Proactive Detection of Collaboration Conflicts

Yuriy Brun , Reid Holmes , Michael D. Ernst , David Notkin

Computer Science & Engineering
University of Washington

School of Computer Science
University of Waterloo

Presentation by:
Hemayet Ahmed Chowdhury

Introduction

- Collaborative development can be hampered when conflicts arise because developers have inconsistent copies of a shared project

- While Version Control Systems permit rapid development progress, loose synchronization can result in textual conflicts, build conflicts and test case failures.

- This paper presents a tool, Crystal, that can help developers identify and resolve conflicts early by reporting conflict errors before commits are pushed into the master repository.

Paper Structure

- The paper first starts with a study of open-source systems and establishes that conflicts are frequent, persistent, and appear not only as overlapping textual edits but also as subsequent build and test failures

- The paper then diagnoses important classes of conflicts using the novel technique of speculative analysis over version control operations.

- The paper finally describes the design of Crystal, a tool that uses PROACTIVE speculative analysis to make concrete advice available to developers, helping them identify, manage, and prevent conflicts.

Sample Scenario

- George and Ringo are adding features to a project.
- They both publish their changes to the master repo.
- No Textual conflicts occur but the program does not build / the regression tests fail
- George and Ringo go through a lot of trouble to recollect the changes and rework their code.

Potential Solutions

Awareness Tools Vs Speculative Analysis Tools

- An awareness tool can alert developers about where other developers are working on. Problematic because it can result in a lot of false positives in situations where no conflict takes place.

- Speculative Analysis Tools can however proactively inform developers of version control conflicts.

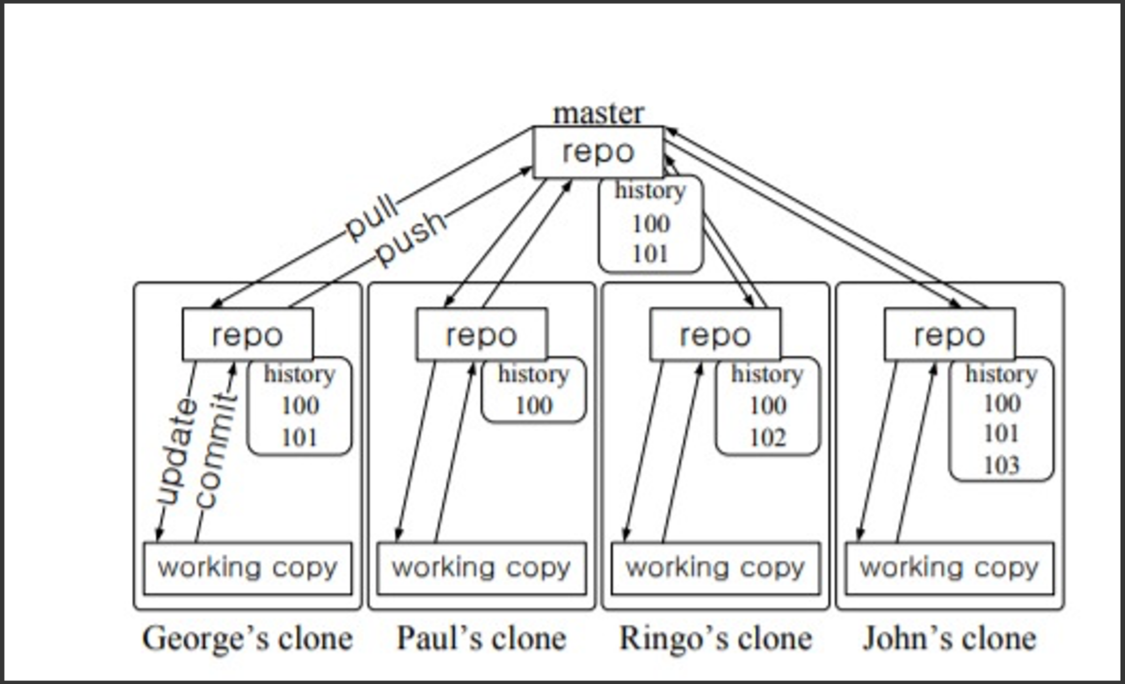
Speculative Analysis Tool

- Does not guess conflicts.
- Executes VCS operations (merging, building, running tests) on the background on the clones of the project.
- Reports the conflicts generated after execution of the merging, building and test cases.
- Crystal, presented by this paper, is a speculative analysis tool.

