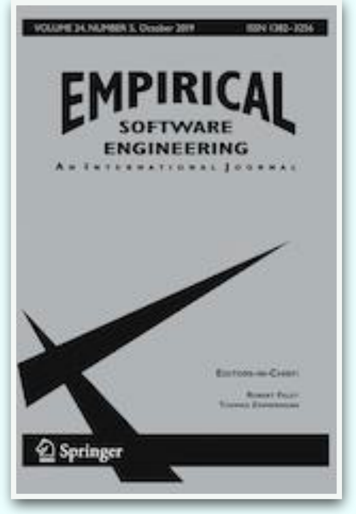
University of Zurich UZH

Swiss National Science Foundation


EMPIRICAL SOFTWARE ENGINEERING

Springer

FSE 25

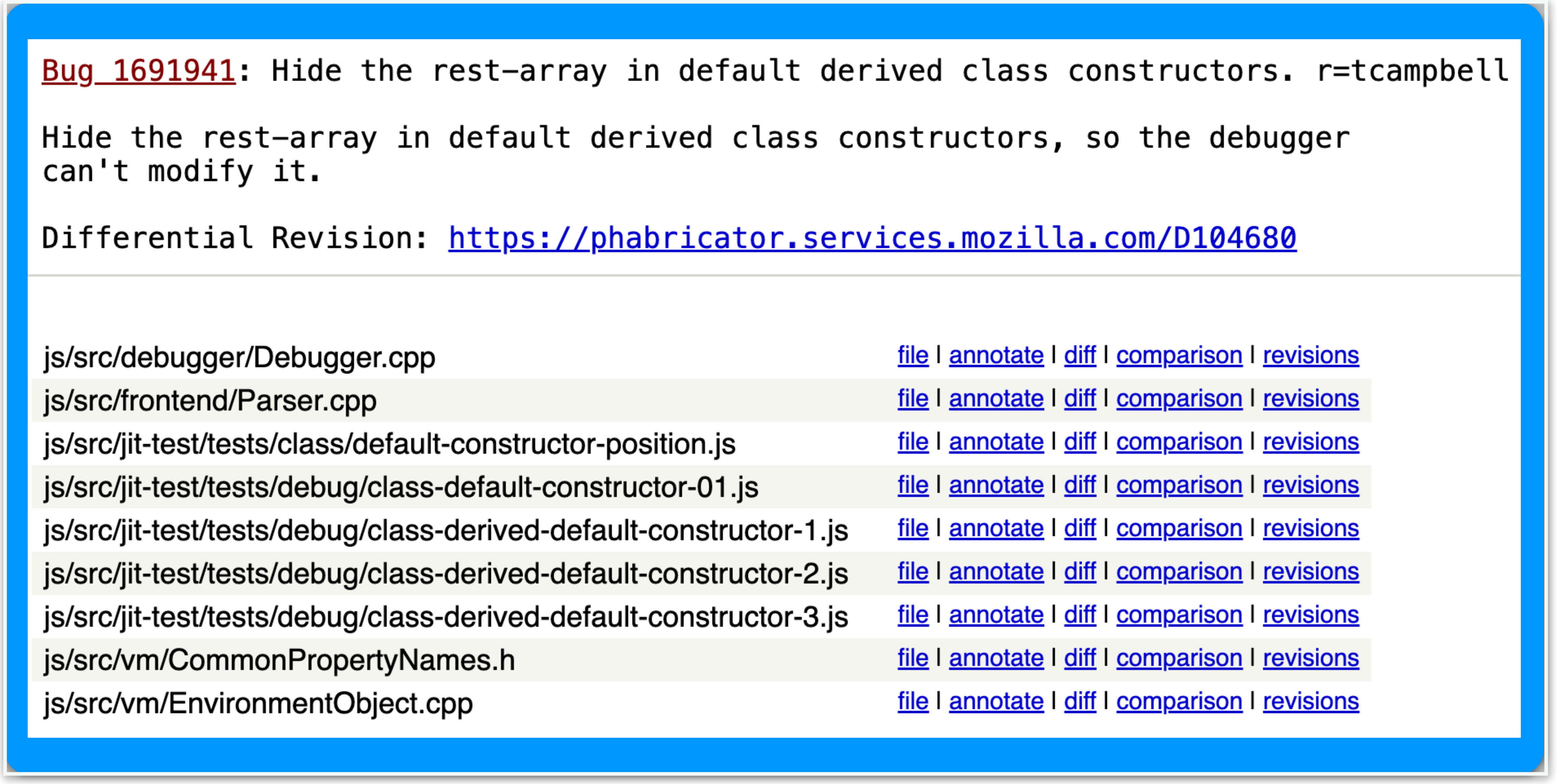
Pooja Rani, Fernando Petrulio, Alberto Bacchelli

University of Zurich, Switzerland

Bugs are everyday occurrences

```
Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger
can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680
```

```
js/src/debugger/Debugger.cpp                                      file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp                                        file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js       file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js       file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h                                   file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp                                   file | annotate | diff | comparison | revisions
```



```
Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger
can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680
```

