

Software Tools Subversion and PDStore

Part II - Lecture/Lab 5

Today's Outline

2009

YEAR

COMPSCI 732

The University of Auckland | New Zealand

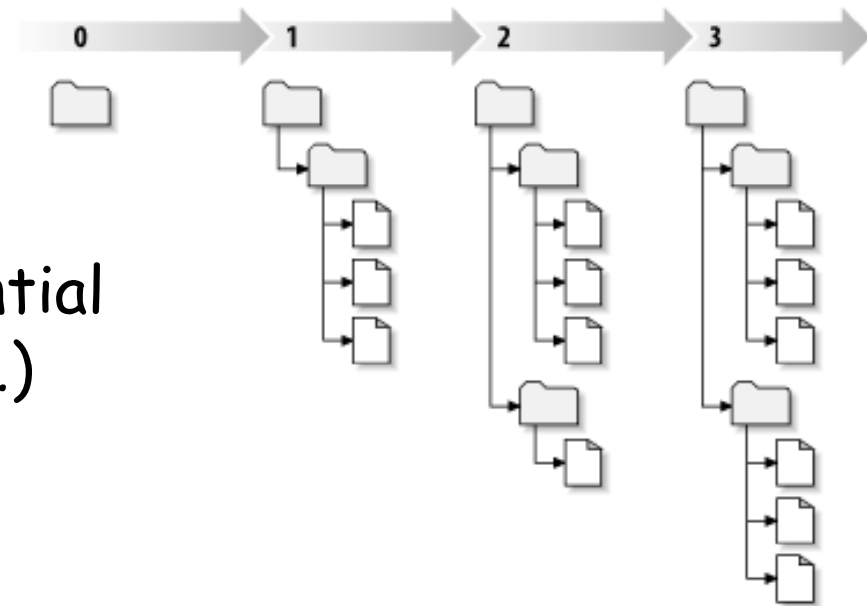
- Subversion (SVN)
- Setting Up Your Project
 - Subversion
 - PDStore

Subversion (SVN)



Subversion (SVN)

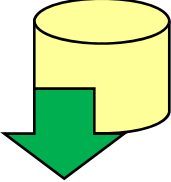
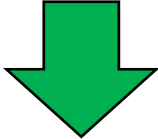

- Centralized open-source VCS; started in 2000
- Developed as replacement for the Concurrent Versions System (CVS) which started 1986
- Subversion filesystem versions files and folders ("three dimensional" filesystem)
- Each change creates new revision of the whole file/folder structure
- Revision names are sequential natural numbers (0, 1, 2, ...)



Subversion Features

- Supports merging (recommended) as well as locking
- Changes are transactions
 - Atomic: completely or not at all
Change is either committed and becomes the latest revision, or is aborted
 - Interrupted commits do not corrupt repository
- Complete file/folder structure is versioned, including renames and file metadata
- Delta encoding and merge algorithms work also with binary data
- Costs are proportional to change size, not data size
- Works with HTTP server: WebDAV/DeltaV protocol makes it possible to read repository with just a web browser

Basic SVN Operations

- **Checkout:** create a working copy of a repository
 - Choose local folder for working copy
 - Enter the URL of the repository
 - Choose the revision to check out (HEAD revision is latest one)
- **Update:** update your working copy to the latest revision
 - If no newer revision exists: no effect
 - If you have changed your working copy: latest revision will be automatically merged into your working copy
 - Textual merging conflicts have to be resolved manually
- **Commit:** write local changes to the repository
 - Fails if your local revision is out of date; update first
 - Creates a new revision on success

