



An Empirical Examination of the Relationship Between Code Smells and Merge Conflicts

Iftekhhar Ahmed, **Caius Brindescu**, Umme Ayda Mannan, Carlos Jensen, Anita Sarma



Oregon State
University

Software development is a non-trivial activity

It is a mix of *social* and *technical* factors.

Oftentimes, things go wrong, and the observable impacts are bugs, merge conflicts etc.

While some factors have been studied in isolation, we want to look at them together, and study their *interaction*.

Merge conflicts

Merge conflicts

Not 4.7 for sure. I need to see such a beast in action before incorporating.



 **Some checks were not successful** [Hide all checks](#)
1 errored and 1 successful checks

	 continuous-integration/travis-ci/pr — The Travis CI build could not complete ... Details
	 continuous-integration/appveyor/pr — AppVeyor build succeeded Details

 **This branch has conflicts that must be resolved**
Only those with [write access](#) to this repository can merge pull requests.

Conflicting files

`runtime-testsuite/test/org/antlr/v4/test/runtime/java/api/TestVisitors.java`

Merge conflicts

Not 4.7 for sure. I need to see such a beast in action before incorporating.

  **Some checks were not successful** [Hide all checks](#)
1 errored and 1 successful checks

-   continuous-integration/travis-ci/pr — The Travis CI build could not complete ... [Details](#)
-   continuous-integration/appveyor/pr — AppVeyor build succeeded [Details](#)

 **This branch has conflicts that must be resolved**
Only those files with conflicts are shown. Files without conflicts are hidden.

Conflicts:
runtime-t

```
Adso:molgenis caius$ git merge feature/global-exception-handling
Auto-merging molgenis-web/src/test/java/org/molgenis/web/ErrorMessageResponseTest.java
Auto-merging molgenis-web/src/main/java/org/molgenis/web/ErrorMessageResponse.java
Auto-merging molgenis-security/src/main/java/org/molgenis/security/MolgenisWebAppSecurityConfig.java
CONFLICT (content): Merge conflict in molgenis-security/src/main/java/org/molgenis/security/MolgenisWebAppSecurityConfig.java
Auto-merging molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/controller/DataExplorerController.java
CONFLICT (content): Merge conflict in molgenis-dataexplorer/src/main/java/org/molgenis/dataexplorer/controller/DataExplorerController.java
Auto-merging molgenis-data-rest/src/main/java/org/molgenis/data/rest/v2/RestControllerV2.java
CONFLICT (content): Merge conflict in molgenis-data-rest/src/main/java/org/molgenis/data/rest/v2/RestControllerV2.java
Auto-merging molgenis-core-ui/src/main/java/org/molgenis/ui/admin/user/UserAccountController.java
Automatic merge failed; fix conflicts and then commit the result.
Adso:molgenis caius$
```

Merge Conflicts

Conflicts are a challenge to collaborative development.