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions

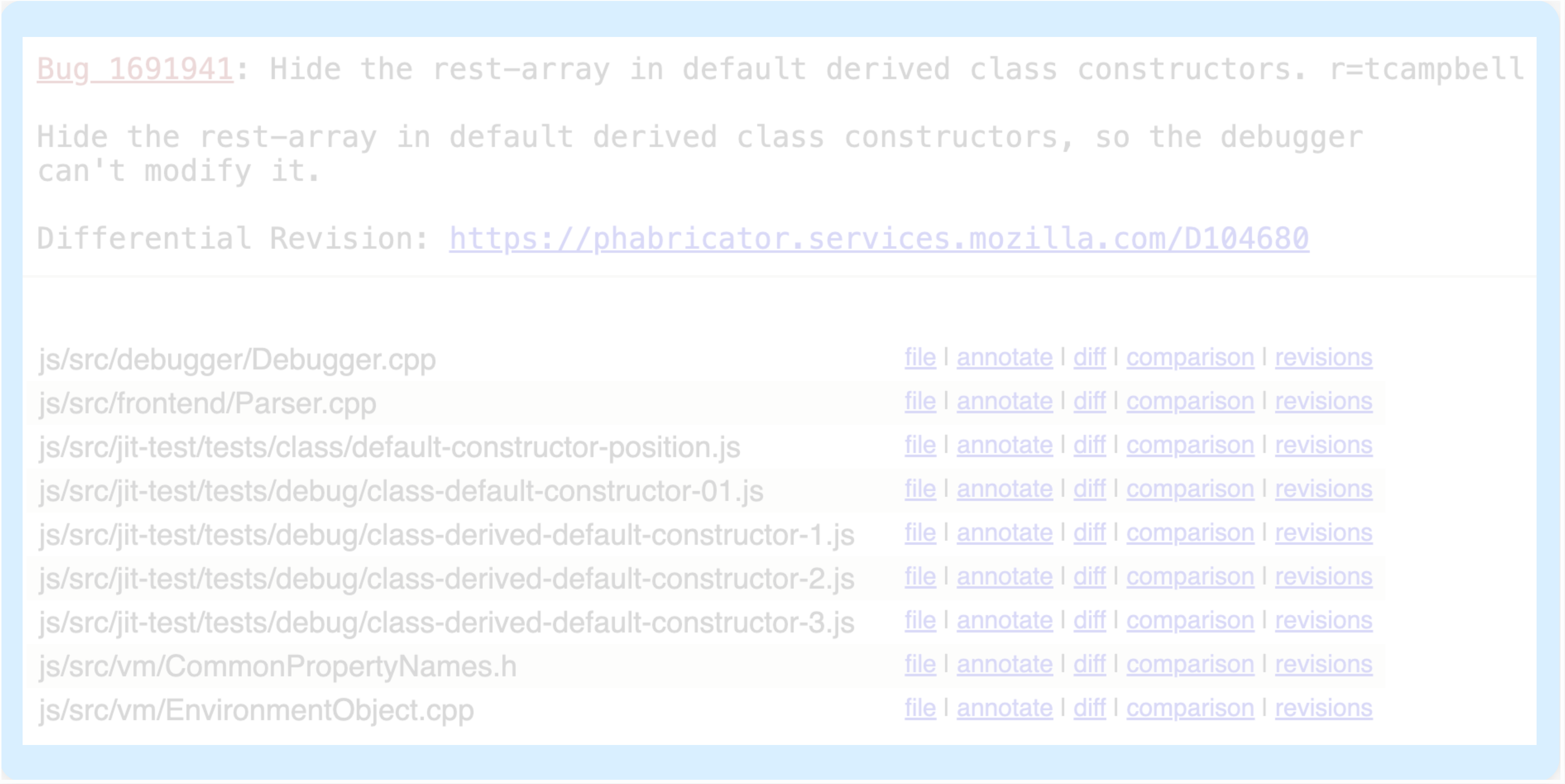
Bugs are costly to fix

Prevent before they happen

Test smarter, not harder

Build robust code

What code introduced this bug?



