Lecture 14: Configuration Management & Midterm Review

Kenneth M. Anderson

Software Methods and Tools

CSCI 3308 - Fall Semester, 2004

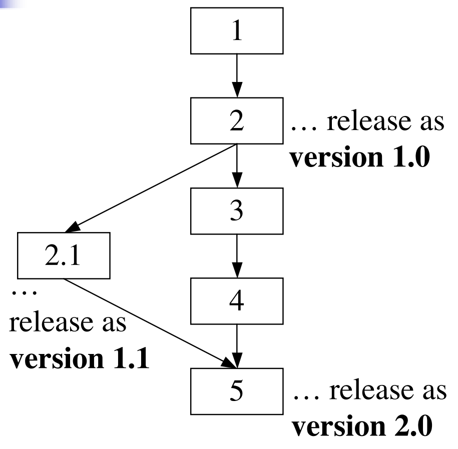
Review of Versioning

- Versioning involves
  - tracking the changes to a file between editing sessions
  - providing services that make each version persistent and retrievable
  - providing support for complex dependencies between versions such as extensions, splits, and merges
- Note: the emphasis is on a single file
- What about collections of files?

Configuration Management

- Versioning a collection of files is known as configuration management
  - A collection can occur at many levels of granularity
    - the collection of files that make up a module
    - the collection of files that make up a library
    - the collection of files that make up a subsystem
    - etc.
- NOTE: each file is still individually versioned, but now we can track the configuration to which a particular version belongs

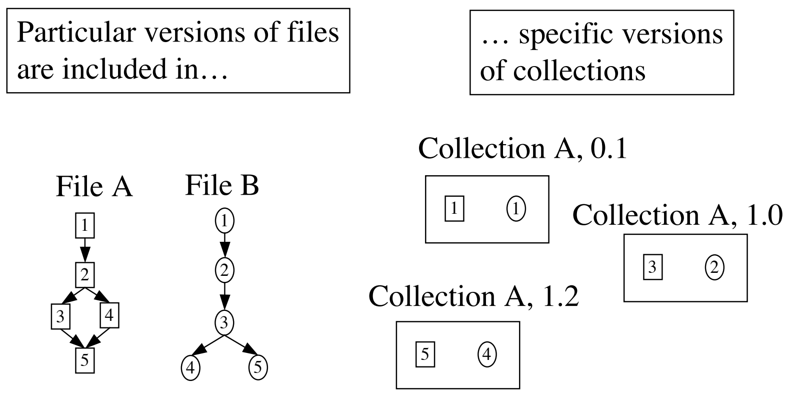
Relation of Versioning to CM


1
2
… release as **version 1.0**
3
2.1
… release as **version 1.1**
4
5
… release as **version 2.0**

Remember, in last lecture, how the **version** number (1, 2, 3, etc.) had nothing to do with the **release** number (1.0, 1.1, etc.)?

The release number is the version number of a configuration!

Configuration Management Example


Particular versions of files are included in…
… specific versions of collections
File A
1
2
3
4
5
File B
1
2
3
4
5
Collection A, 0.1
1
1
Collection A, 1.0
3
2
Collection A, 1.2
5
4

Configuration Management, cont.

- Configurations become first-class objects that can be manipulated by explicit commands
  - (Versions of) Files can be added/removed from configurations
  - Configurations can be checked in and checked out
    - This helps with bug tracking, if a customer reports a bug on release 1.3, the software engineer can check out a clean copy of release 1.3 without affecting the current release
    - Each developer can have their own copy of a configuration; changes to collections are handled similarly to changes to individual files
  - Configurations can be automatically built and packaged for deployment

Configuration Management Tools

- Unfortunately, most configuration management tools are commercial systems
  - ClearCase, Continuus, Razor, TrueChange
- Tools like RCS and CVS are versioning systems
  - CVS has only one feature that provides a configuration management-like capability
    - Its called “tags” and it allows you to tag a particular version of a file with a release number…
    - … but that’s it! It does not have an explicit notion of collections that can be versioned independent of its individual files
- However, the open source community has recently released a new configuration management system called *subversion:* <http://subversion.tigris.org/>

