Topics for this Lecture

- Software maintenance in general

- Source control systems (intro to svn)

Topic 1: Software Maintenance

- This is not how software engineering works:
  - First, design happens,
  - Second, implementation happens,
  - Third, testing confirms implementation and design were successful
  - Fourth, the entire thing is frozen in carbonite forever and released to customers

Software Maintenance

- A second false idea is that the process of generating a working system is much more complicated than that, but
  - At some point “the working version” is complete, and after that most changes are (small) bug fixes.
  - This is the “software maintenance is like an oil change” theory: once in a while you have to make a small corrective action, but that’s it

Software Maintenance

- In fact, up to 80% of maintenance efforts are **not** bug fixes
  - Software systems that *are not* used simply die
  - Software systems that *are* used evolve to match their user environment
    - Adding features
  - Adding features adds complexity
  - Adding complexity requires occasional *re-factoring* unless you want to end up with code that requires a "software archaeologist" to understand

Topic 2: Source Control

- How do we manage all these changes?
  - Real-world projects are not a set of three .java files, or two .c files and two .h files
  - Real-world systems are complicated trees of source files, support files, documentation, test cases, and configuration files
  - Multiple developers work on the tree and make changes to it
    - May want to go back to an old version
    - May want to work on a file when you don’t have access to a shared network location
    - Dropbox/copying around a zipped version is clumsy and prone to disaster

Source Control Systems

- Use **source control systems** (also known as revision control or version control)
- Common examples:
  - rcs
  - cvs
  - svn (subversion)
  - git
- We’re going to use a small subset of svn, which I will introduce on the next few slides
  - svn is widely available and has a simple model
  - svn is what OSU’s beaversource website (a repository for hosting software projects) uses

