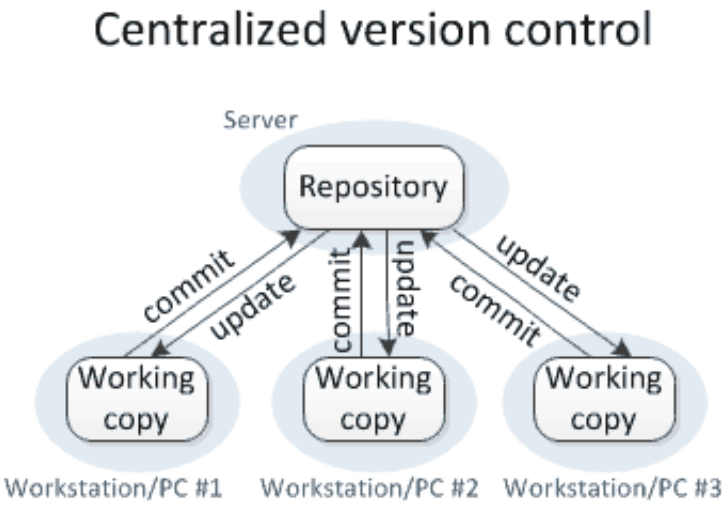
CS 315 - Lecture 5 - Sep 2, 2015

Software Versioning

- Software Change
  - Whenever a product is maintained, there will be a version
- Staged Model
  - Initial
    - First running version
  - Evolution
    - Evolutionary changes
    - End of evolution
  - Servicing
    - Service patches
    - Servicing discontinued
  - Phaseout
    - Switch-off
  - Closedown
- Version Staged Model
  - Used with software that has a large user base
  - Evolution is the backbone of this process
  - For each version, the normal staged model occurs
- There's a problem
  - How do we keep track of all these versions?
    - Folders?
    - What about large projects?
  - What about pushing bug fixes into multiple versions?
  - Collaboration?
    - Manually merge changes?

        - Disallow parallel work on the same files?

- Solution: Version Control



    - Project Tracking
        - Who did what?
        - When were artifacts created/removed?
        - Provides complete evolution

    - Collaboration
        - Easy Sharing
        - Guarantee of synchronization

    - Protection
        - Data loss
        - Bad/undesired code

    - What gets versioned?
        - Text
            - Truly versioned
            - Differences between version can be shown and documented
            - Line by line tracking

        - Binary (non-text)
            - Each version must be completely stored
            - No line by line granularity

    - Repository (aka "repo")
        - Stores all data associated with a project
        - Centralized
            - Canonical copy is stored on the server

        - Decentralized
            - Each user has their own local repository

- Trunk, Branch, and Tag
    - Trunk
        - Main body of the program

    - Branch
        - A copy of the trunk made at the time of branch
        - Major revisions
        - Experimental features
        - Prototyping

    - Tag
        - A stable release/milestone of the project

- Diff
    - Produces a patch file that identifies what changes the programmer made in a file
    - Comparison of the new and existing file

- Merge
    - Takes original file and patch file
    - Merges the changes in the old file, thus creates a new file
    - Types
        - Automated
            - Often works, but may not be available

        - Semi-automated
            - Requires some input on what changes to keep

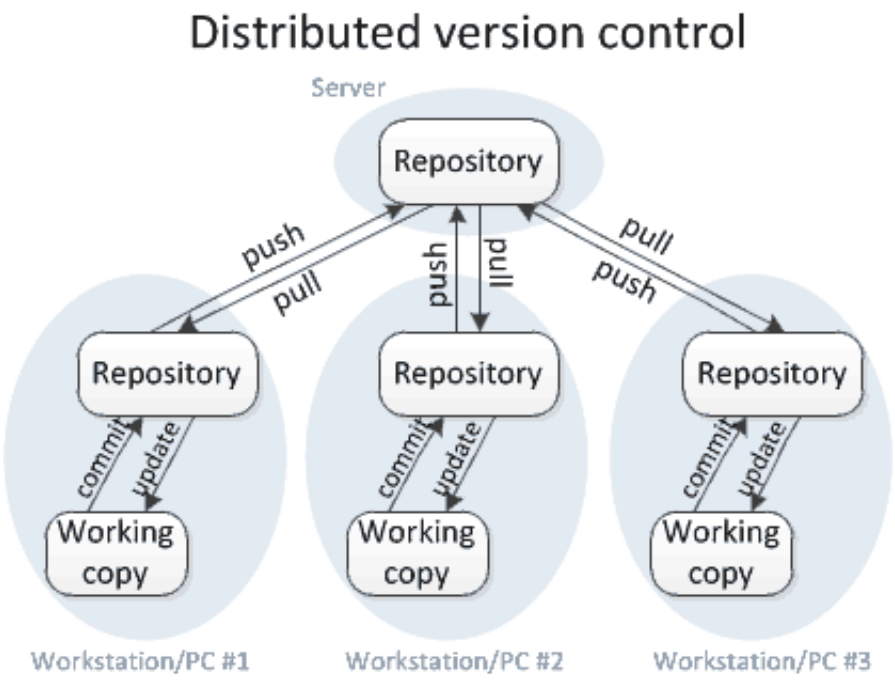
        - Manual
            - Tedious and error-prone

- Checkout/Update
    - Copies a file from a repo to local storage
    - Programmer changes the copy in local, not the repo version
    - Protects the code in repo until it's ready to be merged
    - First Time = checkout
    - After = update

- Commit
    - Writes the modified file back to the repository

        - Generally handled using diff and merge
            - Diff creates a patch of changes between the repo version and local version of a file
            - Merge applies the patch to the repo version
            - Repo is now up-to-date
        - Commit tracks the changed files as a new "revision" or "version" of the file
    - Lock
        - Prevents the marked files from being committed to
        - Useful For
            - Making critical updates
            - Merging may be difficult
            - Exclusive access is required
    - Conflict
        - Happens when programmers work on the same file
        - Must be resolved
            - Merge files using semi-autonomous or manual tools
            - Clobber the upcoming changes
            - Lose your own change, get the new file and make your changes again
            - A better practice: Always try to work on different files
                - Separate source cods into files
                - Virtual Paradigm Projects into different files

- Examples of VCS
    - Centralized
        - SVN
            - Mature VCS, released in 2000
            - Native support for binaries
            - Easy to use
            - Must be connected to the central VCS server
    - Decentralized
        - Git



- New, 2005
- Distributed
- Do not have to be connected to the server
- Repos everywhere (not just on the server)
- Terminology
  - Clone
    - Same as checkout but also creates a repo for your local computer
  - Fetch
    - Check if anything changed in remote repo
  - Stage (or add)
    - Approve changes to be committed
  - Commit
    - Commit changes to local repo
  - Push
    - Applies the local commits to remote repository
  - Pull
    - Pull down changes from the remote repository