Investigation of Repository Reprecation Models in Globally Distributed Configuration Management

Kazuhiro Fujieda Koichiro Ochimizu
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Nomi, Ishikawa, Japan
{fujieda,ochimizu}@jaist.ac.jp

Abstract

The replication of repositories is an important mechanism for the distributed configuration management in open source software development (OSSD) projects. There are various styles of replication taken place in OSSD projects. On investigation we found tools or systems supporting them should have the following functions : partial replication, both of pull and push model in change propagation, and condensation of successive versions into one version. This paper proposes a primitive tool grafting version trees between repositories to satisfy these requirements. This tool will become the basis of the environment to support various styles of repository replication.

1. Introduction

Open source software development (OSSD) projects are typical cases of global software development. Most of all such projects use CVS [2] for their software configuration management. CVS allows developers to access its repository via the Internet. It doesn't provide file locking. Developers can check out the same part of the repository, work on it concurrently, then resolve inconsistency in their work. This style of development can decrease the cost of negotiation among developers separate globally and has supported the rapid evolution of open source software.

CVS, however, provides only a single repository model. It leads two problems. 1) It requires a server connected to the Internet with wide bandwidth and an organization maintaining it. 2) It coerces a single set of management policies of the repository. Such a set includes how to give the write permission to developers, how to commit changes to the repository, how to use branches, and so on. You can see examples of such policies in [5] [7] [6]. The former problem can be solved by hosting services typical of SourceForge[1], The latter problem can decrease concurrency of the work.

Developers or teams in one project sometime need to proceed their work to a considerable extent separately with the overall project. Although they need configuration management in this case, they can't use the project's repository unless their work meets management policy in the project.

If it doesn't meets the policy and they think the negotiation with other members troublesome, they often create full or partial replicas from the repository. They will work on the replicas inconsistently with the original, and incorporate the changes of other members in the original sporadically. Then they will propagate their own changes from their replicas to the original according to the project's policies. This style of development with replicas can augment concurrency of their work.

The repository replication is often taken place because of network bandwidth or server resources. CVS allows read-only access to a repository by anonymous users. The server hosting it can have too heavy load to trouble developer's work. In this case, developers like to separate repositories for them and read-only access with replication. They also like to settle replicas in their local hosts when they have only sporadic or narrow connectivity to the server.

Although repository replication is important in OSSD projects, CVS doesn't support it at all. Developers create replicas by the supplementary tool named CVSup [3] or by hand. CVSup support only one-way mirror naturally, so it can't fully support their own work on replicas. There are two configuration management systems supporting the replication: BitKeeper [1] and ClearCase Multisite [4]. Each of them supports one style of development with replicas. These styles can't necessarily cover various styles of development with replicas taken place in OSSD projects.

In this paper, we first investigate these styles of development with replicas, then discuss problems in the above two configuration management systems. Finally, we will show the design of the replication tool as a supplement to CVS.

[1] http://sourceforge.net/

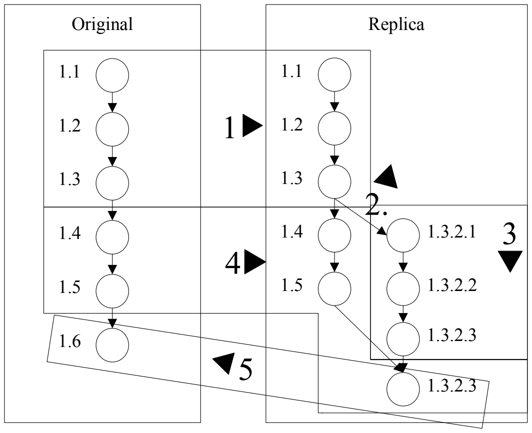

Figure 1. Basic scenario of replication and change propagation.

2. Replication Models in OSSD

2.1. Basic Scenario

We present the basic scenario of replication and change propagation really taken place in OSSD projects with CVS and CVSup. Figure 1 illustrates this scenario.

1. Create a replica.

   A developer create a replica from a repository by hand or CVSup. He/she often create it from part of files and/or part of versions in the original to save costs of network bandwidth and storage.

2. Create a branch in the replica.

   Branches are traditionally used in version control systems to isolate changes from the main line of development. Although he/she work on the replica, a branch is necessary to easily incorporate changes from the original.

3. Work on the branch.

   He/she adds his/her own changes to the branch in the replica.

4. Pull changes from the original and merge them into the replica.

   He/she sporadically pulls changes from the original and add them to the trunk in the replica. CVSup can support this operation as long as the version number of the branch doesn't crash any branch in the original. He/she merges them into his or her branch after that if necessary.

   When he/she can't use CVSup, he/she doesn't reproduce the version history in the original. He/she add one version including all changes in the history by hand. Such condensation of versions is often taken place intentionally when he/she doesn't need intermediate changes between specific versions, for example, stable or snapshot releases.

