International Journal of Advancement in Engineering Technology, Management & Applied Science

Volume1, Issue 1 June 2014 ISSN No: 2349-3224

Title
A study of tools and techniques for Software configuration Management

Agha Salman Haider

Sr. Lecturer, Jazan University, Saudi Arabia

Abstract:

Software Configuration Management has been defined as the regulation of controlling the development of multifaceted software systems. As a vital task for professional software development it also has to provide for software projects carried out using Lyee methodology. In an advance to categorize Software configuration Management functionality is made.This paper compares the tools for software configuration management which help for better process in IT organization.

Keywords:CASE,Configuration, Networks

Introduction:

Tools for module interconnection languages are concerned with analysisand system construction. Analysis refers to syntax and static semantics of module interfaces.System construction comprises version selection and code generation.Object-oriented modeling languages are supported by integrated CASE tools whichprovide graphical editors, analysis tools, code generators, and interpreters (if the modelinglanguage is executable). Many commercial CASE tools are available, includinge.g. Rational Rose (for the UML), ROOM, and SDT for SDL. CASE tools have oftenbeen criticized as drawing tools. Clearly, the functionality of a CASE tool depends onhow well the syntax and semantics of the underlying modeling language are defined. Inthe case of an executable modeling language, CASE tools go far beyond drawing tools.

Software Configuration Management Tools:

1. Subversion:

Pilato, Fitzpatrick and Sussman (2009) have explained that “Subversion is a free open source version control system. That is, subversion manages files and directories and the changes made to them over time. This permits them to recover big versions of their data or inspect the history of how their data changed. In this regard a lot of people think of a version control system as a sort of time machine. Subversion can work across networks, which allows it to be used by people on special computers. At several levels, the ability for various people to adapt and manage the

same set of data from their respective locations fosters association. And as the work is versioned they need not fear that quality is the trades off for losing that medium if some wrong change is made to the data just unfasten that change. Subversion is so often used in software progress environments working on a development team is an inherently social activity and subversion makes it simple to work together with other programmers.

2. Perforce:

Jacobsen, Schlenker and Edwards (2005) have explained that "Perforce is a very quick, easy to use yet fairly powerful multi-platform Software Configuration Management system. Its undisputed speed and its aptitude to firmly incorporate into software development workflows make perforce the preferred Software Configuration Management of many programmers. Perforce is driven by a lock/ modify/ unlock model. This means a customer must get a lock for a file previous to it can be edited and unlock the file afterwards. Locks are typically non-exclusive so that manifold customers can edit a file at the same time. Each user can see which files are currently under modification by other users. It's also probable to lock a file wholly denying other users the ability to alter the file until the lock is lifted again. This provides a streamlined way of evading merging binary files however wrecked or obstructive clients could seriously thwart the project as in these cases administrative action is required to lift the lock.

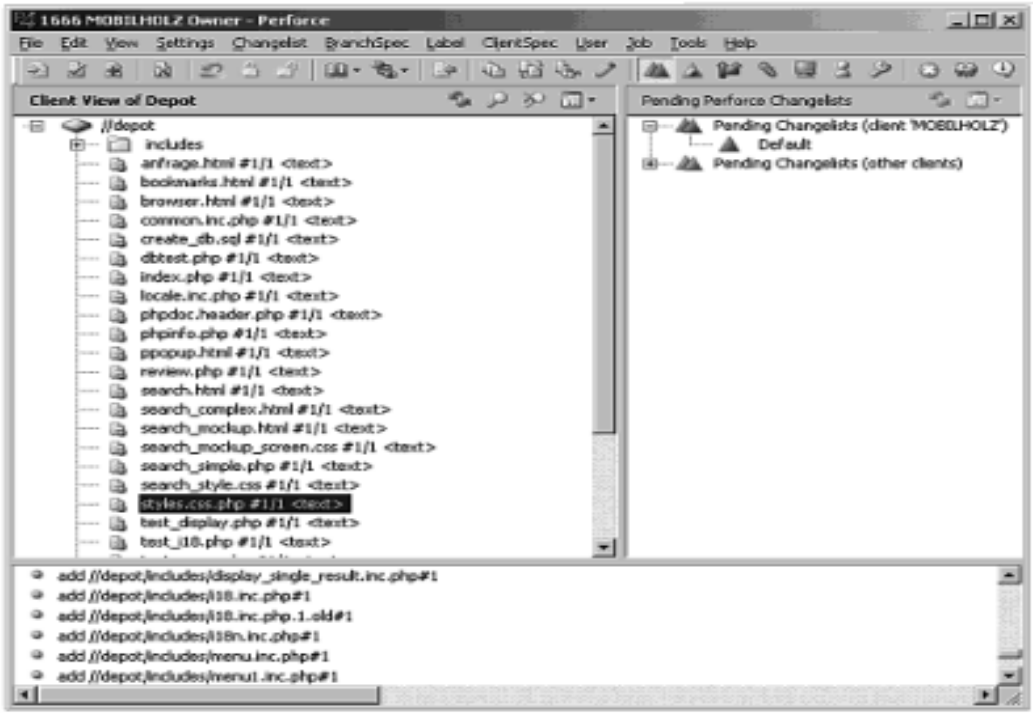

Figure 2.1: The Perforce Windows Client.