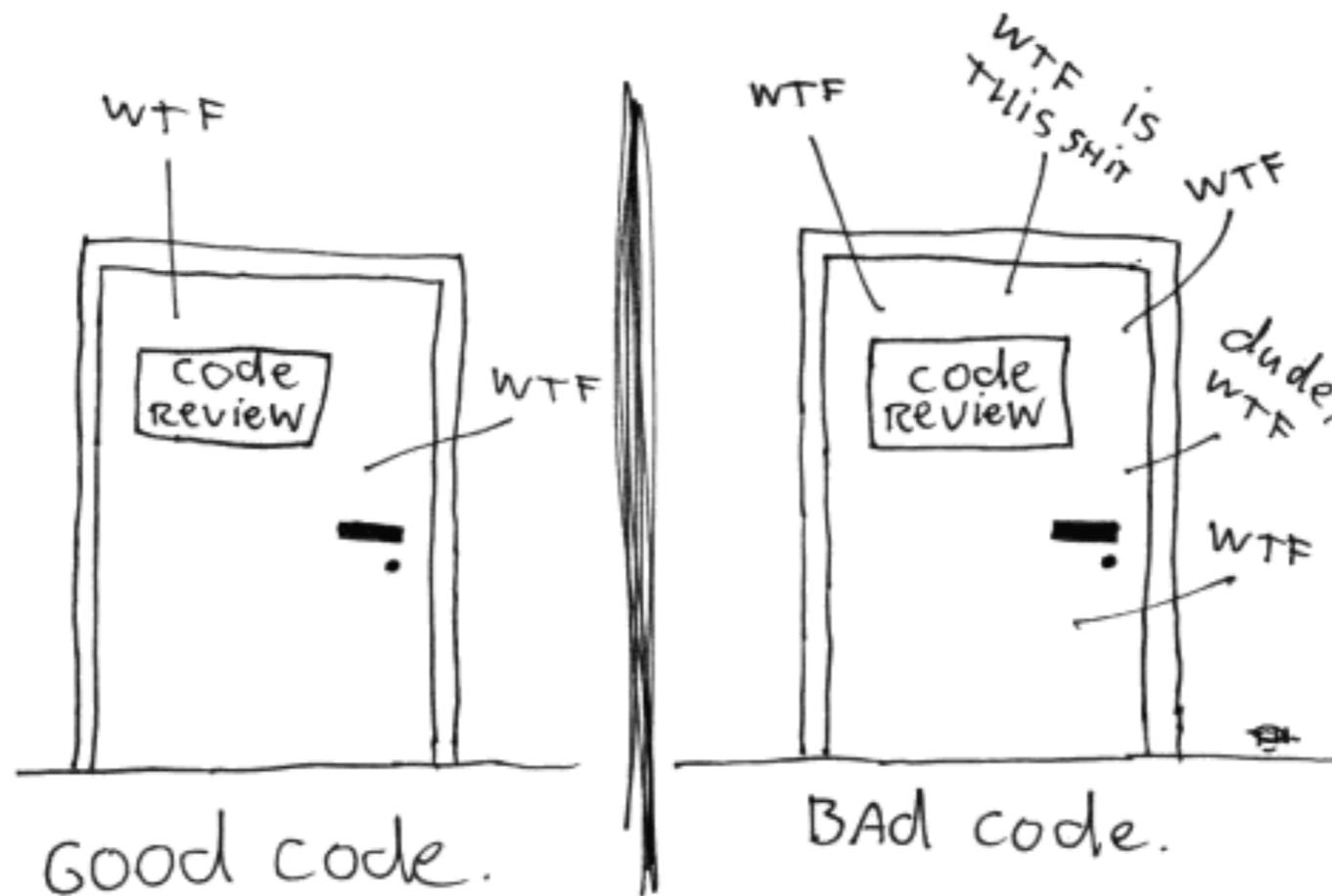
A developer has to interrupt their work, and focus on solving it before they can move on.

It is an ***immediate concern*** for the developer.

They are a common occurrence. In our corpus we find that ***over 19% of merges result in a conflict.***

Code Smells

The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



(c) 2008 Focus Shift

Code Smells

Code smells are ***symptoms*** of poor design or implementation choices.

Code smells are associated with ***future*** maintainability problems.

What Are We Missing?

By bringing these 2 factors together, we can study their interplay, in a more *holistic* way.

Do problems like code smells or merge conflicts *compound* each other?

How do they manifest themselves in the end product?

Our contribution

RQ1: Do program elements that are involved in merge conflicts contain more code smells?

RQ2: Which code smells are more associated with merge conflicts?

RQ3: Do code smells associated with merge conflicts affect the quality of the resulting code?

Corpus

143 Java projects

556,911 commits.

36,111 merges

6,979 (19.32%) merges resulted in a conflict.

RQ1: Do program elements that are involved in merge conflicts contain more code smells?

Code smells

We used the *inFusion* code smell detector.

It detected 22 types of code smells in our corpus.

We used inFusion to identify code smells for each merge conflict.

Code smells and merge conflicts

Are program elements that have more code smells more likely to be involved in merge conflicts?

Program elements involved in a merge conflict have an average of **6.54** smells, while those that don't have an average of **1.92**.

Code smells and merge conflicts

Are program elements that have more code smells more likely to be involved in merge conflicts?

Program elements involved in a merge conflict have an average of **6.54** smells, while those that don't have an average of **1.92**.

Elements involved in a conflict contain 3x more code smells than element not involved in a conflict.

RQ2: Which code smells are more associated with merge conflicts?

Identifying Merge Conflicts

We look back in history and identify all the merge conflicts.