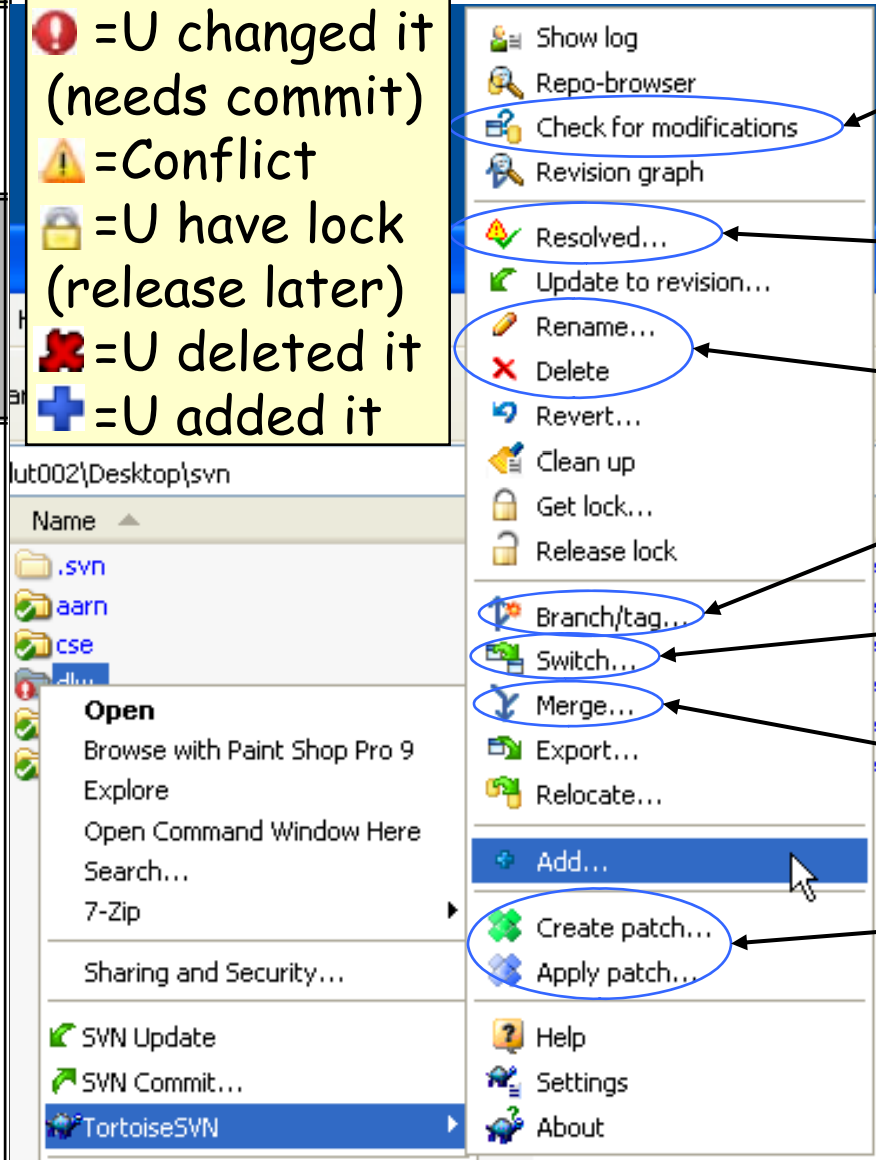
TortoiseSVN

2009
YEAR

COMPSCI 732

The University of Auckland | New Zealand

- ✓ = Unmodified
- ! = U changed it (needs commit)
- ⚠ = Conflict
- 🔒 = U have lock (release later)
- ✖ = U deleted it
- ⊕ = U added it



Check if somebody else has modified files or has acquired lock; also for stealing locks

Tell SVN that conflict in files has been resolved

Use these instead of normal ones!!! Also updates version info.

Create cheap copy of a folder.



Switch to the version in a cheap copy (like updating to it).

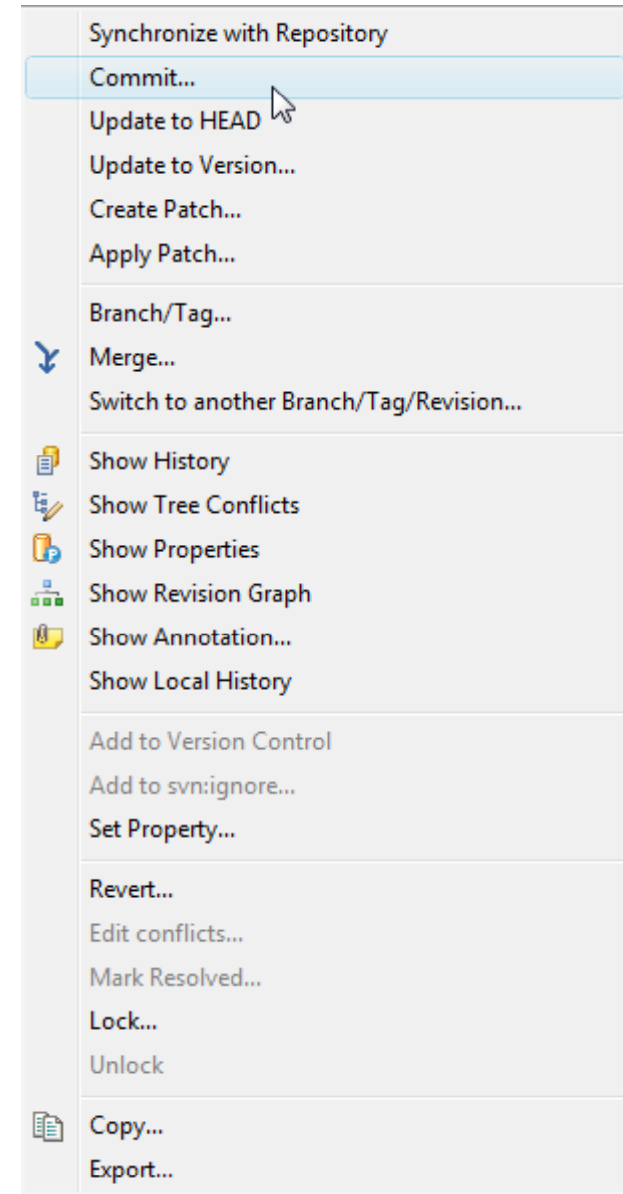
Merge revision range of branch into other branch.