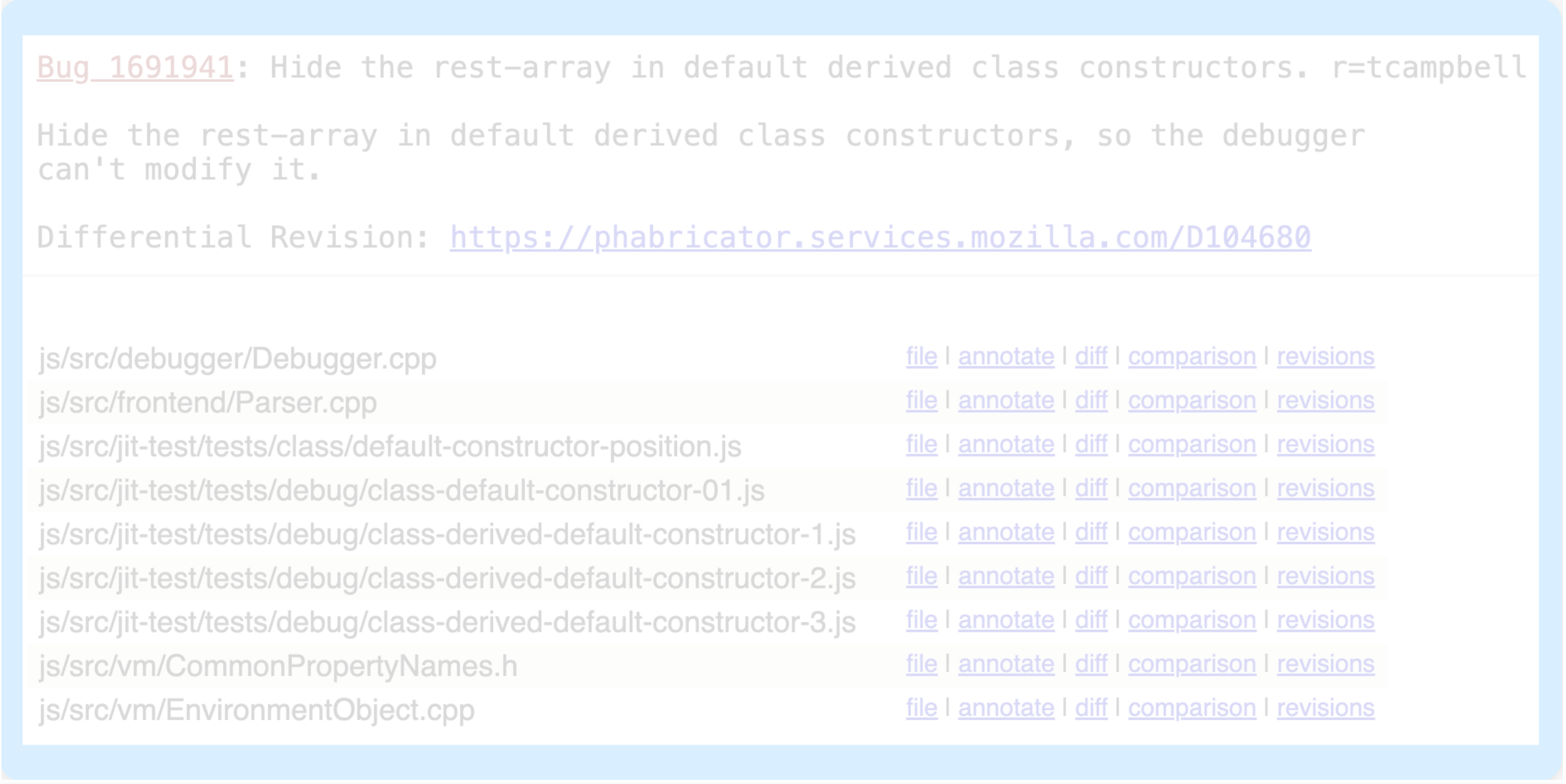
Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions

SZZ

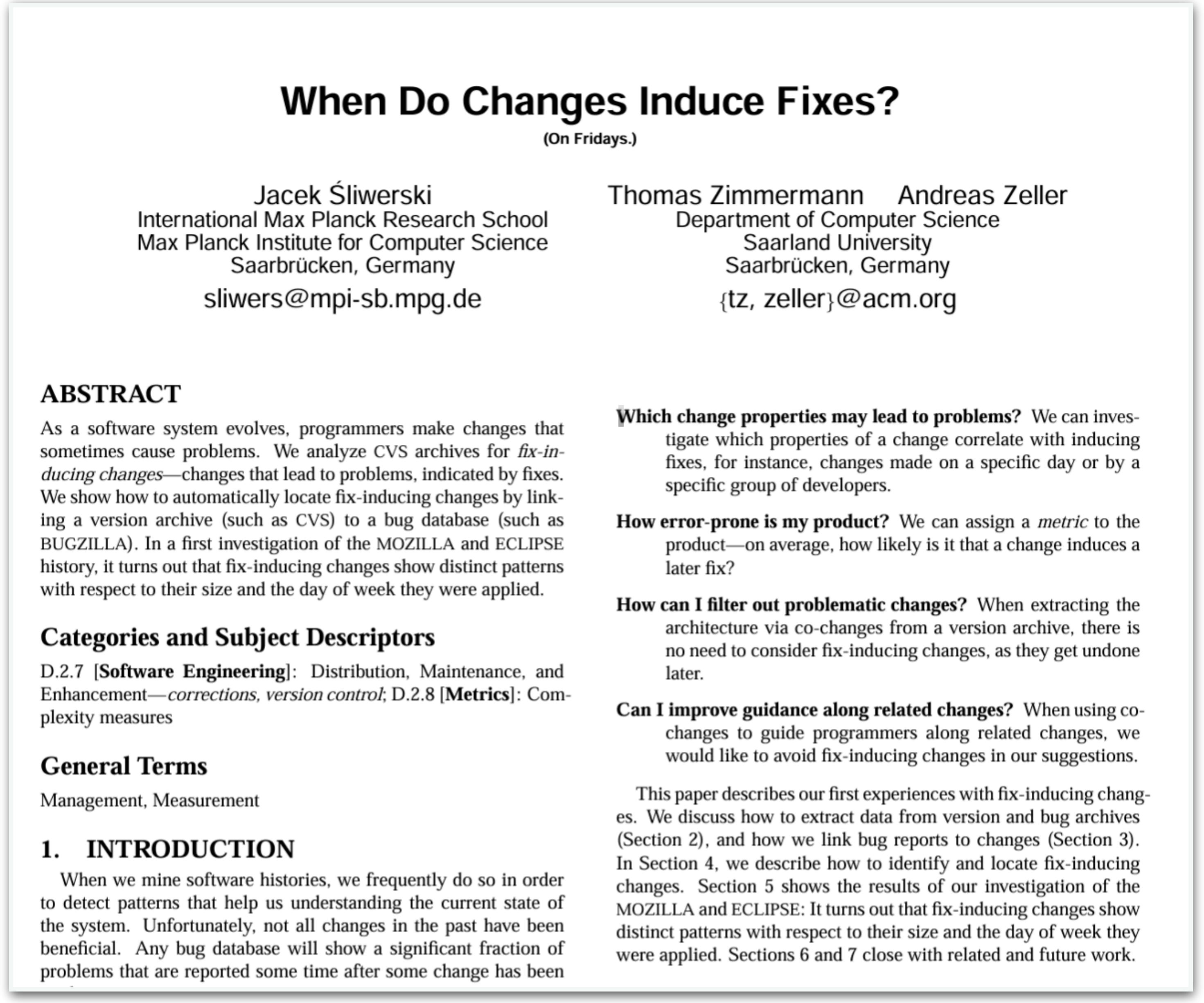


Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions



When Do Changes Induce Fixes?

