

Proactive Detection of Collaboration Conflicts

Authors: Y. Brun, R. Holmes, M.D. Ernst, D. Notkin

Conference: 19th ACM SIGSOFT symposium & the 13th European conference on Foundations of Software Engineering, 2011

Presented By: Pronnoy Goswami

Overview

- ❑ Background
- ❑ Problem Statement
- ❑ Approach
- ❑ Motivation Behind the Approach
- ❑ Contributions of the work
- ❑ Terminology Used
- ❑ Research Questions and Analysis of the Approach
- ❑ Related Work
- ❑ Conclusion
- ❑ Discussion Questions
- ❑ References
- ❑ Appendix

What is all the **fuss** about?

Let's gain some **background knowledge**.

Version Control Systems (VCS)

What is a “**version control system**”, and why should you care?

- ❑ **Version control system (a.k.a, VCS)** is a system that records changes to a file or set of files over time so that you can recall specific versions later

- ❑ **Features** that a VCS provides:
 - ❑ Allows us to revert selected files back to previous state
 - ❑ Revert the entire project back to a previous state
 - ❑ Compare changes over time
 - ❑ See who last modified something that might be causing a problem
 - ❑ Who introduced an issue and when and more..

Some commonly used VCS



Git



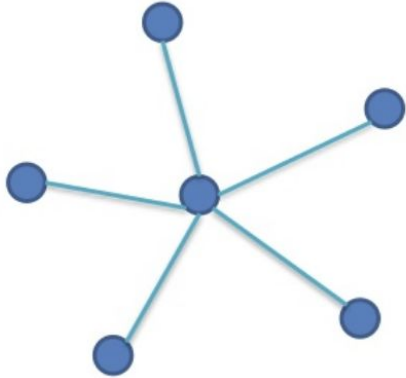
Visual Studio – Team Foundation
Server



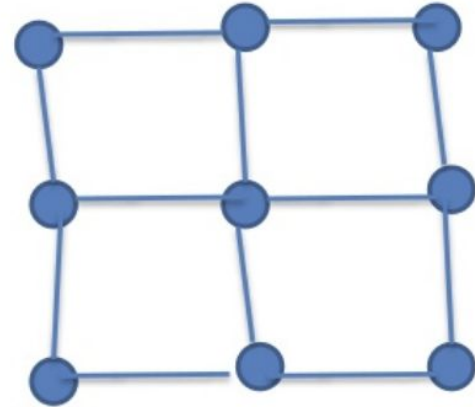
Plastic SCM



Types of VCS



Centralised Version Control
System (CVCS)



Distributed Version Control
System (DVCS)

Centralised VCS (CVCS)

- ❑ Have a **single server** that contains all the versioned files
- ❑ Clients check out files from that central server

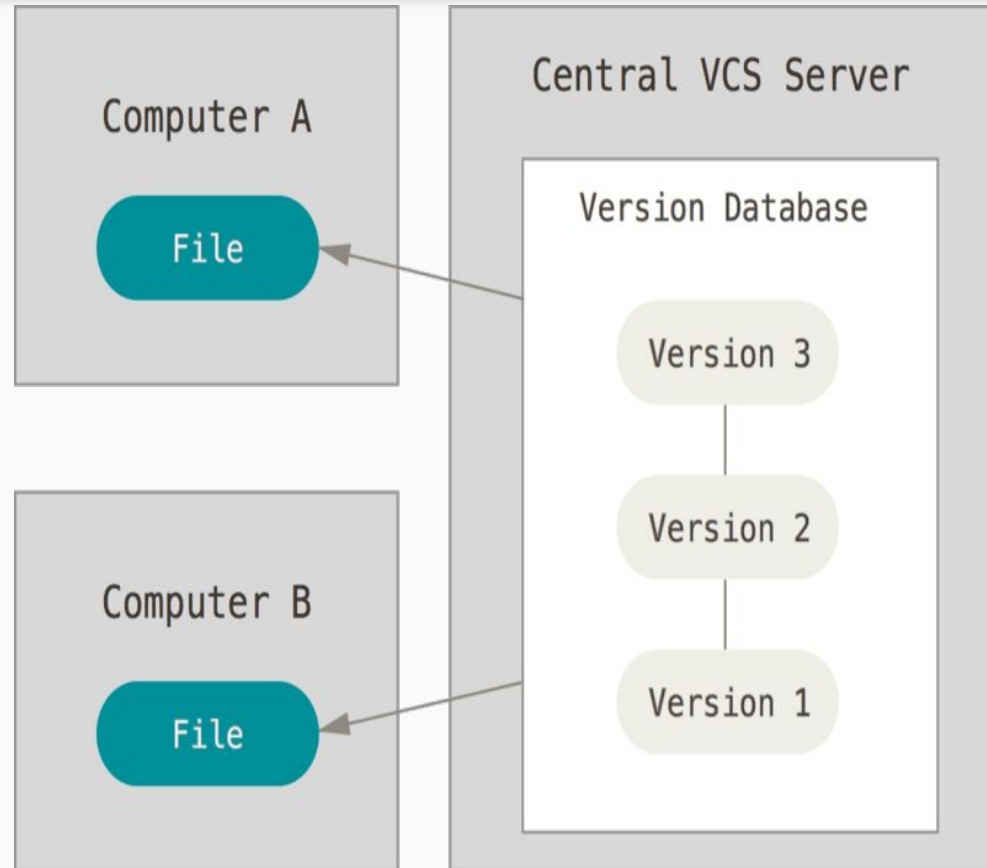
Advantages

- Developers are aware to some extent of what their peers are working on
- Administrators have fine-grained control over who can do what

Disadvantages

- Single point of failure that the centralized server represents

Examples - Subversion, Perforce, CVS etc.



Distributed VCS (DVCS)

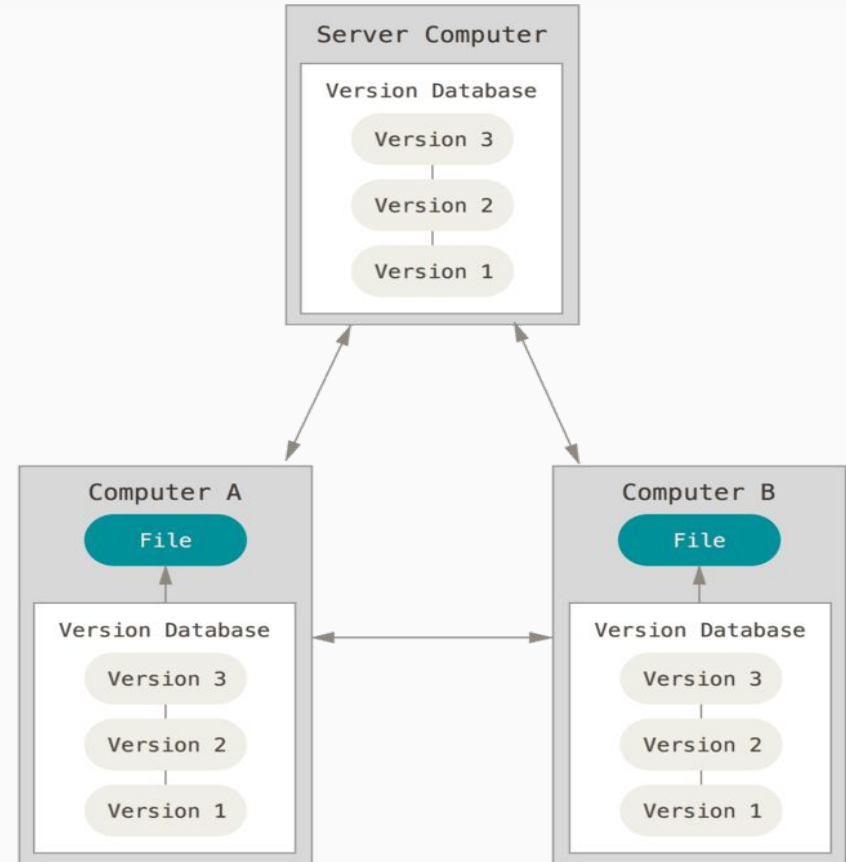
- ❑ Clients **fully mirror** the repository, including its full history.
- ❑ **No single-point of failure** as in CVSCs

Advantages

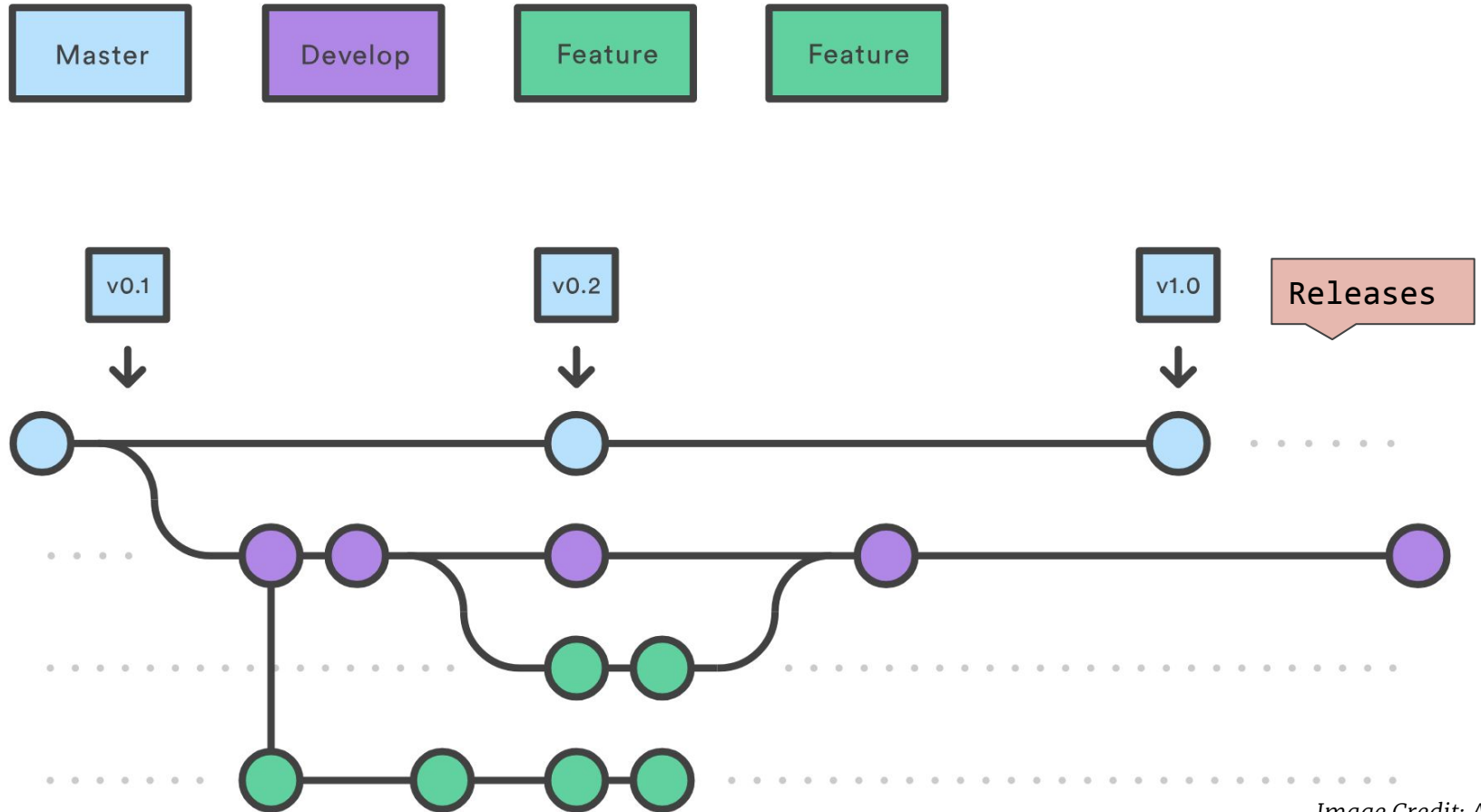
- Can deal pretty well with having several remote repositories
- Allows you to set up several types of workflows that aren't possible in centralized systems

