UCL DEPARTMENT OF COMPUTER SCIENCE

Wolfgang Emmerich, Dr rer-nat, CEng, MIEE
Professor of Distributed Computing
Director of Research

GS04: Tools and Environments

Lab Session 5: Using Subversion

The aim of this lecture is to experience to apply the principles of SCM using an IDE. We will use Subversion (http://subversion.tigris.org), which is installed on most CS machines. Subversion has a command line interface but software developers find it more convenient to use it through their interactive development environment. We will use an Eclipse plug-in that provides a user interface to Subversion for the JDT platform. Because we want to explore how Subversion enables teamwork work on this lab session in pairs.

Checking a project into Subversion for the first time

We have set up a repository for use on this course. Subversion uses URLs to identify repositories and our repository is at svn+ssh://collins.cs.ucl.ac.uk/cs/student/misc0/stud/mscssep/2007/repo. Use the SVN Repository view to create a repository directory for your projects. You might have to open using the SVN Repository view using the Window -> Show View menu. Give it your name (e.g. I would have chosen 'w.emmerich'). Create a new repository location with your Subversion directory for use below.

Now use the model solution of Lab 2 that you should still have in your Eclipse workspace and upload it into Subversion. You can do this using the "Share Project…" menu item of the Team pop-up menu on the project in the Explorer view. Select SVN rather than CVS and use the repository location you have created above. Use the project name followed by '/trunk' as the Subversion directory name. Note how the explorer view has now been augmented with version and configuration information.

Shared access to a project via Subversion

On a second machine and under the login of another team member delete the project with the Model solution of Lab 3 from the workspace and reload it from subversion. To do this you want to create a new project and select to check it out from Subversion. Use the URL you have chosen above to identify the project source.

Controlled update of configuration items

Now work on two different Java source files in parallel on the two machines. For example add a number of Java comments to improve the documentation. Then the first team member commits their changes (using the commit command from the Team menu). And then the second team member commits their changes. Note how the second team member sees the changes made by the first member (but not vice versa). To allow the second member to see the changes of the first member use the update command.

Next we see how subversion behaves if we create an update conflict. To force such conflict, modify the same class in the two different workspaces in parallel. To start with perform modifications in two different methods. Then commit the changes one after the other. Note how the first commit succeeds and that the second commit fails. What is required now of the second commit is to merge the changes. To do this issue an update command from the Team menu. Because the two modifications were done in different parts of the file the update was able to merge the two changes automatically. Now commit the merged class and update the other workspace, too.

Next we create an update conflict that cannot be solved automatically. Create two modifications in the same source line. Committing the first modification will succeed and the second modification will fail. Now update the second workspace and edit the conflict (from the Team Menu). You will see the two versions side-by-side with the conflicting changes highlighted. You will then need to manually merge in the changes of the committed version and declare that you have resolved all conflicts (again from the Team Menu).

P.T.O.

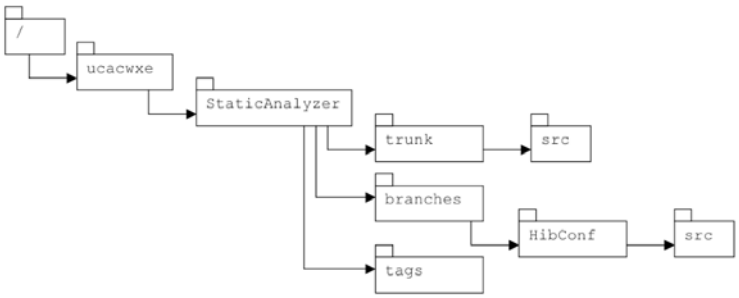
University College London Gower Street London WC1E 6BT
Tel: +44 (0)20 7679 4413 (Direct Dial) Fax: +44 (0)20 7387 1397
www.cs.ucl.ac.uk/staff/W.Emmerich
w.emmerich@cs.ucl.ac.uk

Branching

Now assume that one of you wants to implement a new static analysis feature that also checks the integration with the Hibernate configuration file while the other one continues to improve the documentation of the main trunk. You therefore want to create a branch for the new feature so that the two of you can work in complete isolation for a while.

First of all follow the Subversion convention to store any development configurations that are not part of the main development stream in the 'branches' remote folder. Use the SVN Repository browser to create a 'branches' remote folder and a further 'NewMetrics' remote folder therein for the feature development configuration you want to undertake.

So one of you who wants to develop the new feature can use the 'Branch/Tag' command in the Team menu to establish a repository structure that looks somewhat like the one shown on the right. Then modify the classes and note that now both of you can commit independently of each other. Try this out by performing a concurrent update that would lead to a conflict and note that you can commit independently. Now add the new metrics feature in the branched workspace.



When the feature development is complete you will have to merge the changes done on the NewMetrics branch back into the main development trunk. You do this using the 'Merge' Command from the Team menu. Choose the two different URLs of the trunk configuration and the branch configuration and merge the head revision of the branch into the trunk. It is important to commit both the branch and the trunk workspace before this operation.

University College London Gower Street London WC1E 6BT
Tel: +44 (0)20 7679 4413 (Direct Dial) Fax: +44 (0)20 7387 1397
www.cs.ucl.ac.uk/staff/W.Emmerich
w.emmerich@cs.ucl.ac.uk