

CS 350, slide set 12

M. Overstreet
Old Dominion University
Spring 2005

Reading

- TSP text, Appendix B: Software Configuration Management

Typical Problems

- Uncontrolled, unreviewed changes
- Corrupted or unrecoverable source code
- Many slight variants of "current" version
 - For different OSs
 - For different hardware/software environments
 - For different customers
 - For different in-house developers/users
 - Stable, infrequently changing version for other developers
 - Most current and changing version of some developers

Major components

- A Software Configuration Plan
- A system baseline
 - Submission policy
 - Backup procedures
 - Configuration audit procedure
- Testing facilities
- Specialized
 - Hardware
 - Support tools

Config. Mgmt in CS 350

- Minimal – no CCRs!, no CSRs! required
 - Each group can decide what is appropriate for the group
- Next several slides discuss how this this might be done

Typical plan components

- Must depend on size/complexity of project
- Issues
 - Steps in process
 - Approval decisions
 - Reduce bureaucratic overhead
 - Avoid disasters
 - Baseline submissions
 - Automated support tools
 - Simplify manual tasks
 - Reduce errors; impact of proposed changes
 - When are backups done?
 - When are builds done?

Contents of Configuration Change Request (CCR)

- Product/Component Identification
- Change Information
 - Reason for change
 - Change description
 - Impact of change
- Status & Approvals

Configuration Change Board

- Evaluate proposed changes to baselined products

Contents of Configuration Status Report – TSP version

- Config. Change Process: Activity
 - CCRs submitted, approved, rejected, deferred, outstanding, reversed
- Config. Change Process: Status
 - Volume under SCM control: pages, Pseudocode lines, LOC—total, LOC—new and changed, Test case LOC, Test material pages, Test results pages

Baseline steps: TSP version

- Individual completes and tests approved changes
- Approval by quality/process mgr
- CCB reviews submissions and approves or disapproves
- Support manager retains backups of all baselined items
- Team members can normally obtain baselined items at any time
- Development manager ensures that only baseline products are used in builds
- Support manager produces weekly CSRs

RCS: UNIX Revision Control System Overview

- Free with UNIX (Linux)
 - Only partial solution
 - Commercial systems provide many more services
- Consistent with UNIX philosophy: programmers are smarter than software systems
 - Does not **enforce** rules
 - Designed to be used by knowledgeable, cooperating programmers
 - That is, control is voluntary; really easy to "cheat" – and sometimes necessary.

RCS continued

- Works with make
 - Make knows about RCS and will look in for missing source when needed
- Makes it easy to recover old versions of files
 - Can see exact changes made to a file, along with date of changes and who made changes
- Alerts if two people try to change a file at the same time.

Basic RCS structure

- Assumes related source code is in several files in one directory
- Assumes the existence of an RCS directory in the source directory
 - Use "mkdir RCS" to create this directory
- Tries to reduce size of files by only recording changes made

Basic RCS commands "ci" - 1

- "ci <filename>" to remove file from current directory and put a version under RCS control
- This is "checking in" a file to RCS
 - A structured version of the file, along with some documentation, is placed in the RCS directory called <filename>,v
 - You can edit this file. Don't.

Basic RCS commands "ci" - 2

- When you check a file in:
 - RCS checks to see if you really made changes
 - No reason to keep two identical versions
 - RCS prompts you to document your changes
 - But doesn't force it.
 - RCS assigns a version number (by adding 1 to the old number) so that you can recover exactly this version in the future.

Basic RCS commands "co"

- If a file has been checked in, it can later be "checked out" with the "co" command.
 - `co <filename>`
 - Everything that can be checked out will be a file in the RCS directory
 - If I forget filenames, I often start with
 - `ls -l RCS`
- Use of "ci" means that you promise **not** to change the file, but just need to read it
 - For, say, compilation of your component with baselined versions of other components
- You can edit the file anyway. Don't.

Basic RCS commands, "co -l"

- Unlike use of "co", "co -l" (for "lock") is used when a programmer intends to modify a file.
 - `co -l <filename>`
- The file is then "locked."
- If another team member uses "co -l" for the same file, he/she will get a message that the file is already checked out.
 - UNIX assumes that if you really need it, you will contact the person who checked it out and make some arrangements.
 - You can easily edit the file anyway (this is UNIX). But you should now know that what you're doing may be dangerous.

More info

- `man rcsintro` or `rcs` or `co` or `ci` or `ident` or `rcsclean` or `rcsdiff` or `rcsmmerge` or `rlog` or `rcsfile`
