Configuration Management and Open Source Projects

André van der Hoek
Institute for Software Research
Department of Information and Computer Science
Irvine, CA 92697-3425 USA

andre@ics.uci.edu

Abstract

Configuration management tools are at the heart of every software project. Thus, it should not be surprising that they play a central role in Open Source projects as well. Most prominent in use is CVS, which is—indeed—an Open Source system in its own right. In this position paper we examine why CVS plays such a major role in the management of Open Source projects. Furthermore, we raise some areas in which we believe CVS should be improved, both in the short and long term.

Concurrent Versions Systems

As one of the essential tools needed during the development of a software product, configuration management tools were among the first faced with the reality of having to operate in a distributed setting. In response, many distributed CM systems have been developed in the past few years (e.g., ClearCase MultiSite [1], Gradient [3], ScmEngine [5], DISC [12], WWCM [13], DSCS [16], Perforce [17]). Despite this broad availability, it is clear that a single system has emerged as the *de-facto* configuration management system used in Open Source projects. In fact, this CM system, CVS [4], has been adopted by almost every major Open Source project. Further evidencing its popularity is the fact that CVS is the only CM system that has a book dedicated to its use in Open Source projects [9].

Several reasons can be identified for this intriguing phenomenon.

- **The CM policy embedded in CVS closely matches the Open Source process.** An Open Source projects is typically organized around a central repository from which individual developers retrieve copies of the project. Developers make their changes within one of these copies. Once the changes are complete, they update the repository with new versions of the artifacts that have changed. Other developers synchronize their copies of the project by periodically downloading updated versions of artifacts and resolving any conflicts that may exist. This is *exactly* the CM policy that CVS excels in supporting: CVS is based on a single level of transactions that each are based on an optimistic scheme of conflict resolution. This one-to-one correspondence between the actual Open Source process and the process supported by CVS makes it very appealing to use CVS in Open Source projects.
- **CVS supports decentralized software development.** Since Open Source projects involve developers that are located all over the world, a requirement for a configuration management tool to be used in an Open Source project is

that the tool operates in a decentralized and distributed setting. Preferably, even, the tool supports intermittent disconnected operation in that each developer does not continuously have to be connected to a main repository with artifacts. Although initially not devised as a distributed CM system, CVS has been enhanced over the years to provide several methods of access to its central repository with artifacts. Together with its optimistic method of resolving conflicts, CVS, thus, precisely matches the distributed capabilities needed in an Open Source project

- **CVS is free, yet well maintained.** Since Open Source projects typically have little to no funding and commercial CM systems tend to be rather expensive in nature, the use of a commercial CM system in an Open Source project is usually impossible. CVS is free. Despite being free, however, CVS is well maintained and rather complete in functionality when compared to the other freely available CM systems. In fact, CVS is an Open Source project itself and has been in widespread use for many years now. As a result, many of its initial problems have been solved and CVS is currently one of the best freely available CM systems.

Given this combination of factors, it should come as no surprise that CVS is so widely used in Open Source projects. It provides the necessary functionality at a more than reasonable price: it is free, yet easy to install, setup, learn, and use.

Potential Short-Term Enhancements to CVS

With its unparalleled success, complacency seems to have settled into CVS and the functionality that it provides to its users. In fact, the functionality and CM policy that form the core of CVS have not been enhanced for quite some time now. Unfortunately, a close examination of the Open Source process seems to indicate that several enhancements to CVS could greatly enhance the applicability of CVS in the future to come. Specifically, we believe that the following four enhancements are important to be made in the near future.

- **An infrastructure that supports multiple repositories.** More and more Open Source projects are based upon other pieces of software from other Open Source projects. Currently, these pieces of software need to be periodically incorporated via the vendor-code management functions of CVS. Although certainly usable, this solution becomes unwieldy if many subcomponents are present that each have a different release schedule. It would be preferable to link various CVS repositories together to directly and continuously import source code for subcomponents. In essence, this brings an automated and enhanced version of a tool like SRM [20] to the software development process.

- **Versioning of directories.** One obvious improvement to be made to CVS is its handling of directory versioning. Currently, directories are not versioned at all, even though their contents can change over time. This not only leads to a rather crude way of handling these types of changes, but also to an overuse of tags to label the various configurations in which a project may exist over time. Given that more and more Open Source projects create a large number of configurations and regularly reorganize their project structure, this is a rather serious problem that deserves immediate attention. Fortunately, the well-

known solution of versioning directories solves this problem: it provides a clean way of dealing with the changing content of a directory and it provides a convenient and natural way of dealing with configurations. This is demonstrated by, for example, PRCS [14] and COOP/Orm [15], both of which are CM systems that intrinsically support the versioning of directories.

