Process Documentation

CVS and Software Builds

This document looks at CVS and how it was used throughout the project. It also looks at how software builds from CVS were automated as part of the development cycle.

Revision History

28/03/01	Initial creation				
		Committed by:	tdb1	Verified by:	ajm4
				Date:	29/03/01
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	

Introduction

CVS was introduced at the start of the project as a method of centrally storing all files, whilst maintaining a full revision history. It was rapidly built up to become the central resource for the project, along with the website. As the project progressed, automated scripts made use of CVS to generate latest builds of software to be placed on the website.

Use of CVS

The group decided to use CVS from the start of the project, as an early start would save hassle later on. It was decided that everything would be put in CVS; source code, documentation, minutes, webpages and any other generated files.

There were some key benefits to doing this. First off it gave a central location to store all files, which resided on a secure and backed up machine. It was important, from a risk analysis point of view, that all group members had access to all the files. This also ensured that files were kept in a neat and tidy structure, rather than a scattered arrangement of files in user home directories.

The second major benefit was accountability and tracking. It was possible for any group member to view the progress of other group members, and also view the actual changes they had made.

Being able to view the changes between revisions meant that problems could more easily be pinpointed. It was possible to check the changes since the file in question last worked, and review this more closely.

Documentation

Early on in the project documentation was produced on how to get going with CVS. This document became a referral page for using CVS, and proved quite useful when setting up a CVS client on other computers.

Research was done into how to setup WinCVS to work with a unix CVS repository, both on and off the campus using ssh. This information was made available on the website for all group members to make use of, and indeed for anyone else wanting to do the same.

Setup and Structure

The CVS repository was stored on raptor, which was a central and neutral location. It was mirrored to a second machine, killigrew, to ensure constant availability if raptor became unavailable.

Use was also made of the web based interface, cvsweb, which allowed the group to view changes between files in a much more aesthetically pleasing manner. This was also made available to people outside of the group, as no similar projects existing to cause plagiarism issues.

The CVS repository was divided into key sections, each containing their own subsections. This made organisation of files much better, and made navigation a simpler task.

The following sections existed; documentation, experimental, source and web.

documentation

The documentation section of CVS was purely for textual documents, such as minutes and specifications. Towards the end of the project it quickly became populated with user and maintenance documents as they were produced.

As most of these documents were wanted on the website, the document subtree was automatically placed onto the website by the CVS update scripts. These documents were mostly in ASCII text form, and were presented through the document viewer CGI script.

experimental

The experimental tree was used for testing out new subsections of code before plugging them into the overall system. An example of such a piece of code is the Queue. This was developed in the experimental tree, and underwent testing. When complete, it was moved into the main source tree and added to the system.

It was envisaged the experimental tree would be used more than it was, even for simple things such as testing out ideas. This didn't happen as much as was hoped, but good use was still made of it.

source

The source tree contained all the code from the project. The ultimate aim was that the source tree would contain the "final product" files, and nothing else. This section was broken down into subsections for each part of the system, and then further if required.

Areas of the source tree were automatically built using Makefile's. This will be discussed later.

web

The web section of CVS contained, as you would expect, the project website. All the files, including CGI scripts were kept in here, and updated whenever the website needed changing. A set of scripts were made available to the group members to automatically transfer this tree from CVS to the website, allowing changes to be made available quickly.

Ease of remote use

CVS works by having a local copy of the files in the repository (or a subsection of it). You then edit these files and send changes back to the server. This method of working comes into it's own when you don't have an ethernet link to the CVS server. You simply update your local copy, and then access the files locally, which saves all the hassle of manually uploading files using ftp (or similar).

Backup and Mirroring

As was previously stated, the main CVS repository was located on raptor. This machine is backed up by the computer science department who manage and run it. However, was this machine to become unavailable, it would have caused considerable problems. As such, backups of the CVS repository (and other files in the project directory) were done to two independent machines in another building. These machines in turn have their own backup procedures, ensuring even further redundancy. There was also a continuous mirror of the CVS repository on to killigrew, which allowed work to immediately continue if a major fault had occurred on raptor.

With all these backup strategies in place the group were confident that they would not run in to any major problems caused by system failures.

CVS Logs

The CVS Logs (viewable using the cvsweb interface) allowed anyone to view changes to any file in the repository, including who made them, when, and why. This information was useful if you knew what you were looking for, but if you wanted to see the latest changes, it wasn't so straightforward. To solve this, the following were implemented.

Daily E-Mail Logs

At midnight every day a script ran to generate CVS Logs for the previous 24 hours. These were broken down into sections (e.g. source, documentation) and sent out as e-mails to the group mailing list. This ensured that all group members were aware of progress during that day, and could keep better track of what each other were doing.

Web-viewable Latest Logs

After the e-mail logs had been running for a while it became necessary to see the absolute latest changes, especially as the end of the project neared. To enable this to be a done, a similar output to the e-mail logs was generated dynamically by a CGI script, and was made web visible. The only difference being that it showed from midnight at the start of the current day up until the present moment. This enabled group members to clearly see what was going on, which further aided collaborative working.

Software Builds

An area of the i-scream website is devoted to software downloads. From this page people can download the latest builds of the various components of the i-scream system. This proved invaluable when testing the product amongst other people at UKC, as we only needed to point them towards the download page. This helped encourage people to use the software, and gave invaluable feedback on the use of the various programs, and generated helpful bug reports.

The whole process was automated, and this will be described here.

Daily releases

When the project became stable it was decided to put daily releases onto the website for people to download and fiddle with. This process could obviously not be completed manually, so automated scripts were produced to do this. They were setup to run at 7am, a time in-between coding sessions, which was when code was most stable. The built archives were automatically uploaded to the i-scream website, and made available for download.

This process worked by making use of the common Makefile setup used throughout the system. Whilst this meant it would only work on unix based platforms, it did provide a powerful solution which worked well for the group.

Common Makefile Setup

The majority of the i-scream system had a Makefile system setup. These were all designed to have the same external behaviour, such that their use could be automated easily.

Each Makefile provided a cvsbuild target which would extract and build a given revision from the CVS repository, then place an archive of this build in an expected location with an expected filename. The scripted build process then just needed to run this target and copy the archive file away on completion. This process would be repeated for every component in the i-scream system that supported this Makefile setup.

Javadoc

A further feature of the Makefile system, for Java components, was the ability to automatically produce javadoc webpages for all the code. This functionality was attached to the build script and archives of the javadoc pages left in an expected location. These archives would be picked up by the scripts used to automate managing of the website from CVS.

Automated website management with CVS

As has been discussed, the website was entirely stored in the CVS repository. This introduced a significant dilemma; how the website would get from CVS to the off site webserver.

This was solved by producing a set of scripts to automatically manage this. The first script would fetch the latest copy of the website from CVS and place it into an archive file. It would then transfer this file to the off campus webserver to await extraction into the appropriate directories. Including in this archive would be the javadoc archives, extracted, that were generated from the daily build process.

The next step is to extract this archive onto the webserver. This is done through a second script that simply extracts the files over the existing ones. The update would then be complete.

This process wasn't ideal, but solved the problem quickly and easily. Firstly it would have problems with files deleted from the repository; these would never get deleted from the website. Secondly it uploads the entire website every time this process is run, but this wasn't a major issue as the website wasn't that large.

These scripts were made available from a CVS administration page on the website, used only by the group members.