We split the conflicts into 2 categories:

Semantic: solving the conflict requires understanding and changing the logic;

Non-Semantic: e.g. formatting, adding a method at the end of a line.

Identifying Merge Conflicts

Figuring out the classification requires a *human touch*.

We manually classified 606 conflicts, and then trained a machine learning classifier.

We used 24 features collected for each conflict.

The classifier achieves a precision of 75%

Identifying Merge Conflicts

	# of conflicts	% of total (classified)
Semantic	5,250	75.23%
Non-semantic	1,729	24.77%

Identifying Merge Conflicts

	# of conflicts	% of total (classified)
Semantic	5,250	75.23%
Non-semantic	1,729	24.77%

Most conflicts have an underlying semantic cause.

Are all code smells equally problematic?

Smell	Correlation with # of conflicts
God Class	0.18
Internal Duplication	0.17
Distorted Hierarchy	0.13

Are all code smells equally problematic?

Smell	Correlation with # of conflicts
God Class	0.18
Internal Duplication	0.17
Distorted Hierarchy	0.13

These 3 smells are indicative of bad code structure, at a class level.

What about semantic conflicts?

Smell	Correlation with # of Semantic Conflicts	Odds ratio
Internal Duplication	0.07	1.55
Blob Operation	0.05	1.77

What about semantic conflicts?

Smell	Correlation with # of Semantic Conflicts	Odds ratio
Internal Duplication	0.07	1.55
Blob Operation	0.05	1.77

Methods with code smells are more likely to be involved in Semantic merge conflicts

What does this mean?

Code smells are a ***symptom*** of bad design.

Merge conflicts are ***associated*** with code smells.

What does this mean?

Code smells are a ***symptom*** of bad design.

Merge conflicts are ***associated*** with code smells.

Code smells have an impact on the near future!

RQ3: Do code smells associated with merge conflicts affect the quality of the resulting code?

Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.

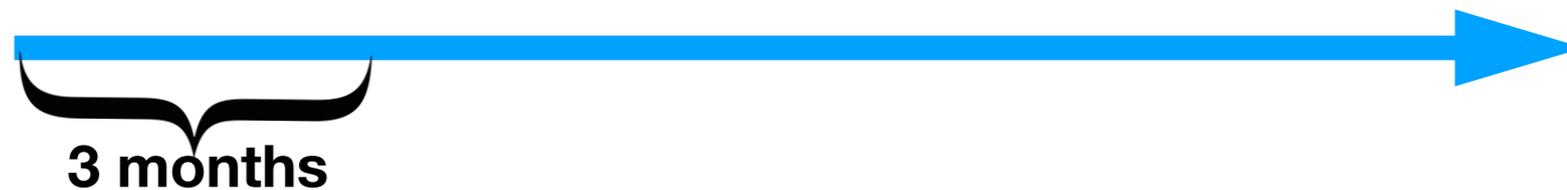
Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.



