Lecture 9: Configuration Management

CS190: Software System Design

February 22, 2002
Steven P. Reiss

I. Today's Class

A. Configuration management

B. How to write code as a team

C. How to keep your code safe

II. Configuration Management

A. Why configuration management

1. Keeping track of all the files in a project

2. Keeping track of previous versions of a project

a) To allow recovery if you do something wrong

b) To allow debugging of a released version

3. Allowing multiple programmers on a project

a) So that changes don't interfere

b) So that they know who is doing what

4. To provide a change history of a project

5. To support multiple versions of a system simultaneously

B. Basic concepts

1. Files have versions

a) A version reflects the file's contents at a point in time

b) Versions can be open or closed

c) Versions can be organized in a sequence or a tree

2. Files have locks

a) Files are locked by a particular user for a particular purpose

b) Locks are used to ensure cooperation

3. File versions can be merged

a) To handle changes from multiple sources

b) To handle merging of version trees

III. RCS

A. History

1. SCCS/RCS are some of the earliest CM systems

2. They are still widely used

B. Basic principles

1. All users work in the same directory hierarchy

2. Locking is done at the individual file level

3. Check-in, check-out model

4. Logs are maintained in the file

5. Files are stored as deltas (all versions in one file)

C. Problems

1. Difficult to have multiple versions simultaneously

2. Difficult to support multiple programmers on same file

3. Granularity doesn't match programmer's

IV. CVS

A. History

1. CVS atteempts to fix the problems with RCS

2. Actually built on top of RCS

B. Basic principles

1. Each user works in their own directory hierarchy

a) Users can have multiple hierarchies for different versions

b) Users can work independently

2. Snapshot/Commit model

a) User gets a snapshot of the system

b) cvs update grabs that snapshot, updating local files

c) After changes are made, user commits all changes in a directory (or directory hierarchy)

d) cvs commit commits a directory

3. Commit does a merge of new and old versions

        a) Detects changes since snapshot
        b) Asks user to help if changes overlap
    4. **Still maintains logs, stores files as deltas**

V. Other systems

A. ClearCase

1. **Based on work done at Apollo**
2. **Directory-based approach**
3. **But versioning is done as part of the file system**
    a) Environment variables indicate which version should be used
    b) File system provides that version when accessing file by name
    c) Check-in, check-out model

B. Shape

1. **Integrate configuration management and make**
2. **Make rules for specifying which version**
3. **Shape automatically accessed that version**