Why BitKeeper

Elliott Wolin
Gluex Collaboration
7-Jul-2004

Abstract

I argue that Gluex should adopt the commercial (but free for us) code management product BitKeeper instead of CVS. Many advances in code management software have been made since CVS was designed, especially in the areas of development, workflow and release management. Advances include hierarchical repository structures, changesets, and support for the "gatekeeper" software development model. Use of these will allow Gluex to save considerable time and manpower in our software development effort.

I. Introduction

For Gluex to succeed a great deal of software must be developed and debugged quite early, years before the first beam arrives. This can only be accomplished if we adopt carefully chosen state-of-the-art strategies, techniques, and tools. We must break with HENP tradition and follow the lead of the few experiments and many open-source efforts that have managed to rapidly develop high-quality software.

Critical to our success is the choice of software development model and the tools chosen to implement it. I argue that the appropriate model for Gluex is the hierarchical "gatekeeper" model. Under this model, gatekeepers at each integration level in a hierarchical development tree ensure the quality of the submitted software before it is pushed up to the next level in the tree. This is in contrast to the single-level "free-for-all" model where developers update the code at their will and whim, with little formal quality control. For larger projects this leads to constant problems at the single integration level and a great deal of wasted time and effort.

As I describe below, BitKeeper is a third-generation code management product designed specifically to implement the hierarchical gatekeeper model. CVS, while an excellent second-generation product, is designed for the old "free-for-all" single-level repository model. While gatekeepers can be imposed on the single-level model in an ad-hoc manner, I argue we are much better off using a product designed for this purpose.

Choosing an appropriate development model and implementation tools will allow us to save a great deal of time and manpower in the software development effort. I estimate that by adopting the hierarchical gatekeeper model plus BitKeeper, Gluex should save one or perhaps more FTE's per year. Although estimation is difficult, I base mine a number of considerations including: the simplification of the work of the release and code managers and major developers, that problems should be caught earlier in the development cycle, the more formal development structure and more clearly defined task breakdown, and the more rapid feedback to developers from integrators and release managers.

A large number of open-source projects agree with this choice of model and BitKeeper, especially the Linux kernel and MySQL projects:

"BitKeeper has made me more than twice as productive, and its fundamentally distributed nature allows me to work the way I prefer to work - with many different groups working independently, yet allowing for easy merging between them." -- **Linus Torvalds, February 2004**

"We, the developers at MySQL AB, have had nothing but pleasant surprises all along with using BitKeeper. We would never want to go back to using CVS! If it's good enough for us, it's good enough for you." -- **Jorge del Conde, MySQL AB**

The key question to answer is whether Gluex needs to adopt a hierarchical gatekeeper model. If you believe the answer is no, then CVS is a good choice. If the answer is yes, and in this note I argue that the answer is a resounding "yes", then BitKeeper is the appropriate choice.

In the following I first describe the two revolutions in the history of code management in HENP and important new developments. Next I summarize relevant BitKeeper history, concepts, and features. Then I describe who will be affected, and how. Finally I argue that adopting the latest generation of code management software offers us compelling advantages that far outweigh the minor inconvenience of switching.

II. History and Recent Developments

I have used nine code management products over the past 27 years[1]. During this period there were two "revolutions" in code management, at least within HENP. In the beginning code management mainly focused on revision control of files in a single directory. Recognition of the need for multiple developers working on a directory tree full of files resulted in the first revolution, during which CVS was developed. This revolution occurred about 15 years ago, and CVS has served the community well since then, becoming the standard against which all other second-generation products were compared.

[1] PATCHY, CMS, RCS, SCCS, CMZ, CM, CVS, Teamware, and BitKeeper

But during this period the limitations of CVS became apparent, resulting in the second revolution. For example, for sizeable projects, managing the entire software development, maintenance, and release lifecycles was found to be critical: old releases still in use had to be patched, patches had to be propagated both forward and backward to older or more recent releases (including the development area,), and development branches and releases had to be accommodated in a much simpler way.