(On Fridays.)

Jacek Śliwerski
International Max Planck Research School
Max Planck Institute for Computer Science
Saarbrücken, Germany
sliwers@mpi-sb.mpg.de

Thomas Zimmermann Andreas Zeller
Department of Computer Science
Saarland University
Saarbrücken, Germany
{tz, zeller}@acm.org

ABSTRACT

As a software system evolves, programmers make changes that sometimes cause problems. We analyze CVS archives for *fix-inducing changes*—changes that lead to problems, indicated by fixes. We show how to automatically locate fix-inducing changes by linking a version archive (such as CVS) to a bug database (such as BUGZILLA). In a first investigation of the MOZILLA and ECLIPSE history, it turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections, version control*; D.2.8 [**Metrics**]: Complexity measures

General Terms

Management, Measurement

1. INTRODUCTION

When we mine software histories, we frequently do so in order to detect patterns that help us understanding the current state of the system. Unfortunately, not all changes in the past have been beneficial. Any bug database will show a significant fraction of problems that are reported some time after some change has been

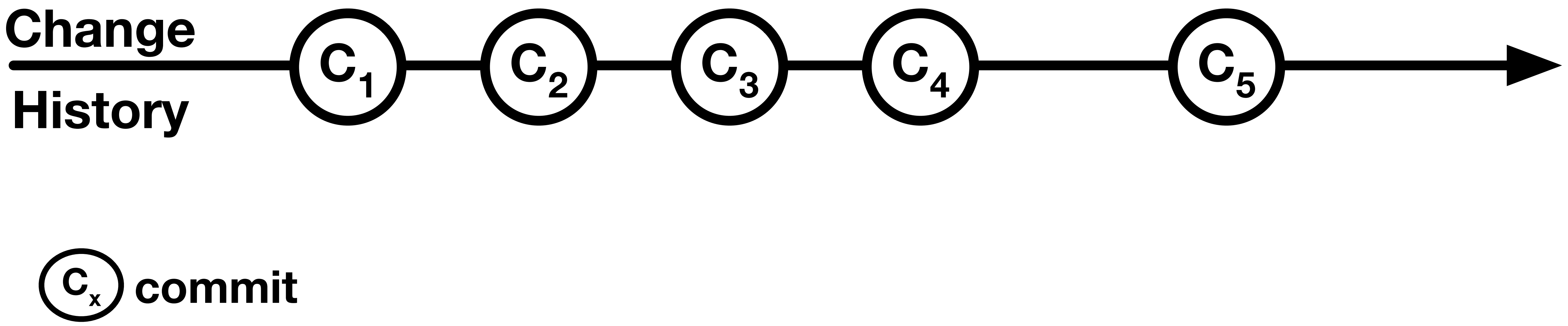
Which change properties may lead to problems? We can investigate which properties of a change correlate with inducing fixes, for instance, changes made on a specific day or by a specific group of developers.

How error-prone is my product? We can assign a *metric* to the product—on average, how likely is it that a change induces a later fix?

