

TCSS 360: SOFTWARE DEVELOPMENT AND QUALITY ASSURANCE

Scrum Software Process, Version Control, Git/Github, Postman

Wes J. Lloyd
 Institute of Technology
 University of Washington - Tacoma

OBJECTIVES

- From chapter 10: Engineering SaaS
 - Scrum
 - Version control
 - Git / Github
 - Postman

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.2

SOFTWARE PROJECT PHASES

1. Enthusiasm
 - Great! Can't wait to start
2. Disillusionment
 - Oh, there's a lot of requirements...
3. Panic
 - The customer wants everything! and the scope just keeps increasing ...
4. Search for the guilty
 - Who agreed to this anyway?
5. Punishment of the Innocent
 - No one knew...
6. Praise for non-participants

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.3

SCRUM

- Agile like process
- Feature daily short meetings ~ 15 minutes
 - Often called "STAND UPS"
 - By standing we encourage meetings to be short
- Basic agenda
 - What have you done since yesterday?
 - What are you planning to do today?
 - Are there any impediments or stumbling blocks?


January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.4

SCRUM - 2

- Scrum uses "Sprints" instead of agile iterations
- Scrum Team – two pizza team that delivers software
 - Teams of 4 to 9 people
 - Two pizzas can feed the team in a meeting
- Scrum Master
 - Acts as a buffer between the team and external distractions
 - Keeps team focused
 - Enforces rules, coding standards, testing, walkthroughs
- Product Owner
 - Represents customer (not the Scrum Master)
 - Serves as the customer's voice, prioritizes user stories

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.5

PAIR PROGRAMMING



- **GOAL:** Improve software quality
- **Automobile analogy:** two developers work jointly at the same workstation on the same code
- **Driver Role**
 - Enters code, thinks tactically about how to complete the current task, explains thoughts out loud while typing
- **Observer/navigator**
 - Reviews each line of code as it is entered
 - Acts as a safety net for the driver
 - Thinks strategically about future problems
 - Makes suggestions to the driver
- Pairs alternate between driving and observing

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.6

PAIR PROGRAMMING - 2



Pivotal Tracker Developers

- Sit side-by-side facing screens together
 - Not personal computer; pair programming workstation
 - To avoid distractions, no email reader, browser

PAIR PROGRAMMING - 3

- Can produce more readable code
- Requires more effort than solo programming
- Supports knowledge transfer between pair:
 - programming idioms
 - tool tricks
 - company processes
 - latest technologies


January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.8

PAIR PROGRAMMING - 4

- Software quality improvements: measured by study (Cockburn and Williams 2001)
- Faster development with PP
 - Code development in 20% to 40% less time
- Fewer failures with PP
 - Only 15% of tests failed
 - 30% with solo programming
- But, higher development costs
 - On average 15% more "programmer hours"
- Requires more focused attention
 - Two programmers "close off" interrupts for hours
 - Can be exhausting, less net surfing, smartphone use

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.9


VERSION CONTROL SYSTEMS



January 25, 2017 L6.10

VERSION CONTROL SYSTEM

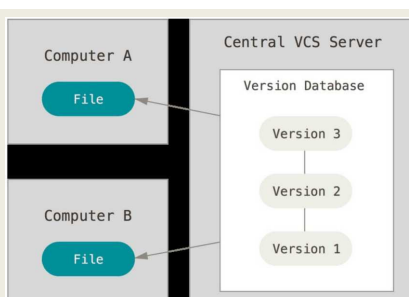
- Records changes to files or filesets over time
 - Allows reversion to previous state
 - Compare changes
 - Creates a version history
 - Supports team collaboration on code
 - Supports versioning
 - Releases are tagged
 - Supports branching
 - Developers work on branches
 - Merge code into trunk



January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.11

CENTRALIZED VERSION CONTROL

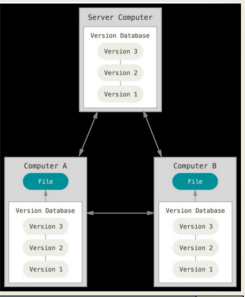
- SVN (subversion)
- CVS
- PVCS



January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.12

DISTRIBUTED VERSION CONTROL

- Distributed copies of the source code repository
- Peer-to-peer approach to VC
- Git, mercurial (hg)
- Users work on local filesystem
- Commit code to local repository
- Push code to shared repository
- Shared repositories: Github.com, privately hosted git servers



January 25, 2017
TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma
L6.13

CENTRALIZED VS. DISTRIBUTED VERSION CONTROL TRADEOFFS

	Centralized VC	Distributed VC
Branching and Merging	Not easy. Users manually track which revisions have been merged between branches	Creating, managing and removing branches is simple
Synchronization	No way to share changes between users except through central server	Users can exchange code and collaborate directly: peer to peer
Offline commits	Not supported. Requires network connection to central server	Supported to local repositories
Backup	If central server fails, all data is lost. Reliant on server backup	Source code copies are everywhere. User copies are "remote" backups. Provides security against data loss
Performance	Slow operations to central server. Becomes overloaded with large teams	Operations are fast because central server communication is minimized
Complexity	Very easy to setup and use, conceptually simple	More complex and requires time to learn generally

