CVS and Other Tools

Matthew Worcester

Background and Simulation Meeting

December 11, 2004

Outline

- CVS
- Code Browser
- Thoughts

CVS

- CVS has all versions of files that have been checked into the server
  - no code ever lost
  - different versions of files can be “tagged” into frozen releases
  - files are retrieveable by version, tag, date
- Server for ReactorFsim: cp4.uchicago.edu
- Basic user instructions available online: braidwood.uchicago.edu/private/software/cvs.html

CVS Tools

- cvs diff: diffs local files against the repository
- cvs log: provides history of files from the repository
- cvs update: merges in changes from the repository more recent than when the local version was checked out
- cvs commit: checks changes in local files into the repository (requires a comment for log)

CVS Tools

- When code has reached a certain milestone versions of the files can be “tagged” together into a release:
  - provides a common point of reference for all developers and users
  - users have stable code to work with
- Current ReactorFsim system: whenever Matt feels that enough has changed and doesn’t know about any (new) bugs

Now available

- CVS log messages and file modification times are updated automatically online: braidwood.uchicago.edu/private/software/cvsweb/index.html
  - developers must make comments specific and detailed when committing code
- Automatic email sent to developers whenever new updates are checked in
  - all developers must be on the email list
- Goal: prevent conflicting code changes

CVS Guidelines

- provide specific and detailed comments when committing code (make the log files useful)
- always check the online logs and update your code before committing (don’t burn others’ updates)
- users start from frozen releases

Code Browser

- Generated by Doxygen
- Scans any types of file for doxygen-compatible markers:
  - based on C++ comment markers: /** or ///
  - developers can easily keep their C++ comments doxygen-compatible
  - encourages documentation in the code
- Outputs html and latex documentation

Thoughts

- How complicated to make the BW software?
- Design a software package:
  - installed on users' PCs or a set of collaboration accessable analysis machines
  - sets up a known software/analysis tool structure
  - contains pre-built libraries for each frozen release
  - development code can be checked out of cvs and built on top of a software release
- Probably not necessary… yet

Thoughts

- How much automatic checking is needed?
- Code in the CVS repository head should be checked out and built every night
- run resulting executables as standard jobs?
  - check job log files for errors
  - define baseline histograms to compare to output from automatic jobs
- Who is notified of failures and what is the expected time of response?