# CVS / SVN Practise

Wouter Borremans, Maarten Michels, Thijs van den Berg

October 6th, 2004

This document describes the findings on using CVS / SVN during the ESA practise.

## 1 Getting a working CVS

- To make sure everything is working as it should we need to create a directory: mkdir cvs
- export ="/Users/mmichels/cvs"
- cvs init
- Go to a directory with all the files that you would like to have a repository of and do: cvs import <cvs module name> <author> <ver1.0>
- Make a new directory in a higher folder (not in the directory where you created the cvs import or cvs root): mkdir google
- cvs checkout <cvs module name>

Now you can edit your files. To get the file with the changes back into the original folder sync first the repository (when working with more users on the same project)

- cvs update

after this you can add the changed file to the project

- cvs commit <filename>

Now comes the nicest thing ever, you your self must copy the changed files to the directory where your original files are.
That's IT!!!!

## 2 CVS vs SVN

This chapter describes some differences between CVS and SVN.

- *Version numbering* One of the biggest differences between CSV and SVN is that the numbering of files is handled differently. CVS uses a file independent numbering, this means that every file in the repository gets its own version number. SVN uses another version scheme which only numbers a whole project tree. This may come in quite handy looking at projects on themselves.
- *Conflict resolution* Once a conflict has been detected between files that are committed, these files are skipped by CVS. CVS continues committing other files that may be present. This can confuse project teams significantly, CVS simply continues with only small error messages. SVN uses another approach in this case, it simply will stop committing all the files to the repository.
- *Introduction of more flexible commands* The "remove" command in SVN has been expanded in its capabilities, it is now capable to remove directories instead. The move commando is newly implemented in SVN, this makes the user able to rename files without using the version numbering.
- *Usage of Metadata* SVN offers metadata functionality. In practice this means that additional properties can be added to files or directories within the repository.

## 3 Setting up an CVS server

Since we used freebsd for our cvs server we where able to use a document found at : $http://www.sullust.net/docs/quickcvs.html$ This however exposed our server to several security risks. There for we did not use the cvs server option and started to use cvs over ssh immediately.

On the server side we created a user cvs. In the directory $/usr/exports$ we have created a directory cvsroot. In this directory we used cvs init to make a new CVSROOT directory. With cvs import $< cvsmodulename >< author >< ver1.0 >$ we created a new module.

On the client side the following export variables where set:

export CVS_RSH=$'/usr/bin/ssh'$
export CVSROOT=$:ext:cvs@freebsd.sicilia.os3.nl:/usr/exports/cvsroot$

Now we can use all the usual commands like

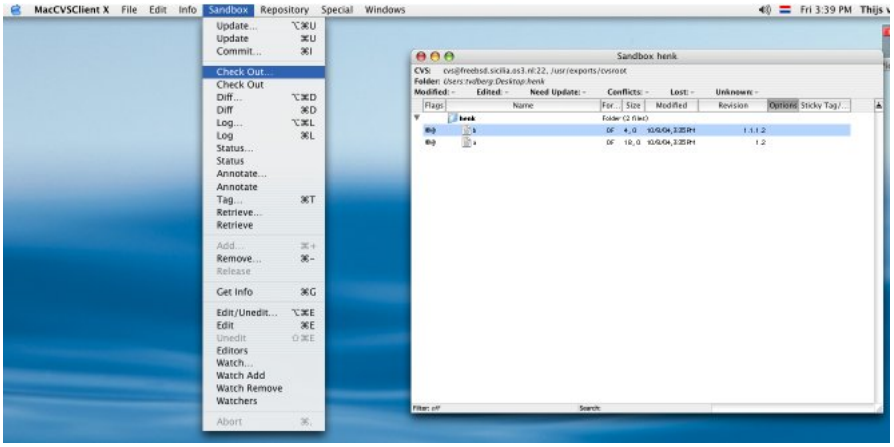
cvs checkout henk
cvs update
and cvs commit

on the client to get the the module contains, make changes and write changes back to the cvs.

# 4 Research into CVS/SVN GUI implementations

For this assignment we downloaded and installed the program MacCVS-Client 1.9 for Mac OS x.

### MacCVSClient 1.9




Figuur 1: Screenshot van een MacCVSClient

The first thing to do when using MacCVSClient is to set the login profile. We have MacCVSClient setup to connect to our freeBSD machine through SSH. After the client is connected we are able to checkout a module from the server to our local machine. This is done in the following way:

- In the menu we go to “Repository” and then to “Check out”.
- Then we choose a module, a local directory to export the module to and click on the ok button.
- Now we go in the menu to “File” and then to “open Sandbox”.
- When we have done this the screen, as shown in the screenshot, appears.

- Now we can use all the main CVS functions like add files, commit files etc.

We found MacCVSClient very easy to use.