5. Push changes from the replica to the original.

   If necessary, he/she negotiates with other members about merging his/her changes into the original. If the negotiation succeeded, he/she adds one change condensed from his/her changes to the trunk in the original. In most cases, there is no need to reproduce the branch in the original because such branch rarely meets the management policy of the original. Even though he/she want to do so, it is hard because there is no tool supporting it.

ClearCase MultiSite can't support this scenario at all. It performs automatic change propagation in the push model to keep consistency between the original and replicas. It can't help policy separation, but solve the problem about network resources. BitKeeper and CVS with CVSup can't support this style of replication.

BitKeeper naturally performs this scenario. Developers must create replicas when they intend to make any change in a repository. They don't need to create branches on their replicas at the step 2. BitKeeper automatically creates branches and merges the trunks into them at the step 4. Additionally it makes version trees consistent between the replica and the original at the step 5.

2.2. Advanced Scenario

BitKeeper can work on the above basic scenario. But it always replicates all files in a repository and synchronize all files between a replica and its original. So it can't support some styles of repository replication in OSSD.

When a development project use outcome of another ongoing project, it is desirable its repository contains the full or partial replica of another repository. For example, Cygwin project[2] uses a part of outcome by Mingw project[3]. Its repository contains the partial replica of the repository of Mingw project (Figure 3).

A project depends on outcome of multiple ongoing projects, its repository can be composed full or partial replica of the repositories of these projects. For example, KAME Project[4] depends on FreeBSD, NetBSD and OpenBSD. Its repository contains partial replicas of the repositories of those projects.

[2] http://www.cygwin.com/

[3] http://www.migw.com/

[4] http://www.kame.net/

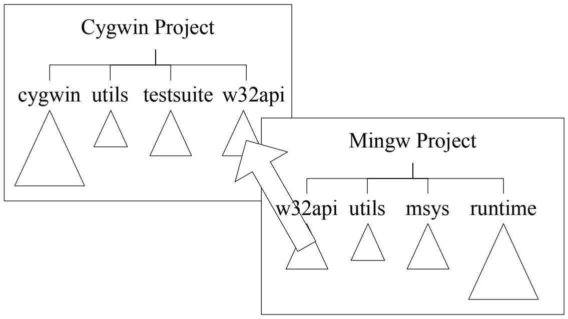

Figure 2. An example of partial replica

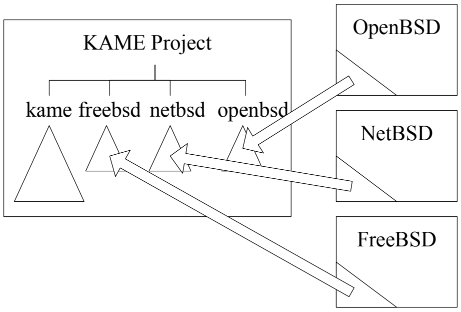

Figure 3. An example of multiple parents

3. Our Approach

On the above investigation, A tool or a configuration management system needs the following functions to cover various style of development with repository replication.

- Partial replication in files and versions.
- Both of push and pull models to propagate changes in both directions between a replica and its original.
- Condensation of several changes into one change on propagating changes.

We believe it isn't proper solution to realize a configuration management system supporting all of the above functions. Developers working on replicas need a specific style such as provided by BitKeeper and ClearCase MultiSite to handle them without confusion.

Our approach is rather simple. We will provide a supplement tool grafting version trees from a repository to another for CVS. At first, it obtains a part of version trees from the source repository, then store them to destination repository or grafts them on existing trees in it. It records how version trees are transmitted between repositories. At the next time, it grafts growing part of the trees in the source onto the corresponding trees in the destination. This tool is a primitive to realize a style specific to a replica or a project. CVS supports some triggers to invoke other tools on specific timing. We estimate the combination of these triggers and our tool can realize various styles.

4. Conclusion and Future Work

OSSD projects need a tool supporting various styles of repository replication. We proposed a tool grafting version trees between repositories as a primitive to cover these styles. Now we are implementing it in the object-oriented scripting language 'Ruby'. We will provide the environment allowing users to write their own style of replication in Ruby and automate their work with their replica to some extent.

References

[1] BitMover, Inc. BitKeeper - the scalable distributed software configuration management system. `www.bitkeeper.com`, 2002.

[2] K. Fogel. *Open Source Development with CVS*. CoriolisOpen Press, 1999.

[3] J. D. Polstra. CVSup: The CVS-optimized general-purpose network file distribution system. Available at `http://www.cvsup.org/`, 2002.

[4] Rational Software Corporation. *Administrator's Guide for Rational ClearCase MultiSite*, Oct 2001.

[5] The Apache Software Foundation. Apache project guidelines and voting rules. `http://httpd.apache.org/dev/guidelines.html`, 2002.

[6] The FreeBSD Documentation Project. Committer guide. `http://www.freebsd.org/doc/en_US.ISO8859-1/articles/committers-guide/`, Feb 2003.

[7] The Mozilla Organization. Mozilla hacking in a nutshell. `http://www.mozilla.org/hacking/`, Feb 2003.