Version Control Systems

Phil Pratt-Szeliga

Fall 2010

Version Control Systems: Overview

- Definitions
- Motivation
- Client-Server Version Control Systems
  - Subversion
- Distributed Version Control Systems
  - Git
  - Mercurial
- Version Control with Test Driven Development
- References

Definitions

- Revision Control
  - “Management of changes to documents, source code or other information stored as computer files”
- Version Control System
  - Software tools to support revision control

Motivation

- Individual Usage
  - It is 10pm. You are two hours before the deadline of project 1 code.
  - You are trying to get something to work better and you make quick change to your reflection library
  - It doesn't work so make yet another change, and another, and so on
  - You tried so many things to make your project perfect that it no longer works at all
  - You need to redo your work
  - How many people have been here?
  - With Version Control Systems and correct usage this can disappear

Motivation

- Team Usage in the same physical location
  - The new person on your team at work makes a change and breaks everything
    - It will take days or weeks to put things back to how they where
    - With Version Control Systems it will less of a problem
  - You are a manager and an experienced person makes a great change that solves a long standing problem
    - What did he or she do?
    - With Version Control Systems you can easily look at differences from one version to the next

Motivation

- Team Usage over the Internet
  - How can we easily have one place to put files?
  - How can we easily get new versions of files?
  - How can we review changes that contributors make?
  - Version Control Systems answer all of these questions

Motivation

- Team Usage in the same or different physical locations
  - How do we let two people work on the same file
  - Do they coordinate with each other
    - Over the phone?
    - Through email?
    - Instant messanger?
  - How do you know who is editing a file?

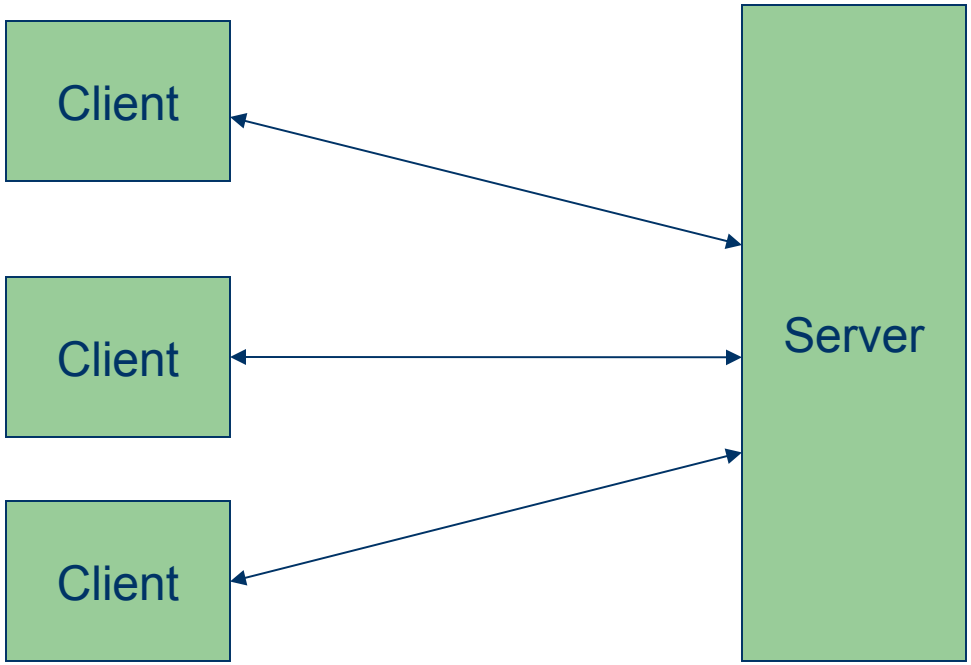
Motivation

- Releasing a product to the customer
  - Whenever you give product code to a customer you should keep an exact copy of all source for yourself
  - When they call you up and there is a problem you need to be able to go to the version you sent them and fix the problem
  - Version control systems can facilitate this

Client-Server VCS

- In Client-Server VCS there is one centralized server and a number of clients
  - The centralized server holds all the files under revision control
  - The clients access the files from the centralized server
- Subversion is the most popular Client-Server VCS designed to replace CVS