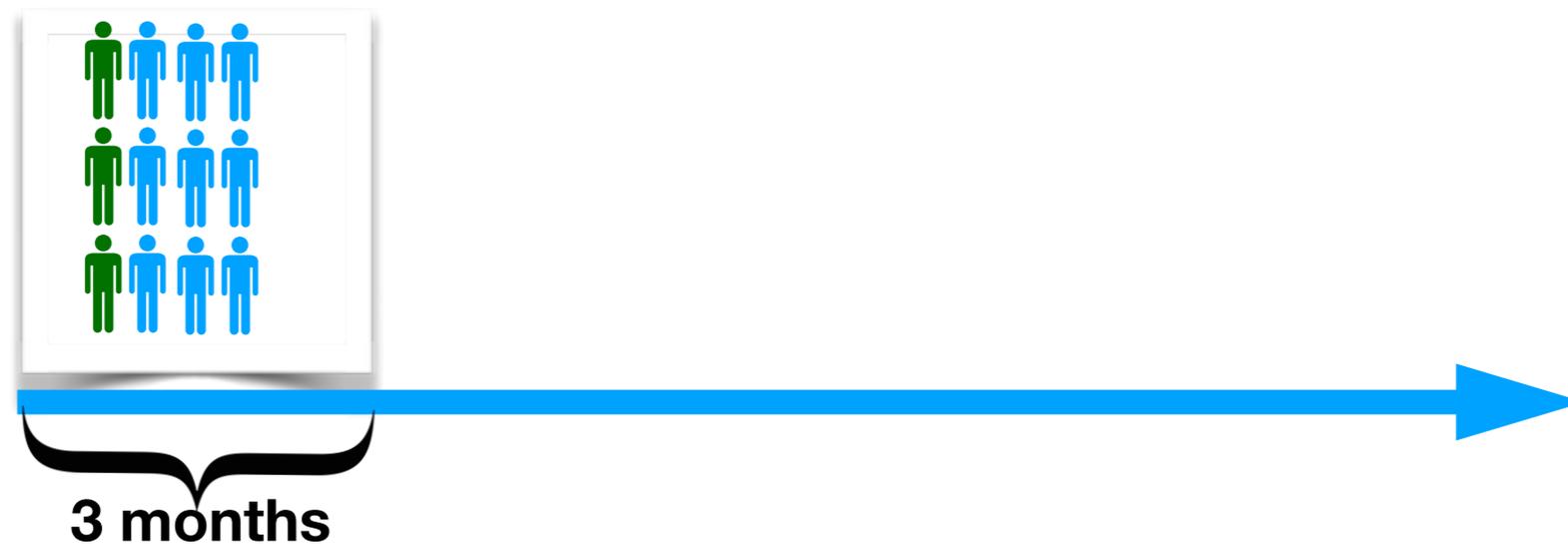
Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.



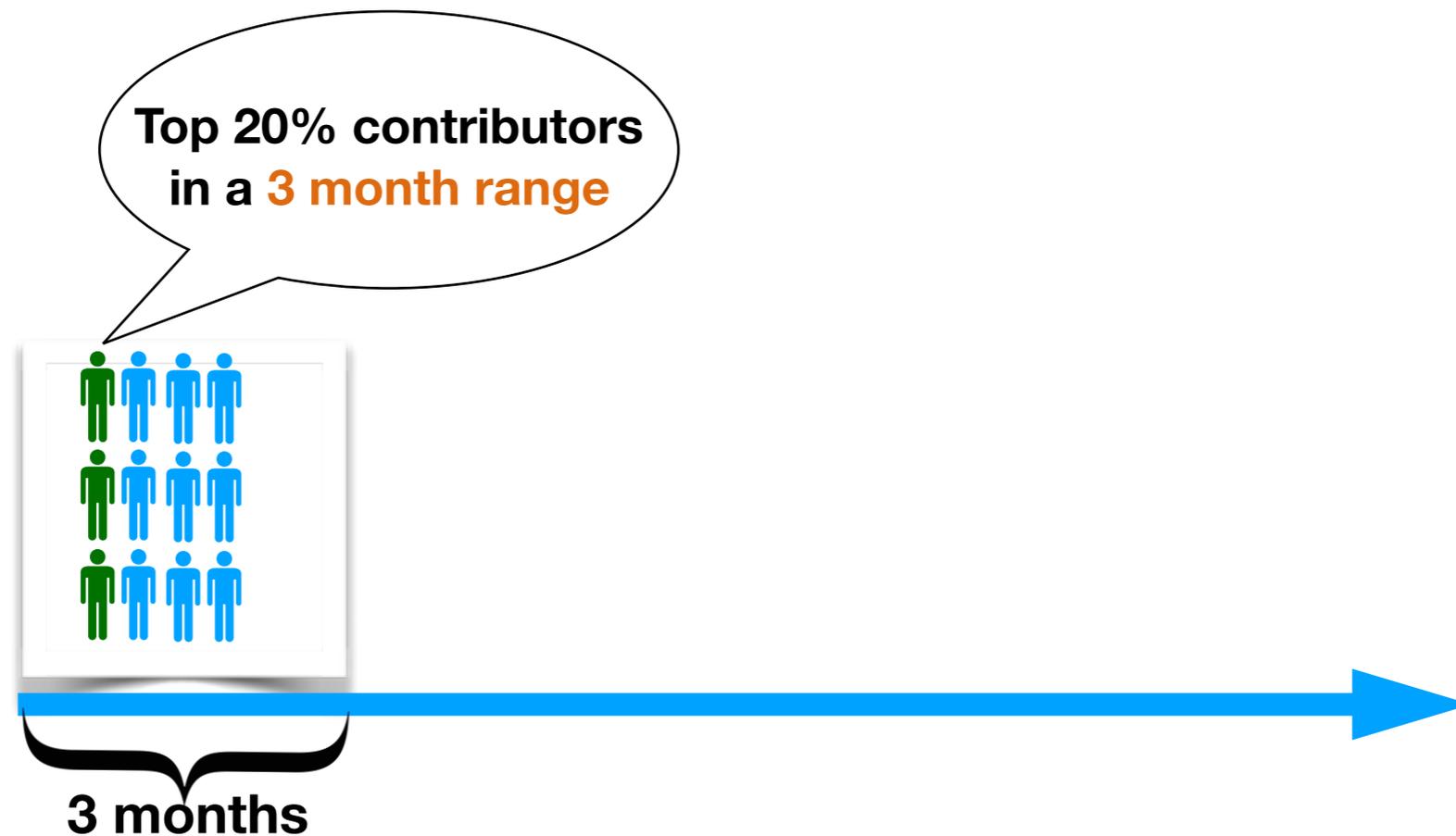
Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.