Disadvantages

- Developers are less aware what their peers are working on
- Steeper-learning curve



Snapshot of a VCS from a typical project



Conflicts

What are conflicts?

- ❑ A **conflict** occurs when two different users make simultaneous, different changes to the same line of a file.
- ❑ In this case, the VCS cannot automatically decide which of the two edits to use (or a combination of them, or neither!)
- ❑ Manual intervention is required to resolve the conflict.
- ❑ Conflicts are **costly!**

Real-world example of a **merge conflict** in a Java project

```
private static IInstaller CreateInstance(Type type)
{
    return (IInstaller)Activator.CreateInstance(type);
}
```

```
public static IEnumerable<Type> TryGetExportedTypes(this Assembly assembly)
{
    try
    {
        return assembly.GetExportedTypes();
    }
    catch (Exception ex)
    {
```

```
<<<<<< HEAD
```

```
    Debug.WriteLine(ex.Message);
```

```
=====
```

```
    Debug.WriteLine(ex.InnerException);
```

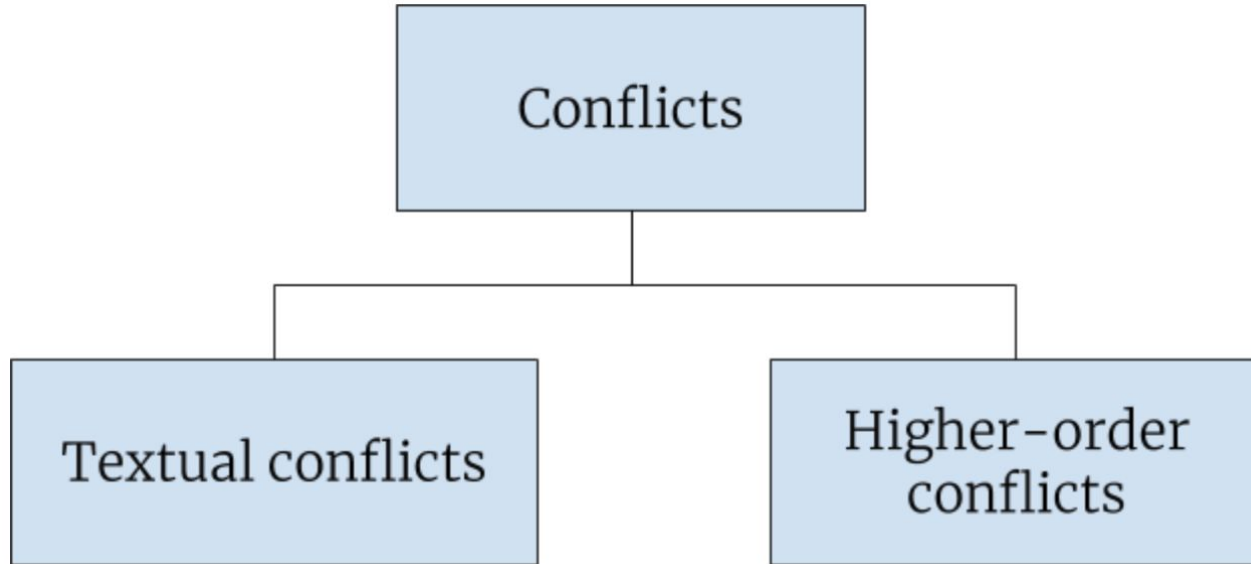
```
>>>>>> refs/heads/Feature1
```

```
}
```

```
    return new Type[] { };
```

```
}
```

Types of conflicts



Textual Conflicts

Why do they happen?

- ❑ Arise when two developers make inconsistent changes to the same part of the source code

How VCS handles it?

- ❑ VCS allows the first developer to publish changes, but precludes the second developer from publishing until the conflict is resolved automatically (by the VCS) or manually (by a developer)

Impact of textual conflicts?

- ❑ Although they can be costly to resolve but are less severe

Higher-order Conflicts

Why do they happen?

- ❑ Arise when the VCS can integrate the developers' textual changes, but the changes are **semantically incompatible** and can cause compilation errors, test failures, or other problems

How VCS handles it?

- ❑ One idea can be to integrate the test-suite and language semantics to run in background and inform developers about errors
- ❑ Any other ideas??

Impact of higher-order conflicts?

- ❑ More damaging than textual conflicts

Conflicts are the **norm!**

BRACE YOURSELF



MERGE CONFLICTS ARE COMING

Problem Statement

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

While working within a **team**, have you ever made a mistake while programming and only realized it later?

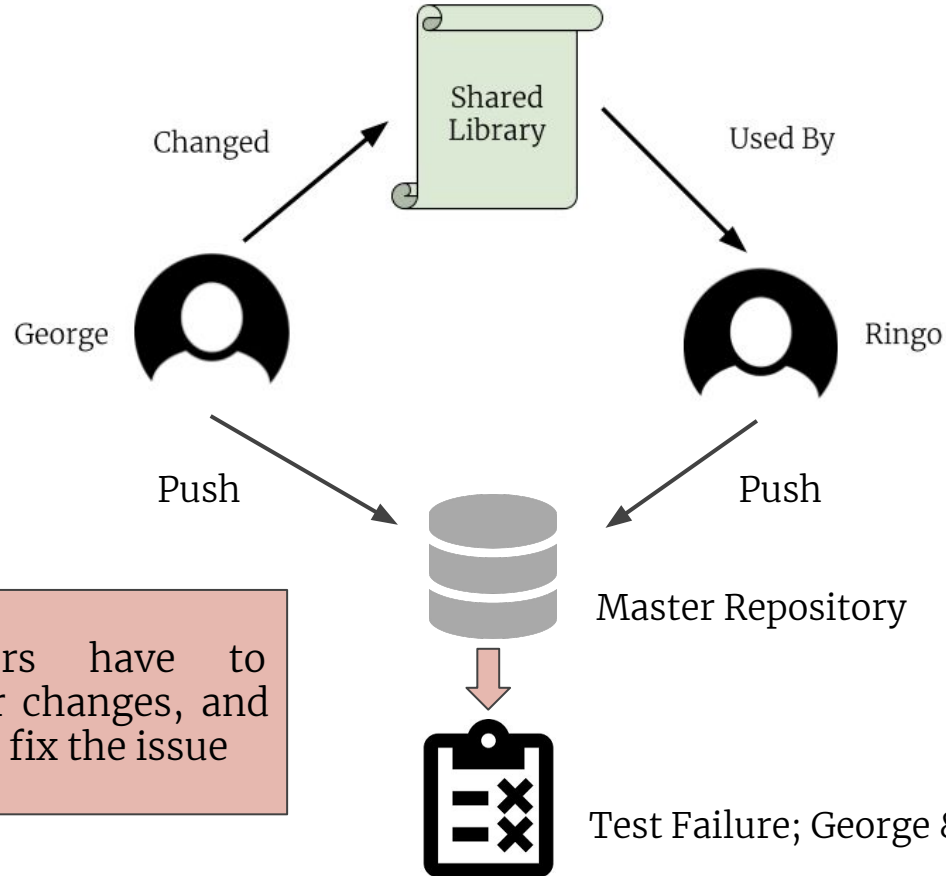
Mistakes can be related to:

- ❑ Version Control Issues
- ❑ Design decision
- ❑ Code Refactoring
- ❑ Repeating someone else's work
- ❑ API Usage

This leads to **conflicts** and **bugs** within the project.

Let's look at a **scenario..**

Scenario comprising **two developers**, George and Ringo



Problem: Developers have to recollect their earlier changes, and rework on the code to fix the issue

Few **ideas** to solve this problem?

- ❑ Use an **awareness tool**, which reports where in the code base teammates are working
- ❑ Help developers to make better informed decisions about **how** to share changes with the team
- ❑ Help developers to understand **when** is the best time to share changes with the team
- ❑ **Setting norms** among developers about the development practices for a project
- ❑ **Up-skilling developers** about the good practices related to conflict resolution
- ❑ **Simulate the possible actions** that a developer performs on the source code. **How?**

Limitations of using an awareness tool

- ❑ Using an awareness tool may lead to a lot of **false-positive warnings**
- ❑ Developers might have been exploring some ideas and changes, without ever intending to share the intermediate changes with his team
- ❑ A lot of false warnings can be counter-productive for developers

We have a problem at hand. Solution?

Speculative Analysis approach to the rescue!