VCS Terminology

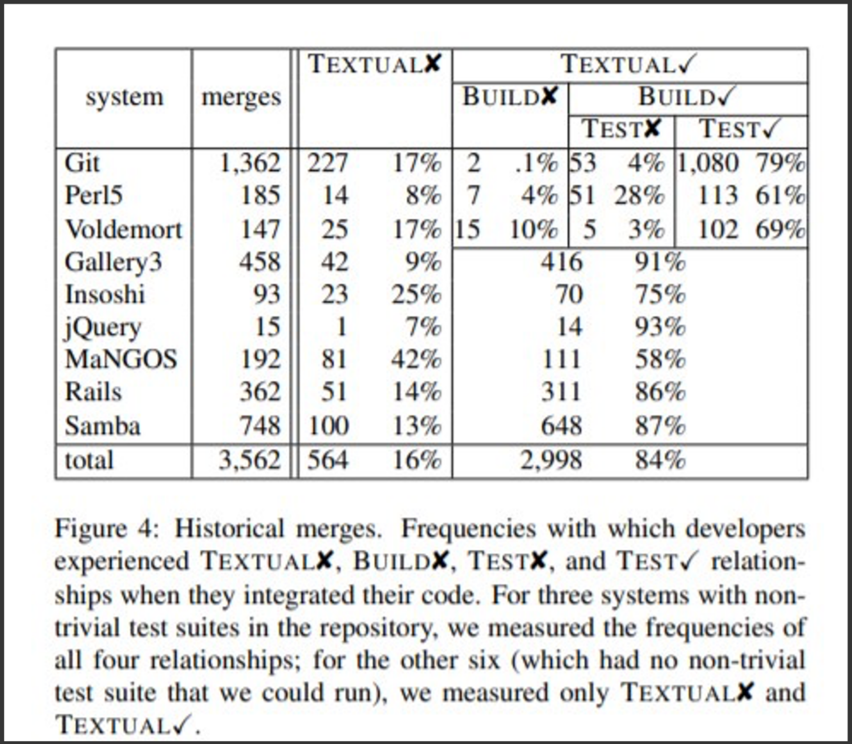
- SAME: 2 local repos of the developers have the same change sets
- AHEAD: One repository has a superset of the other repository’s changesets.
- BEHIND: The inverse of AHEAD
- Textual Conflict : Overlapping lines of code between 2 repos
- Build Conflict : Failure to build after merging of 2 repos
- Test Conflict : Failure to pass test cases after building
- Textual Pass, Build Pass, Test Pass : The repos can be merged, built and pass the test cases

VCS Overview



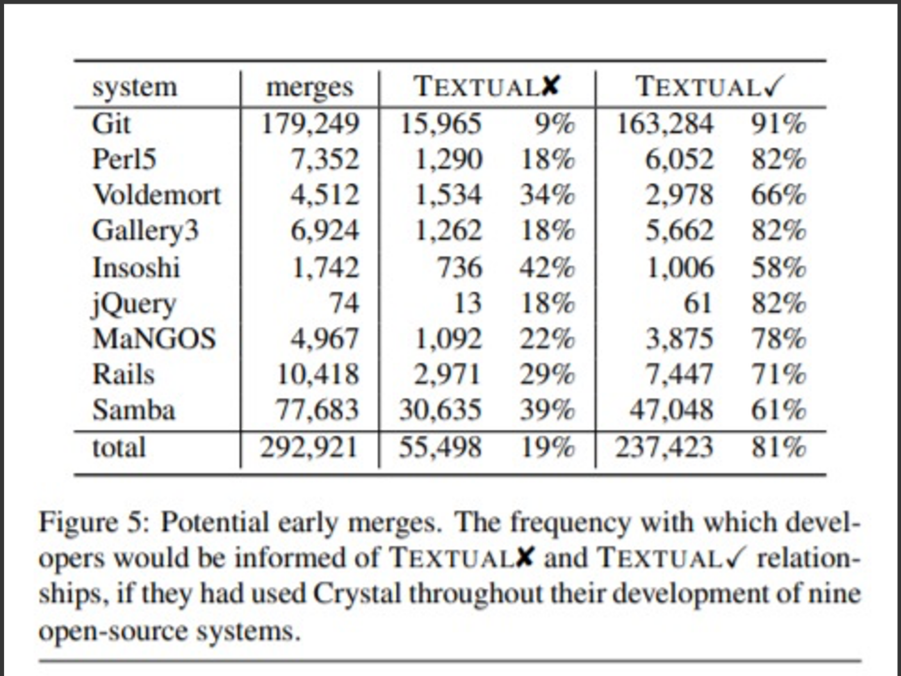
Crystal has access to and analyses each local clone of the developers.