Creating a file containing the local changes or use it to update working copy.

Right-click & drag = copy/move & update version info

Subclipse

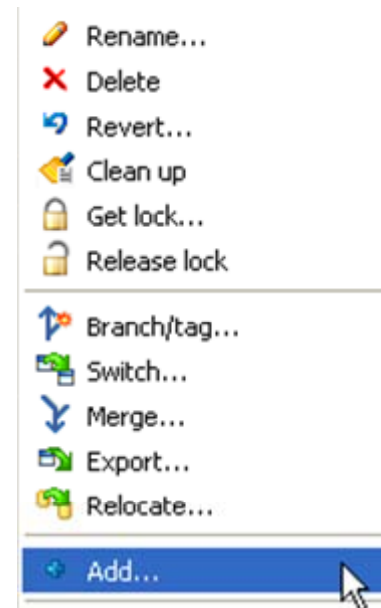
- SVN plugin for Eclipse, very similar to TortoiseSVN
- Package Explorer will show overlay icons for files, e.g.  
- Right-click on a folder or file and select SVN command from the "Team" submenu
- The commands are just the normal SVN commands, i.e. the way you have to use it is the same (still need to add files to repo explicitly)



Add, Delete, Rename, Revert

Add file/folder to the repo

- All new files/folders need to be added explicitly to the repo
- Only add source files (e.g. not `.class` files)
 - Other files are generated from them
 - Take up space and are hard to merge



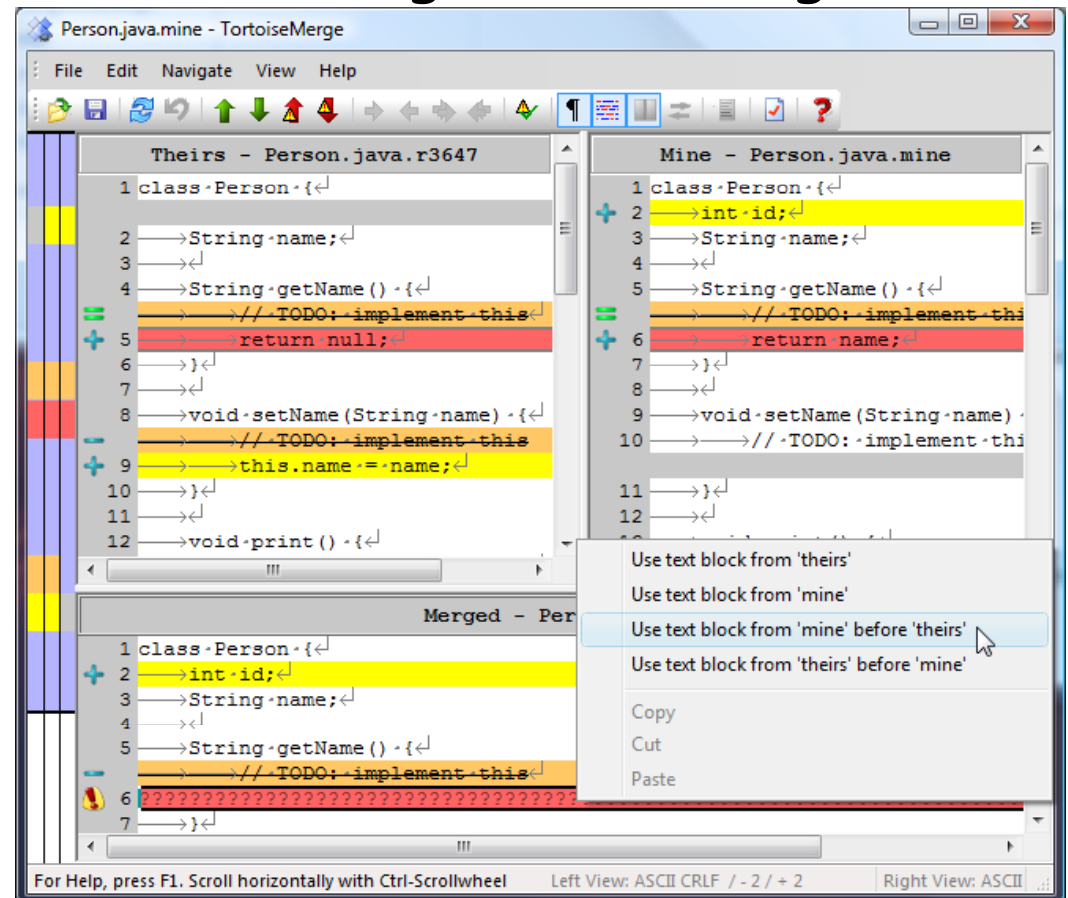
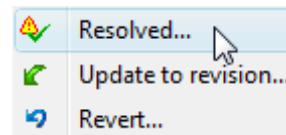
For **deleting** and **renaming** files/folder in the repo use the SVN commands (don't delete/rename directly)

