Tools for Configuration Management

Jon Malmquist
D01, Lund Institute of Technology, Sweden
d01jm@efd.lth.se

24th February 2004

Regardless of what methodology a project is using, good tools for configuration management (CM) can make the average day of a developer a whole lot better.

Even though the developers of a project may be the smartest persons to ever walk the earth, problems can arise from such simple things as misunderstandings or lack of sleep. This study is about how CM tools can help projects go smoother and with less hassle for the people involved.

This study concerns exploration of tools for CM, what their most interesting features are and an evaluation from both a developer's and a coach's point of view. The main tools that will be evaluated are the Concurrent Versions System (CVS), the Perforce System and Subversion.

Contents

1 Introduction

As software systems today become larger and larger, it is necessary to involve more and more people in the development. When a lot of people are working together, there is a high demand for good tools in order for the work to be as efficient and unhindered as possible.

In response to this, several tools for CM has been created over the last years. While they all have the same goal, to let people work together more efficiently, they all come from different sources and different developers with different goals and visions.

In this study, I will examine a few of the largest tools available today and point out the advantages they have. I will also include the experiences I have gathered from using them. The study will not go deep into the technical aspects of how you do something, merely that it can be done.

The part of the study concerning CVS is based on experiences gained in project courses during my studies on LTH. Mainly, this and last years Enduro project and another project called the WebCam project.

The Perforce system was tested during the Christmas break and some time into January, when the author was participating in a project outside of LTH called AirSoftMaster. This project involved several people who have not studied any form of CM.

The Subversion system was tested on a private server with a dummy project.

2 What is Configuration Management?

The development of software is a task that often involves a lot of people. And the more people that are involved, the more questions will arise. CM is the art to minimize the technical questions and making it easier for people to work together on a technical level. CM is the art to try and keep developers from having to ask what has and what hasn't been changed and who is working on what.

CM is the art of keeping the developers calm, not needing to worry too much about the technical part of their jobs and letting them focus on the developing part of their job. This while ensuring that they are as efficient as possible.

2.1 Getting people to work together efficiently

When a lot of people is going to work together, you naturally can't just throw them all in there and hope that they'll get the work done. Even with good project leaders, a project can come to a sudden halt just because of a minor technical detail.

The first step for efficient work is that everyone knows the basic rules for the project. Should the code be written in a specific way? Are there steps to be taken before adding changes to the existing code? Are there tests to be run?

If everybody is working in their own way, things are bound to clash sooner or later. Just a small thing as different editors that use different ways to handle tabs can make merging a file a lot worse. This part of CM is more about discipline and obeying the rules, but there are a number of tools out there that can help smooth the edges. For example, there is a program called CheckStyle that goes through the code to see if the code obeys predetermined coding standards. If everyone is coding in the same way, everyone has an easier time reading what others have done.

If more than one person needs to work in the same file, it will be a good idea to have them coordinate their work. This can limit the number of unpleasant surprises and confusing situations that otherwise can occur. The Perforce system (to name one - more about this system later), does a good job in helping with this, since it lets others know that a file is being worked on. Thereby, you can look that person up to see what he is doing.

2.2 Versioning control

One important aspect when multiple programmers are working on the same piece of software is the changes everyone does. For what can be more irritating to a programmer, than when he has successfully written all those hundreds of lines of code (during that long night, working overtime) only to realize the next day that his coworker, who was working on the same file, overwrote it all when he saved the file? To know how many changes that has been introduced into a file, version numbers can be used. Whenever a change is made, the version number is increased.

However, when working on a file, it's important to know whether you have the latest version of the file or not. Even if you use version numbers, there must still be some central place where anyone involved with the project can see what version contains the most recent changes. And even if this is used, there is always the risk of changes being lost and overwritten.

Here are some of the methods that can be used when several people are working on the same file.

2.2.1 One works at a time

This approach works on the notion that any file can only be worked upon on one machine at a time. This prevents anyone from accidently destroying the work of another developer, but also means that the development itself is severely hampered. The developers who are not working has to wait until the torch is passed to them, which can take quite some time. In the meantime, they may not be adding much to the project.