RQ 1 : How frequently do conflicts arise across developers' copies of a project?



system	merges	TEXTUAL✗		TEXTUAL✓					
				BUILD✗		BUILD✓			
						TEST✗		TEST✓	
Git	1,362	227	17%	2	.1%	53	4%	1,080	79%
Perl5	185	14	8%	7	4%	51	28%	113	61%
Voldemort	147	25	17%	15	10%	5	3%	102	69%
Gallery3	458	42	9%	416	91%				
Insoshi	93	23	25%	70	75%				
jQuery	15	1	7%	14	93%				
MaNGOS	192	81	42%	111	58%				
Rails	362	51	14%	311	86%				
Samba	748	100	13%	648	87%				
total	3,562	564	16%	2,998	84%				

Figure 4: Historical merges. Frequencies with which developers experienced TEXTUAL✗, BUILD✗, TEST✗, and TEST✓ relationships when they integrated their code. For three systems with nontrivial test suites in the repository, we measured the frequencies of all four relationships; for the other six (which had no non-trivial test suite that we could run), we measured only TEXTUAL✗ and TEXTUAL✓.



system	merges	TEXTUAL✗		TEXTUAL✓	
Git	179,249	15,965	9%	163,284	91%
Perl5	7,352	1,290	18%	6,052	82%
Voldemort	4,512	1,534	34%	2,978	66%
Gallery3	6,924	1,262	18%	5,662	82%
Insoshi	1,742	736	42%	1,006	58%
jQuery	74	13	18%	61	82%
MaNGOS	4,967	1,092	22%	3,875	78%
Rails	10,418	2,971	29%	7,447	71%
Samba	77,683	30,635	39%	47,048	61%
total	292,921	55,498	19%	237,423	81%

Figure 5: Potential early merges. The frequency with which developers would be informed of TEXTUAL✗ and TEXTUAL✓ relationships, if they had used Crystal throughout their development of nine open-source systems.

RQ 2: How long do these conflicts persist

- On average, the TEXTUAL conflict relationships between repos persisted for 9.8 days and involved 23.2 changesets — 11.6 per developer.
- On average, the build conflicts/test failure relationships between repos persisted for 11 days and spanned 23 changesets.

RQ3: Do clean merges devolve into conflicting changes?

- Of all conflict relationships , 93% developed from a TEST pass relationship; the other 7% of conflict relationships developed from a BEHIND relationship
- 20% of TEST pass relationships later devolved into a conflict

RQ4: What information could developers use to reduce the frequency and duration of conflicts?

What information could help developers make better decisions such as whether

- to perform a particular operation such as a push/pull
- to wait for a coworker to perform one
- to communicate directly with a coworker

Guidance by Crystal

Crystal can provide information of 5 different categories to guide developers into better conflict resolving.

- Committer: Who made the relevant changes?
- When: Can an action that affects the conflict relationship be performed now, or must it wait until later?
- Consequences: Will an action — perhaps one on a different repository — affect a relationship?
- Capable: Who can perform an action that changes the relationship?
- Ease: Has anyone already made changes that ease resolving an existing conflict?

Committer : Who made the relevant changes?

Consider three developers George, Ringo and Paul

- George realises he is in a textual conflict with Ringo
- However, Ringo may not have made the conflicting changes
- instead, Paul may have made and pushed the changes to the master, and Ringo then pulled them from the master
- In this case, George should likely discuss the conflict with Paul rather than with Ringo
- Crystal can provide this guidance with the information it collected from the repos and commits of all three developers.

When : Can an action that affects the conflict relationship be performed now or later?

- Ringo is behind George in terms of commits in their local repos
- But Ringo can’t incorporate his changes because George hasn’t pushed his commit to master yet
- Even though Ringo has to pull the master repo to incorporate his changes, he will have to do it after George pushes his commit
- Crystal can help developers know when to perform such an action like a pull from the repo or when to wait

Consequences: Will an action — perhaps one on a different repository — affect a relationship?

- Ringo is behind George in terms of commits in their local repos
- Crystal can help him know if he is behind
  - Because George has pushed his changes but Ringo hasn't pulled from the master Repo yet. In which case, Ringo should perform a pull.
  - Or because George hasn't pushed his commits yet. In which case, even if Ringo does pull from master, it's not going to help him much.

Capable: Who can perform an action that changes the relationship?

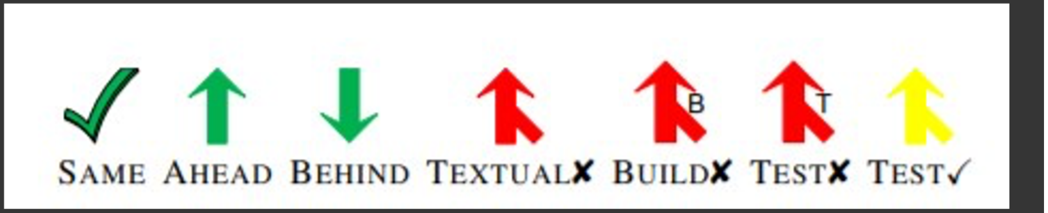
- George and Ringo are in a textual conflict.
- If Ringo has already pushed his changes to the master, George must be the one to resolve this commit.
- If George has already pushed his, Ringo must be the one to resolve the commit instead
- If neither has pushed, any one of them can resolve it.
- Crystal can help both of them have a clear picture of scenarios such as these.