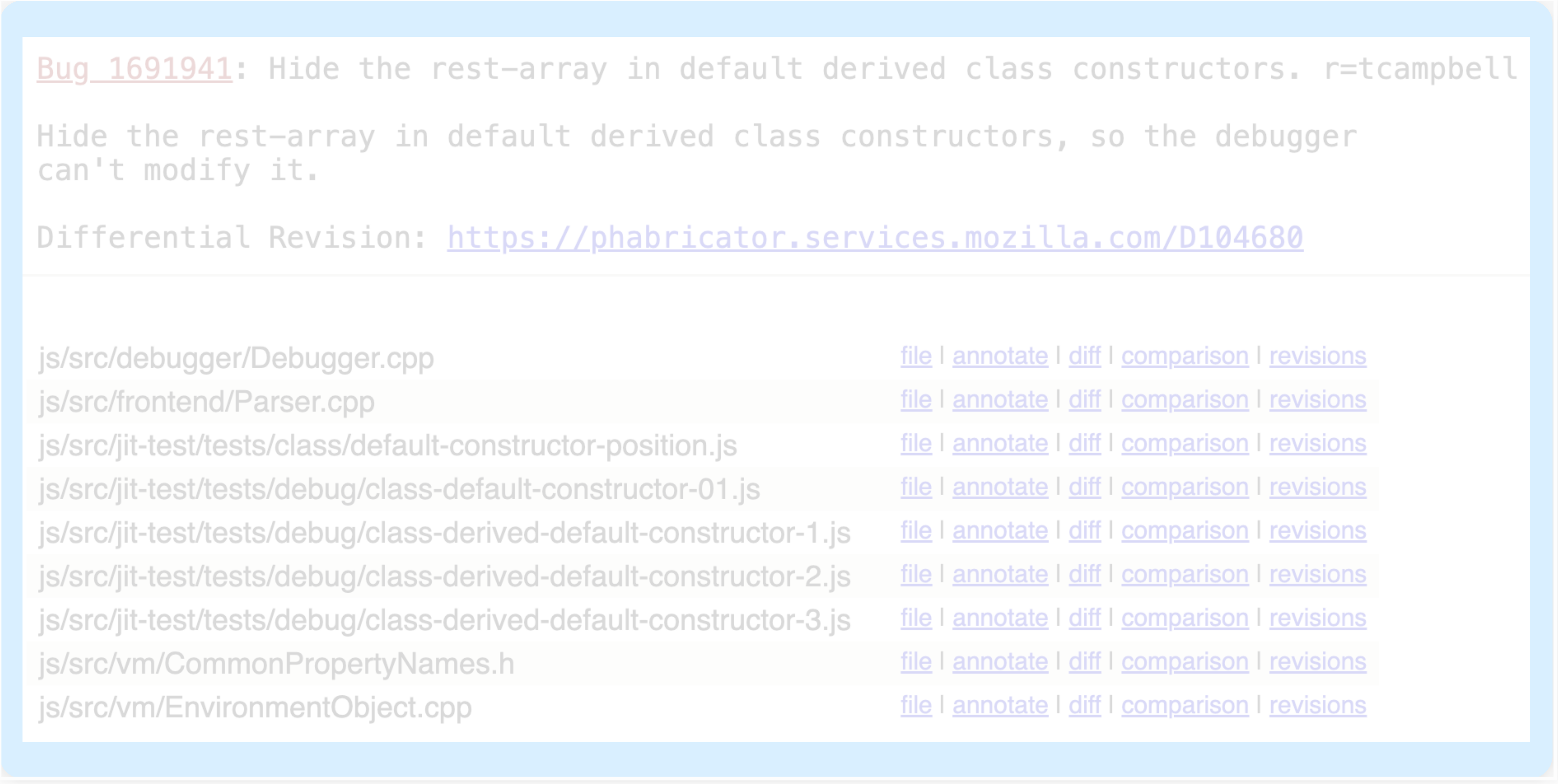
How can I filter out problematic changes? When extracting the architecture via co-changes from a version archive, there is no need to consider fix-inducing changes, as they get undone later.

Can I improve guidance along related changes? When using co-changes to guide programmers along related changes, we would like to avoid fix-inducing changes in our suggestions.

This paper describes our first experiences with fix-inducing changes. We discuss how to extract data from version and bug archives (Section 2), and how we link bug reports to changes (Section 3). In Section 4, we describe how to identify and locate fix-inducing changes. Section 5 shows the results of our investigation of the MOZILLA and ECLIPSE: It turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied. Sections 6 and 7 close with related and future work.