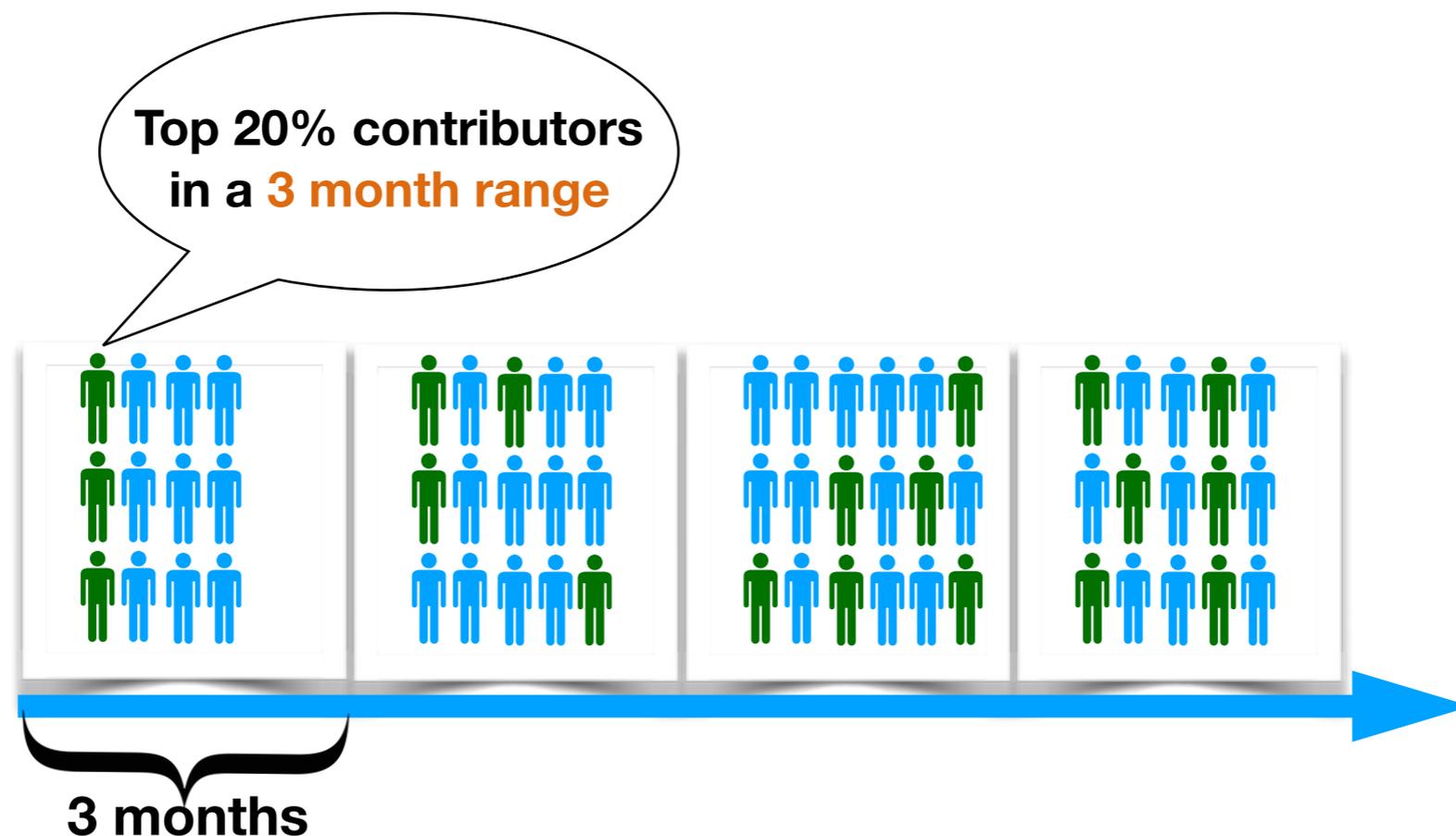
Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.



