LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	**LIGO-T080241-00-Z**	23 October 2008

VCS Committee Report

Oliver Bock, Franco Carbognani, Kipp Cannon, Ed Maros, Greg Mendell, Adam Mercer (chair), Reinhard Prix, Jameson Rollins, Keith Thorne

Distribution of this draft:

LIGO Scientific Collaboration

Processed with $\LaTeX$ on 2008/11/07

Contents

1 Charge

1. Recommend version control systems for software development and document preparation.
2. Consider the software development models currently in use within the LV collaboration.
3. Consider the document preparation and sharing mechanisms currently in use within the LV collaboration.
4. Account for support of the VCS on multiple operating systems including Windows, Linux, and Macs.

2 Current Version Control Usage

The LIGO Scientific Collaboration (LSC) currently uses the Concurrent Versions System (CVS) as the "official" Version Control System (VCS) for both software development and document management. Currently there are several other VCSs in usage throughout the LSC for smaller projects. Therefore in order to determine the exact VCS usage within the collaboration the questionnaire included in Appendix A was sent out to the collaboration.

2.1 Questionnaire Responses

In total, 26 responses were received from a wide range of people spanning the entire collaboration. A summary of the received responses can be found in Appendix B.

2.2 Version Control in the Virgo Collaboration

CVS constitutes the "official" VCS for software development and partially for document management for the Virgo Collaboration. The centralized CVS repository is hosted at the Virgo detector site in Cascina and is managed by the Virgo Software Librarian. The repository contains 100% of the Virgo Detector Control Software, and the inclusion of all Data Analysis software packages, and corresponding documentation, is currently in progress. Other software/specification/design documents are stored in the repository as separate modules, along with webpages.

The repository has around 110 active users and is accessed mainly from the command line on Linux/Unix platforms, access from Windows using a dedicated GUI (tortoise) is currently being tested. Web access is made available via CVSweb. The need for VCS features not supported by CVS has not yet been raised as an important issue. The use of others VCSs, such as git, is rare. On those few cases developers work with local archives and then commit their released package versions to the central CVS repository.

In order to facilitate VCS adoption the Virgo Software Librarian developed a thin layer of shell commands on top of CVS, called SCVS [1], that is currently used by a fair fraction of the user community. SCVS defines a simplified CVS use mode (mainly based on package level lock) tailored on the Virgo Software development practises (e.g. package definition and identification).

3 Recommendation

As is clear from the questionnaire responses there is not a single VCS that will satisfy the needs of all projects, librarians, developers, and users within the collaboration. Also from the responses there are currently three different VCSs in active use within the collaboration: Git [2], Subversion [3], and CVS [4].

Subversion, like CVS, is a centralized VCS and according to the Subversion project webpage it "was originally designed to be a better CVS"[5], it therefore targets the same development and social models as CVS. Git is a distributed VCS, and therefore its strengths are more evident when used in distributed development and social models - it does not however need to be used in a distributed setup.

In order to provide the most flexibility to project librarians it is the recommendation of the Version Control Subcommittee that either Git or Subversion should be used for new software development and the support of CVS for legacy projects should be provided, i.e. no new software project should use CVS. The final decision on what VCS, and which social/development model, to use is left up to the librarian of the project in question.

3.1 Authentication and Authorization Project

As the collaboration is currently in the process of migrating to a new authorization and authentication system it needs to be ensured that any recommendation from this committee is compatible with this new system. More information on the authorization and authentication project can be found on the AuthProject Wiki [6].

At a basic level the AuthProject provides two authentication methods: Kerberos tickets and X.509 (RFC 3820) proxy certificates. As sshd can be configured to authenticate against both Kerberos tickets and X.509 proxy certificates and Git, Subversion, and CVS can all be used over ssh, they are therefore compatible with the AuthProject. In addition, both Subversion and CVS can be configured to authenticate using Kerberos tickets directly.

3.2 Bug/Issue Tracking

Another important issue to consider when investigating VCSs is how they interact with various bug and issue tracking systems (BTS). As CVS and Subversion are relatively mature, it is generally safe to say that if a given BTS supports VCS integration at all then CVS and Subversion are supported. Git is a much younger VCS and therefore integration support varies. However, it's rate of popularity increase is quite high and BTS integration support is improving steadily.

At the moment Trac [7] seems to have the most complete Git support, with Redmine [8] being close behind. Git support for other BTS like GForge [9], Savane/Savannah [10], or Bugzilla [11] is still limited but under active development. Another option to consider is SCMbug [12], a project that is striving to provide a generic BTS–VCS integration interface that explicitly supports Git.

References

[1] https://www.lsc-group.phys.uwm.edu/daswg/wiki/VCSComm?action=AttachFile&do=get&target=scvs_usr_man.pdf.

[2] http://git.or.cz.

[3] http://subversion.tigris.org.

[4] http://ximbiot.com/cvs.

[5] http://subversion.tigris.org/features.html.

[6] https://www.lsc-group.phys.uwm.edu/twiki/bin/view/AuthProject.

[7] http://trac-hacks.org/wiki/GitPlugin.

[8] http://www.redmine.org/wiki/redmine/RedmineRepositories.

[9] http://gforge.com/gf/project/scmgit.

[10] https://savannah.gnu.org/maintenance/Git.

[11] http://code.istique.net/?p=git-bugzilla.git;a=summary.