- ❑ In short, predict the future and analyze it
- ❑ **Unobtrusively** provides information about the presence or absence of conflicts in a continual and accurate way

How does it work?

- ❑ Speculatively performs the work and executes the VCS operations in the background on clones of the program
- ❑ It merges the committed codes of two different developers, builds it, and runs its tests in the background

Motivation behind the approach

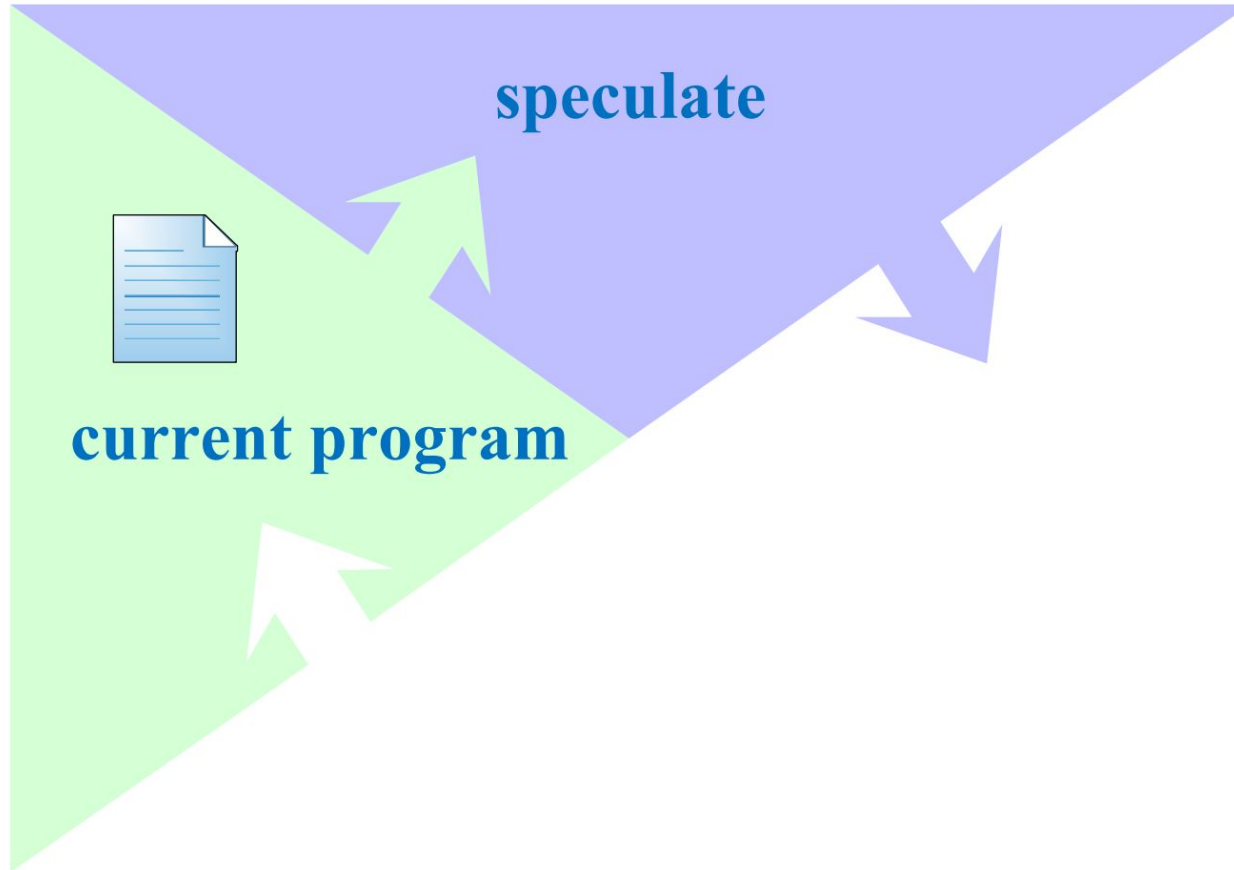
Use the information provided by the speculative analysis:

- ❑ To allow developers make better-informed decisions about version control issues
- ❑ To help developers understand how and when to share changes
- ❑ Reducing the need for human processing and reasoning
- ❑ Thus, reducing the cost of conflicts in the software development lifecycle