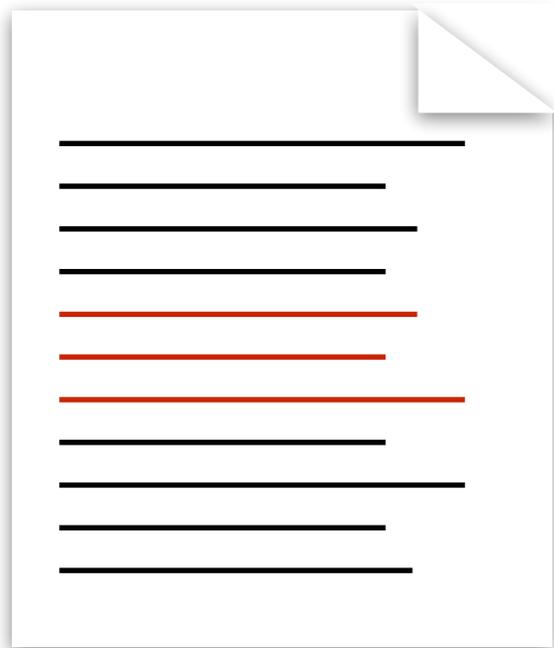
Author classification

Not all authors have the same experience level, we categorize authors as **core** and **non-core**.



Identifying buggy lines

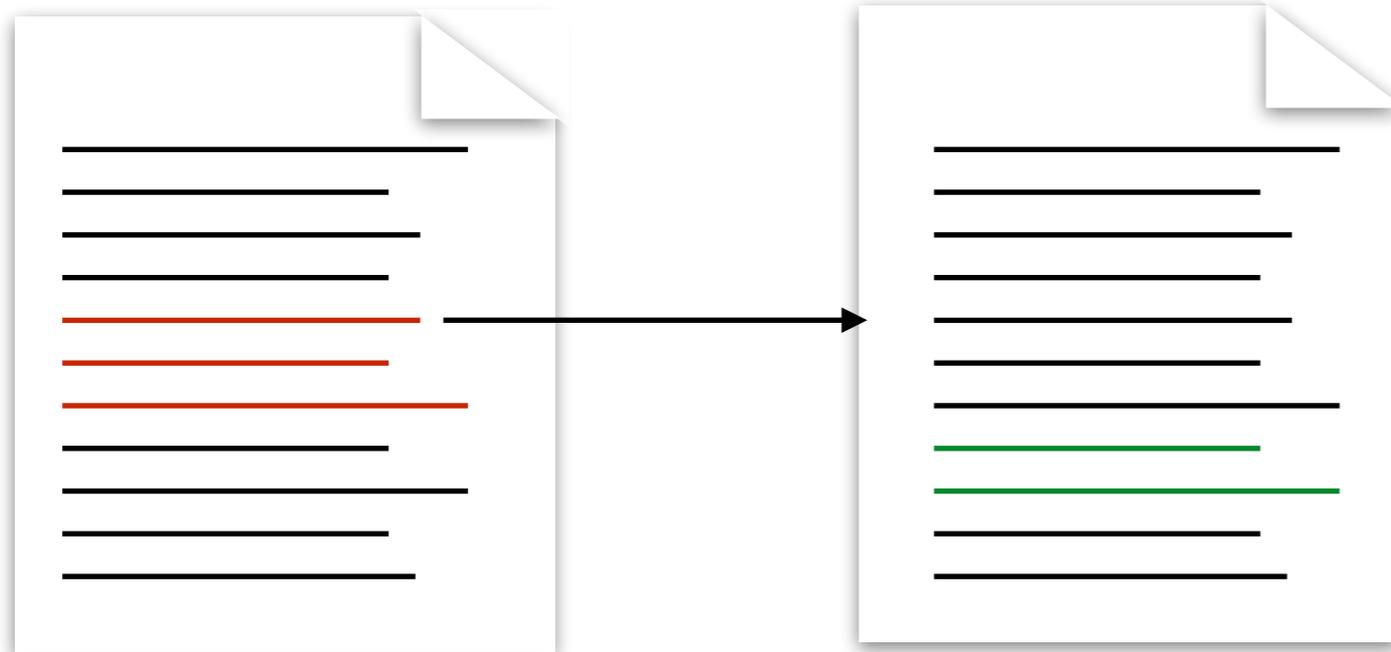
How do we determine if a merge conflict is associated with a future bug?



