Working in Teams

Today

- Talk about Teamwork
- 4:00-5:00 Guest lecture in CS 150
  - DLS talk on testing software
  - Free snacks right after class!
  - Part of class material (will be on test)
  - Recorded, in case you cannot attend.

Lecture outline

- Why is teamwork hard?
- Not getting into each other's way
- Positive teamwork

Team pros and cons

- Benefits
  - Attack bigger problems in a short period of time
  - Utilize the collective experience of everyone
- Risks
  - Communication and coordination issues
  - Groupthink: diffusion of responsibility; going along
  - Working by inertia; not planning ahead
  - Conflict or mistrust between team members

Communication: powerful but costly!

- Communication requirements increase with increasing numbers of people
- Everybody to everybody: quadratic cost
- Every attempt to communicate is a chance to miscommunicate
- But *not* communicating will *guarantee* miscommunication

What about conflicts?

What can cause conflicts?

- Two people want to work on the same file
  - Google docs lets you do that

But…

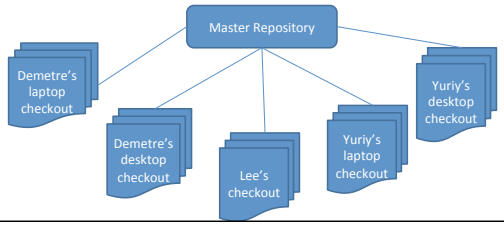
- What about same line?
- What about timing?
- What about design decisions?

Version control

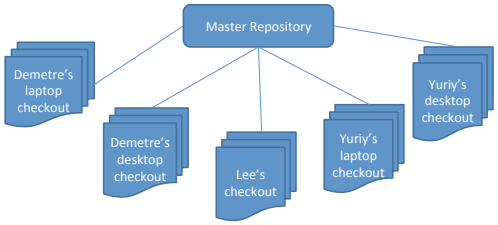
Version control aims to allow multiple people to work in parallel.

Centralized version control

- (old model)
- Examples: Concurrent Versions System (CVS)
  Subversion (SVN)



Doing work



- I update my checkout (working copy)
- I edit
- I update my checkout again
- I merge changes if necessary
- I commit my changes to the Master

Problems with centralized VC

- What if I don't have a network connection?
- What if I am implementing a big change?
- What if I want to explore project history later?

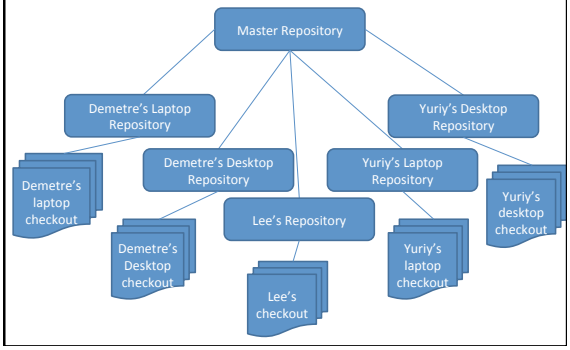
Distributed version control

(new model)

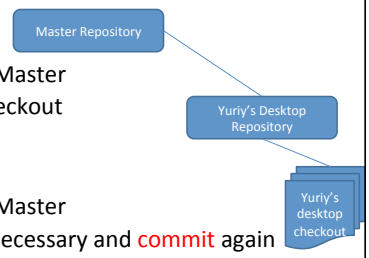
- Examples: Mercurial (Hg), Git, Bazaar, Darcs, …

- Local operations are fast (and possible)
- History is more accurate
- Merging algorithms are far better

Distributed version control model



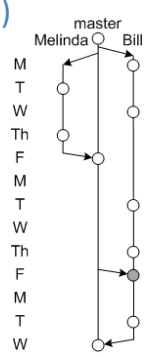
Doing work



- I pull from the Master
- I update my checkout
- I edit
- I commit
- I pull from the Master
- I merge tips if necessary and commit again
- I push my changes to the Master

History view (log)

- Bill and Melinda work at the same time
- At the end, all repositories have the same, rich history



What do conflicts look like?

Crystal tool

What VC does the cloud provide?

- code.google.com has SVN and Hg
- bitbucket.org has Hg and git
- github.com has git
- sourceforge.net has SVN, CVS, git, Hg, Bazaar

- You can run whatever you want on EDLab

Lecture outline

- Why is teamwork hard?
- Not getting into each other's way

➔Positive teamwork

Team structures

- Tricky balance among
  - progress on the project/product
  - expertise and knowledge
  - communication needs

"A team is a set of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable."