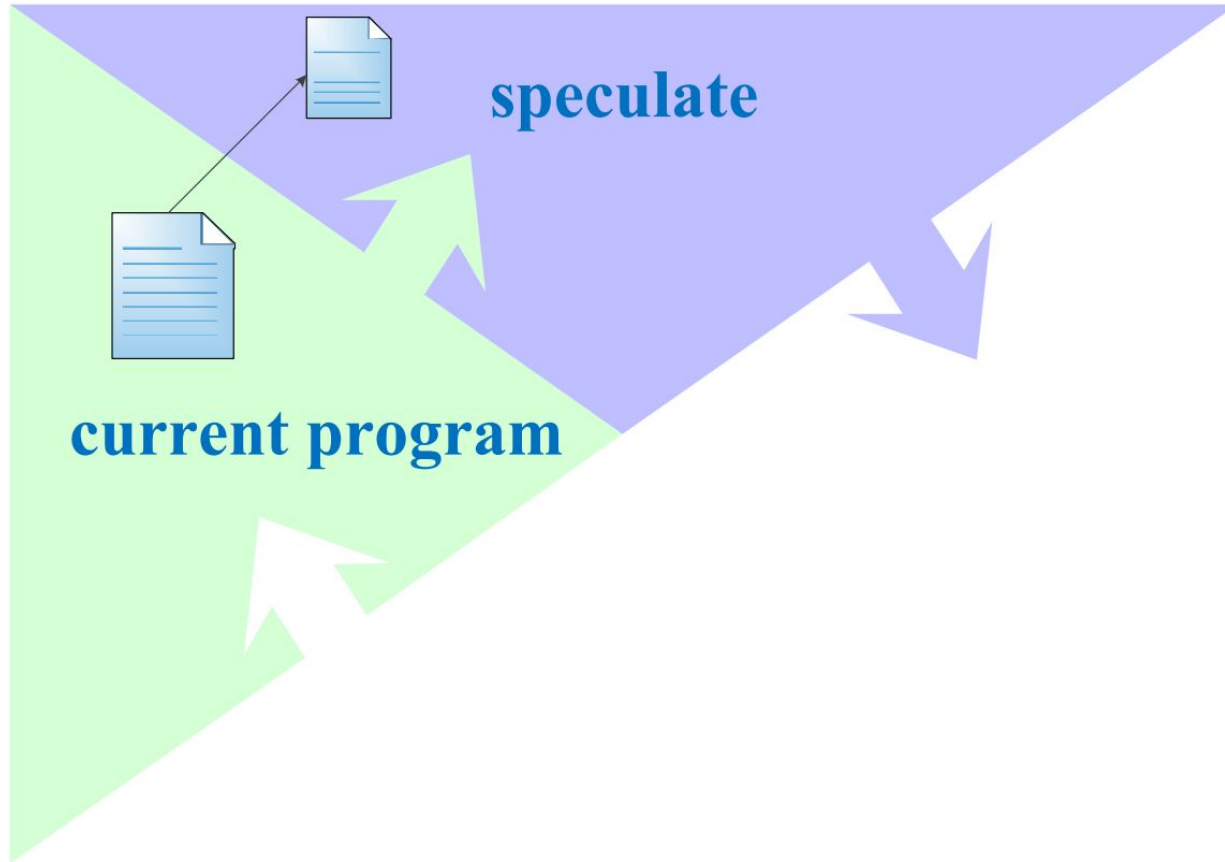
Speculative Analysis: Workflow



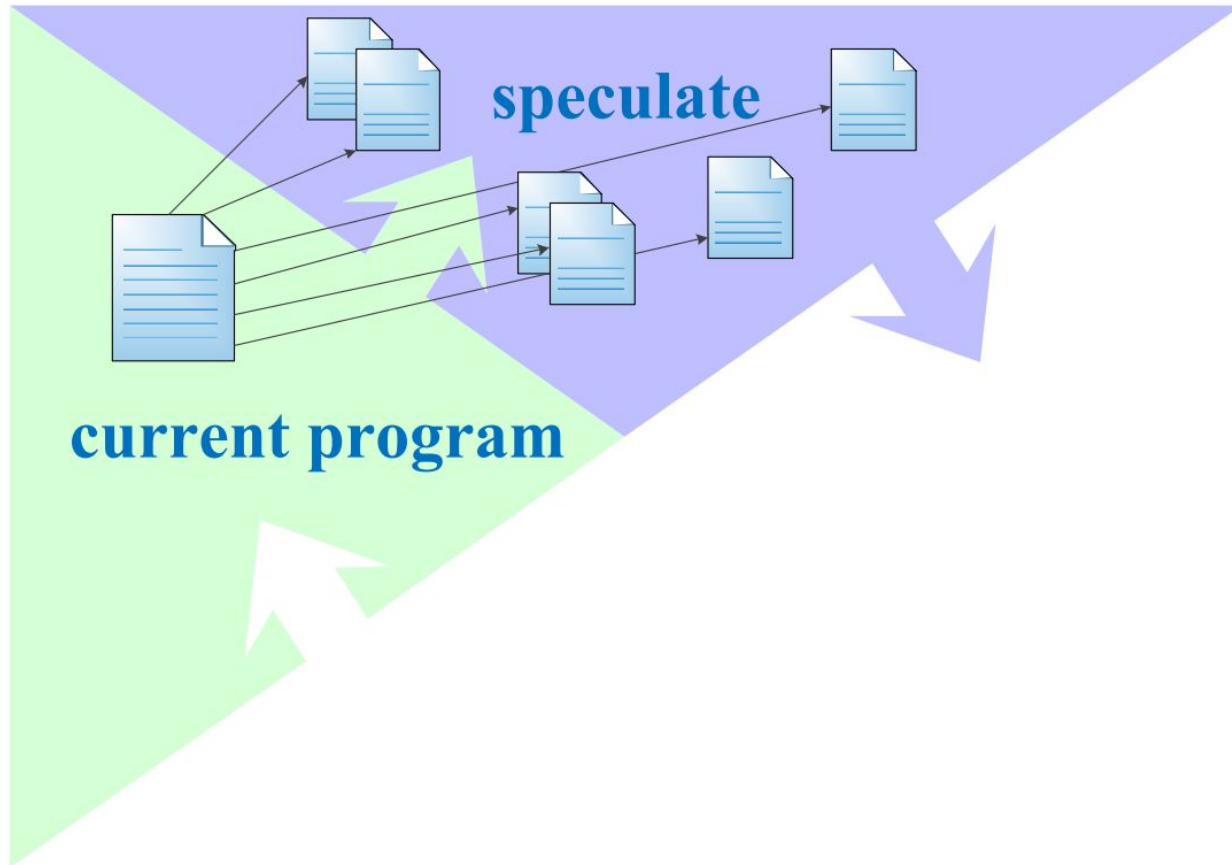
Speculative Analysis: Workflow



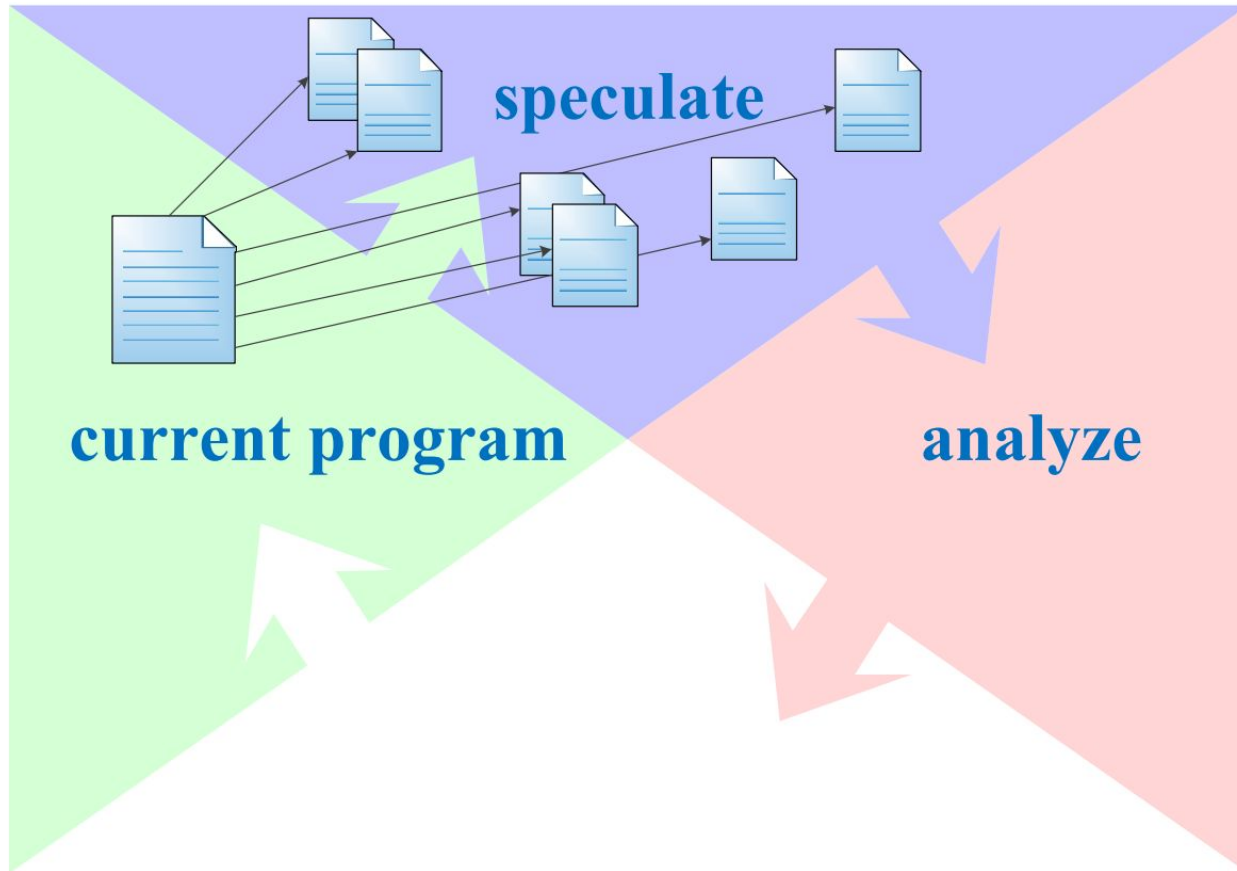
Speculative Analysis: Workflow



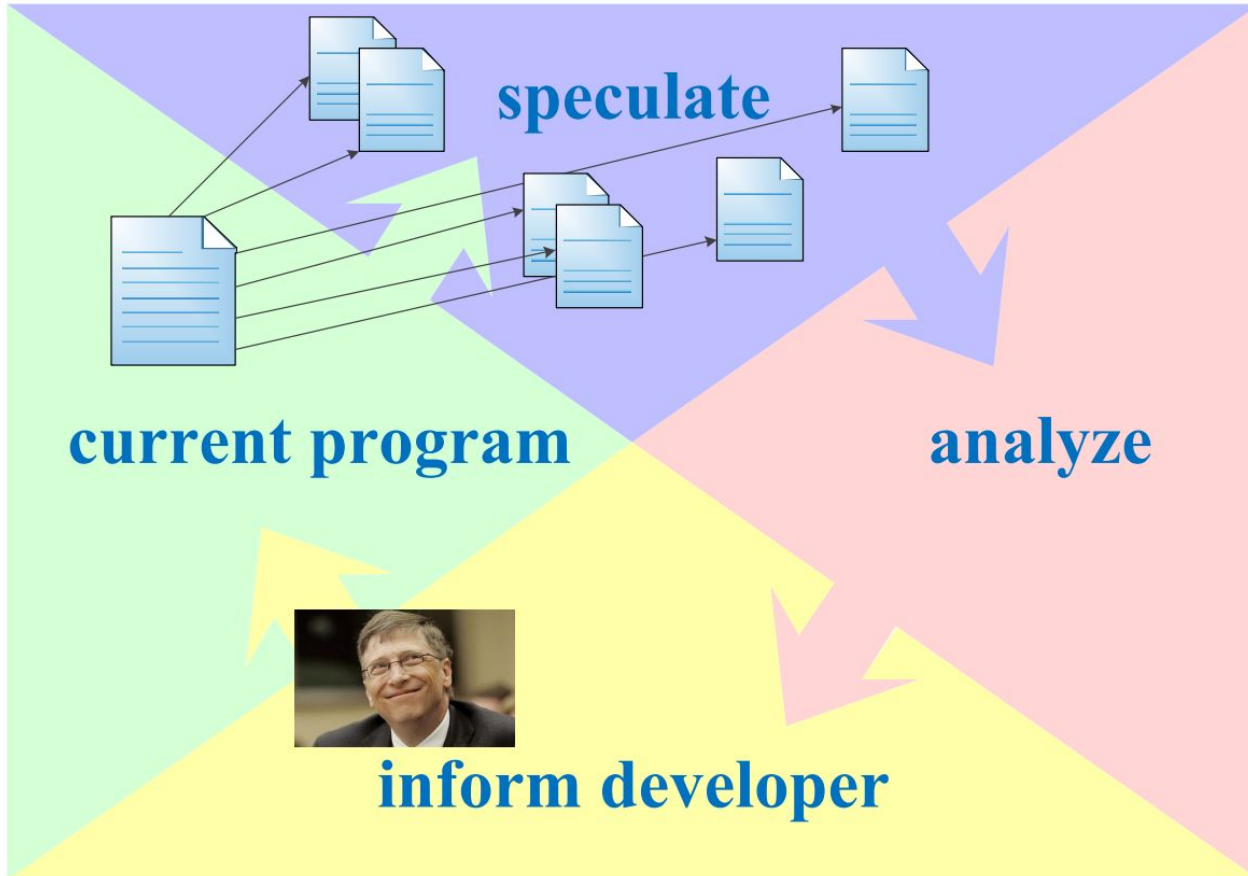
Speculative Analysis: Workflow



Speculative Analysis: Workflow



Speculative Analysis: Workflow



Contributions of the work

Contributions of the work are **threefold**.

1. Analyzed **9 open-source systems** and drew insights about the conflicts between developers' copies of a project
2. Used information given by **speculative analysis**, to reveal important classes of conflicts and offer advice about addressing them
3. Designed and implement an open-source, publicly-available tool called **Crystal** (<http://crystalvc.googlecode.com>). It implements the analyses and unobtrusively presents advice to developers, to aid them in identifying, managing, and preventing conflicts

Previous scenario after using the Crystal tool

- ❑ The **green arrow** informs George that his changes can be published (uploaded) without conflict to the master repository.
- ❑ The **red merge** symbol indicates that Ringo's changes, if combined with George's, would cause a test ("T") failure.

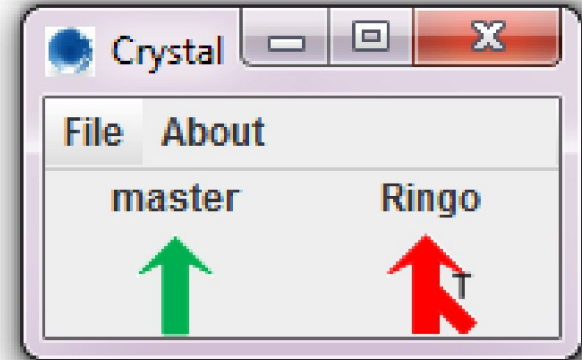


Fig: Screenshot of the Crystal tool run by George

Let's understand some **terminology**.