2.2.2 Only one may merge

Here, a lot of developers can work at the same time. Whenever the developers feel that they are ready, they send their work to a designated merger, who's job it is to make sure that their work gets incorporated into the main project. While this makes the development faster than the previous example, the merger can quickly become overloaded if many are sending in their work at the same time. And if there is a lot of developers, the project may have to be broken down into several subprojects, which are then merged later. Also, the merger has to make sure that the developers know when the files have been altered, since any alteration may (and probably will) affect them.

2.2.3 Using a tool

A good tool which keeps track of the latest files and logs any changes made to them can be very helpful - especially for larger projects. While not removing the need for the team members to communicate, it can let the developers feel confident that the features they are adding won't mysteriously disappear because someone didn't get enough sleep last night. Since it also let's everyone take part of the whole project, it helps the developers to see the whole picture at any time.

3 The tools

By now, there is a sea of different CM tools to use. They all have their own features which have been added to satisfy the visions of their developers,

which may or may not be the same as the real users. Here is a couple of them and a mention of some of their features.

3.1 Concurrent Versioning System

The Concurrent Versioning System (CVS) is a server-client system intended to allow multiple developers to work together on a project. The system stores all development files on a central server, from which the developers then can retrieve any file. This repository also stores all the versions of any file along with the changelogs of the files. CVS is Open Source software, which means that all development is on a voluntarily basis.

CVS is based on command line inputs, and has no graphical components. However, there are third-party programs that adds the possibility of using CVS in a graphical environment.

In CVS, the client is the one that does all the real work. It is responsible for checking that the files it sends to the server are the latest version. It's the client that is responsible for the merging of different files and alerting the user if something goes wrong.

3.1.1 Working with CVS

When a developer wishes to work on a project, he uses the CVS client to check out the project from the server. He will then receive the latest version of all files associated with the project. These files are then stored locally in his workspace. He can then make changes to the files.

After working on a file, the developers commit the files to the repository. CVS will then compare the new files against those in the repository, after which it adds the changes. If this is successful, the files are stored on the server and the version history on the server is upgraded with the new version numbers and the comments from the developer.

If CVS finds that the files on the server has been altered since the developer checked them out, the developer must update the files in his workspace. CVS then retrieves the new files from the server and tries to merge the new file with the developers changed code. This can lead to a merge conflict.

3.1.2 Merge conflicts

If two developers have been editing the same piece of code, a merge conflict may arise. It is then up to the developer who tries to commit last to solve

the conflict. When CVS detects a conflict, it will add both options into the file, and marking it. The latter developer must then look through the code and choose which ones that are correct, or rewrite the code so that it works.

This can be a time consuming task and there is always a risk that a programmer manages to break the code while trying to resolve the conflict. Also, if a programmer has worked on a file for a long time without updating and committing, the number of merge conflicts in a file can be quite a few. On a bad day, a developer can get stuck for hours just trying to resolve merge conflicts. To help with merging, there are a number of third-party tools that can aid in the task.

3.1.3 The author's comments

The CVS system is a good and reliable tool. It has been the dominant tool during the projects I have worked with and if anything, I have trusted it to do it work. When working with it, I have primarily used it through a graphical editor, Eclipse. Using CVS in this way spares you the need to remember the command syntaxes and makes CVS easier to use in general.

However, CVS has a downfall which has been the source of much wailing and gnashing of teeth during the projects. When doing a update of the code (and especially when it's been a while since the last update), the merge conflicts are unavoidable. Since CVS isn't always doing a great job when analyzing the changes made, there are often merge conflicts to deal with.

Also, CVS posts the merge conflicts directly into the code (since there is no graphical interface) and with many conflicts, the work of developers can grind to a halt while they try to resolve them. Since this on occasion can take several hours, developers can easily get stuck in an evil circle. Why? Because when the developers has resolved the conflicts after two hours of work and do another update, others are bound to have checked something new in and the poor developers are back to square one, with several merge conflicts.

