，ユーザ A の sample.txt には反映されていないためである．そこでユーザ A は，コミットを行う前に update コマンドにより，ユーザ B の変更を sample.txt に反映させなければならない．

```
userA$ vi sample.txt
userA$ cat sample.txt
      int a,b,c;
      a=10;
      b=5;
userA$ svn status
M sample.txt
userA$ svn commit -m 'edit sample.txt'
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
Sending trunk/sample.txt
Transmitting file data .svn:  Commit failed (details follow):
svn:  Out of date:  '/trunk/sample.txt' in transaction 'f'
userA$ svn update
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
G sample.txt
userA$ cat sample.txt
      int a,b,c;
      a=10;
      b=5;
      c=a*b;
Updated to revision 13.
userA$ svn commit -m 'edit sample.txt'
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
Sending trunk/sample.txt
Transmitting file data .
Committed revision 14.
```

- resolved
  コンフリクトの際に，その修正を Subversion に知らせる．

```
svn resolved URL[path]
```

update の例でユーザ B が，コミットした後，以下のようにユーザ A が sample.txt に下線部分を付け加えたとする．

int a,b,c;
a=3;
b=5;
c=a/b;

すると，ユーザ B が変更した場所と同じ所を変更しているため，update をすると衝突が生じ，ユーザ A の sample.txt が以下のように変更される．そのため手作業により，sample.txt を正しく変更しなければならない．

```
userA$ svn update
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
C sample.txt
Updated to revision 15.
```

```
userA$ cat sample.txt
      int a,b,c;
      a=3;
      <<<<<< .mine
      b=5;
      c=a/b;
      =======
      b=5;
      c=a*b;
      >>>>>> .r16
```

ここではユーザ A が変更したように，上記の sample.txt を訂正する．訂正が終わると，resolved コマンドにより，衝突が解消されたことを知らせる．するとコミットを行うことができる．

```
userA$ vi sample.txt
userA$ cat sample.txt
      int a,b,c;
      a=3;
      b=5;
      c=a/b;
userA$ svn resolved sample.txt
Resolved conflicted state of 'sample.txt'
userA$ svn commit -m "clear conflict > sample.txt"
Enter passphrase for key '/home/Takehiro/.ssh/id_dsa':
Sending trunk/sample.txt
Transmitting file data .
Committed revision 17.
```

3.5 修正の取り消し

- revert
  ワーキングコピーでの修正を取り消す．

  ```
  svn revert [ファイル名]
  ```

  ワーキングコピーのファイルを修正前の状態に戻すことができる．コミットを行っている場合は，コミット直後の状態に戻る．コミットを行っていない場合は，チェックアウト直後の状態に戻る．

3.6 ヘルプ

- help
  Subversion の help を表示する．

  ```
  svn help [コマンド名]
  ```

  引数のコマンドの help を見ることができ，そのコマンドに使用可能なオプション一覧も確認できる．

参考文献

1) 木田 清香，千田 智治，村上 耕平．2006 年度 UNIX ゼミ Subversion マニュアル．

2) 荒久田 博士，折戸 俊彦，市川 親司．2003 年度 LINUX 開発アプリケーションゼミ資料．

3) サブバージョンによるバージョン管理．http://subversion.bluegate.org/doc/book.html#svn.branchmerge.using.create