Study Design of the terminology

- ❑ The results of this work are applicable to both CVCS and DVCS
- ❑ The paper focuses on DVCS to simplify the presentation
- ❑ Briefly presents accepted DVCS terminology
- ❑ Uses those terminologies to infer 7 relationships between repositories

A typical DVCS snapshot

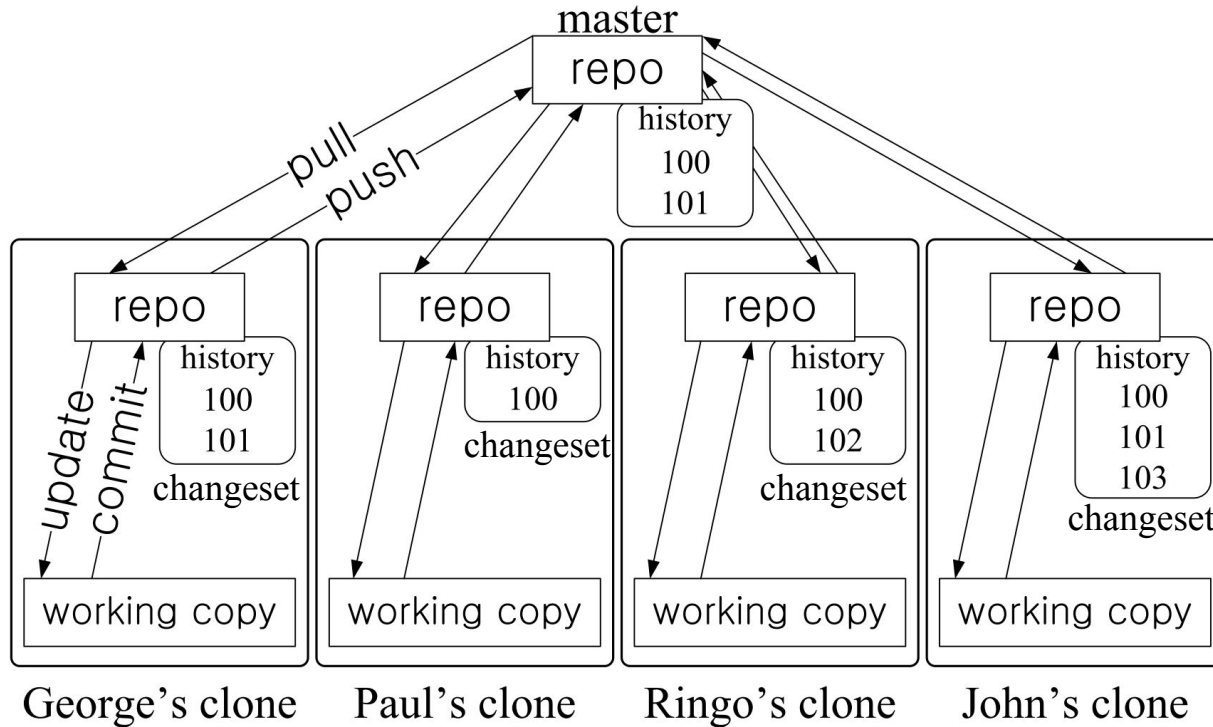


Fig: A DVCS with four clones of a master repository

Repository Relationships

Researchers have identified 7 relevant relationships that can hold between two repositories

1. **SAME:** The repositories have the same changesets
2. **AHEAD:** The repository has a superset of the other repository's changesets
3. **BEHIND:** The opposite of AHEAD
4. **TEXTUAL ✗** (textual conflict) : The distinct changesets need human intervention as they cannot be automatically merged by the VCS
5. **BUILD ✗:** The repositories can be automatically merged by the VCS, but the resulting merged code fails to build
6. **TEST ✗:** The repositories can be automatically merged by the VCS and the resulting merged code builds but fails its test suite
7. **TEST ✓ :** The repositories can be automatically merged by the VCS and the resulting merged code builds and passes its test suite

Assumption(s) behind the approach

- ❑ When a developer commits code to the VCS, the developer has decided to share that code with other developers
- ❑ Developers push to and pull from only the master repository
- ❑ Developers only make a commit when all their tests pass

Let's **deep dive** into the study & analysis

Dataset

- ❑ 9 open-source projects
- ❑ 3.4 million lines of code
- ❑ 550,000 development versions analyzed
- ❑ KNCSL stands for thousands of non-comment source lines
- ❑ The version control history ends on Feb 13, 2010

system	KNCSL	devs	changesets	days	description
Gallery3	57	24	4,838	437	Web-based photo album
Git	267	27	20,785	1,741	Version control system
Insoshi	173	15	1,316	629	Social networking platform
jQuery	26	23	2,183	1,393	JavaScript library
MaNGOS	643	27	3,511	626	Online game server
Perl5	660	51	34,653	8,061	Programming language
Rails	141	50	12,342	1,875	Web application framework
Samba	1,363	59	58,802	5,001	File and print services
Voldemort	103	22	1,219	375	Structured storage system
Total	3,433	298	138,549	20,138	

Fig: Dataset consisting of the 9 subject projects

Criteria to choose the dataset

- ❑ 9 most active projects on Github
- ❑ Project has at least **10 developers**
- ❑ Project has at least **1000 changesets**
- ❑ Project is not just a Git copy of a CVC repository. Because, it would not contain sufficient information to answer the research questions

Research Questions

RQ1: How frequently do conflicts, textual and higher-order, arise across developers' copies of a project?

RQ2: How long do conflicts persist?

RQ3: Do clean merges devolve into conflicting changes?

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

Research Question 1

Explores: Frequency of conflicts

RQ1: How frequently do conflicts, textual and higher-order, 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%			

Fig: Historical merges. Frequencies with which developers experienced 4 relationships when they integrated their code

RQ1: How frequently do conflicts, textual and higher-order, arise across developers' copies of a project?

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%

Fig: Potential early merges. The frequency with which developers would be informed of TEXTUAL✗ TEXTUAL✓ relationships, if they had used Crystal

RQ1: How frequently do conflicts, textual and higher-order, arise across developers' copies of a project?

Findings about textual conflicts

- ❑ Conflicts are the **norm**
- ❑ Of all the merges, one in six, or **17%**, had textual conflicts
- ❑ An unrecognized TEXTUAL✗ between the repositories of two developers may cause problems
- ❑ Had the developers been using Crystal, it would have informed them about TEXTUAL✗ relationship that resulted from **19%** of the commits
- ❑ But does the TEXTUAL✓ relationship mean that everything is going as planned? **Maybe**
Not

Interviews with managers portray a serious problem

“The remote guys tend not to commit frequently enough to get leverage out of our continuous integration builds, even after prompting. It is a real challenge to know how far out of sync [the remote teams] are [with the local team] when their commits are not being merged in regularly.

...

I want [my developers] to at least initiate a conversation with the relevant parties when the system says they have, or are just about to, walk into a conflicting situation. I also want the system to give them a certain level of **trust** of other developer's changes so that if [a merge] won't cause a problem, they should sync up.”

RQ1: How frequently do conflicts, textual and higher-order, arise across developers' copies of a project?

Findings about higher-order conflicts

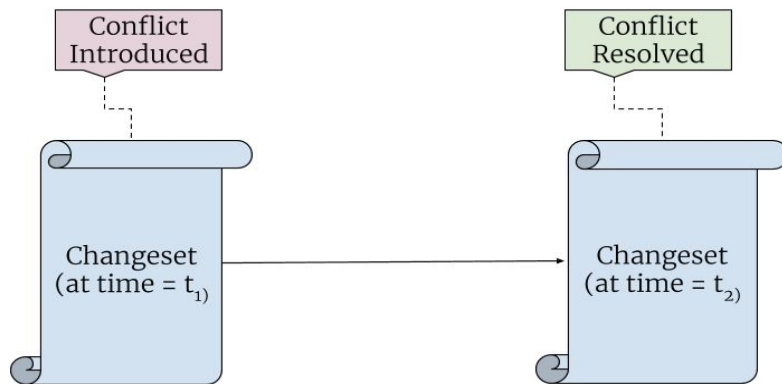
- ❑ Analyzed only **3 projects** Git, Perl5 and Voldemort. Because, only these had the a non-trivial test suite that could be run
- ❑ **5,355 merges** analyzed in 3 projects
- ❑ **76%** of merges completed cleanly, **16%** resulted in TEXTUAL✗, **1%** resulted in a BUILD✗, and **6%** resulted in TEST✗
- ❑ **33%** of the **399 merges** that the version control system reported as being a clean merge, actually were a TEST✗ or BUILD✗ conflict

Research Question 2

Explores: Persistence of conflicts

RQ2: How long do conflicts persist?

Study Design

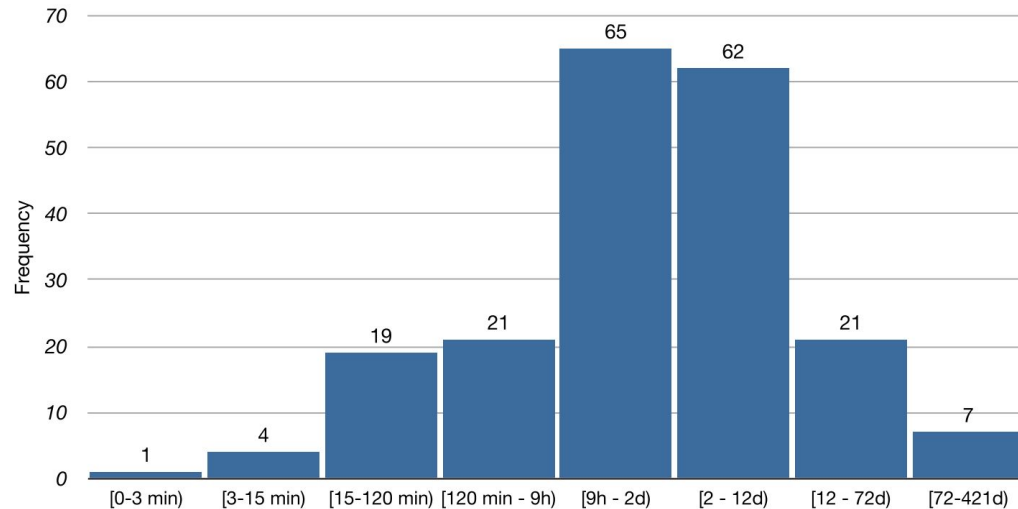


$$\text{Lifespan}(\text{conflict}) = t_2 - t_1$$

- ❑ Omitted all conflicts between changesets that were never actually merged in the history
- ❑ Omitted all conflicts on dead-end or stale branches
- ❑ Analyzed **4 projects** – Gallery3, Insoshi, MaNGOS, and Voldemort. Due, to sheer volume of the data and computation costs.

