3C05: Advanced Software Engineering

Unit 10: Configuration Management

Stefanos Zachariadis
http://www.cs.ucl.ac.uk/staff/s.zachariadis/

Unit 18: Configuration Management

Objectives

- To introduce the concept of configuration management and the key problems which can arise from configuration management failures and collaborative development.
- To present some techniques for configuration management.
- To consider tools for automated configuration management support.

Scale and Complexity

- Projects tackled so far:
  - Few (usually one) developer
    - In the same location
  - Hundreds/few thousands lines of code
  - Small Scale (tackle limited problems)
- Increase the scale
  - Group Projects (2-? developers)
    - Distributed around the world
  - Tens, hundreds of thousands lines of code
  - Operating Systems, Office Suites, Web Browsers, Middleware systems, etc.

Issues

- Multiple people working on code & documentation
- Access to other people's changes
- Conflicting updates
- Simultaneous Updates
- Tracking/Logging of Changes
- Tracking bugs
  - When did it arise?
  - What changes were made to trigger it?

Configuration Management

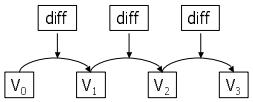
- Systematic way to control changes to large (or small) software systems. Allows for
  - Keeping track of all files in a project (including additions and deletions)
  - Groups working on same project
  - Tracking changes (documents, source code etc)
  - Branching
  - Selecting versions
  - Merging changes

Basic Concepts

- Repository
  - Where (usually all) files for the project are located
- Committing a change/file
  - Uploading your version of the file to the repository
- Versions of a file
  - Different *revisions* of a file, committed by a developer.
  - Has an identifier (e.g. 1.1, 1.2, 1.3…)
  - Can be stored as
    - Distinct files (more space required)
    - Deltas (more processing required)

Versioning Deltas

- More commonly used
- A list of changes/differences from one version to another
  - A "diff"
- Forward Deltas

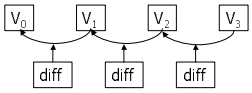


- Processing required to fetch the latest version

7

Versioning Deltas (2)

- More commonly used
- A list of changes/differences from one version to another
  - A "diff"
- Reverse Deltas
  - Easy to fetch the latest version from the repository



8

How to Efficiently Use CM Tools

- Use Open (Text) Formats
  - .TEX, .TXT, .JAVA, .CPP, .XML, .HTML, etc.
  - Binary data do not work well with CM tools
    - Difficult to track changes (create diffs) to binary data
  - Open formats can be processed in multiple platforms by various programs.
    - Do not assume all people in the group use the same software as you
- Modularise Project
  - Makes it easier for more people to develop
  - Use Interfaces and Abstract Classes in coding.
    - Allows the programmers to use the API expected without worrying about the implementation (that is being worked on by another person).
- Standardise on coding/naming conventions

9

Example Tool: RCS

Revision Control System

- Works over individual files
- Reverse Deltas (all versions in the same file with a ,v suffix)
- Logs of changes are maintained in the file
- "Check out the Latest Version" to process it.
- "Check in the changes"
- Developers work on the same directory hierarchy
- When a file is checked out, it is locked.
- Others cannot checkout the file if locked
- Difficulty in supporting multiple programmers
- Difficulty in having multiple versions of a file concurrently
- Good for single-user projects (e.g. documents)

10

Example Tool: CVS

Concurrent Versions System

- Widely used on the internet
- Based on RCS
- Network Oriented
- Works on Projects (uses RCS for the individual files)
- Users work in their own directory hierarchy
- User "checks out" a snapshot of project
  - Works on it independently
  - Can "update" the snapshot with changes other users made
- User commits changes
  - If changes conflict with those made by other users, CVS prompts on how to resolve conflict.
- Can have different branches of the project
  - E.g. experimental branch, stable branch

11

Example Tool: CVS (2)

Problems with CVS:

- Lack of Atomic Commits
- Difficult to change structure once created
  - E.g. Need shell access to the repository to delete directories
- Does not support symbolic links
- Special treatment needed for binary files
  - Sometimes necessary, e.g. graphical images.

12

Example Tools: Other

- BitMover BitKeeper
  http://www.bitkeeper.com/
- Rational ClearCase
  http://www.rational.com/products/clearcase/index.jsp
- Microsoft Visual SourceSafe
  http://msdn.microsoft.com/ssafe/
- Some of the tools integrate with development environments (Emacs, Jedit Visual Studio, etc)

13

Summary

- Configuration management is the most important component of information management in software development.
- Automated tools such as RCS/CVS can be used to provide configuration management and solve the issues arising with multiple developers working on the same project
- The key elements of CM software is the logging of changes and versioning.

14

Links for Casual Reading

- A collaborative development environment for open source projects which includes a bug tracking system, forums, mailing list, CVS support etc.
  http://www.sourceforge.net
- Karl Fogel, "Open Source Development With CVS"
  http://cvsbook.red-bean.com/cvsbook.html
- The GNU Project, RCS.
  http://www.gnu.org/software/rcs/rcs.html
- Ant, a Java-Based Build Tool.
  http://jakarta.apache.org/ant/

15