Project updates

- Everything on schedule?
- Requirements specifications are due Thursday, Feb 14, at noon
- Submit on Moodle, 1 per group

- AWS funding has come through! Each team can count on $800 in free AWS credit.

Working in Teams

TEAMWORK
Large ambitious goals usually require that people work together.

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

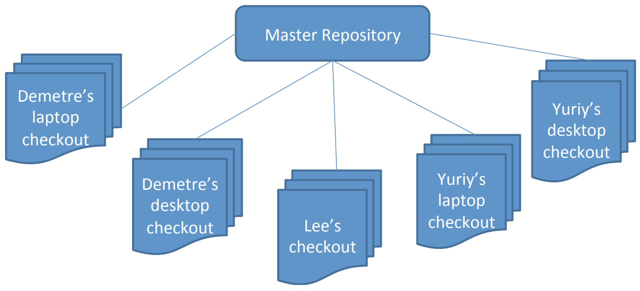
But…
- What about same line?
- What about timing?
- What about design decisions?

Version control

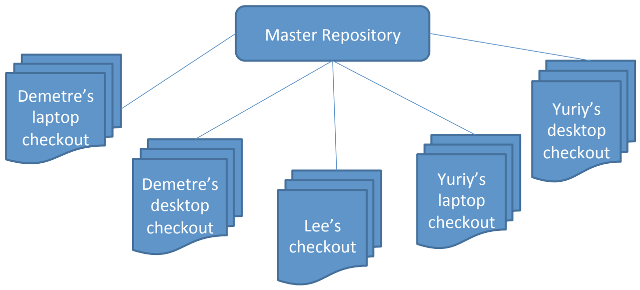
Version control aims to allow multiple people to work in parallel.

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?

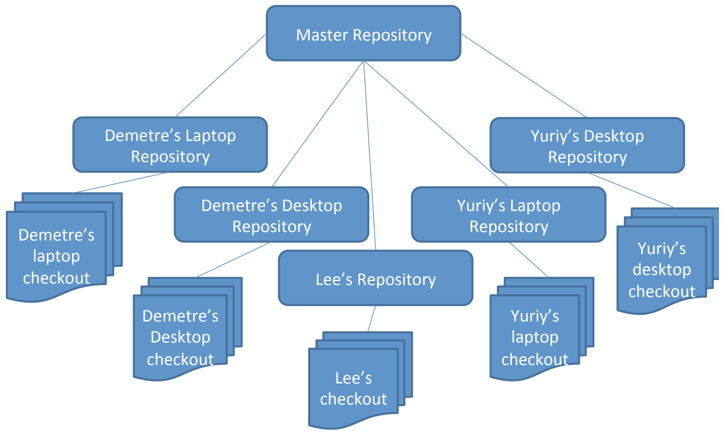
Distributed version control

(new model)

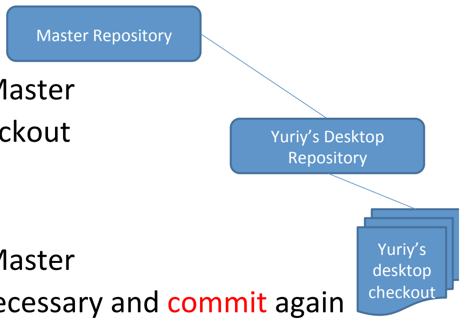
- Examples: Mercurial (Hg), Git, Bazaar, Darcs, …

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

Distributed version control model


Master Repository
Demetre's Laptop Repository
Yuriy's Desktop Repository
Demetre's Desktop Repository
Yuriy's Laptop Repository
Lee's Repository
Demetre's laptop checkout
Demetre's Desktop checkout
Lee's checkout
Yuriy's laptop checkout
Yuriy's desktop checkout

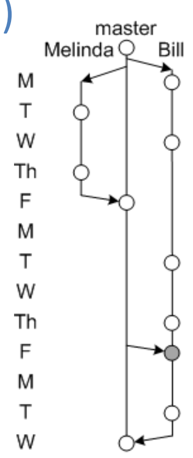
Doing work



- 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)

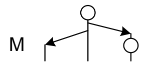
- Bill and Melinda work at the same time
- At the end, all repositories have the same, rich history



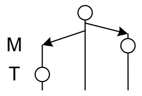
What do conflicts look like?