Conflicting lines

Identifying buggy lines

How do we determine if a merge conflict is associated with a future bug?



Conflicting lines

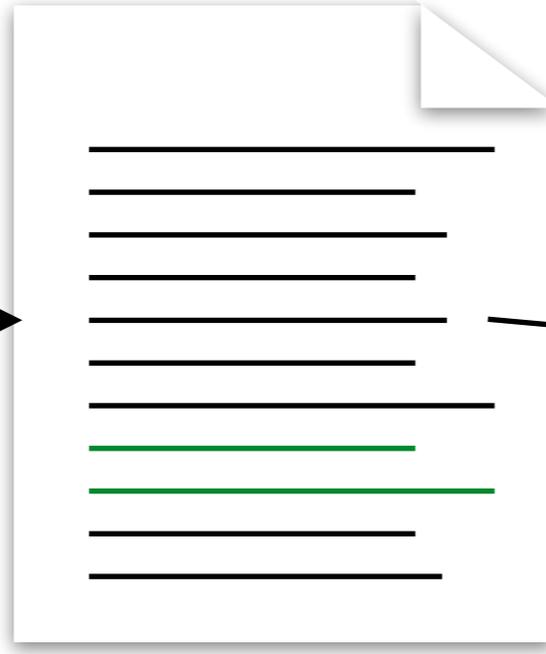
Commits that
do not touch the
line

Identifying buggy lines

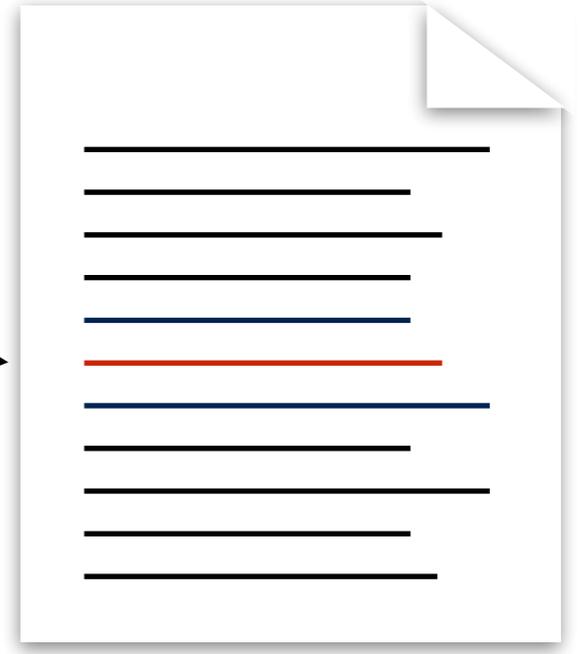
How do we determine if a merge conflict is associated with a future bug?