- **Private versioning capabilities.** Despite the fact that developers may store intermediate versions of artifacts in a CVS repository, it is generally encouraged that only complete and working changes are committed. Therefore, developers are left without any versioning support in their private workspaces. As Open Source projects are becoming larger and changes more complex, such a capability is much needed. As demonstrated, for example, by Continuus [6] and Perforce [17], the availability of such functionality enhances the development experience and typically leads to the creation of many intermediate versions before the final changes are stored in the main repository. These intermediate versions remain private to a developer and they neither interfere with changes from others, nor clutter the version history in the main repository.

- **Repository Replication.** It is common to use CVS in combination with a replication program like rsync [19] to improve access times for developers that are physically located in different continents than the main project repository. Although certainly beneficial, this solution has the problem that conflicts arising during synchronization cannot be resolved by rsync. Instead, the synchronization fails and manual intervention is needed to integrate and merge the changes from developers that use different instances of a replicated repository. Since rsync and CVS share many pieces of functionality, it should be possible to merge both into an integrated solution that resolves conflicts during synchronization in the same way CVS resolves conflicts during regular development. This would lead to a solution like ClearCase MultiSite [1].

It should be observed that each of these enhancements is based on solutions that already exist in commercial CM systems. They, in effect, can be seen as bringing CVS up-to-date with some of the advanced functionality that not only has emerged in today's CM systems, but also has proven to be very beneficial.

It should also be noted that none of the above suggestions involves changing the core policy or functionality of CVS. The basic premise of a transaction-oriented CM system that resolves conflicts in an optimistic way remains. The functionality suggested merely increases the applicability and utility of CVS, it does not change its fundamental principles.

Potential Long-Term Enhancements to CVS

Even with the short-term enhancements suggested in the previous section, it remains an open question as to how long CVS, in its current incarnation, will survive as the myriad of CM systems that are available continue to evolve and incorporate more advanced functionality. Therefore, it may be time to look into a complete redesign of CVS that radically advances its functionality. In fact, we believe it is possible to leapfrog most of the existing CM systems in terms of functionality and popularity if a

new, reincarnated CVS supports a complete cycle of the Open Source process and not just version control. In particular, we suggest that CVS be redesigned to include not only the changes suggested in the previous section, but also changes that lead to the incorporation of such activities as release management (automatically packaging software, creating and maintaining a change log, and publishing a package on a Web site), bug tracking (filing bug reports, keeping an archive of resolved bugs, and associating bug reports to those versions of the source code that fix each bug), and deployment (installing the software at the client side, periodically polling for updates, and actually upgrading the version of the software on the installed base). Although ambitious, several pieces of infrastructure exist that have proven to be beneficial in their respective domains and that may help in realizing the vision of an integrated and rejuvenated CVS.

- **SRM.** SRM is a software release management system that manages software releases stemming from multiple different sites [18,20]. SRM integrates the development process with the deployment process. It supports groups of distributed software development organizations with a simple release process that hides distribution. Specifically, developers are supported by allowing the specification of cross-site dependencies and users are supported by allowing the retrieval, via the Web, of a system of systems in a single step and as a single package.

- **Software Dock.** The Software Dock is a deployment system that manages software systems after they have been released [10,11]. Based on precise deployment instructions that are captured in a Deployable Software Description (DSD), specific agents install, update, and reconfigure software at a consumer site. The DSD is expressive enough to be able to handle dependencies among software, even if the software stems from different release sites.

- **RPM.** RPM [2] is a system that can be viewed as an intermediate between SRM and the Software Dock. Although not as fully functional as the Software Dock in managing a deployed software system, it is more advanced in its integration between the release site (similar to SRM) and the customer site (similar to the Software Dock).

In addition to these systems, others can be adapted for bug-tracking (such as GNATS [7]), repository synchronization (such as rsync [19]), and providing a distributed CM infrastructure (such as NUCM [21,22] or Adele [8]).

Basically, our vision is for a component-based, fully distributed and decentralized configuration management system that manages artifacts from their incarnation to their eventual destination as an installed piece of software at a consumer site. Although much work needs to be done to fulfill this vision, we believe it would be a great advance, not only for the functionality of CVS, but also for the Open Source community at large which will be served with a CM system that intimately supports its software process.

Conclusions

Although CVS is still the de-facto CM system to use in Open Source projects, it is in danger of losing its edge. Its functionality is slowly but surely becoming out-of-date with respect to the current state-of-the-art in commercial CM systems. In this paper we have suggested some short-term fixes for CVS to be able to continue to hold its ground, but it may be time for a radical redesign that incorporates some of the long-term changes we have suggested. CVS is here, but hopefully it is here to stay as a ubiquitous, free, easy-to-use tool that intrinsically supports the Open Source process.

Acknowledgements

The author wishes to thank Compaq for their gracious donation in support of his research.