Crystal tool

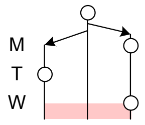
The Gates conflict



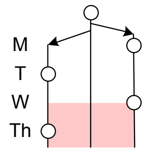
The Gates conflict



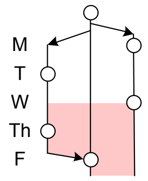
The Gates conflict



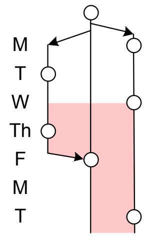
The Gates conflict



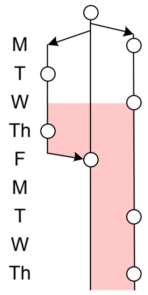
The Gates conflict



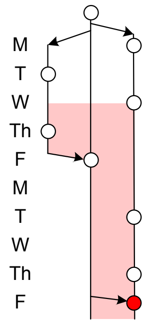
The Gates conflict



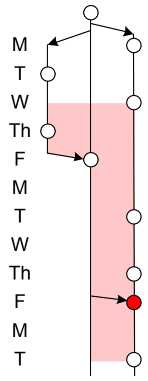
The Gates conflict



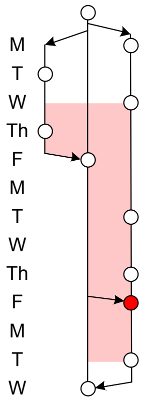
The Gates conflict



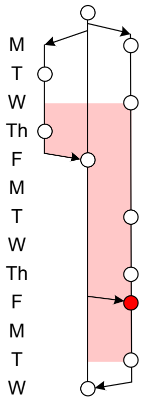
The Gates conflict



The Gates conflict

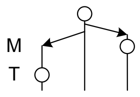


The Gates conflict



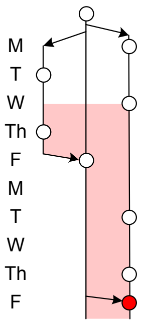
The information was all there, but the developers didn't know it.

What could well-informed developers do?



- avoid conflicts

What could well-informed developers do?



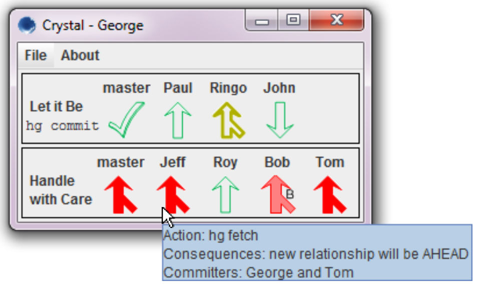
- avoid conflicts
- become aware of conflicts earlier

Introducing Crystal: a proactive conflict detector

DEMO

Introducing Crystal: a proactive conflict detector

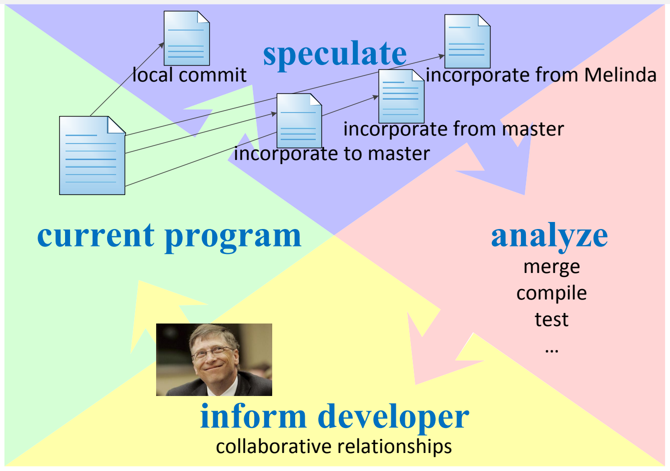
DEMO



Action: hg fetch
Consequences: new relationship will be AHEAD
Committers: George and Tom

http://crystalvc.googlecode.com

Speculative analysis in collaborative development



Reducing false positives in conflict prediction

Collaborative awareness

- Palantír [Sarma et al. 2003]
- FASTDash [Biehl et al. 2007]
- Syde [Hattori and Lanza 2010]
- CollabVS [Dewan and Hegde 2007]
- Safe-commit [Wloka et al. 2009]
- SourceTree [Streeting 2010]

Reducing false positives in conflict prediction

Collaborative awareness

- Palantír [Sarma et al. 2003]
- FASTDash [Biehl et al. 2007]
- Syde [Hattori and Lanza 2010]
- CollabVS [Dewan and Hegde 2007]
- Safe-commit [Wloka et al. 2009]
- SourceTree [Streeting 2010]

Crystal analyzes **concrete artifacts**,
eliminating false positives and false negatives.

Utility of conflict detection

- Are textual collaborative conflicts a real problem?
- Can textual conflicts be prevented?
- Do build and test collaborative conflicts exist?

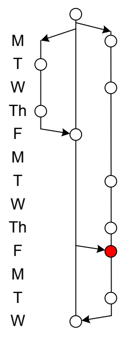
Are textual collaborative conflicts a real problem?

histories of 9 open-source projects:

size:	26K–1.4MSLoC
developers:	298
versions:	140,000

Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

Are textual collaborative conflicts a real problem?