When working with CVS as a developer, the only thing I really worry about is merge conflicts. This worry has continued on during my work as a coach since you know how much these conflicts can hamper work.

3.2 Subversion

Subversion is an offspring from the CVS program, initiated in early 2000 by a company called CollabNet, Inc. At the time, CollabNet was offering a CM software suite called SourceCast which included CVS for version control.

CVS had however serious limitations and it became apparent for CollabNet that they had to find something else that could take it's place. The decision was made to rewrite a new version control system, which while retaining the basic ideas of CVS, contained a lot of improvements. Subversion is however, like CVS, open source.

In order to ease migration to Subversion, it is purposely made to handle and act a lot like CVS. Anyone who has been working with CVS will have no problems trying out Subversion.

3.2.1 Improvements over CVS

Subversion boasts with a number of improvements over CVS, including:

Renaming is possible - In CVS, if you rename or move a file, the history of that file is gone. In Subversion, it's retained.

Commits are atomic - Either the whole commit gets stored in the repository, or nothing at all. This is to protect against the situation where an error causes some files to be left out, and thereby compromising the validity of the repository. CVS operations are atomic with respect to failure at the granularity of individual files, not at the broader granularity of multi-file operations.

Efficient handling of binary files - Subversion uses a binary algorithm to store revisions of a file. By doing this, the project claims to be equally efficient on binary files as on text files.

Parsable output - All output is designed for automatic parsing, which makes scripting easier.

Client side diff plug-ins - Subversion makes it possible to create client side plug-ins that makes merging of certain files (for example, Java source files) a lot more manageable by analyzing the contents of the file. *NOTE! This feature has not been incorporated into Subversion when this study was written. It is a planned feature.*

3.2.2 Disadvantages of subversion

First of all, when this study was written, the subversion system was still in a beta state. However, for a beta, the system is surprisingly stable which also bears witness by the fact that the developers of Subversion is, in fact, using Subversion itself as the version control system for the project. Nevertheless,

judging by the number of reported bugs on Subversions forums and the fact that it is beta, the author of this study would wait until a full version was released before using it in a project.

3.2.3 The author's comments

Since the majority of the differences against CVS has to do with performance, we didn't really see a big change handling Subversion to handling CVS. The single largest difference, which sadly is not implemented yet, is the plug-in feature for merging files. If this can be made successfully, it can mean a big advantage over CVS since it may greatly reduce the number of merge conflicts when updating (and thus minimizing programmer frustration).

3.3 Perforce

The Perforce system is a commercial client-server system which in a way acts a lot like CVS. Perforce is mainly based on graphical interfaces and ships with a range of graphical tools for merging files. These tools can be used alone outside of the Perforce environment. Perforce is available for several operating systems such as Windows, Linux, Macintosh and more. Perforce also exists as plug-ins for popular editors such as Eclipse and IDEA IntelliJ. Before a plug-in for Perforce can be released, it must be validated by Perforce Software, thus ensuring that it works as expected.

3.3.1 The merge function

The merge function is what stands out the most in Perforce. A graphical merging tool is shipped with Perforce and is launched automatically whenever a merge conflict is discovered. The graphical tool displays three versions of the code. The code on the server, the code in your workspace and the code after the automatic merge. All conflict are highlighted with color coding and can be altered at will. Perforce will not, however, analyze the code based on code syntax. There is also the option to let Perforce do an auto-resolve and make a best guess what lines to use. Since Perforce can't analyze the syntax this auto-resolve is not really that usable, and any developer is probably better off doing the merge himself.

3.3.2 Focus on the server

The Perforce system works mainly on the server. If you aren't working on a file, the file is not stored in the client workspace by default. However, the user can choose to retrieve the files anytime he wants. This feature is intended to reduce the number of files altered and thereby reducing the history logs. As an extra feature, any server can have a number of proxy servers that work as extra file caches and speeds up the server-client connections. This can be useful if different development teams in a number of countries are working together.