Revert local changes in a file/folder if you want to go back to the last version you got from the repo (i.e. throw away local modifications)

Resolving Conflicts



- After updating SVN might tell you someone committed a change that conflicts with your local changes
- Resolving the conflict means deciding how to merge the conflicting changes
- Supported by editor that shows conflicting changes and gives options to resolve it (e.g. use only one of two changes)
- When conflict is resolved, you must tell SVN



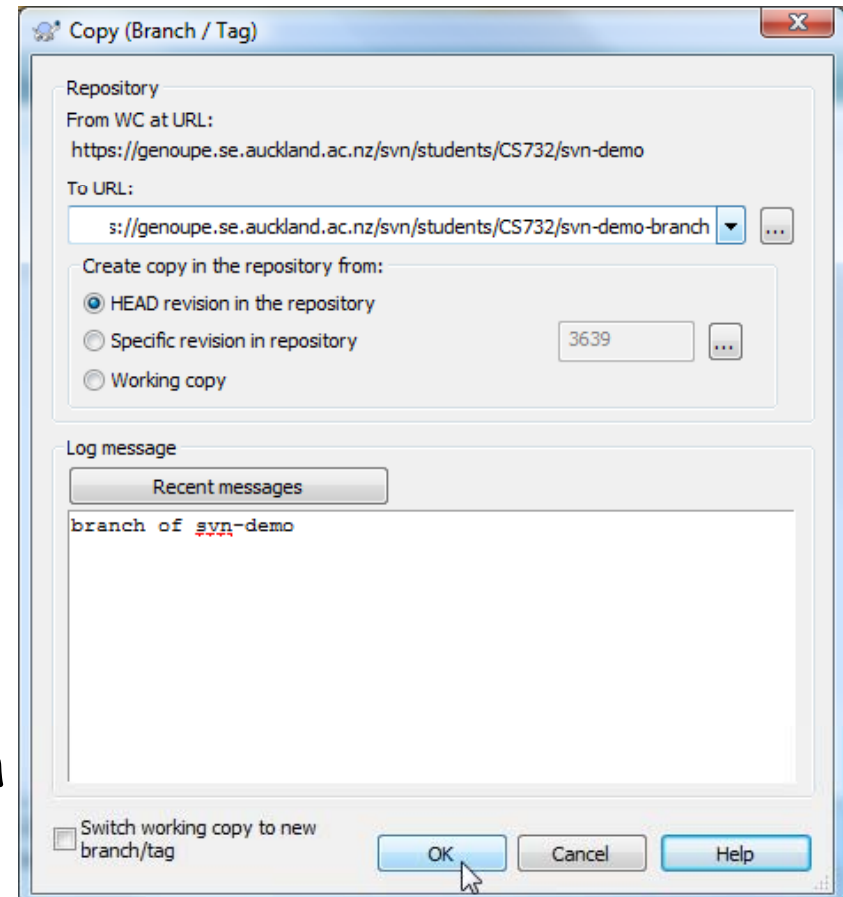
Branching / Tagging



- Creates a copy of a folder in your repository
- **Branch:** the copy will be used for further development
- **Tag:** the copy is just for archival and will remain unchanged