3. CVS:

Concurrent Version System (CVS) is the Software Configuration Management all others are deliberate against. It is very mature, scalable and steady software. Appropriate to its broad sharing there are integrations available for almost every software development tool and for various other software packages. As in most Software Configuration Managements the version the past is stored on a central server and the client machines can get restricted copies of all the files the developers are working on. Client and server may run on the similar machine. Network communication.

4. ClearCase:

Preston(1999) has defined that "ClearCase is a software configuration management tool developed and promoted by rational, Inc. ClearCase is most usually used in software projects linking at least more than a few developers but frequently in settings with hundreds of developers in a lot of sets. ClearCase offers complete configuration management together with version control, build administration, workspace management and progression control without forcing them to modify their obtainable environment, their tools or the way they work. As with approximately some Database Management System, ClearCase have to be inactive previous to backups can be executed. Therefore, the common plan is to catch or create engaged the objects being backed up, perform the backup and after that hasty the ClearCase admission to the individual objects. This may seem like a simplistic approach, although it is the most dependable. A lot of the conditions that are used to explain ClearCase may be untried. ClearCase has a "multilevel" planning:

- The primary layer is the UNIX files system with all the common mechanisms and limitations.
- The second layer is Rationale's proprietary Multi Version File system, it is the layer which proceeds very much like a DBMS and permits the multi-version functionality of ClearCase.
- The third layer is the Versioned Object Base; the VOB is the fundamental data container for items below control of ClearCase. A ClearCase installation characteristically has more than a few VOBs at any given time. Each VOB is owned and used by exacting groups in order to do their work.
- The fourth and the last layer is a ClearCase outlook. Just as a database analysis selectively accesses exacting database items, so does a ClearCase view decide which items a worker can access at any given time. View also have their own data storage containers that hold temporary items such as program files that are at present being customized. The mechanics of how these containers are accessed is not relevant to this conversation, although if they are engrossed they can discover more information on the balanced Clear Case website.

5. RCS:

RCS is one of the oldest SCM tools. It is very easy and contains rudimentary SCM functions applied on files. The effortlessness of RCS is marvelous and the motive for its extensive use. RCS is also used as a necessary tool for a lot of other SCM tools. Still additional interesting is that the main beliefs initiated in RCS are still practical in further more complied SCM tools. RCS is file oriented and the SCM repository is merely a directory that includes versioned files. One versioned file includes one or numerous versions of that file. Baselines are attained by setting a

label on accurate versions of the records. RCS contains dissimilarity and join functions, which can be used by clients and by, sign in and sign out instructions. The main features of RCS are:

- Realistic for using on individual bases or in little sets.
- Uncomplicated and easy to utilize.
- Can be used as a base for building more complicated tools.
- Encloses essential SCM functions.

Comparison of various SCM Tools:

We described the features of various SCM tools as below:

	CVS	Subversion	Perforce	Rational ClearCase	BitKeeper	VSS
Scalability	+++	+++	+++	++++	++++	–
Speed	+++	++++	+++++	++	+++	–
Security	++++	+++++	++++	++	++++	–
Stability	++++	+++++	+++	++++	++++	–
Handling of binary data	+	++	+++	+++	+++	+

+++++, Perfect; ++++, outstanding; +++, good; ++, sufficient; +, some; –, totally insufficient.

Conclusion:

We examined the features between software configuration management tools. To this end,we delineated their roles and try to find out the best tool based on scalability, Speed, Security, Stability and handling of binary data for better IT organization process. Finally, we summarized our conclusions in features table described above.

References:

1) Heydon (2006), Software configuration management using Vesta, Birkhäuser, p 167.

2) Appleton and Berczuk (2003), Software configuration management patterns, Addison-Wesley Professional, p 183.

3) Christensen (2010), Flexible, Reliable Software, CRC Press, 2010, p 394.

4) Johanneson and Fujita (2002), New Trends in Software Methodologies, Tools and Techniques, IOS Press, p 317.

5) Kishore (2005), ISO 9001:2000 For Software Organizations, Tata McGraw-Hill Education, p 215.

6) Conradi, Estubier, Sommerville, International Conference on Software Engineering and International Workshop on Software Configuration Management (1995), Springer, p 35.

7) Pilato, Fitzpatrick and Sussman (2009), Subversion 1.6 Official Guide, Fultus Corporation, p 27.

8) Jacobsen, Schlenker and Edwards (2005), Implementing a Digital Asset Management System, Elsevier, p 203.

9) Merriam S B (2002), Qualitative research in practice, Jossey-Bass, San Francisco.

10) Taylor G R (2005), Integrating Quantitative and Qualitative methods in research, University Press of America, USA.

11) Bordens K S and Abbott B (2006), Research Design and Methods, Tata McGraw Hill, New York.

12) Kirsch G (1992), Methods and Methodology in composition research, Illinois University, USA.