References

[1] L. Allen, G. Fernandez, K. Kane, D. Leblang, D. Minard, and J. Posner. ClearCase MultiSite: Supporting geographically-distributed software development. In *Software Configuration Management: ICSE SCM-4 and SCM-5 Workshops Selected Papers*, number 1005 in Lecture Notes in Computer Science, pages 194—214, New York, New York, 1995. Springer-Verlag.

[2] E.C. Bailey. *Maximum RPM.* Red Hat Software, Inc., February 1997.

[3] D. Belanger, D. Korn, and H. Rao. Infrastructure for wide-area software development. In *Proceedings of the Sixth International Workshop on Software Configuration Management*, number 1167 in Lecture Notes in Computer Science, pages 154—165, New York, New York, 1996. Springer-Verlag.

[4] B. Berliner. CVS II: Parallelizing software development. In *Proceedings of 1990 Winter USENIX Conference*, Washington, D.C., 1990.

[5] J.X. Ci, M. Poonawala, and W.-T. Tsai. Scm Engine: A distributed software configuration management environment on X.500. In *Proceedings of the Seventh International Workshop on Software Configuration Management*, number 1235 in Lecture Notes in Computer Science, pages 108—127, New York, New York, 1997. Springer-Verlag.

[6] Continuus Software Corporation, Irvine, California. *Continuus Task Reference*, 1994.

[7] Cygnus Support. *The GNU Problem Report Management Systems*, January 1996.

[8] J. Estublier, editor. *Proceedings of the Ninth International Symposium on System Configuration Management*, number 1675 in Lecture Notes in Computer Science, New York, New York, 1999. Springer-Verlag.

[9] K. Fogel. O*pen Source Development with CVS.* The Coriolis Group, Scottsdale, Arizona, 1999.

[10] R.S. Hall, D.M. Heimbigner, A. van der Hoek, and A.L. Wolf. An architecture for post-development configuration management in a wide-area network. In *Proceedings of the 1997 International Conference on Distributed Computing Systems*, pages 269—278. IEEE Computer Society, May 1997.

[11] R.S. Hall, D.M. Heimbigner, and A.L. Wolf. A cooperative approach to support software deployment using the Software Dock. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 174—183, New York, New York, May 1999. ACM Press.

[12] B. Hoang and F. Maurer. DISC: A distributed web based software configuration management system. May 1999.

[13] J.J. Hunt, F. Lamers, J. Reuter, and W.F. Tichy. Distributed configuration management via Java and the World Wide Web. In *Proceedings of the Seventh International Workshop on Software Configuration Management*, number 1235 in Lecture Notes in Computer Science, pages 161—174, New York, New York, 1997. Springer-Verlag.

[14] J. MacDonald, P.N. Hilfinger, and L. Semenzato. PRCS: The project revision control system. In *Proceedings of the Eighth International Symposium on System Configuration Management*, number 1439 in Lecture Notes in Computer Science, pages 33—45, New York, New York, 1998. Springer-Verlag.

[15] B. Magnusson and U. Asklund. Fine grained version control of configurations in COOP/Orm. In *Proceedings of the Sixth International Workshop on Software Configuration Management*, number 1167 in Lecture Notes in Computer Science, pages 31—48, New York, New York, 1996. Springer-Verlag.

[16] B. Milewski. Distributed source control system. In *Proceedings of the Seventh International Workshop on Software Configuration Management*, number 1235 in Lecture Notes in Computer Science, pages 98—107, New York, New York, 1997. Springer-Verlag.

[17] Perforce Software, Alameda, California. *Networked Software Development: SCM over the Internet and Intranets*, March 1998.

[18] R.A. Smith. Analysis and design for a next generation software release management system. Master's thesis, University of Colorado, Boulder, Colorado, December 1999.

[19] Tridgell and P. Mackerras. The Rsync algorithm. Technical Report TR—CS—96—05, June 1996.

[20] A. van der Hoek, R.S. Hall, D.M. Heimbigner, and A.L. Wolf. Software release management. In *Proceedings of the Sixth European Software Engineering Conference*, number 1301 in Lecture Notes in Computer Science, pages 159—175, New York, New York, September 1997. Springer-Verlag.

[21] A. van der Hoek, R.S. Hall, D.M. Heimbigner, and A.L. Wolf. A generic, reusable repository for configuration management policy programming. Technical Report CU—CS—864—98, Department of Computer Science, University of Colorado, Boulder, Colorado, September 1998.

[22] A. van der Hoek, D.M. Heimbigner, and A.L. Wolf. A generic, peer-to-peer repository for distributed configuration management. In *Proceedings of the 18th International Conference on Software Engineering*, pages 308—317. Association for Computer Machinery, March 1996.