SZZ

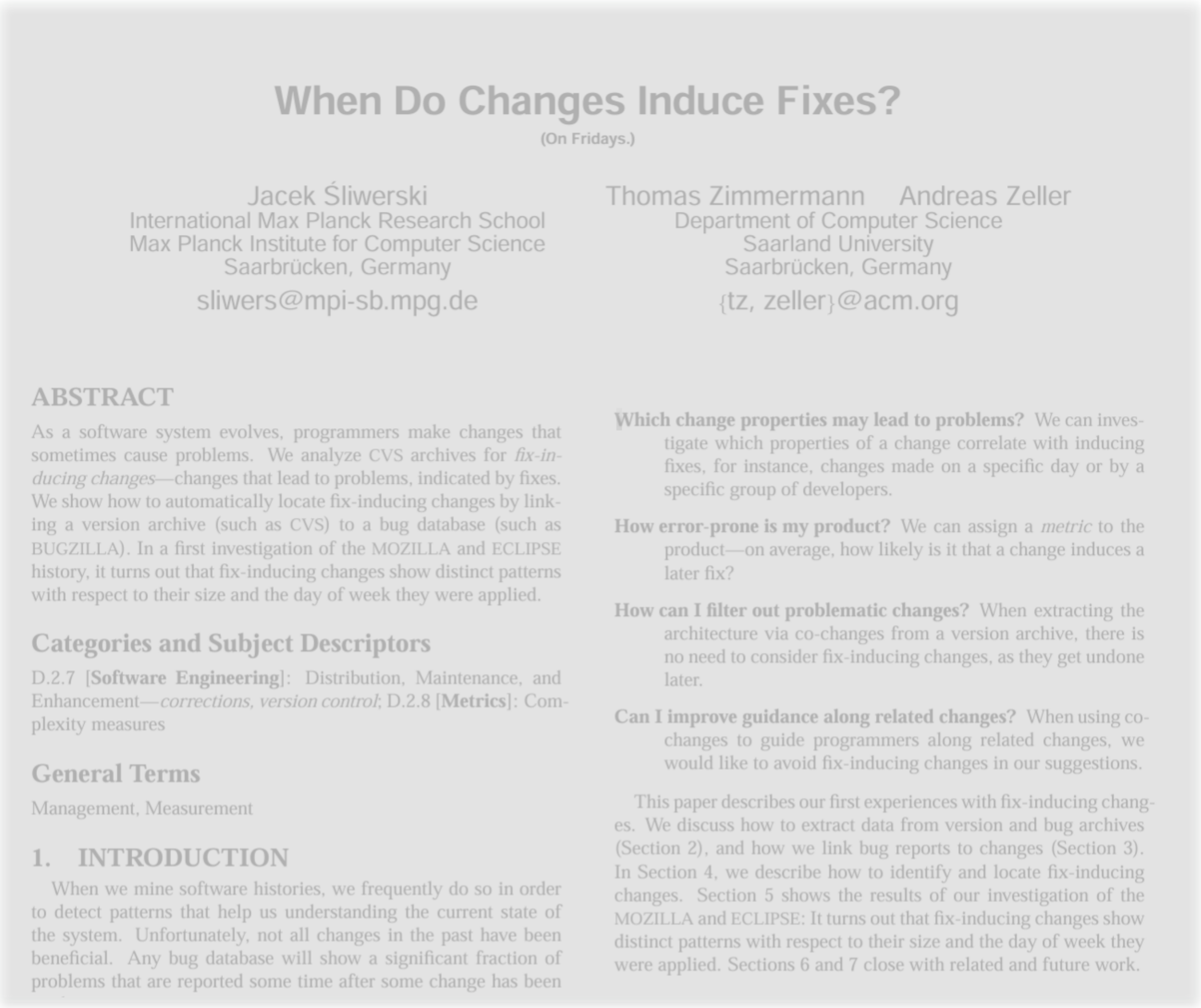


Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger
can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions



When Do Changes Induce Fixes?

(On Fridays.)

Jacek Śliwerski
International Max Planck Research School
Max Planck Institute for Computer Science
Saarbrücken, Germany
sliwers@mpi-sb.mpg.de

Thomas Zimmermann   Andreas Zeller
Department of Computer Science
Saarland University
Saarbrücken, Germany
{tz, zeller}@acm.org

ABSTRACT

As a software system evolves, programmers make changes that sometimes cause problems. We analyze CVS archives for *fix-inducing changes*—changes that lead to problems, indicated by fixes. We show how to automatically locate fix-inducing changes by linking a version archive (such as CVS) to a bug database (such as BUGZILLA). In a first investigation of the MOZILLA and ECLIPSE history, it turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections, version control*; D.2.8 [**Metrics**]: Complexity measures

General Terms

Management, Measurement

1. INTRODUCTION

When we mine software histories, we frequently do so in order to detect patterns that help us understanding the current state of the system. Unfortunately, not all changes in the past have been beneficial. Any bug database will show a significant fraction of problems that are reported some time after some change has been

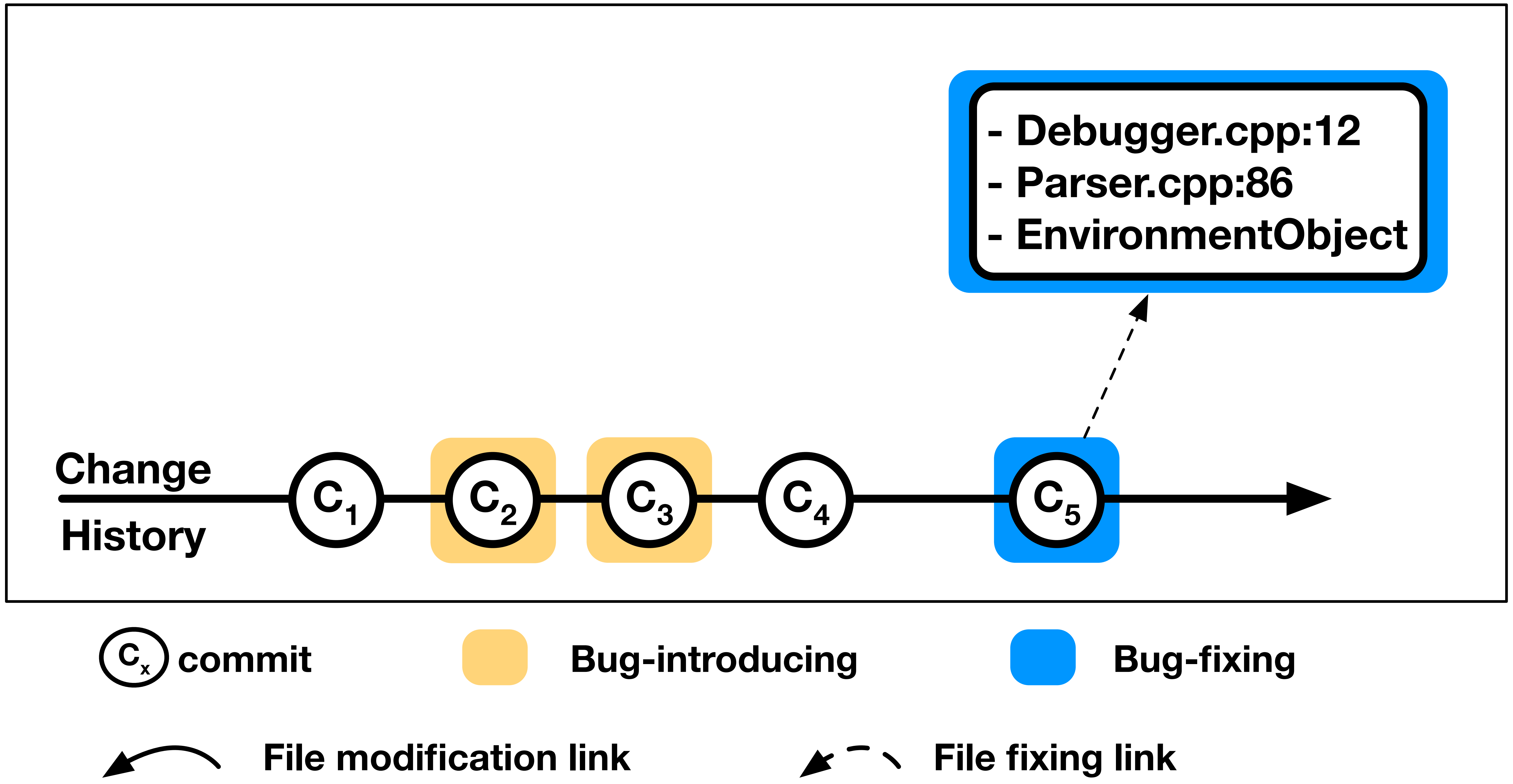
Which change properties may lead to problems? We can investigate which properties of a change correlate with inducing fixes, for instance, changes made on a specific day or by a specific group of developers.

