Objectives

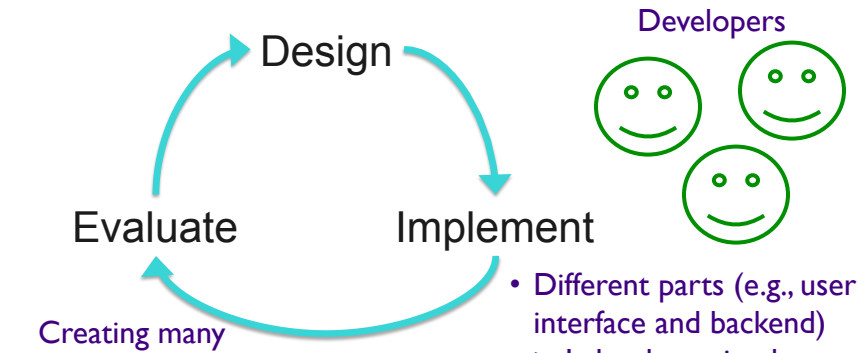
- JUnit
- Collaboration
- Software Testing Issues

JUnit

- Review from last Wednesday

VERSION CONTROL

Problems in Collaborating on Code



Creating many prototypes
- What if don't like recent prototype? Need to go back to older version

- Different parts (e.g., user interface and backend)
- > 1 developer implementing concurrently
  - *What if one introduces a bug?*

Version Control Features

- Synchronization
  - Lets people share files
  - Stay up-to-date with the latest version
- Backup and Restore
  - Files are saved as they are edited
  - Revert to a specific version/checkpoint
- Track changes to code
  - Save comments explaining why change happened
  - Stored in the VCS, not the file
  - Track how, why a file evolves over time
- Track ownership
  - Tags every change with the name of the person who made it

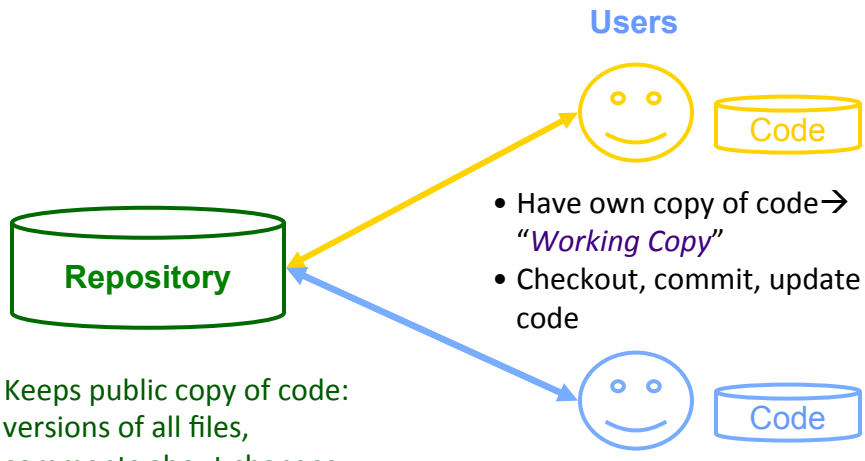
Version Control Features

- Short-term undo
  - Messed up a file? Go back to the last good version
- Long-term undo
  - Created a bug a year ago? Jump back to see change you made.
- Sandboxing
  - Making a big change? Make temporary changes in isolated area, test, work out kinks before "checking in" your changes
- Branching and merging
  - Branch a copy of your code into a separate area, modify it in isolation (tracking changes separately)
  - Later, merge work into common area

Version Control Systems

- Popular Version Control Systems
  - CVS, Subversion, Git, …
- Terms used are common for most version control systems
- We will use Subversion with Subclipse
  - Mark Phippard, a W&L grad works on both
    - Director of the Subversion engineering team at CollabNet, the company that founded Subversion

Using Version Control

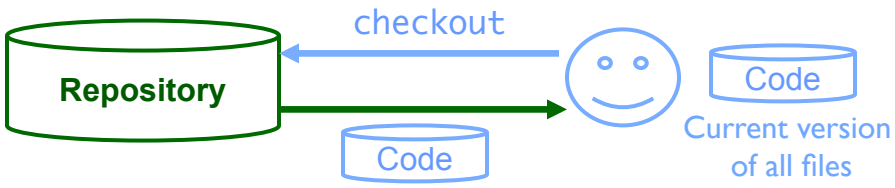


- Have own copy of code→ "*Working Copy*"
- Checkout, commit, update code

- Keeps public copy of code: versions of all files, comments about changes, who made changes