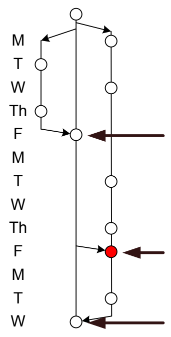


histories of 9 open-source projects:

size:	26K–1.4MSLoC
developers:	298
versions:	140,000

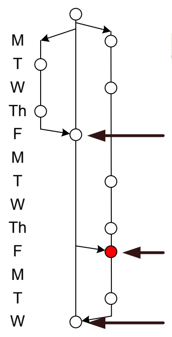
Perl5, Rails, Git, jQuery, Voldemort, MaNGOS, Gallery3, Samba, Insoshi

Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

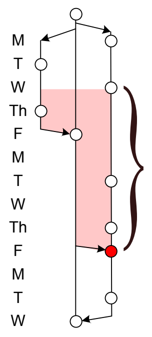
Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

16% of the merges have textual conflicts.

Are textual collaborative conflicts a real problem?

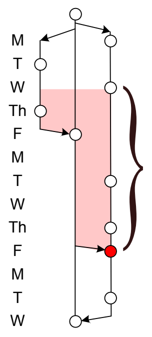


How frequent are textual conflicts?

16% of the merges have textual conflicts.

How long do textual conflicts persist?

Are textual collaborative conflicts a real problem?



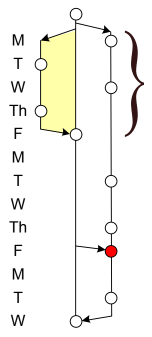
How frequent are textual conflicts?

16% of the merges have textual conflicts.

How long do textual conflicts persist?

Conflicts live a mean of 9.8 and median of 1.6 days. The worst case was over a year.

Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

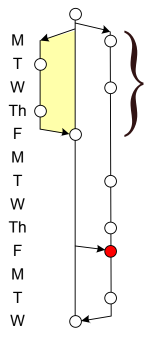
16% of the merges have textual conflicts.

How long do textual conflicts persist?

Conflicts live a mean of 9.8 and median of 1.6 days. The worst case was over a year.

How long do textually-safe merges persist?

Are textual collaborative conflicts a real problem?



How frequent are textual conflicts?

16% of the merges have textual conflicts.

How long do textual conflicts persist?

Conflicts live a mean of 9.8 and median of 1.6 days. The worst case was over a year.

How long do textually-safe merges persist?

Textually-safe merges live a mean of 11.0 and median of 1.9 days.

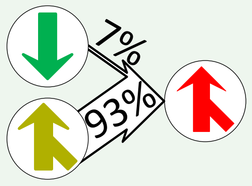
Can textual conflicts be prevented?

Where do textual conflicts come from?

Can textual conflicts be prevented?

Where do textual conflicts come from?

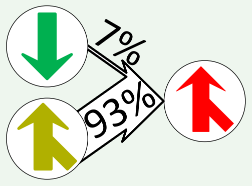
93% of textual conflicts developed from safe merges.



Can textual conflicts be prevented?

Where do textual conflicts come from?

93% of textual conflicts developed from safe merges.



The information Crystal computes can help prevent conflicts.

Do build and test collaborative conflicts exist?

program	conflicts			safe
	textual	build	test	merges
Git	17%	<1%	4%	79%
Perl5	8%	4%	28%	61%
Voldemort	17%	10%	3%	69%

Does merged code fail to build or fail tests?

One in three conflicts are build or test conflicts.

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")

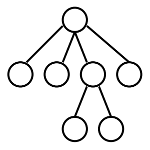
- These could be all different team members, or some members could span multiple roles.
- Key: Identify and stress roles **and** responsibilities

Issues affecting team success

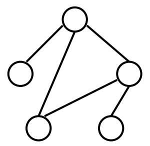
- Presence of a shared mission and goals
- Motivation and commitment of team members
- Experience level
  - and presence of experienced members
- Team size
  - and the need for bounded yet sufficient communication
- Team organization
  - and results-driven structure
- Reward structure within the team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

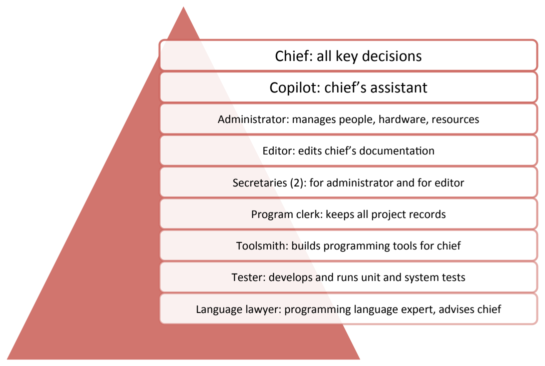


Team leadership

- 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]



Chief: all key decisions
Copilot: chief's assistant
Administrator: manages people, hardware, resources
Editor: edits chief's documentation
Secretaries (2): for administrator and for editor
Program clerk: keeps all project records
Toolsmith: builds programming tools for chief
Tester: develops and runs unit and system tests
Language lawyer: programming language expert, advises chief

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