Conflicting lines



Commits that do not touch the line



First commit that touches the line

Identifying buggy lines

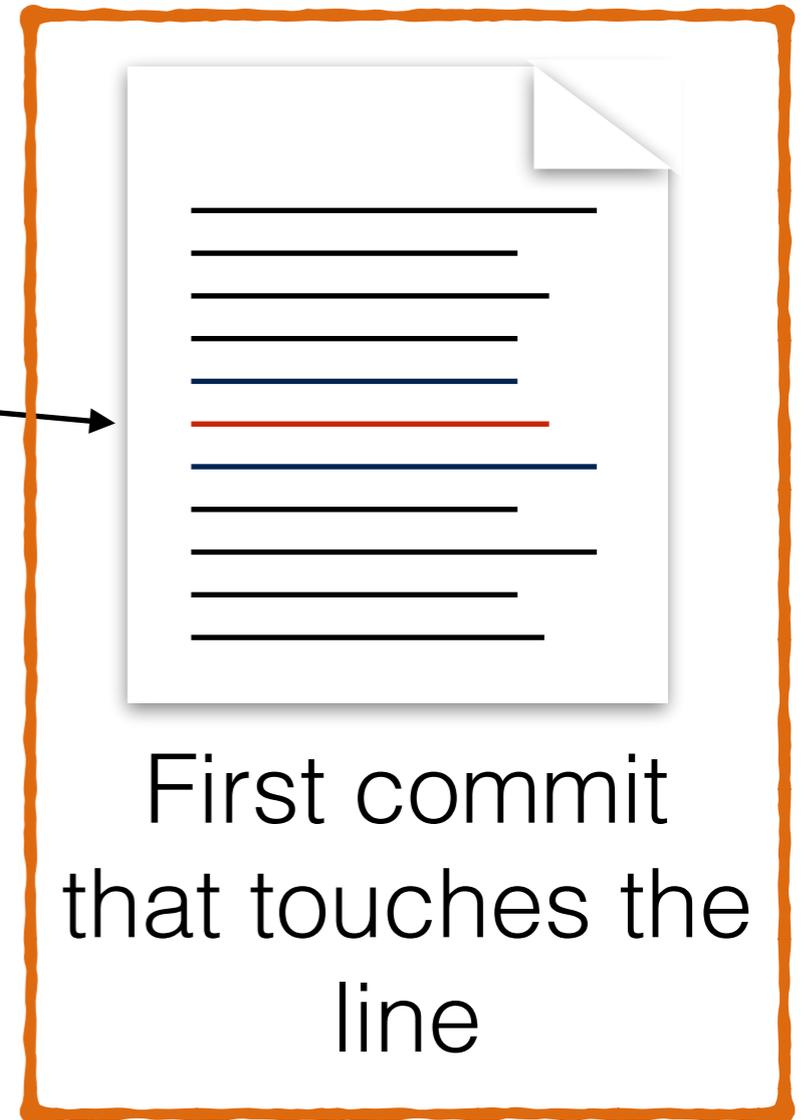
How do we determine if a merge conflict is associated with a future bug?



Conflicting lines



Commits that do not touch the line



First commit that touches the line

Identifying buggy lines

Whether the future commit is a bug fix was determined using a "bag-of-words" classification approach.

We trained a Naive-Bayes classifier with 1.500 manually classified commits.

We obtained a precision of 0.75 and a recall of 0.86.

What about the impact on bugs?

We analyzed lines that were involved in a merge conflict.

We used factors that have been showed to affect the bug proneness, and added the # of code smells and author type.

Factor	Estimate
In Deps	3.195
Out Deps	-0.053
Noncore author	-3.799
No. Authors	0.129
No. Classes	-0.373
No. Methods	0.244
AST diff	0.001
LOC diff	0.00002571
No. of Smells	0.427

Factor	Estimate
In Deps	3.195
Out Deps	-0.053
Noncore author	-3.799
No. Authors	0.129
No. Classes	-0.373
No. Methods	0.244
AST diff	0.001
LOC diff	0.00002571
No. of Smells	0.427

What does this mean?

The number of code smells is an indicator for bugginess, if the line is involved in a conflict.

The interaction between code smells and merge conflict has an effect on the final product!

Limitations

The precision of the classifiers;

Looking at code smells and bugs in isolation;

The 3 month period for identifying core contributors.



Conclusions

Over 75% of all conflicts are semantic in nature.

Methods that exhibit code smells are over 50% more likely to be involved in a semantic merge conflict.

When looking at lines involved in a conflict, code smells are an accurate predictor of bugginess.

This work was funded by NSF through grants IIS-1559657, and CCF-1560526, and by an IBM fellowship.