How to do it:

1. Select folder to copy from (right-click on it, use menu)
2. In the dialog: select new folder to copy to
3. Select revision of that folder (usually HEAD)
4. Enter log message
5. Update parent folder of branch into the local working copy

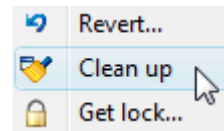
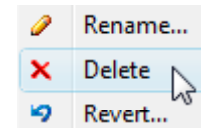




Subversion Tips



1. Don't forget to add your files/folders to the repo
2. Don't add files that are generated (e.g. `.class`)
3. Careful with binary files, they can be impossible to merge (so they should be locked)
4. Delete and rename only using SVN operations
5. If two SVN clients are running at the same time, there might be errors like "working copy locked"
6. If something is wrong with working copy, use cleanup command
7. If nothing else helps, delete local working copy and check out a new one
8. Write log messages!!! (or you won't find versions)



Subversion References

Version Control with

Subversion

- B. Collins-Sussman, B.W. Fitzpatrick, C.M. Pilato. *Version Control with Subversion*. 2008.

<http://svnbook.red-bean.com/>



TortoiseSVN

- TortoiseSVN Manual
 - Press F1 key in any TortoiseSVN dialog for help
 - Manual can also be found online:
<http://tortoisesvn.net/support>

Setting Up Your Project



Setting Up Your Project

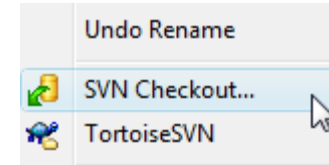
1. Choose and download a grammar from

<http://wwwantlr.org/grammar/list>

2. Checkout your own local working copy of

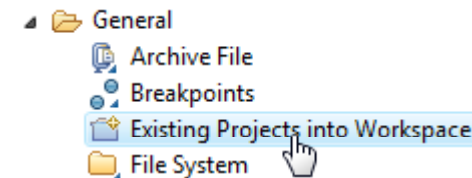
<https://genoupe.se.auckland.ac.nz/svn/students/CS732/projects/>

3. Create a branch of `projects/Example` in `projects`
Name you branch after the language you chose.



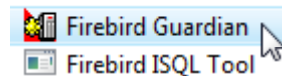
4. Open the project in Eclipse

- Choose `projects` as workspace
- Import the project into the workspace



5. Start the Firebird Guardian:

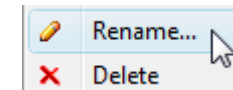
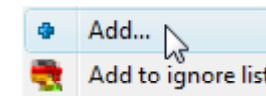
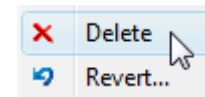
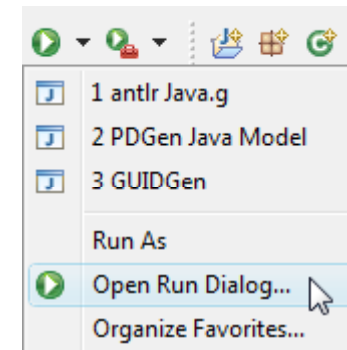
Start → Development → Dev. Env. → Firebird → Fb. Guardian



6. Run the `reset-pdstore.bat` in your project folder
A database `pdstore.fdb` will be created

Working on Your Project

1. Have a look at the Eclipse run dialog
 - Run PDGen on the Java Model
 - Run GUIDGen
2. Delete the `Java.g` grammar file
3. Add your downloaded grammar file
4. Rename `java.sql` to `yourlanguage.sql`
5. Change `reset-pdstore.bat` so that it refers to *yourlanguage* instead of `java`
6. Start changing `yourlanguage.sql` so that an AST model for *yourlanguage* is created (using fresh GUIDs)
7. Run the `reset-pdstore.bat` in your project folder
8. Run PDGen on the *yourlanguage* Model



```
del .\pdstore.fdb
fsql\fsql -i pdstore.sql 2> pdstore-errors.txt
fsql\fsql -i java.sql 2> java-errors.txt
```