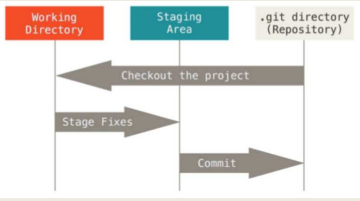
GIT DESIGN

- Snapshots of source files are kept
- Stored as checksums (incremental)
- Operations are local (offline)
- Push syncs local repository with remote repository

January 25, 2017
TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma
L6.15

FILE STATES IN GIT

- Committed → Stored safely in local repository
- Modified → Local copy changed but not committed
- Staged → Marked as modified file in this current version to go into the next commit snapshot



January 25, 2017
TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma
L6.16

KEY TERMS

- **Origin**
Authoritative repository. Acts as the master. Changes are all eventually "pushed" there. Git doesn't care which repository is authoritative. Can be on a central server, users laptop, etc.
- **Forking**
Used to make a copy of another project in your own account. Creates your own copy for contributing.
- **Cloning**
Download a repository to your local machine

January 25, 2017
TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma
L6.17

KEY TERMS - 2

- **Pull request**
When committing changes, you send a PULL request to the master contributor so they know that there are new changes to incorporate
- **Merge request**
The original owner merges the request and it becomes part of the master.

January 25, 2017
TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma
L6.18

MERGE CONFLICTS

- Amy and Bob each have a current copy of the repository
- Amy makes and commits a set of changes to file A
- Amy makes and commits a separate set of changes to file B
- Amy pushes her commits to the origin repository
- Bob makes and commits his own changes to file A, but doesn't touch file B.
- Bob tries to push his commits, but is prevented from doing so since file A has changed since Bob's last pull. Bob must bring his copy of file A up-to-date with respect to the origin before he can push his changes to the origin
- Bob merges Amy's file A changes, and can then commit/push

January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

L6.19

MERGE CONFLICTS - 2

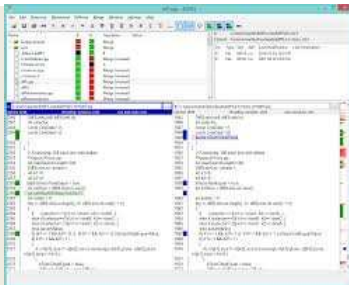
- Automatic merge
 - When differences are minimal and there are no conflicting changes the system will automatically perform the merge
 - Differences by line, word
- Manual merge
 - When changes conflict with each other, they must be manually inspected and dealt with
 - Sometimes can be confusing
 - Requires communication with team members who changed to code
- Visual Diff tools
 - Show differences between file versions to help observe differences
 - Sometimes integrated directly into tools, IDEs

January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

L6.20

GRAPHICAL DIFF



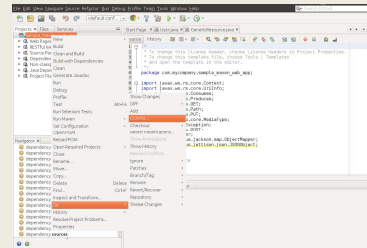
January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

L6.21

IDE INTEGRATION

- Git support already integrated into Netbeans & other IDEs



January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

L6.22

CODE REVIEWS

- Pair programming eliminates the need
- What if not programming in pairs?
- Plan and document processes: formal code reviews
- Agile source management: PULL requests
 - Developers perform SHORT reviews on new code when merging pull requests
 - If concerns arise, short discussions ensue
 - Reviews becomes **mandatory** when merge conflicts arise

January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

L6.23

POSTMAN

- Chrome plug-in
- Browser based generic webservicess client
- Alternative to command-line curl
- Create/invoke webservicess tests
- Provide files, raw text as an input, etc.
- Also, other similar browser plugins exist

January 25, 2017

TCSS360: Software Development and Quality Assurance [Winter 2017]
Institute of Technology, University of Washington - Tacoma

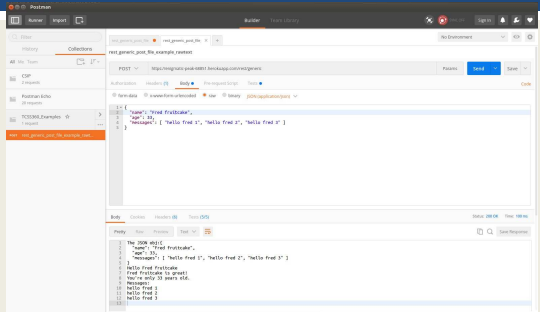
L6.24

POSTMAN - DEMO

- Installed into chrome as an extension
- Extension menu
 - Visit: <chrome://apps/>
- Postman runner: batch tests, batch POSTs

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.25

POSTMAN DEMO




January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.26

TUTORIAL #2

- http://faculty.washington.edu/wlloyd/courses/tcss360/tutorials/TCSS360_w2017_Tutorial_2.pdf

January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.27

QUESTIONS



January 25, 2017 TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma L6.28