Principles of Software Construction: Objects, Design, and Concurrency

Software engineering in practice

Configuration management and version control systems

Josh Bloch    Charlie Garrod    Darya Melicher
Administrivia

• Homework 6 due next Wednesday
  – Checkpoint deadline Monday night
Key concepts from Thanksgiving
Key concepts from last Tuesday
Streams design discussion

• Recall the fundamental API design principles...
Major topics in 17-313 (Foundations of SE)

• Process considerations for software development
• Requirements elicitation, documentation, and evaluation
• Design for quality attributes
• Strategies for quality assurance
• Empirical methods in software engineering
• Time and team management
• Economics of software development
Test-driven development (TDD), informally

From Growing Object-Oriented Software by Nat Pryce and Steve Freeman
http://www.growing-object-oriented-software.com/figures.html

@sebrose http://cucumber.io
Empirical methods in software engineering

• How do we study the effectiveness of test-driven development compared to other methodologies?
This week: DevOps (Development operations)

- Introduction to devops
- Configuration management and version control
Consider: timelines of traditional software development

e.g., the Microsoft* OS development history

Source: By Paulire - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=46634740
Compare to the Ubuntu release cycle
Compare to the Facebook release cycle
DevOps: Development / Operations
DevOps toolchain

- **Code**: code development and review, source code management tools
- **Build**: continuous integration tools, build status
- **Test**: continuous testing tools provide feedback
- **Package**: artifact repository, application pre-deployment staging
DevOps ecosystems...
Consider: Continuous integration (CI)

- Advantages and disadvantages of CI?
Real-world software development challenges

• Imagine: You discover a bug in version 8.2.4 of your software
  – You want to discover, fix, and deploy updates to old versions
  – You want to fix the bug for new versions in ongoing development
Configuration management (CM)

- Definition (Pressman): Configuration management “is a set of tracking and control activities that are initiated when a software engineering projects begins and terminates when software is taken out of operation.”
Reasons for configuration management

- Software evolution
- Separate development
- Audits (legal, regulatory)
- Product lines
- Market variation (e.g., U.S., Europe, Asia)
- Platform variation (e.g., Android, iOS)
Configuration management in the modern world

Version control + workflows ➔ Build managers ➔ Package managers ➔ Deployment managers + VMs/containers ➔ App markets + update managers
Aside: Semantic versioning for releases

• Given a version number MAJOR.MINOR.PATCH, increment the:
  – MAJOR version when you make incompatible API changes,
  – MINOR version when you add functionality in a backwards-compatible manner, and
  – PATCH version when you make backwards-compatible bug fixes.

• Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

http://semver.org/
Branches within software repositories
Centralized version control

- Single server contains all the versioned files
- Clients check out/in files from that central place
- E.g., CVS, SVN (Subversion), and Perforce

Distributed version control

- Clients fully mirror the repository
  - Every clone is a full backup of all the data
- E.g., Git, Mercurial, Bazaar

SVN (left) vs. Git (right)

- SVN stores changes to a base version of each file
- Version numbers (1, 2, 3, ...) are increased by one after each commit

- Git stores each version as a snapshot
- If files have not changed, only a link to the previous file is stored
- Each version is referred by the SHA-1 hash of the contents

Aside: Git file status

© Scott Chacon “Pro Git”
Aside: Git internals

© Scott Chacon “Pro Git”
Aside: Git object graph

© Scott Chacon “Pro Git”
Aside: Which files to manage?

• All code and noncode files
  – Java code
  – Build scripts
  – Documentation

• Exclude: generated files (.class, …)
  – Most version control systems have a mechanism to exclude files (e.g., .gitignore)
Next time...

- Practical Git