Midterm Review

- In-Class Midterm on Monday
  - worth 100 points
- This review is presented at a high-level
  - We can go back to slides from previous lectures in response to questions

No Silver Bullet

- Fred Brooks claims there is no silver bullet to solve the “software crisis”
  - A silver bullet would be a single technique that leads to an order of magnitude improvement in the production of software
- He divides the problems facing software engineers into accidental and essential difficulties
  - The essential difficulties include complexity, changeability, conformity, and invisibility

Fred Brooks Continued

- Other chapters covered in MMM
  - Tar Pit
    - Programming System ; Joys and Woes of Craft
  - The Mythical Man-Month
    - Adding people to a late project…
  - The Surgical Team
    - Formalizing communication paths
  - Aristocracy, Democracy, and System Design
    - Conceptual Integrity

Fred Brooks, continued

- The Second System Effect
  - Architects need extra self discipline on second system in a class of programs
  - Beware changes in assumptions between versions
- Why Did the Tower of Babel Fail?
  - Communication, Project Workbook, Director and Producer
- Software Tools
  - Generic vs. Specific Tools

Deployment

- Deployment is the process of delivering software to a user after it has been created
  - We want this process to be “engineered”
  - We need to support the deployment lifecycle
    - Producer Side
      - New Release and/or Update
      - Retirement (of obsolete versions)
    - Consumer Side
      - Install/Uninstall
      - Update
      - Adapt (to changing environment)
      - Reconfigure (to meet new needs)

Unix and the Shell

- The Unix Architecture is split
  - between user-level programs, the kernel, and devices
- The Shell is a user-level program that provides an interpreted programming environment
  - Shell Variables/Environment Variables
  - Math Operations/C Operators
  - Input/Output Redirection
  - Job Control
  - Control Flow Constructs

Pattern Matching

- Wildcards
  - Used to match sequences of characters, digits, etc.
  - “a*.c” - all files that start with a, have any number (including zero) of characters or digits after the a, and end in .c
    - abc.c, a.c, a123.c, …
- Regular Expressions
  - Used to match sequences of patterns
  - ab*c, matches zero or more instances of the pattern “ab” followed by the pattern “c”
    - c, abc, ababc, abababc, etc.

Find & Grep

- Find
  - Tool to search directories and files
    - via sequences of boolean operations
  - Makes use of wildcards and can invoke external operators
- Grep
  - General Regular Expression Processor
  - Tool to search the contents of files using regular expressions
- Both help software engineers deal with large systems (that is, they are scalable)

Build Management

- An engineered process for building software systems
- Process can be supported by tools
  - e.g. Make
  - These tools attack accidental difficulties
    - They free developers from having to remember code dependencies

Make

- Makefiles are specifications that provide precise control over build management
  - If something changes, only those files impacted by the change are recompiled (as opposed to the entire system)
- Make is well-integrated with Unix/C and provides
  - rules: targets, dependencies, and actions
  - **macros** (variables), VPATH, and **automatic variables**
  - **pattern matching** and implicit rules

Software Reuse

- Software consists of
  - source code, binaries, requirements and design documents, etc.
- Any of these parts can be re-used
  - Requirements and Design re-use is especially powerful since we are attacking essential difficulties when we create this type of information
- Source code and object code re-use
  - Pros: Source code can be modified, Object code does not need to be recompiled
  - Cons: Source code has to be modified(!), Object code can not be extended and is architecture specific

Unix Libraries

- A technique for re-using collections of object code
- Enabled by marshalling
  - rules for passing parameters to object code; requires object code and .h files
- ar is used to create libraries
  - naming convention: **lib***name***.a**
- Compilers provide -I, -L, -l flags to use libraries

Versioning & RCS

- Version Control
  - Track changes to a file between editing sessions
  - Version Graph supports extension, split, and merge and is stored in a version control file'
  - Version control files make use of deltas to save space
  - Version control systems provide check-in, check-out, and other capabilities
- RCS: backward-delta version control system
  - numbering scheme: branch number.version number
  - ci and co are primary commands; rcs, rlog, rcsdiff
  - Provides Keywords like $Author$