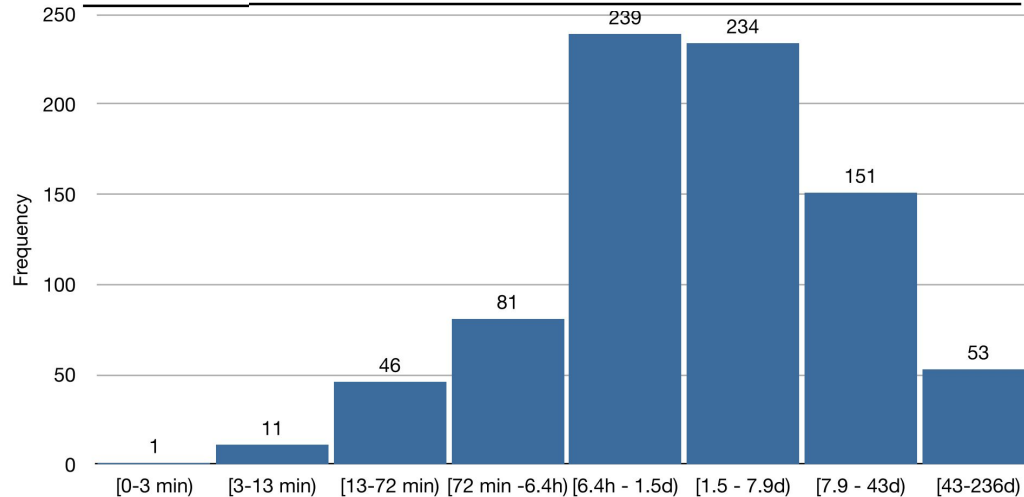
RQ2: How long do conflicts persist?

system	TEXTUAL✕ relationships						
	number	length (days)			length (changesets)		
		mean	stddev	median	mean	stddev	median
Voldemort	39	25.7	35.0	8.9	12.8	16.4	6
Gallery3	80	3.1	9.4	0.7	7.5	19.7	3
Insoshi	27	11.8	21.2	4.8	9.4	16.3	3
MaNGOS	58	8.2	44.0	1.8	17.6	83.0	3
Total	204	9.8	30.3	1.6	11.6	96.9	3



RQ2: How long do conflicts persist?

system	TEXTUAL✓ relationships						
	number	length (days)			length (changesets)		
		mean	stddev	median	mean	stddev	median
Voldemort	128	35.0	46.6	7.3	9.7	15.1	3
Gallery3	483	7.1	12.5	1.0	14.3	30.0	3
Insoshi	87	7.7	13.5	3.7	6.3	8.5	3
MaNGOS	118	2.4	2.1	1.7	5.8	7.1	3
Total	816	11.0	23.8	1.9	11.5	24.2	3



Research Question 3

Explores: Escalation of clean merges into conflicts

RQ3: Do clean merges devolve into conflicting changes?

- ❑ Of all conflict relationships (TEXTUAL✗, BUILD✗, and TEST✗), **93%** developed from a TEST✓ relationship
- ❑ Rest **7%** of conflict relationships developed from a BEHIND relationship
- ❑ In other words, in almost every case, both developers had already committed (but not shared) changes before the conflict developed
- ❑ **20%** of TEST✓ relationships resulted into a conflict
- ❑ This suggests that what we call “safe merges” are actually at risk of devolving into conflicts that require human effort to resolve

Research Question 4

Explores: Information about conflicts

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

Questions Explored to answer RQ4

- ❑ What kind of unexploited information is available from a VCS that is not yet leveraged to smooth collaboration?
- ❑ What information could help the developer make better-informed decisions?

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

Assumptions

- ❑ Considered all situations with **3 developers**. Third developer was used to represent arbitrary other developers and repository hierarchies
- ❑ Limit lookahead to **2 rounds** of version control operations. One developer performs one VC operation and then the other developer may or may not perform one.

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

Analysis Methodology

- ❑ Analyzed a hypothetical global perspective of all version control information across repositories
- ❑ Then systematically analyzed the above perspective by
 - ❑ enumerating all possible **version control situations** or **states**
 - ❑ determining the **best course of action** for the team
 - ❑ identifying what **information that decision depended** on
 - ❑ **classifying the advice** for the team

RQ4: Local States

The paper describes 5 possible local states.

STATE	DESCRIPTION
UNCOMMITTED	There are uncommitted changes in the working copy
IN CONFLICT	The local repository is in conflict with itself
BUILD FAILURE	The repository's version of the code fails to build
TEST FAILURE	The repository's version of the code builds but fails its test suite
OK	The repository's version of the code builds and passes its test-suite

RQ4: Possible Actions

Given 2 repositories A and B, the relationship between A and B, and local states determine the possible actions developers can perform.

INITIAL STATE	POSSIBLE ACTION(S)	FINAL STATE
SAME	Nothing to do	SAME
AHEAD	Push	SAME
BEHIND	Pull	SAME
TEXTUAL✗	Pull; Push	CONFLICT; B will be in TEXTUAL✗
BUILD✗	Pull and Merge; Push	BUILD FAILURE; B will be in BUILD✗
TEST✗	Pull and Merge; Push	TEST FAILURE; B will be in TEST✗
TEST✓	Pull and Merge; Push	AHEAD; B will be in TEST✓

Motivation

- ❑ Obtain information about how each action may affect the developer's state
- ❑ Understand how different relationships can help developers make better-informed decisions

Assumptions

- ❑ Developers perform actions in a **tree hierarchy**, pushing only to and pulling only from a parent
- ❑ Considered information relevant to **2 developers** who share a common parent repository

RQ4: Guidance

- ❑ **Committer:** Who made the relevant changes?
- ❑ **When:** Can an action that affects the 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 made changes that ease resolving an existing conflict

Crystal: A tool for version control advice

Crystal: User Study

- ❑ Authors surveyed **50 DVCS users** before building Crystal.
- ❑ They focused on the following aspects:
 - ❑ What are their **developments habits** while working collaboratively?
 - ❑ Which operating systems, IDEs, VCSs, and programming languages the developers use?
 - ❑ What are their habits while working with the VCS like, Git?

Crystal: User Interface

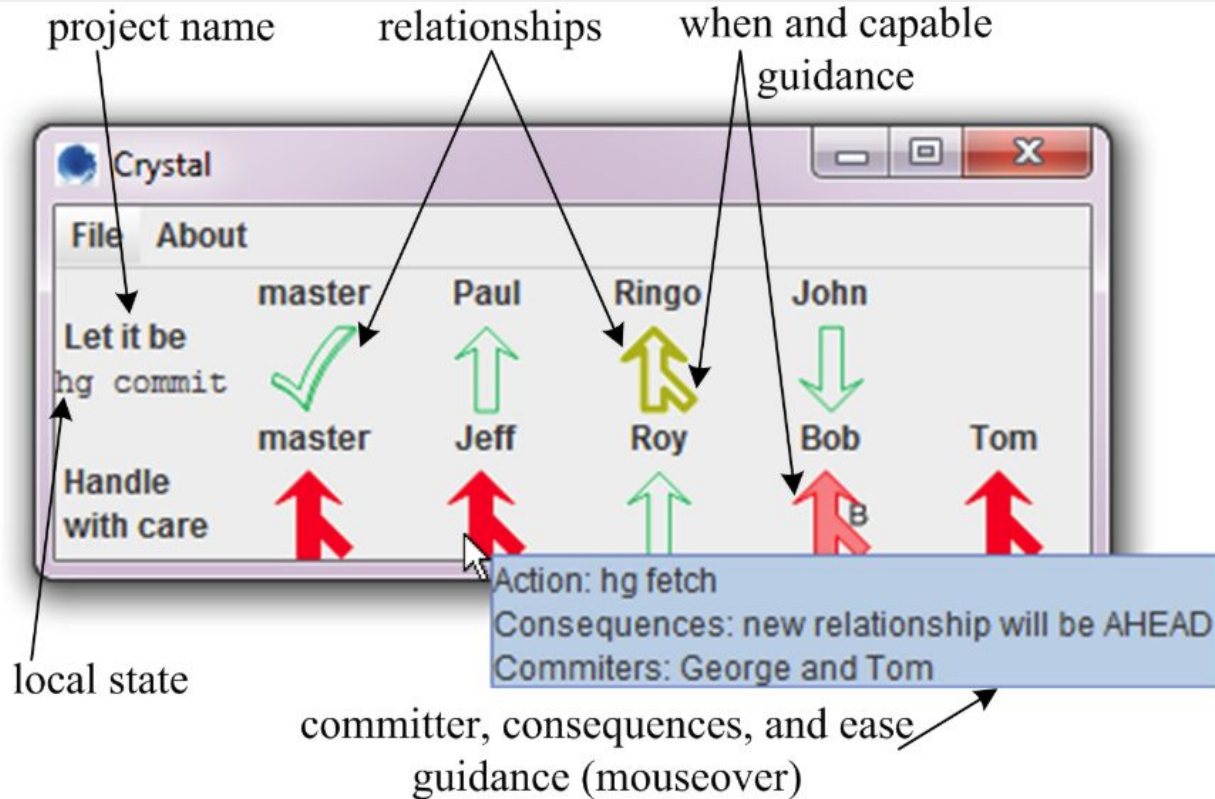





Fig: Screenshot of a developers' (George) view of Crystal

Crystal: User Interface



Fig: Crystal-Icon Color Scheme

-  No merging required
-  Manual merging required
-  Can be automatically merged

Crystal: Features

- ❑ Developed on **Java**
- ❑ Allows developers to select the repositories they want
- ❑ Provides information about relationships with developers even if they are not using it
- ❑ Allows developers to execute a subset of the tests related to their feature
- ❑ At that time, Crystal worked with **Mercurial DVCS**
- ❑ Available for Windows, Mac OS, and Linux distributions

Crystal: Beta Testing

- ❑ Beta version tested by a small number of developers
- ❑ Tested on Windows, MacOS X, and many Linux distributions
- ❑ One co-author used Crystal to monitor **49 clones** of **10 projects** that belonged to **8 actively working developers**
- ❑ Received frequent feedback from the users
- ❑ Incorporated the feedback into Crystal

Let's explore some **related works**

Related Work

- ❑ **Dewayne et al.**[5] show that as the parallel work increases the number of defects in the software rises.
- ❑ **Grinter et al**[3]. reveal that developers avoid parallel work to avoid conflict resolution while committing code.
- ❑ **Palantír**[7], shows which developers are changing which artifacts and by how much.
- ❑ **FASTDash**[1], gives a spatial representation of which files each developer is editing.
- ❑ **Syde**[4], reduces false positives using fine-grained analysis of the ASTs
- ❑ **CollabVS**[8] detects a potential conflict between dependent program elements
- ❑ **Safe-commit**[2] performs deepest analysis by identifying changes that are guaranteed not to cause test-failure

Let's summarize!