Further, better control over submission and inclusion of code from developers was required. Sub-system integration levels were needed where small groups could test their code before submitting it up a hierarchical development tree so as not to break the entire product if their changes were incompatible with changes made by other groups.

Finally, to simplify management of changes from many developer groups, individual changes needed to be logically grouped together so that they were manageable as a single unit, e.g. allowing a higher level integrator to incorporate or reject such "changesets" as appropriate.

Many third-generation products were developed to address the deficiencies of the second-generation products, and most are expensive[2]. A relative newcomer is BitKeeper, a commercial product that is being offered free to academic/research and open-source projects such as Gluex. I know of no other widely used, state of the art third-generation product that is as rich in features and free for us.

III. Important BitKeeper Concepts

III.1 Background

BitKeeper is basically a cross-platform, improved version of Teamware, a third-generation product created by Sun to manage the development of Solaris, and subsequently sold to Solaris customers. The BitKeeper founder, Larry McVoy, originally worked on the Teamware project before creating his own company to develop BitKeeper. Also, he was a major player in the open-source world, making important contributions to many projects. His goal in creating and marketing BitKeeper was both to provide a state of the art product to business at a much reduced cost compared to similar products, and to make the product available to the open-source world at no cost.

III.2 Licensing and Openlogging

McVoy's innovative but controversial BitKeeper license allows him to do this. It is innovative in that business customers have an incentive to pay, to protect their privacy, whereas open-source projects sacrifice unneeded privacy to get BitKeeper for free. The

[2] Note that some freeware efforts, such as Subversion, aim to create a better second-generation product. Subversion fixes many CVS problems and adds some new features, but does not address the major problems of second generation software.

key to this is "openlogging", described below. The license is controversial because a few purists in the open-source world refuse to use anything that does not have a GPL or similar license. I have read the license very carefully and find no problems with it as far as Gluex is concerned[3].

With openlogging all checkin administration information is logged to a publicly accessible site (the code itself is not logged). Open-source projects have no problem with this, and indeed find it an advantage. But business customers absolutely do not want this information accessible as competitors could potentially glean important information about the company's software effort from it. Even worse, competitors might try to hire away the programmers who seem to be doing the best work (names are logged along with checkin comments!). Thus under the BitKeeper license businesses pay to disable the openlogging feature; their logging is done only to a private, internal site.

III.3 Code Development Model

CVS supports only a single repository from which developers check out copies of the code, work on it, then check everything back in, usually with a single comment. In BitKeeper, as in Teamware, developers "clone" an existing repository and work within it. The original is called the "parent' and the cloned repository is called the "child". Since the child is also a repository, others can clone from it, making it both a child and a parent. Thus a hierarchical tree structure of repositories can be constructed, with intermediate levels serving as integration areas for child repositories below them. Both the depth of this development tree (number of integration stages), and the width of the tree (how the software effort is divided up), can be tailored for each development effort.

The tree structure can be reorganized through "reparenting", or resetting the parent of a child repository, a feature that is extremely useful for release management. For example, one possible release strategy is to clone the development repository, then reparent the clone to the current release, then reparent the development tree to the clone.

This results in a time-ordered chain of releases, with each release repository being the child of the previous release repository, and with the development repository always being the child of the most recent release (many other release strategies are possible). Developers can clone and patch any release repository, and the patches can be propagated backwards or forwards to all relevant releases, including the development tree. Similarly, features developed in the development tree can be propagated back to previous releases if needed.

Developers generally work within a single, private repository, usually one of the leaves of the development tree. All checkins are done locally, and changes are private until the developer "pushes" the changes up the tree to the parent integration repository (the inverse is where the developer "pulls" new code from the parent into the child repository). Developers are encouraged to check their code in locally (with comments) frequently, resulting in a fine-grained history record. In contrast, in CVS checkins

[3] The current BK public license is reproduced in Appendix B.

modify the single main repository, and thus developers are discouraged from checking files in until they think the code is ready, resulting in an extremely coarse-grained history record.

