What Can We Learn From Software Engineers?

Part 3 – Version Control

“Everything changes and nothing stands still.” - Heraclitus (535 BCE to 475 BCE)

Colloquially, “Change is the only constant.”
Does this sound familiar?

- Scattered files
- Many, many versions
- Unidentifiable releases
- Unknown baseline
- Cannot fallback
- Terrified of deleting anything
- Clumsy backup
Version control could be right for you.
The repository keeps a complete history of files and revisions.
The repository keeps a complete history of files and revisions.
Version control has its own lingo.

- Working files – *your files*
- Repository – *where everything is stored*
- Remote repository – *secondary storage*
- Add – *incorporate new file*
- Checkout – *retrieve a file*
- Commit – *save your changes*
- Trunk – *main line of development*
- Tip – *latest changes*
- Branch – *split development off trunk*
- Tag – *create label*
We visualize the version history like a tree.

- trunk
- branch
- merge
- tip
A version control system solves many problems.

- System of record
- Version history
- Facilitate collaboration
- Fallback
- Local backup
- Off-site backup
Using a version control system adds new steps to the dev. process.

- Design
- Code
- Debug and test
- **Add new files** (if any)
- **Commit changes**
- Repeat
What's in the repository?

- File-level granularity
- Source code, documentation, configuration
- Complete history
- Comments on every commit
- Branches for parallel work
- “Deleted” files
What file-level operations are possible on a repository?

- add – *add new file*
- commit – *save changes*
- checkout – *get current version*
- diff – *what changed?*
- merge – *combine two sets of changes*
- list – *what's in there?*
- log – *history of changes*
- remove – *nominally delete a file*
What project-level operations are possible on a repository?

- status – *what's changed?*
- tag – *define a release*
- checkout by tag – *fetch release*
- branch – *split from trunk*
- merge – *merge branch with trunk*
- revert – *undo botched commit*
Steps in creating a new repository

- Identify artifacts
- Get organized; declutter, declutter, declutter
- Create empty repository
- Add files (artifacts)
- Commit current versions
- Enable sharing
Your turn: Create a repository

- Create a local repository
- Add your package (library) code
- Commit changes
- View repository contents
Your turn: Update your package

• Edit the package code
• View “diff” of changes
• Commit changes
• View history of changes
Some version control systems use a centralized repository.
Some systems use a distributed repository.
Reprise: What project-level operations are possible on a repository?

- status – what's changed?
- tag – define a release
- checkout by tag – fetch release
- branch – split from trunk
- merge – merge branch with trunk
- revert – undo botched commit
- push – share with community
- pull – get community's changes
BTW, there are many synonyms for “version control system”.

- Code management system
- Source code management
- Revision control system
- Change management system
Version Control
Best Practices #1

D.R.Y. applies to change manage systems, too:
Let the VCS record the creator, changes, versions, etc.

Don't keep this stuff in your files:

# Name: models.R
# Author: Fred
# Created: 2010-4-1
# Modified: 5/3/2013
Version Control
Best Practices #2

Track your source files, not your outputs and not the computer generated artifacts.

Yes: Source code, config files
No: Reports, generated PDFs, intermediate data
Version Control
Best Practices #3

Keep everything you need in the repository.

Examples: Source code, configuration files, documentation

Ask yourself, if we lost every file tonight, could we recover fully tomorrow?
Version Control
Best Practices #4

Commit changes frequently, like a checkpoint.
But share only clean, working code.

Example: Commit accumulated changes daily, but share them only when clean and tested.
Version Control
Best Practices #5

Keep your latest, greatest stuff on the main trunk. Put releases on a either own branch.
Version Control
Best Practices #6

Keep working files and the repository on separate storage devices.

Example: Working files on PC, repository on file server.

Example: Working files locally, repository on GitHub.