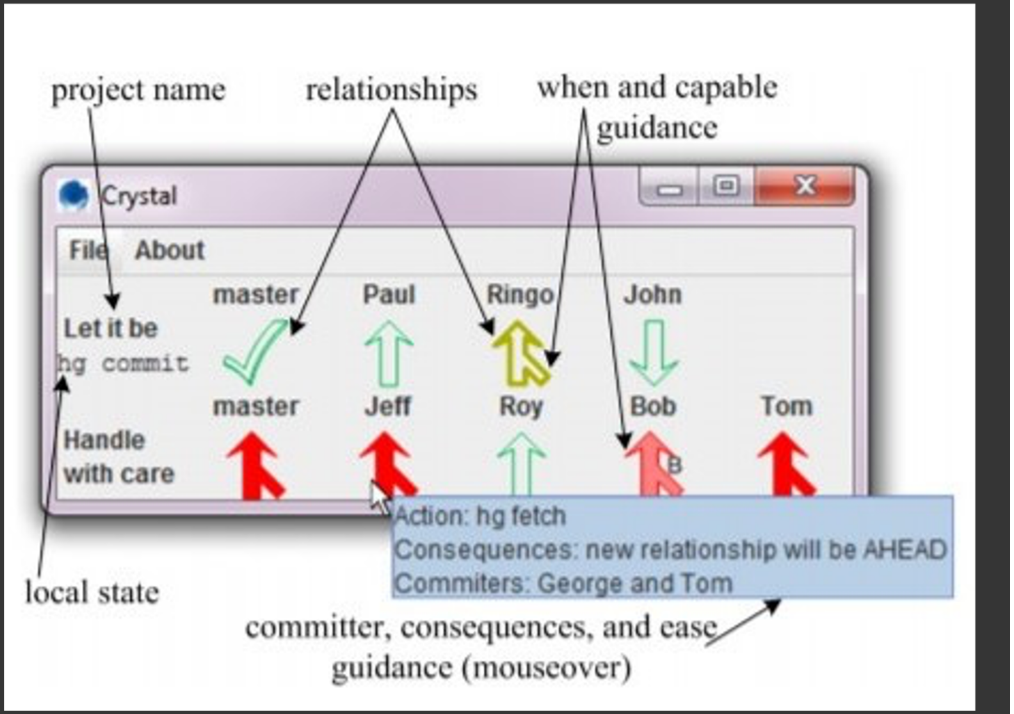
Ease: Has anyone already fixed the issue but hasn't pushed to master yet?

- George and Ringo are in a textual conflict.
- Ringo pushed to the master
- If George pulls, he will have to resolve the conflict.
- However, if Ringo has already worked on resolving the conflict, George can wait till Ringo pushes them.
- Crystal can help George know if Ringo has already made follow up changes to his commit, so George doesn't have to put in the unnecessary effort to fix things.

Crystal's UI

- UI that George can see.
- Hollow icons mean George has to wait. Solid icons mean George can perform action with the VCS
- John has made more commits that George hasn't.
- George is behind John but he must wait till John pushes to Master in order to pull from the Master. This is represented by a hollow icon.





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

Related Work

Most current research similar to this paper focus on awareness tools.

- Palantír shows which developers are changing which artifacts by how much
- Syde is another awareness tool that reports on textual changes but reduces its false positives via a fine-grained analysis of the abstract syntax trees (ASTs) modifications
- CollabVS detects a potential conflict when a user starts editing a program element that has a dependency on another program element that has been edited but not checked-in by another developer.

Crystal differs in the sense that

- It provides guidance along with precise conflict information
- Reports much fewer false positives
- Most importantly, can handle higher level conflicts apart from textual changes such as build conflicts and test failures.

Threats To Validity

- The empirical study does not take into account when and how the developers found out the conflicting relationships.
- The experiments are performed in the context of Distributed VCs, which differ from CVCs. So the same assumptions and analyses may not hold for all version control systems.
- The study is focused on nine open-source systems which may not be characteristic of all other systems.
- Usability and developer style may effect how Crystal helps users. For instance, in projects where conflicts don’t occur much at all, Crystal may end up being a distraction for the developers.

Thank you