Using Version Control: `checkout`

- To start, need to **`checkout`** your working copy of the code



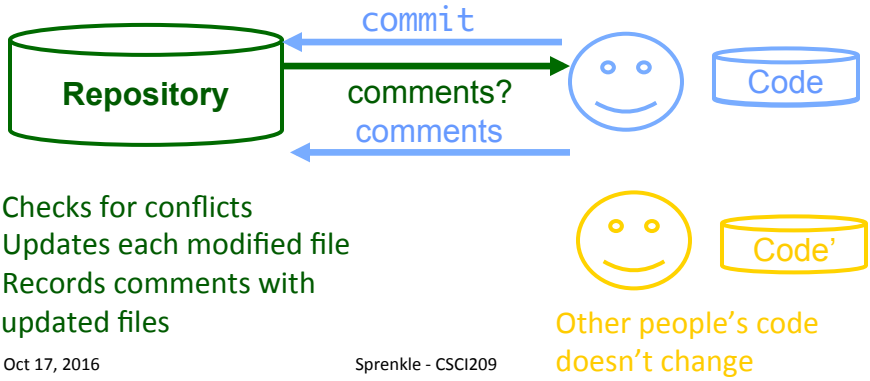
Using Version Control: `commit`

- After you make changes that you *want others to see*, **`commit`** your version
  - Include comments about what changes you made and why

- Checks for conflicts
- Updates each modified file
- Records comments with updated files

Using Version Control: `commit`

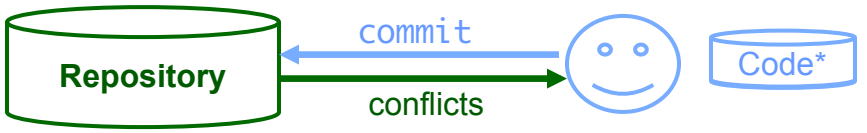
- After you make changes that you *want others to see*, `commit` your version
  - Include comments about what changes you made and why



- Checks for conflicts
- Updates each modified file
- Records comments with updated files

Using Version Control: `commit`

- After you make changes that you *want others to see*, `commit` your version



- Checks for conflicts: code conflicts with recent changes in the public copy

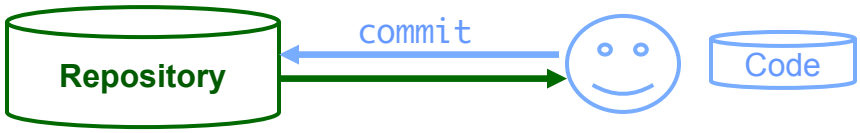
- Update code, fix conflicts
- Try commit again

Using Version Control: `update`

- To see the *current* version of the code, **`update`** your repository
  - Resolve conflicts

Using Version Control: `add, delete`

- You need to **`add`** and **`delete`** files and directories to the repository, then **`commit`**



- Create new records for added files
- Close records for deleted files
  - Files not deleted from repository

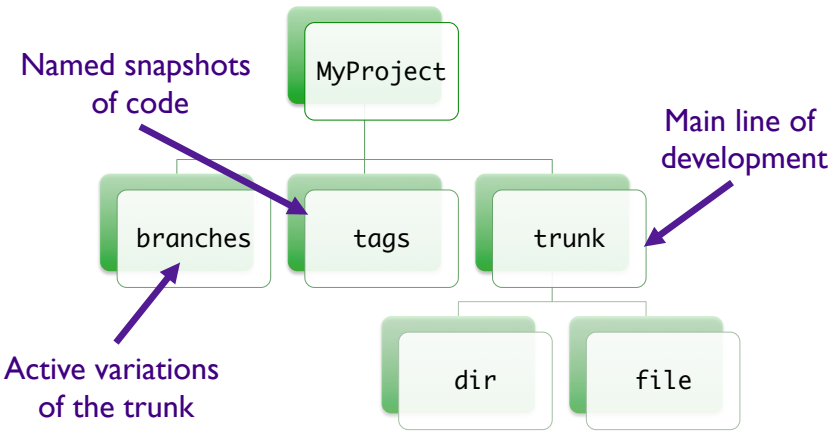
- Add, delete files and directories
- Commit

Version Control Advice

- Does not eliminate need for communication
  - Process becomes much more difficult if developers do not communicate
- Before picking up again, **update** your working copy
- **Commit** only after you've tested code and you're fairly sure it works
  - Write descriptive comments so your team members know what you did and why

Code Organization

- Organize code into appropriate structure



SUBCLIPSE

Subclipse

- Plugin for Eclipse
- **See SVN Practice assignment**
- Installation instructions online
  - Includes Configuration instructions too

Make sure you configure Subclipse after installing

Checking Out Code in Eclipse

- Import code
  - Import → SVN → Checkout Projects from SVN
  - Repository (works from home computer too):

```
svn+ssh://username@atmos.cs.wlu.edu/
  svn/csrepos/cs209_fall2016/
```

- Checkout `Example/trunk/` from repository
  - As a new Java project (Wizard)
  - Java project, named `SVNPractice`

SVN Rules

- Communicate with your teammates
- Describe your committed changes in comments
  - Let's your teammates know what you're doing—what changes to look for
- Don't commit the `bin` directory/class files
- Don't commit files that are specific to your account
  - .classpath, .project, .settings

SOFTWARE TESTING ISSUES

Software Testing Issues

- How do we know if the calculator program is correct?
  - How do we know that we've exposed all the faults?
  - How confident are we in its correctness?
- How do we know if we've tested enough?
  - Our customers want this product soon but we need product to be correct
    - Harder to fix after it has been released