3.3.3 Altering a file

Whenever a user wishes to alter a file, the server is notified of this and downloads a copy of the file to the client's workspace. By doing this, the server can keep track of whether a file is being worked on by another person and can notify other users of this. This does not stop anybody else from working on the file at the same time, but it merely acts as an early warning system. The Perforce client can also be set to check the server for any changes at a specified time interval. Whenever it detects that the files you are working on are no longer the latest, it will tell you so and give you the chance to do an update. Just viewing the contents of a file will however not be notified to anyone else.

3.3.4 Viewing the history

When viewing the file history in Perforce, you can see what alterations has been made to the file from any of the previous versions. The differences are presented graphically and are easy to understand. Lines that have been removed are shown, but in red and are crossed out. Lines that have been added are shown in blue. Much work has been put into making the graphical tools easy to understand and work with.

3.3.5 The author's comments

When setting up a server or a client for the first time, Perforce can be a bit confusing. Luckily, the documentation contains examples of how to do this. If you have been using CVS before, Perforce can take some time to get used to, and at times, the graphical interface can be overwhelming.

Another interesting aspect is that instead of registering users separately, the Perforce server registers clients (The computer). This is so Perforce can

keep track of what files has been checked out where. This can be both an advantage and a disadvantage. If a user is working on a file on one computer, and then moves to another, he can work there independently of what he is working on at the other computer. On the dangerous side, if the server is not configured properly, once a client is registered, ANY user on that computer can log on to the server. If the user and password does not exist, Perforce simply adds the user to the database. Since this is the default behavior, this can be a serious security risk when working in an environment where many users share computers.

In my opinion however, I prefer Perforce over CVS, especially because of the improved merging. Perforce also feels more stable and more thought-through than CVS.

On the downside, Perforce is a commercial product and can be a bit expensive if it isn't intended to be used by a company. Also, the graphical interfaces are limited to Microsoft Windows, which makes the product not as appealing to users who uses other operating systems.

4 The future of Configuration Management

As software development evolves, so must the tools that support it. CM is an important part in the average programmers day, but to ensure that it stays that way, not only must the developers of CM do a good job, but those who develop the development systems must also have to take notice.

While CM tools today allow binary files to be version controlled, they don't really analyze the contents, instead they simply store them for safekeeping. For example, let's take a look at the Specification and Description Language (SDL), which is used when developing telecommunication systems. Here, the development files are stored in a binary format. This makes these files much harder to version control and work upon simultaneously.

Development systems developers may want to keep this in mind when standardizing how a system stores the files in question. If it's possible to use a format that can be version controlled, the people who later will use it can gain much.

Another approach would be to make it possible for the CM software to analyze binary files. This can however be very risky and may do more harm than good if something goes wrong.

Another thing that could make a big change for developers today, would be the possibility for the merging parts to analyze the code based on the code's syntax. For example, if a developer updates his code, and the name of a

variable has been changed, the merging program could recognize this and update the code accordingly. This is already underway in the Subversion project, but it could take some time before this becomes really efficient.

5 Acknowledgments

The author wishes to thank the following people:

- Görel Hedin, Lars Bendix, Christian Andersson and Torbjörn Ekman at The Department of Computer Science of the Lund Institute of Technology for introducing me to the art of configuration management.
- Daniel Flink, Erik Norberg and Fredrik Redgård for reviewing this study and making excellent comments.
- The members of the Enduro projects of 2003 and 2004 for their input and comments.
- The members of the AirSoftMaster project for their input and comments.

6 References

Websites:

- The Concurrent Versions System - http://www.cvshome.org
- The Subversion project - http://subversion.tigris.org
- The Perforce System - http://www.perforce.com
- The CM Crossroads - http://www.cmcrossroads.com

Articles:

- *S M Thompson*
  Configuration management - keeping it all together.
  BT Technology Journal 15 (3) 1997
- *Babich*
  W. Software CM - Coordination for team productivity.
- *Tellioglu H., Wagner, I.*
  Negotiation Boundaries - CM in Software Development Teams.
  CSCW 6(4) 1997