Conclusion

- ❑ Speculative analysis over version control operations provides accurate information about conflicts
- ❑ Conflicts are very frequent within collaborative teams
- ❑ In the dataset, **16%** of all merges require human effort to resolve textual conflict
- ❑ **33%** of merges that were reported as “safe merges” actually contained higher-order conflicts
- ❑ On average conflicts persist for **10 days**. With a median conflict persisting **1.6 days**
- ❑ **Crystal**, a speculative analysis tool provides accurate information and advice about pending conflicts unobtrusively
- ❑ Collaborative development is important but troublesome

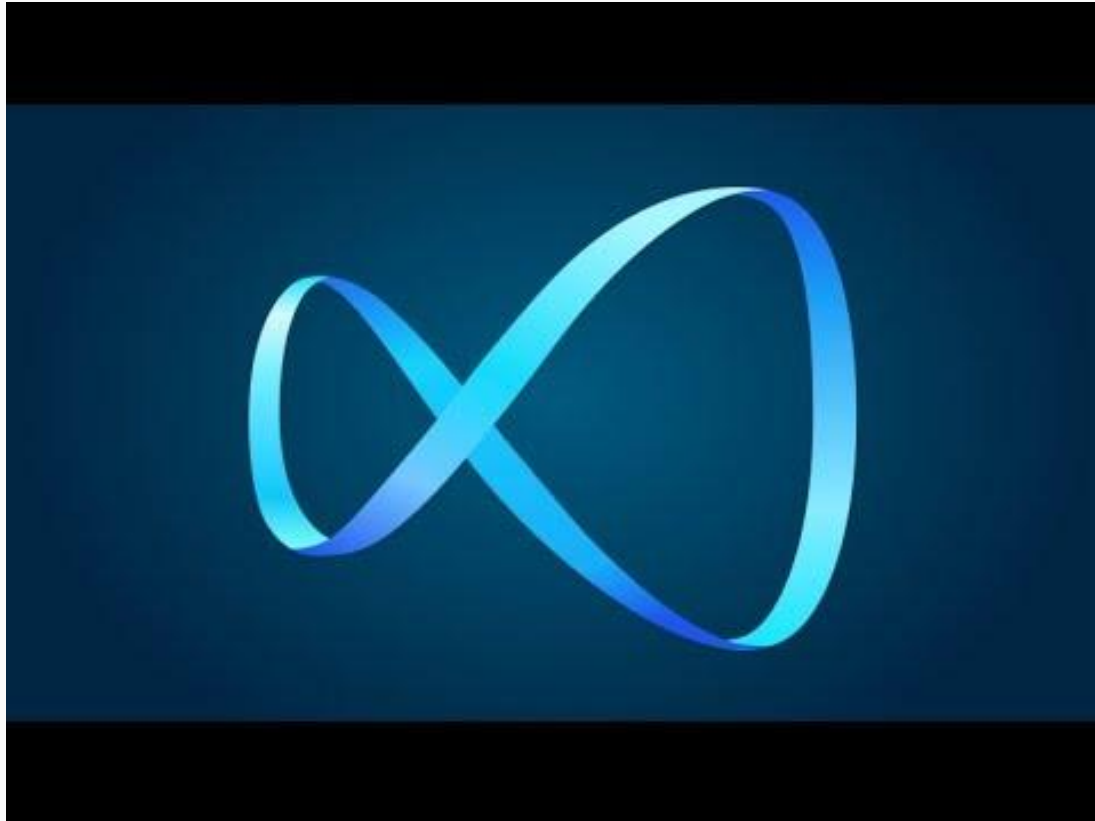
Penny for your thoughts!

Discussion Questions

- ❑ Can the **dataset generalize** well to other projects? Could they have selected more projects that have a non-trivial test-suite?
- ❑ Could they have performed a **larger user study during Beta Testing** with detailed analysis of the user feedback?
- ❑ Does Crystal actually increase developer productivity or distract them? Should we see the actual **adoption rate of Crystal** using a user study?
- ❑ Since, Crystal runs on a developers' local machine. Do you think that some analysis on the **computational cost** and **memory usage** could have been provided?
- ❑ Projects using a DVCS have **lots of branches**. Any ideas on how will Crystal handle branching?
- ❑ VCS history informs us about the TEXTUAL✗ and TEXTUAL✓ relationship. Any ideas how to **handle other relationships**?
- ❑ Will the collaborative development **habit of a developer vary** between DVCS and CVCS? If yes, then any ideas how will Crystal handle it?

Current State-of-the-art

Real Time Collaboration in **Microsoft VS Code**



References

1. Jacob T. Biehl, Mary Czerwinski, Greg Smith, and George G. Robertson. FASTDash: A visual dashboard for fostering awareness in software teams. In CHI, pages 1313–1322, San Jose, CA, USA, Apr. 2007.
2. Jan Wloka, Barbara Ryder, Frank Tip, and Xiaoxia Ren. Safe-commit analysis to facilitate team software development. In ICSE, pages 507–517, Vancouver, BC, Canada, May 2009.
3. Rebecca E. Grinter. Using a configuration management tool to coordinate software development. In CoOCS, pages 168–177, Milpitas, CA, USA, Aug. 1995.
4. Lile Hattori and Michele Lanza. Syde: A tool for collaborative software development. In ICSE Tool Demo, pages 235–238, Cape Town, South Africa, May 2010.
5. Dewayne E. Perry, Harvey P. Siy, and Lawrence G. Votta. Parallel changes in large-scale software development: an observational case study. ACM TOSEM, 10:308–337, July 2001.
6. Anita Sarma. A survey of collaborative tools in software development. Technical Report UCI-ISR-05-3, University of California, Irvine, Institute for Software Research, 2005.
7. Anita Sarma, Zahra Noroozi, and André van der Hoek. Palantír: raising awareness among configuration management workspaces. In ICSE, pages 444–454, Portland, OR, May 2003.
8. Prasun Dewan and Rajesh Hegde. Semi-synchronous conflict detection and resolution in asynchronous software development. In ECSCW, pages 159–178, Limerick, Ireland, Sep. 2007.

Thank You!

Appendix

Appendix: Stale Branches

Stale branches

`initial-setup` Updated 2 years ago by des-des

204 | 5

 New pull request



`authentication` Updated 2 years ago by des-des

204 | 13

 New pull request



`backup-on-install` Updated 9 months ago by SimonLab



130 | 0

 New pull request



`oauth` Updated 8 months ago by RobStallion



6 | 5

 New pull request



Appendix: Git Blame

atom / .travis.yml 

Newer  Older

100644 | 63 lines (53 sloc) | 1.26 KB

Raw Normal view History

 	2 years ago		1 git:
			2 <code>depth: 10</code>
			3
 Only trigger branch builds on master...	5 months ago		4 branches:
			5 <code>only:</code>
			6 <code>- master</code>
			7 <code>- /^[0-9.]+--releases\$</code>
			8
 Fix janky .travis.yml config	2 years ago		9 matrix:
 Explicitly define build matrix	2 years ago		10 <code>include:</code>
 wip	2 years ago		11 <code>- os: linux</code>
 Update Travis manifest to use libsecr...	4 months ago		12 <code>dist: trusty</code>
 Revert "Use a more recent c++ toolc...	2 months ago		13 <code>env: NODE_VERSION=6.9.4 DISPLAY=:99.0 CC=clang CXX=clang++ npm_config_clang=1</code>
 Exclude core and package specs on ...	2 years ago		14
 Update Travis manifest to use libsecr...	4 months ago		15 <code>sudo: required</code>
 Switch to containerized infrastructure	2 years ago		16
 Run main process tests on Linux	a year ago		17 before_install:
			18 <code>- "/sbin/start-stop-daemon --start --quiet --pidfile /tmp/custom_xvfb_99.pid --make-pidfile --background --exec /usr/bin/Xvfb -- :9</code>
			19

Appendix: Crystal Github Repo

brunyuriy / crystalvc

Watch 3 Star 4 Fork 5

Code Issues 21 Pull requests 0 Projects 0 Wiki Insights

Automatically exported from code.google.com/p/crystalvc

672 commits 1 branch 0 releases 4 contributors View license

Branch: master New pull request Create new file Upload files Find File Clone or download

brunyuriy added a lincens Latest commit 2275e69 on Jul 23, 2015

.settings	Updated comment for all the test classes	7 years ago
hgkit-src/org/freehg/hgkit	Updated comment for all the test classes	7 years ago
lib-exc	Updated comment for all the test classes	7 years ago
lib	Updated comment for all the test classes	7 years ago
releases	added a release	4 years ago
src/crystal	Created git log parser test. And still half way to make crystal grace...	7 years ago
test-src/crystal	Updated HgLogParserTest	7 years ago
testDataFile	Updated comment for all the test classes	7 years ago
webpage	Updated comment for all the test classes	7 years ago
.classpath	Updated comment for all the test classes	7 years ago
.fatjar	Updated comment for all the test classes	7 years ago
.gitignore	renamed .hgignore to .gitignore	4 years ago
.project	Updated comment for all the test classes	7 years ago
.project	Updated comment for all the test classes	7 years ago
LICENSE.md	added a lincens	4 years ago
README.md	fixed a link	4 years ago
ToDo.txt	Updated comment for all the test classes	7 years ago
build.xml	Updated comment for all the test classes	7 years ago
coreUpdate.txt	Updated comment for all the test classes	7 years ago
design.txt	Updated comment for all the test classes	7 years ago