Related checkins can be grouped into a single "changeset" (using the "commit" command), a unit that is labeled and can be manipulated. For example, if a changeset is applied to a parent repository and found to be unacceptable, the parent can back-out the changeset and become restored to its previous state.

There are many other interesting and useful features in BitKeeper, especially the many graphical management tools, but the ones described this chapter are the critical ones for Gluex. See the BitKeeper web site (www.bitkeeper.com) for further details, or use BitKeeper's excellent online help system (on any JLab CUE solaris or linux machine type: $ /apps/bk/PRO/bitkeeper/bk helptool).

IV. Who Will Be Affected, and How

Although BitKeeper has a rich and diverse command set, the average programmer needs to learn only a small subset of commands similar to those commonly used with CVS. Thus switching from CVS to BitKeeper will be quite simple for most people. And to be honest, the advantage of BitKeeper over CVS for the average programmer is not large.

On the other hand, integrators and release managers will need to use many of the powerful features of BitKeeper, and the advantages to them are major. Although they will have to learn a lot more, they will be delighted with the simplicity and ease of use of BitKeeper.

For the discussion below I divide Gluex programmers into three groups: casual programmers who produce a few thousand lines of code or less; major programmers and integrators who produce or are responsible for many tens of thousands of lines of code; and release managers who are responsible for integrating and releasing the entire code base.

IV.1 Casual Programmers

Casual programmers familiar with CVS need to understand the differences between the CVS and BitKeeper development models. In particular, BitKeeper supports a tree of repositories, and casual programmers will work at the bottom of the tree. Above them are integration levels, with the topology set by software managers.

Casual programmers start by creating a private clone of the integration level just above them and working within it. After creation all files in their private repository are locked and cannot be edited. To work on a file they must first check out a copy for editing (can be done within emacs, for example). After making a unit of change to the file it should be checked in again with an informative comment. This can be repeated as needed.

When a significant unit of change has been performed on a related group of files these are committed as a changeset. Note that only checked-in files can comprise a changeset, and that as yet all changes are local to programmer's private repository.

Eventually the changesets need to be pushed back to the parent integration repository. This will have to be coordinated with the gatekeeper or integrator of that level, and many strategies are possible. A simple one is for children to push their code back to the parent at will. At an agreed upon time they stop and the gatekeeper/integrator runs benchmark tests. Problems are worked out between them until the code is successfully integrated, at which point all the child repositories are resynchronized or updated by pulling from the parent.

Eventually the gatekeeper/integrator decides to push the code up to the next integration level, and the integration mechanism for the level above is performed. In this way code flows up and down the development tree in a controlled, documented, and robust manner.

In appendix A I present a list of the commands most commonly needed by casual programmers, and their CVS equivalents.

IV.2 Major Programmers and Integrators

Major programmers will do the same as casual programmers but may find use for many BitKeeper features beyond the basic set. BitKeeper provides a rich set of commands and graphical tools to examine the history of files or changesets, to move or delete files, display file differences, etc.

It is likely that major programmers will also act as gatekeepers or integrators. Again BitKeeper supplies numerous additional tools to examine who pushed what back into an integration workspace, the ability to back out of changesets, triggers that can execute scripts when certain actions are performed on the repository (e.g. push from a child), the ability to lock the workspace against update by a child, etc.

I don't think a universal strategy is applicable to all integration workspaces…each group must tailor procedures for their own situation. A variant of the strategy mentioned in the previous section likely will suffice for simple cases.

IV.3 Release Managers

Release managers work at the topmost level in the development repository tree, and a large number of release strategies are possible. The simplest was mentioned earlier, and involves cloning the topmost development repository and renaming it appropriately. This repository is then added to the end of an existing chain of release repositories (via reparenting), and the development tree is made a child of the new release. Note that here the development area is unchanged other than being reparented to the new release.