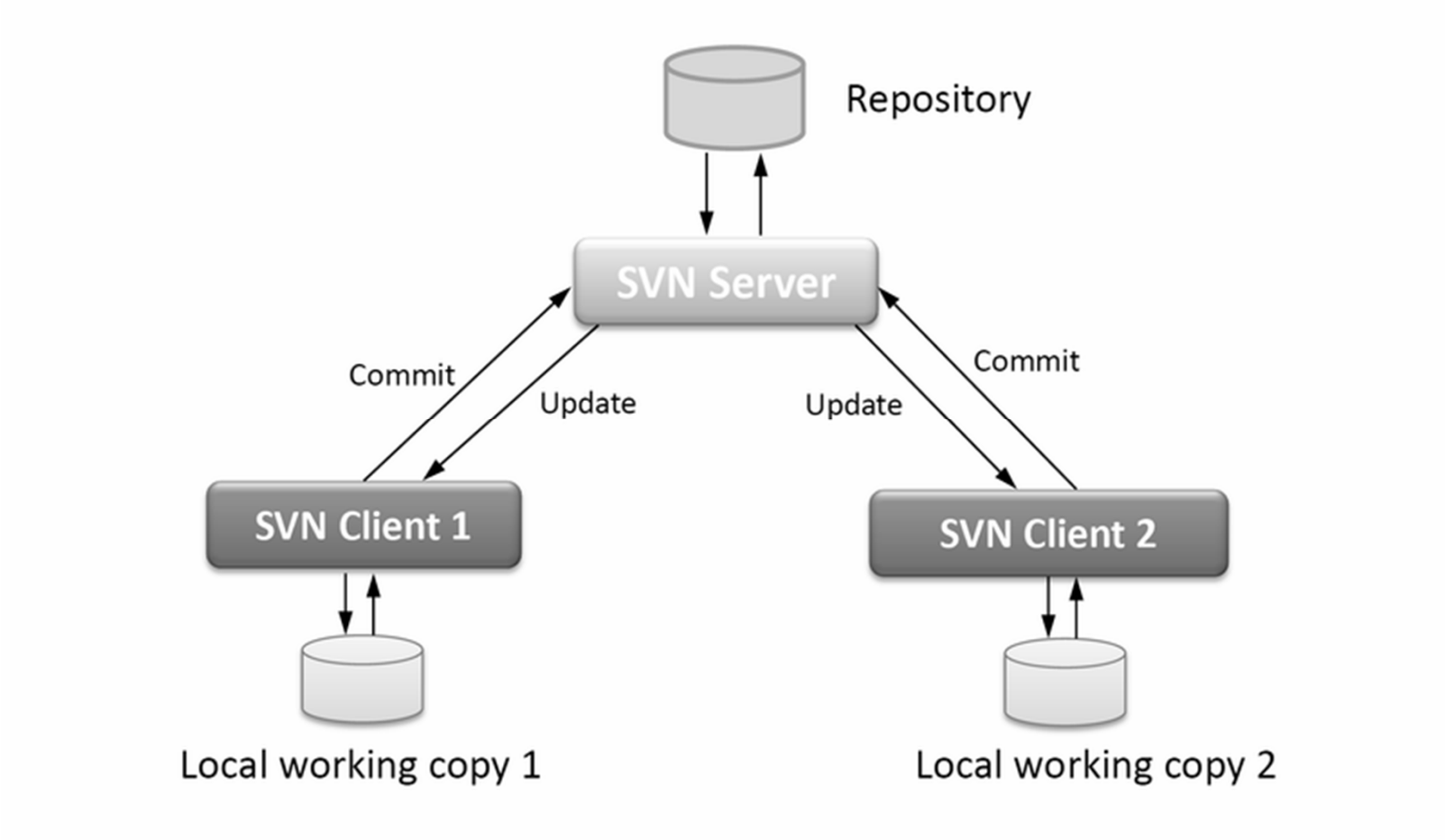
Intro to svn

- All source control systems:
  - Track changes made to a software system
  - Allow merging of changes to a part of a system
  - Allow the development of multiple versions of a system

- We’re going to look at just three operations:
  - Checking out a project
  - Checking changes into a project
  - Updating a project to get the latest changes made by others

  - Also look at the web interface to svn provided by beaversource

Intro to svn


Repository
SVN Server
Commit
Update
Update
Commit
SVN Client 1
SVN Client 2
Local working copy 1
Local working copy 2

Getting Started with svn

- Log in to beaversource at http://beaversource.oregonstate.edu (use your ONID account)

- Go to a command prompt (cygwin or unix/linux)
  - Navigate till your current directory is a location you want to store your class code repository
  - `svn co --username <onid username> https://code.oregonstate.edu/svn/cs362class`
  - A new directory will be created below your current directory location, containing all the cs362 material
  - This is the contents of the repository

Creating Your Own Project Space

- Navigate into the cs362class directory projects section
  - `cd cs362class/projects`
- Create your own directory for your code and tests:
  - `mkdir <onid username>`
- Is your code in the repository where everyone can see it now?
  - We can check by going to http://beaversource.oregonstate.edu/projects/cs362class/browser
  - Navigate in your web browser to the projects directory

Making Your Work Visible to Others

- How do you add your directory to the repository?
  - `svn add <onid username>`
- Is your code in the repository *now?*
  - http://beaversource.oregonstate.edu/projects/cs362class/browser
- Why not??
  - Adding a directory or file doesn't actually put it in the repository, it just tells svn you will eventually *check it in*
  - I've personally held up a Mars mission by forgetting this fact!

- One more step:
  - `svn ci <onid username> -m "Initial checkin"`

Failed Permissions?

- If this didn’t work, you probably don’t have permission to add things to the class repository
  - Need to contact someone else who does, and knows how to add permissions on beaversource

Seeing the Work of Other People

- Other people may change their files, add files, etc.
- How do you get the latest version of everything?
- Navigate to the top of the repository (or where ever you want to update everything below) and try:
  - `svn update`

SVN REMINDERS

- You can look on beaversource to see if your code is actually checked in, and what version it is
- You need to add every file and directory you want other people to be able to look at
  - If you add a directory svn will automatically try to add all the files in that directory
- Just adding things doesn't actually put it in the repository
- You have to *check in* your changes