Client-Server VCS



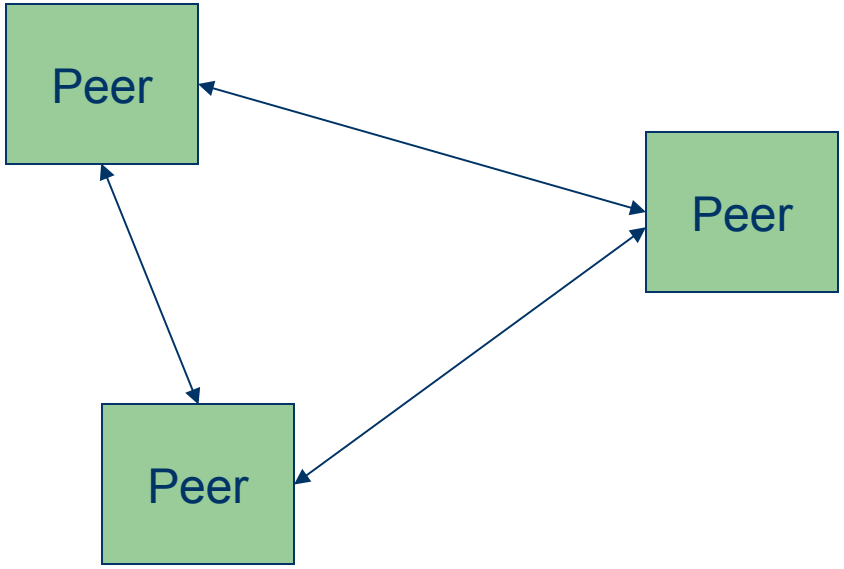
Subversion Client-Server VCS

- Many Open Source Projects are converting from Subversion to DVCS (git, mercurial, bzr)
  - Making a commit requires a network connection
  - Merging is harder than in mercurial
  - Second tier contributers are left with no versioning system
- Free hosting for open source projects:
  - http://code.google.com/

Distributed VCS

- In Distributed VCS there is a peer to peer model
  - When you want to get the code, you download the entire repository contents
  - You can then simply host a repository by allowing incoming connections
  - DVCS is used in Open Source Software Development
    - The maintainer of a project could stop answering his or her emails
    - The other developers already have the full repository.

Distributed VCS



Mercurial DVCS: Definitions

- Init
  - Make a new repository (all mercurial files are stored in the .hg folder)
- Clone
  - Get a full copy of another repository
- Add
  - Tell Mercurial what files you want to be placed in the next commit
- Commit
  - Make a revision from all the added files that has a message of what has been changed

Mercurial DVCS: Definitions

- Pull
  - Get the changes from a peer
- Push
  - Send your changes to a peer
- Make sure you commit before pushing and pulling

Mercurial DVCS: Definitions

- Update
  - Make a selected revision the current revision
  - Files on the filesystem are overridden
  - Make sure your commit or before updating
- Merge
  - Incorporate the changes from two repositories that have a common root into one new repository

First Project Usage Of Mercurial

- Each student will have his/her own folder on a server provided by SU
- You can use Mercurial on your individual computer and make backups on the server
- Sharing code with other students in the first project is forbidden! Doing so can result in an F in the course.

Final Project Usage of Mercurial

- Each student will have his/her own directory on a server provided by SU
- Use this folder to share code with other students
  - Access is configured by HgRepoManager
  - Pull from someone's repository
  - Incorporate the changes you want from them
  - Push to your repository and with their permission, push to their repository

Practices of using Version Control

- When you start the day get the latest version
- Before submitting your code, get the latest version again and compile and test again
- Don’t submit your code unless it at least compiles. Preferable to only submit code that functionally works

Version Control with Test Driven Development

1. Get latest
2. Compile
3. Add a new unit test
4. Compile
5. Develop to make a unit test pass.
6. Get latest (merge latest changes)
7. Compile
8. Run Unit Tests
9. If they pass push to maintainer

(No code gets pushed if it doesn't compile and the unit test passes)

References

- http://en.wikipedia.org/wiki/Revision_control
- http://wiki.netbeans.org/MercurialVersionControlScreenshots
- http://www.python.org/dev/peps/pep-0374/