If a problem is found in a release it can be fixed in numerous ways. It could be addressed in the development area, with the patch being propagated back to the release (possibly through many releases). Alternatively, first a clone can be made of the release itself. Then the problem is fixed in the clone (or a clone hierarchy, if needed), and the patch pushed back to the release. If appropriate, the patch can be propagated back to earlier releases via push or forward to later releases (including the development area) via pull.

A white paper by Sun (http://www.jlab.org/Hall-D/documents/software/teamware.ps, written for Solaris Teamware) discusses a number of development and release strategies possible when using a hierarchical repository system.

V. Summary and Recommendations

I strongly believe that the gatekeeper model, utilizing a tree-structured repository hierarchy, is the appropriate software development model for Gluex. I note that BitKeeper, unlike CVS, is designed to implement it. The gatekeeper/BitKeeper model has been successfully adopted by a large number of commercial and open-source projects, and it seems to work quite well. Indeed use of a hierarchical gatekeeper model is the norm in most large commercial projects.

Adoption of gatekeeper/BitKeeper model by Gluex promises to save us significant time and manpower during the software development effort, and may be essential to allowing us to complete our software on time. This is because this model provides an excellent mechanism to divide the software effort into manageable pieces, to control and document the flow of code back and forth between coders and releases in a robust manner, and to ensure accountability at all stages.

BitKeeper is a full-featured, distributed, multi-platform commercial product with a large customer base, yet it is free for us. It is under active development, and is constantly updated to meet the demands of commercial customers. In contrast, CVS is a frozen product, and the only efforts related to CVS are either layers on top of it, or efforts intended to replace it.

BitKeeper is simple to use and casual programmers will find it similar to CVS. Once a few new concepts are understood they will have little problem switching. Major software developers, integrators, and software managers will find BitKeeper quite rich in management tools designed to implement the hierarchical gatekeeper model, yet still simple to use.

Many strategies are possible within the hierarchical gatekeeper model, and Gluex software experts need to choose among them (see the Sun Teamware note referenced above). Note that different strategies can be applied at the different levels of the repository hierarchy.

I propose that choice of a development model and implementation tools be discussed at the weekly Gluex software meeting and a decision made as soon as possible. I further propose that all new projects be developed with the chosen model and tool set, but that we discuss whether to retroactively convert existing projects (esp. the simulation efforts), as this may be counterproductive at this time. I note that conversion from CVS to BitKeeper is very simple via the "bk import" command.

In summary, I find the hierarchical gatekeeper model implemented via BitKeeper to be an excellent match to our requirements, and the compelling choice for Gluex.

Appendix A: Common BitKeeper Commands

BK Command	Description	CVS Equivalent
bk helptool	Start graphical help system (use this excellent utility to get help)	
man bk man bk-cmd	Display bk man page, or display page for command "cmd" (e.g. man bk-new)	man cvs
bk clone	Create local clone of an existing repository	cvs checkout
bk new	Add new file to local repository	cvs add
bk edit	Unlock local files for editing	
bk ci	Check in local files, include comment (Repeat edit/ci cycle as needed)	
bk commit	Commit local files as changeset (Repeat after edit/ci cycles as needed)	
bk push	Push files back to parent repository	cvs commit
bk pull	Update local repository from parent	cvs update
bk status	Print summary of repository status	cvs status
bk changes –L	List changes in local but not in parent repository	cvs status
bk changes -R	List changes in parent but not in local repository	cvs status
bk diffs	Display file differences	cvs diff

Appendix B: BitKeeper License

bk bkl - display free use BitKeeper license

BitKeeper License version August-18-2003

1. DEFINITIONS

BKL: This license in its entirety, also known as the BitKeeper License.

You: The licensee of the BitKeeper Software.

BitMover: The licensor of the BitKeeper Software.

BitKeeper Software: The complete set of executable programs and any accompanying files, such as documentation, known as the BitKeeper Software. The set of programs and files must include all files and programs distributed by BitMover as part of the BitKeeper Software.

BitKeeper Package: A set of files managed by the same BitKeeper ChangeSet file. There may be multiple instances of the package; each instance is called a repository.