– Katzenbach and Smith

Common SW team responsibilities

- Project management
- Functional management
- Developers: programmers, testers, integrators
- Lead developer/architect ("tech lead")

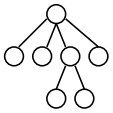
- These could be all different team members, or some members could span multiple roles.
- Key: Identify and stress roles **and** responsibilities

Issues affecting team success

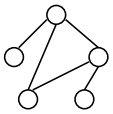
- Presence of a shared mission and goals
- Motivation and commitment of team members
- Experience level
  - and presence of experienced members
- Team size
  - and the need for bounded yet sufficient communication
- Team organization
  - and results-driven structure
- Reward structure within the team
  - incentives, enjoyment, empowerment (ownership, autonomy)

Team structure models

- Dominion model
  - Pros
    - clear chain of responsibility
    - people are used to it
  - Cons:
    - single point of failure at the commander
    - less or no sense of ownership by everyone



- Communion model
  - Pros
    - a community of leaders, each in his/her own domain
    - inherent sense of ownership
  - Cons
    - people aren't used to it (and this scares them)



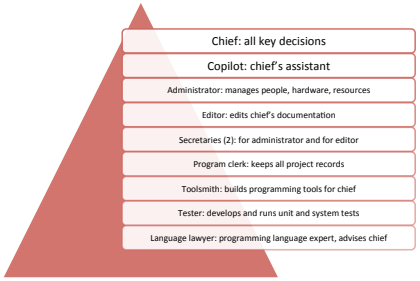
Team leadership

- Who makes the important product-wide decisions in your team?
  - One person?
  - All, by unanimous consent?
  - Other options?...

  - Is this an unspoken or an explicit agreement among team members?

Surgical/Chief Programmer Team

[Baker, Mills, Brooks]



Microsoft's team structure

[microsoft.com]

- **Program Manager**. Leads the technical side of a product development team, managing and defining the functional specifications and defining how the product will work.
- **Software Design Engineer.** Codes and designs new software, often collaborating as a member of a software development team to create and build products.
- **Software Test Engineer.** Tests and critiques software to assure quality and identify potential improvement opportunities and projects.

Toshiba Software Factory [Y. Matsumoto]

- Late 1970's structure for 2,300 software developers producing real-time industrial application software systems (such as traffic control, factory automation, etc.)
- Unit Workload Order Sheets (UWOS) precisely define a software component to be built
- Assigned by project management to developers based on scope/size/skills needed
- Completed UWOS fed back into management system
- Highly measured to allow for process improvement

Common factors in good teams

- Clear roles and responsibilities
  - Each person knows and is accountable for their work
- Monitor individual performance
  - Who is doing what, are we getting the work done?
- Effective communication system
  - Available, credible, tracking of issues, decisions
  - Problems aren't allowed to fester ("boiled frogs")
- Fact based decisions
  - Focus on the facts, not the politics, personalities, …

Motivation

What motivates you?

- Achievement
- Recognition
- Advancement
- Salary
- Possibility for growth
- Interpersonal relationships
  - Subordinate
  - Superior
  - Peer
- Status
- Technical supervision opportunities
- Company policies
- Work itself
- Work conditions
- Personal life
- Job security
- Responsibility
- Competition
- Time pressure
- Tangible goals
- Social responsibility
- Other?

De-motivators

- What takes away your motivation?
  - Micro-management or no management
  - Lack of ownership
  - Lack of effective reward structure
    - Including lack of simple appreciation for job well done
  - Excessive pressure and resulting "burnout"
  - Allowing "broken windows" to persist
  - Lack of focus in the overall direction
  - Productivity barriers
    - Asking too much; not allowing sufficient learning time; using the wrong tools
  - Too little challenge
  - Work not aligned with personal interests and goals
  - Poor communication inside the team

Distinguished Lecturer Series

Automated Support for Reproducing and Debugging Field Failures

Alessandro Orso
Georgia Institute of Technology

Two extremely challenging software maintenance tasks are reproducing and debugging field failures, failures that occur on user machines after release. We have developed a family of techniques that help developers with these tasks. I will overview our work in this area and present two of these techniques: BugRedux and F3. BugRedux reproduces field failures by collecting dynamic data about failing executions and synthesizing executions that mimic the observed field failures. F3 uses those executions to automatically localize faults. I will present the results of an empirical evaluation on a real-world programs and field failures. The results of our evaluation are promising. For all the failures considered, our approach was able to (1) synthesize failing executions that mimicked the observed field failures and (2) use the synthesized executions localize the fault. I will conclude discussing our latest results, open challenges, and future research directions.

BIO: Alessandro Orso is a Professor in the College of Computing at the Georgia Institute of Technology. His area of research is software engineering, with emphasis on software testing and program analysis. He is a senior member of the ACM and of the IEEE Computer Society.

Wednesday, January 28, 2015 @ **4:00pm**
in CS 151

UMASSCS
SCHOOL OF COMPUTER SCIENCE

4:00-5:00 (snacks now)
distinguished lecture
CS 150
on testing software

https://www.cs.umass.edu/speakers/alessandro-alex-orso/2015/jan/28