Using CVS in Eclipse

CVS, or “Concurrent Versioning System” is a popular piece of version control software. Essentially, what this means is that it allows multiple programmers to work on the same project with a minimum of toe-stepping. This makes it useful for projects in 335. Even better, Eclipse provides an easy interface to use with CVS. This tutorial consists of:

1) How to obtain a CVS repository from the lab
2) How to link to that CVS repository from Eclipse
3) How to interact with Eclipse’s CVS interface

Step 1: How to obtain a CVS repository from the lab

In order to create a CVS repository, a unix group must first be designated that has access to that repository, and a location for your repository must actually be created on disk. All of this is taken care of by our lab staff, provided one of your team members does this:

- go to http://www.cs.arizona.edu/computing/services/
- Click Authenticate
- Enter your UANetID and Password
- Select this service option
  Create a Unix GROUP (CVS area and MySQL database) on the CS network

This will bring up the webpage which is shown below. Enter these inputs:

1. Group Name: This represents the project. It must be unique to others hence the letters of each team member’s first name followed by Fin to indicate Final project
2. Members: The UANetIDs (CS logins names) of all team members
3. Select YES for CVS directory
4. Leave MySQL option as NO

On-Line Services

Richard H. Mercer,

---

1 You will have a repository seen from a linux machine in /cs/cvs/<Group name>
2 Please note that you will have to wait for some time--about one hour--before your CVS repository is fully set up
Step 2: How to link to that CVS repository from Eclipse

Be sure that you have waited the proper length of time for the repository to be built before you attempt to plug your repository in. There’s nothing worse than trying to find something that doesn’t exist.

- Start Eclipse
- Select File > New > Other > CVS (expand) > CVS Repository location > Next

The following window should appear.

You now should be all set up with a repository. Note that this step must be done by all team members in each Eclipse workspace you use (on your laptop and in the lab for example), not just the team leader, as above.

Your repository is now known in Eclipse, but it is void of content.
Step 3: How to interact with Eclipse’s CVS interface

Only one Team member needs to do this:

In Eclipse, create a new project if you do already have a start. The name of this project should be something meaningful.

Right click on the project folder and select Team > Share Project

A series of screens will pop up, but you should just use the default options provided. Use the existing repository you created in step 2, and use the module name as the project name.

Congratulations, you have just checked a project into the repository.
For all other members:

In Eclipse…
Go to File > Import > Checkout project from CVS

Select “Use Existing repository location” and when the list of projects pop up, select the name of the project you will be working on (NOT CVSROOT).

Click the Finish Button

Congratulations, you have just checked a project out from the repository.

Step 4b: Working with CVS

There are 2 basic operations in CVS:
   1) Update (replace your local copy with the copy in the repository)
   2) Commit (replace the repository copy with your local copy)

Whenever you want to interact with the repository, you should right click your project in the project explorer, and select Team > Synchronize with Repository. This will bring up the “Team Synchronizing perspective”. To get back to the Java perspectve, look near the upper right corner of the screen and change the option marked there to “Java”.

When the Team Synchronizing perspective is open, you will be presented with several different types of arrows:

A right facing gray arrow: You have code to be added to the repository. This code should be committed

A left facing gray arrow: Someone has deposited code in the repository that you have not yet seen. This code should be updated

A left facing blue arrow: Someone has modified code in the repository that you have not been working on. This code should be updated

A right facing blue arrow: You have modified code and nobody else has checked in a version of that code. This code should be committed.

A double headed red arrow: You have modified a local version of a piece of code, but a different version exists in the repository. Perhaps two people were working on it. There is really no “right way” to resolve these conflicts, so you should discuss it with your team. Though complex resolution tools are available, my advice is to pick the “better” version. If that is the local copy, do an override and commit. If the “better” copy is the repository copy, do an override and update.