How error-prone is my product? We can assign a *metric* to the product—on average, how likely is it that a change induces a later fix?

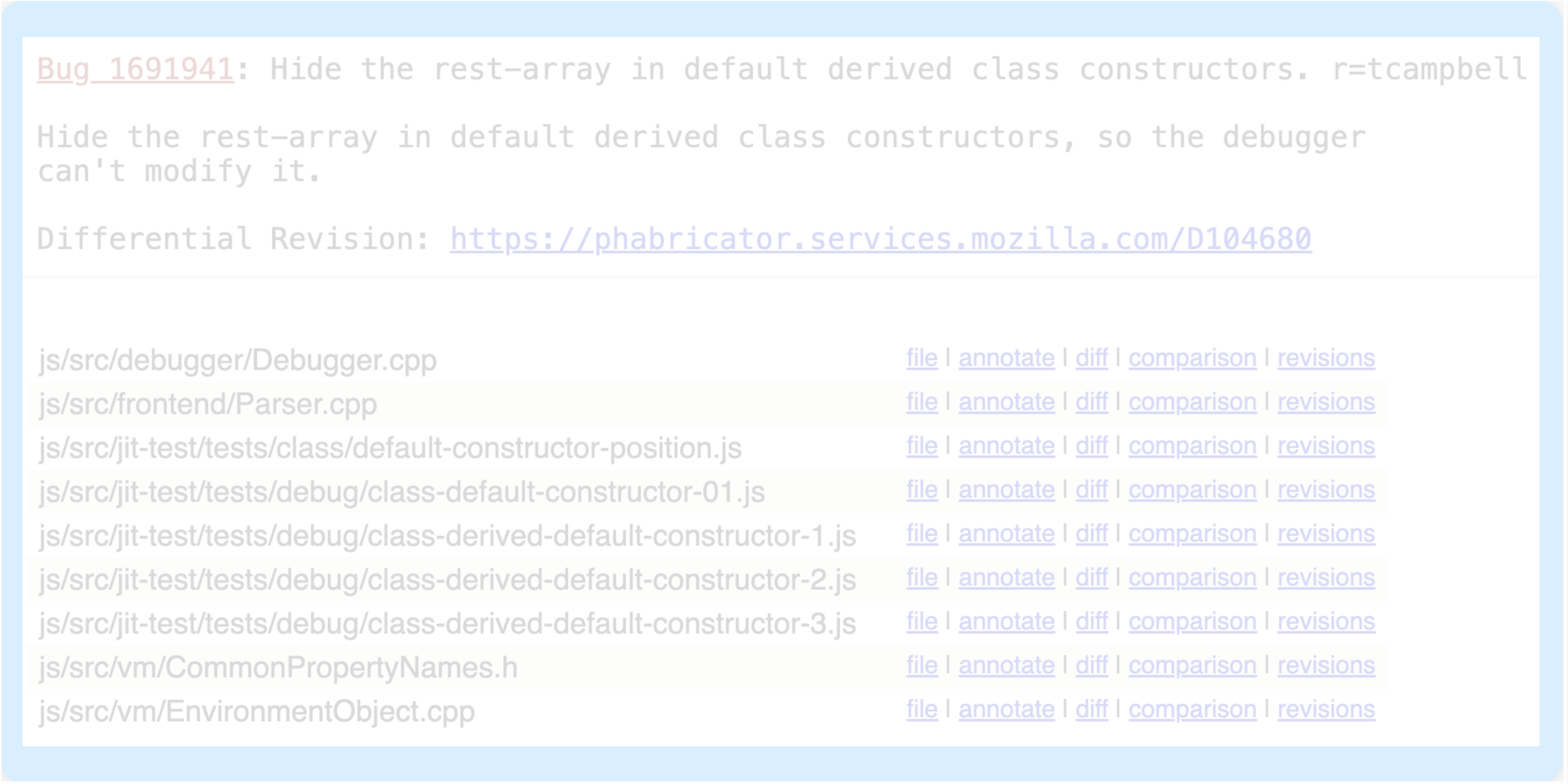
How can I filter out problematic changes? When extracting the architecture via co-changes from a version archive, there is no need to consider fix-inducing changes, as they get undone later.

Can I improve guidance along related changes? When using co-changes to guide programmers along related changes, we would like to avoid fix-inducing changes in our suggestions.