Single user BitKeeper Packages: One or more BitKeeper Packages wherein all changes to all files are made by the same person and the total number of unique files over all BitKeeper Packages does not exceed 1000.

Metadata: Information about the data managed by the BitKeeper Software in a BitKeeper Package, such as

+ The ChangeSet file;

+ The messages which annotate modifications of the data (also known as check in comments, ChangeLog entries, and/or log messages);

+ All infrastructure files contained below the top level BitKeeper directory in a BitKeeper Package. User data files, i.e., files contained in the BitKeeper/deleted directories are explicitly excluded.

Open Logging: The transmission of Metadata about the data managed by the BitKeeper Software, to a functioning Open Logging server in the openlogging.org domain (or an alternative domain as posted on www.bitkeeper.com/logging). Examples of such collected information may be seen at http://www.openlogging.org.

2. LICENSE GRANTS

Licensees may install and use the BitKeeper Software for its intended purpose.

3. LICENSEE OBLIGATIONS

(a) Maintaining Open Logging Feature: You hereby warrant that You will not take any action to disable or otherwise interfere with the Open Logging feature of the BitKeeper Software. You hereby warrant that You will take any necessary actions to ensure that the BitKeeper Software successfully transmits the Metadata to an Open Logging server within 21 days of the creation of said Metadata. By transmitting the Metadata to an Open Logging server, You hereby grant BitMover, or any other operator of an Open Logging server, permission to republish the Metadata sent by the BitKeeper Software to the Open Logging server.

(b) Accessing Others' BitKeeper Package: You may only use the BitKeeper Software to access a BitKeeper Package created by BitMover or third parties if You comply with the license of the BitKeeper Package, which can be found at the BitKeeper/etc/REPO_LICENSE file within the BitKeeper Package and/or by running bk repo_license.

(c) Maintaining Open Source: It is the intent of BitMover that Your use of BitKeeper under this license is for the purpose of maintaining Open Source. By accepting this license, You agree that You are prepared to demonstrate Your conformance, at the request of BitMover, by making your BitKeeper repositories publicly available via the BitKeeper protocol within 15 days from the time of such request. In the event that You do not wish to make BitKeeper repositories publicly available, You have 15 days in which to negotiate a waiver, convert said repositories to closed use, or cease use of said repositories.

(d) No free use for competitors: Notwithstanding any other terms in this License, this License is not available to You if You and/or your employer develop, produce, sell, and/or resell a product which contains substantially similar capabilities of the BitKeeper Software, or, in the reasonable opinion of BitMover, competes with the BitKeeper Software.

(e) No combination with competing products: Inclusion of the BitKeeper Software for use with a system having substantially similar capabilities of the BitKeeper Software requires prior written permission from BitMover.

(f) Staying current: This license is terminated in the event there is a new release of the BitKeeper Software which has associated regression tests and said regression tests would not be passed by this version of the BitKeeper Software. This license is terminated in the event there is a new release of the BitKeeper Software which contains any

changes to any part of the licensing functions, including but not limited to Open Logging.

(g) No reverse engineering: You may not yourself and may not permit or enable anyone to: (i) modify or translate the Software; (ii) reverse engineer, decompile, or disassemble the Software or otherwise reduce the Software to a form understandable by humans, except to the extent this restriction is expressly prohibited by applicable law notwithstanding this limitation; or (iii) provide access to the metadata created and managed by BitKeeper to any person or entity which is not licensed to use the BitKeeper Software.

(h) Public reference: By using the BitKeeper Software, You agree to the public use of your name and/or your companies' name as a user of the BitKeeper Software.

4. NON-CONFORMING USE

4.1. Single user packages

For single user BitKeeper Packages, Open Logging is optional. The total number of allowed single user files is 1000 per licensee.

4.2. Closed Use

Closed use is the use of the BitKeeper Software without participating in BKL licensing restrictions such as Open Logging. Closed use of the BitKeeper Software requires that You (or your organization) purchase closed use licenses for all users of the BitKeeper Software within your organization. This license, the BKL, does not convey authority to make closed use of the BitKeeper Software.