Appendix: FASTDash Tool

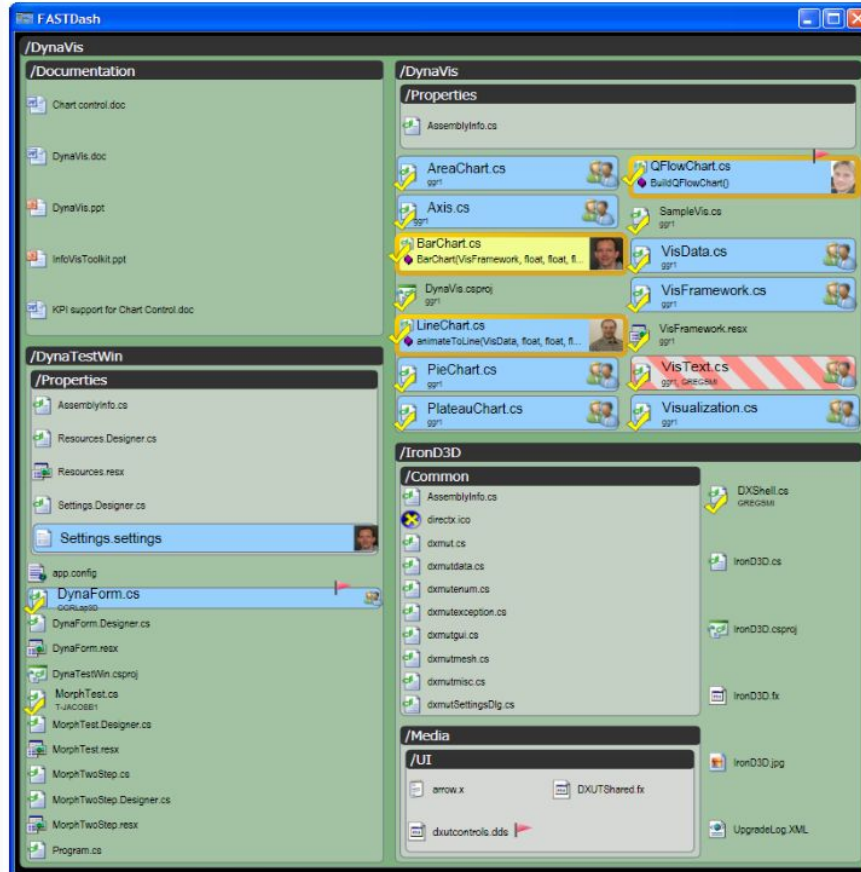


Fig: Three programmers using FASTDash

Appendix: Syde Tool

The image displays the Syde tool interface, which is used for managing code conflicts in a distributed system. The interface is divided into several sections:

- Class Hierarchy (3):** A tree view showing the project's class structure. The root is `Game`, which includes `MainPanel`, `GamePanel`, `Level`, `DefaultBall`, `InteractionPanel`, and `GamePanel` (multiple instances).
- Conflict View (4):** A bar chart showing the number of conflicts for each class. The `Game` class has the highest number of conflicts, followed by `GamePanel` and `Level`.
- Code Editor (5):** A window showing the source code for the `NotifierFacade` class. The code includes a `getChangeAlert` method that returns a `ChangeAlert` object.
- Console (6):** A window showing the output of the tool, including messages about version conflicts and the author of the changes.

```
Entity      Author      Message
-----
ch.unisi.lu.sydeserver.notifier.NotifierFacade    anja      An older version (159) is under edition.
ch.unisi.lu.sydeserver.notifier.NotifierFacade.addChangeAlert(QIOperation.) anja      An older version (159) is under edition.
ch.unisi.lu.sydeserver.notifier.NotifierFacade.addChangeAlert(QIOperation.) anja      Entity has been concurrently added/changed, but not
ch.unisi.lu.sydeserver.notifier.ChangeAlertManager anja      An older version (159) is under edition.
ch.unisi.lu.sydeserver.notifier.ChangeAlertManager anja      An older version (159) is under edition.
```

```
49: Entity has been concurrently added/changed, but not committed. Author: anja
50:   * Gets a new ChangeAlert for a given user
51:   *
52:   * @return ChangeAlert object
53:   */
54: public ChangeAlert getChangeAlert(String author, long timestamp)
55:     throws RemoteException {
56:     return new NotifierFacade().getChangeAlert(author, timestamp);
57: }
```

Fig: Syde Tool

Appendix: Palantír Tool

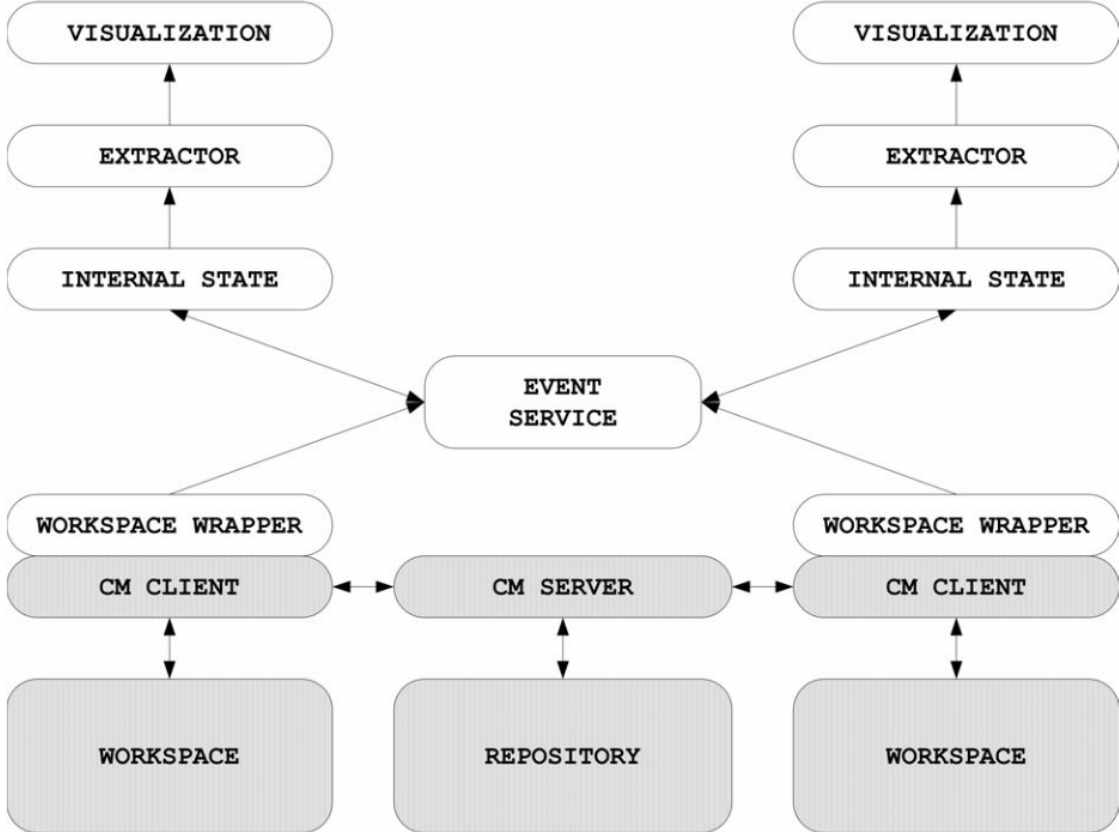


Fig: Palantír Tool Architecture

Appendix: Safe-Commit

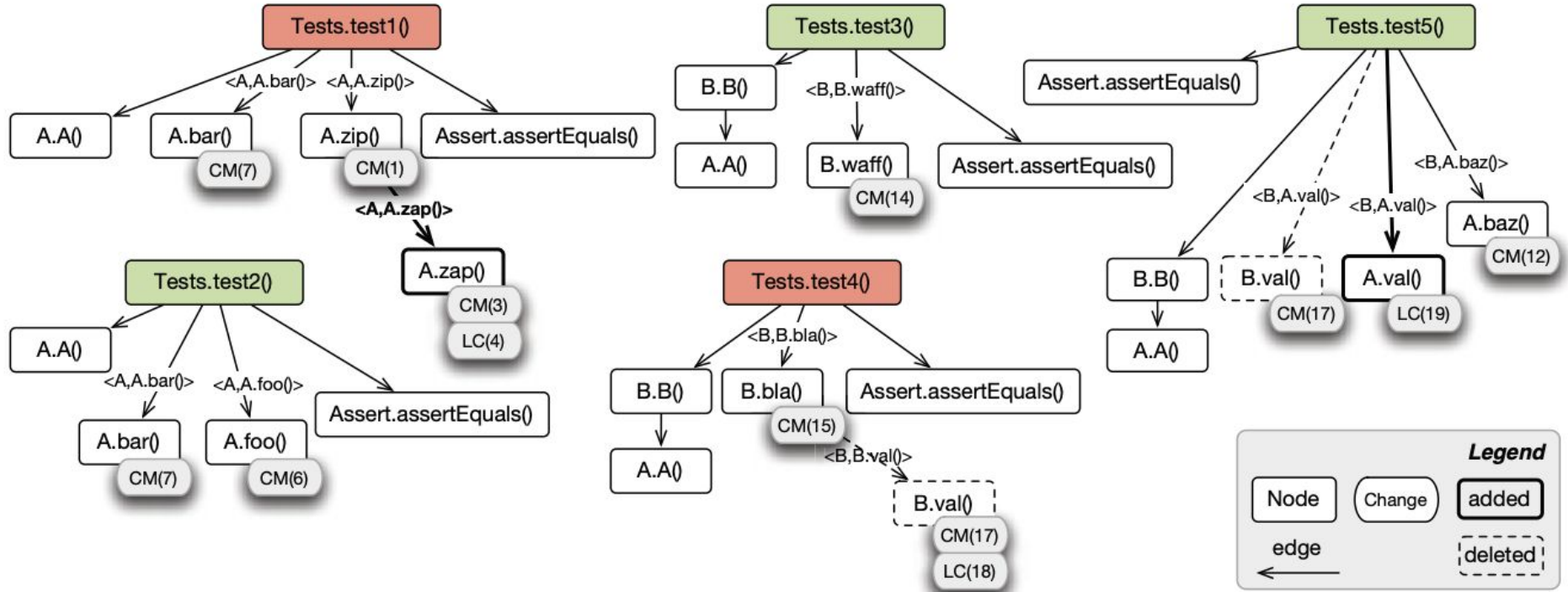


Fig: Call-graph analysis between original and edited versions