

Requirements for CVS and Build Servers

FOCUS

03.05.2001

Presented by: M. Stavrianakou (ATLAS)

Prepared and/or signed by:

C. Arnault (ATLAS), F. Carminati (ALICE), M. Cattaneo (LHCb), S. Fisher (ATLAS), R. Jones (ATLAS), H. Meinhard (ATLAS, CHORUS), A. Pfeiffer (IT/Anaphe), D. Quarrie (ATLAS), M. Stavrianakou (ATLAS), H.-P. Wellisch (CMS), T. Wildish (CMS)

Preamble/disclaimer

- This is a first collection of requirements for CVS and build services, as understood by individuals more or less actively involved in configuration management tasks in several CERN experiments and projects.
- These requirements do not necessarily reflect “official” or even “majority” views of the experiments/projects.
- These requirements are not meant to be conceived of as “requests” to IT or other groups.
- A conscious, if not entirely successful, effort has been made to avoid implementation details or suggestions.
- A similar effort has also been made to focus on the general and common rather than the experiment/project specific.
- The effort to accommodate most, if not all, views and comments contributed may have not been entirely successful; apologies offered in advance.

CVS server (I)

- ✍ files stored on a local file system, backed up to a shared file system
 - ? files accessible **ONLY** via the server
 - ? reliability/security
 - ? understand configurability and limitations on read/write access
 - ? **BUT** definitely need access control (read/commit/tag/import access) via appropriate mechanism; crude password schemes not sufficient
 - ? all possible users must be "known" to the server
 - ? repository access controlled by experiments concerned at user, group, project and collaboration-wide levels
 - ? better performance
 - ? server recommended if developers connected by slow or flaky networks
 - ? login access for a few individuals
 - ? required to fix occasional CVS glitches

CVS server (II)

? system requirements

✍ disk:

✍ rule of thumb: $\sim 3 \times$ (size of code to be in CVS repository)

✍ also: fast disk and file system for many simultaneous small disk operations

✍ memory:

✍ big memory consumption in

? large checkouts:

? rule of thumb: \sim (size of each CVS server) \times (number of servers expected to be active at one time)

? diff esp. when checking in large files; required even for binary files:

? rule of thumb: $(5 \text{ to } 10) \times$ (size of largest file to be checked in)

✍ fast network connection:

✍ CVS sends many small packets...

CVS companion services and add-ons

- WWW access
 - for browsing, checkouts etc
- other WWW services:
 - e.g. dependency scanners and graphers, author/coordinator tables etc
- query manager for applications
 - e.g. show all tags of a particular module
- monitoring, quality control and metric services
 - to be run by authorised individuals in order to avoid conflicts, resource waste etc
 - e.g. cleaning stale CVS locks, code checking at commit time,...
- multi-repository services
- automated documentation services
 - language (in)dependent?
- integration with problem tracking system

Build services: Reference platforms

- one reference machine for each architecture:
 - well defined & stable OS/compilers/libraries
 - where changes are agreed by all concerned
 - ? enough local disk
- ? for regular builds and releases, including nightly builds (experience has shown that it is better to build locally and then install on a shared filed system e.g. AFS)
- ? possibility to set up a temporary certification server quickly and easily, with all the services provided by the reference platform
- ? possibility to replicate easily at regional centres

Build services: Companion services (I)

- WWW services for automated/remote build management on all supported platforms
- automated/remote build job submission
 - at requested granularity (package, subpackage, directory, file level)
 - without interactive login (or even account ownership) requirement for "submitter"
 - with automatic, configurable email notification
- build analysis
 - i.e. scanning of build logfiles and presentation of results (warnings, errors etc)
 - in easy-to-understand visual format
 - with or without automatic, configurable email notification

Build services: Companion services (II)

- other WWW services
 - dependency scanners and graphers, author/coordinator lists etc
- configuration management services
 - needed resources, external libraries, etc
- release distribution services
- multi-site operations
 - GRID related?

Some comments and conclusions (I)

- probably good to have finally collected some if not all requirements for services we have been requesting and/or implementing
- *BUT what for?*
- *And what next?*
- experiment and/or project specific facilities (CVS and build services) either already in place or being implemented
- build and release mechanisms obviously tied to the different tools in use

Some comments and conclusions (II)

- too little in common, too late?
- too limited (experiment and IT) resources to consider convergence and common effort?

it depends...

- obviously commonalities do exist
- identified commonalities, if exploited, may or even should
 - allow sharing of information, experience and specific products
 - promote some common maintenance and support schemes
 - help formulate informed recommendations