4.3. Logging Waivers

Certain sites which do not wish to participate in Open Logging, such as educational or research institutes, may apply for, and may be granted, a written waiver from BitMover, Inc. After applying for a written waiver, such an institution may use the BitKeeper Software without Open Logging, for up to 90 days, or until a response is received from BitMover, Inc., whichever comes first. Should BitMover not grant your waiver request, You have the option of converting to Open Logging, immediately terminating your use of the BitKeeper Software or continuing your use after purchasing closed use license[s].

4.4. Damages

Use and/or copying of modified versions of the BitKeeper Software is a violation of copyrights held by BitMover on the BitKeeper Software. Use of the BitKeeper Software without a license is a violation of copyrights held by BitMover on the BitKeeper Software. Damages for copyright infringement are the greater of actual damages or statutory damages, which are cur-

rently up to $150,000 per infringement.

This license is not available to You if You and/or your company have any unresolved copyright disputes with BitMover.

5. DISCLAIMER OF WARRANTY

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN ``AS IS'' BASIS, WITHOUT WARRANTY OR INDEMNIFICATION OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES OR INDEMNITIES CONCERNING INTELLECTUAL PROPERTIES (E.G. PATENTS OR COPYRIGHTS), WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. SHOULD ANY PORTION OF BITKEEPER SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU ASSUME THE COST OF ANY RESULTING DAMAGES, NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF BITKEEPER SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT SUBJECT TO THIS DISCLAIMER.

6. TERMINATION

+ This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein. Provisions which, by their nature, should remain in effect beyond the termination of this License shall survive.

+ If any of the licensing requirements, such as Open Logging, are found to be unenforceable, then this license automatically terminates unless You continue to comply with all of the licensing requirements.

+ Should You or your organization choose to institute patent, copyright, and/or intellectual property litigation against BitMover, Inc. with respect to the BitKeeper Software, then this License and the rights granted hereunder will terminate automatically as of the date such litigation is filed.

+ If this License is terminated for any reason, You must delete all copies of the BitKeeper Software and cease using the BitKeeper Software.

7. LIMITATION OF LIABILITY

TO THE FULL EXTENT ALLOWED BY APPLICABLE LAW, BITMOVER'S LIABILITY TO YOU FOR CLAIMS RELATING TO THIS LICENSE, WHETHER FOR BREACH OR IN TORT, SHALL BE LIMITED TO ONE HUNDRED PERCENT (100%) OF THE AMOUNT HAVING THEN ACTUALLY BEEN PAID BY YOU TO BITMOVER FOR ALL COPIES LICENSED HEREUNDER OF THE PARTICULAR ITEMS GIVING RISE TO SUCH CLAIM, IF ANY.

IN NO EVENT WILL BITMOVER BE LIABLE FOR ANY INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THIS LICENSE (INCLUDING, WITHOUT LIMITATION, LOSS OF PROFITS, USE, DATA, OR OTHER ECONOMIC ADVANTAGE), HOWEVER IT ARISES AND ON ANY THEORY OF LIABILITY, WHETHER IN AN ACTION FOR CONTRACT, STRICT LIABILITY OR TORT (INCLUDING NEGLI-

GENCE) OR OTHERWISE, WHETHER OR NOT SUCH PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY.

8. MISCELLANEOUS

8.1. Merger

This License represents the complete agreement between You and BitMover regarding the BitKeeper Software covered by this License. BitMover reserves all rights not specifically granted herein.

8.2. Assignment

BitMover may assign this License, and its rights and obligations hereunder, at its sole discretion.

8.3. Severability

If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. If any provision of this License is held to be unenforceable, the enforceability of the remaining provisions of this License will not be impaired thereby.

8.4. Governing Law/Jurisdiction

This License shall be governed by the laws of the US and the State of California, as applied to contracts entered into and to be performed in California between California residents. By using this product, You submit to the jurisdiction of the courts in the Northern District of California.

Licensing