[12] http://www.mkgnu.net/?q=scmbug.

A Version Control System Usage Questionnaire

The LSC currently uses CVS, the Concurrent Versions System, in order to manage the development of analysis software, such as LAL, and in the preparation of publications and web pages. Version Control Systems (VCSs) keep track of all work and all changes in a set of files, and allows many individual to collaborate easily and efficiently.

CVS was first released in 1986, and whilst still under development, is beginning to show its age. Therefore, a sub-committee of the Data Analysis Software Working Group (DASWG) has been charged to investigate the current usage of VCSs within the collaboration, and to recommend VCSs for both software development and document preparation/webpage management.

In order to ascertain the current usage and requirements of VCSs we are soliciting input from all collaboration members on the following questions, repeated for both software development and document preparation.

Software Development

1.1 What VCSs do you currently use for software development?

1.2 For what working group?

1.3 What is housed in these repositories?

1.4 How often do you checkout from these repositories?

1.5 How often do you commit to these repositories?

1.6 How many people actively use these repositories?

1.7 What platforms are these repositories accessed from?

1.8 Can you access the repositories from all platforms you want?

1.9 What VCS features do you regularly use (e.g. tag, branching/merging, etc.)

1.10 Are there any features lacking from the current VCS (e.g. rename, off-line usability, atomic commits, etc.)?

Document Preparation/Webpage Management

2.1 What VCSs do you currently use for the management of documentation and/or webpages?

2.2 For what working group?

2.3 What is housed in these repositories?

2.4 How often do you checkout from these repositories?

2.5 How often do you commit to these repositories?

2.6 How many people actively use these repositories?

2.7 What platforms are these repositories accessed from?

2.8 Can you access the repositories from all platforms you want?

2.9 What VCS features do you regularly use (e.g. tag, branching/merging, etc.)

2.10 Are there any features lacking from the current VCS (e.g. rename, off-line usability, atomic commits, etc.)?

Please send all responses to ram@gravity.phys.uwm.edu by 9am CDT Tuesday 16th September 2008.

B Version Control System Usage Questionnaire Responses

B.1 Software Development

1.1) What VCSs do you currently use for software development?

CVS:	18
SVN:	13
GIT:	10

1.2) For what working group?

Burst:	8
Pulsar:	8
CBC:	7
Detector Characterisation:	3
Calibration:	1
DASWG:	1
Ext. Trigger:	1
GEO:	1
Stochastic:	1

1.3) What is housed in these repositories?

Glue/LAL/LALApps/PyLAL:	12
MatApps:	6
General Code:	5
DMT:	3
Omega Pipeline:	2
Cluster Configuration:	1
Einstein@home:	1
LDR:	1
LIGOTools:	1
QPipeline:	1
XPipeline:	1

1.4) How often do you checkout from these repositories?

Daily:	13
Weekly:	9
Several Times Per Year:	1

1.5) How often do you commit to these repositories?

Daily:	16
Weekly:	5
Monthly:	1
Several Times Per Year:	1

1.6) How many people actively use these repositories?

O(10):	17
O(100):	6

1.7) What platforms are these repositories accessed from?

Linux:	23
Mac:	14
Windows:	8
FreeBSD:	1

1.8) Can you access the repositories from all platforms you want?

Yes:	19

1.9) What VCS features do you regularly use (e.g. tag, branching/merging, etc.)?

Tags:	18
Branching/Merging:	11
Bisect:	1
Email notification:	1
Exporting/Importing patches:	1
Move/rename:	1

1.10) Are there any features lacking from the current VCS (e.g. rename, off-line usability, atomic commits, etc.)?

Better Branching/Merging:	10
Move/rename:	9
Off-line:	5
Atomic Commits:	4
Status:	3
Access Control:	2
Decentralised:	1
No repository wide ID:	1
Remote Tracking:	1
Unable to modify file permissions:	1

B.2 Document Preparation/Webpage Management

2.1) What VCSs do you currently use for the management of documentation and/or webpages?

CVS:	19
SVN:	6
GIT:	3

2.2) For what working group?

Burst:	9
CBC:	5
Pulsar:	5
Detector Characterisation:	3
DAC:	2
Ext. Trigger:	2
DASWG:	1
GEO:	1
LSC P&P:	1
Stochastic:	1

2.3) What is housed in these repositories?

Documentation:	18
Webpages:	11
Papers:	9
Minutes:	3
Talks:	3

2.4) How often do you checkout from these repositories?

Weekly:	8
Daily:	7
Monthly:	7

2.5) How often do you commit to these repositories?

Weekly:	8
Daily:	7
Monthly:	3
Yearly:	1

2.6) How many people actively use these repositories?

O(10):	16
O(100):	5

2.7) What platforms are these repositories accessed from?

Linux:	20
Mac:	12
Windows:	10
FreeBSD:	1
Unix:	1

2.8) Can you access the repositories from all platforms you want?

Yes:	18

2.9) What VCS features do you regularly use (e.g. tag, branching/merging, etc.)?

Tags:	6
Branching/merging:	4
Move/rename:	1

2.10) Are there any features lacking from the current VCS (e.g. rename, off-line usability, atomic commits, etc.)?

Rename/Moving:	6
Off-line:	4
Better Branching/Merging:	3
Status:	2
Access Control:	1
Atomic commits:	1
Automatic binary file recognition:	1
Checkout subset of repository:	1
Decentralised:	1