This paper describes our first experiences with fix-inducing changes. We discuss how to extract data from version and bug archives (Section 2), and how we link bug reports to changes (Section 3). In Section 4, we describe how to identify and locate fix-inducing changes. Section 5 shows the results of our investigation of the MOZILLA and ECLIPSE: It turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied. Sections 6 and 7 close with related and future work.



SZZ

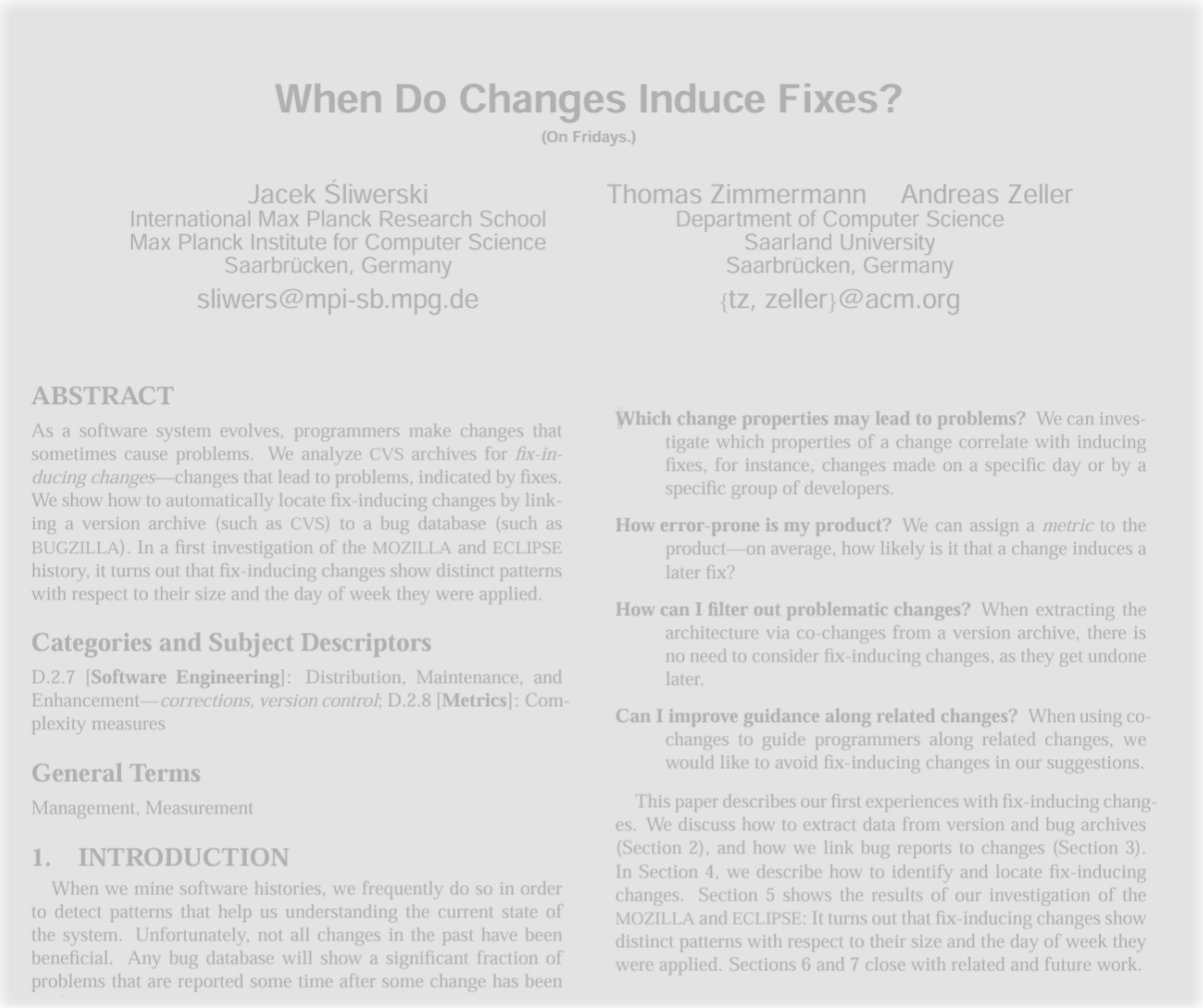


Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger
can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions



When Do Changes Induce Fixes?

(On Fridays.)

Jacek Śliwerski
International Max Planck Research School
Max Planck Institute for Computer Science
Saarbrücken, Germany
sliwers@mpi-sb.mpg.de

Thomas Zimmermann Andreas Zeller
Department of Computer Science
Saarland University
Saarbrücken, Germany
{tz, zeller}@acm.org

ABSTRACT

As a software system evolves, programmers make changes that sometimes cause problems. We analyze CVS archives for *fix-inducing changes*—changes that lead to problems, indicated by fixes. We show how to automatically locate fix-inducing changes by linking a version archive (such as CVS) to a bug database (such as BUGZILLA). In a first investigation of the MOZILLA and ECLIPSE history, it turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections, version control*; D.2.8 [**Metrics**]: Complexity measures

General Terms

Management, Measurement

1. INTRODUCTION

When we mine software histories, we frequently do so in order to detect patterns that help us understanding the current state of the system. Unfortunately, not all changes in the past have been beneficial. Any bug database will show a significant fraction of problems that are reported some time after some change has been

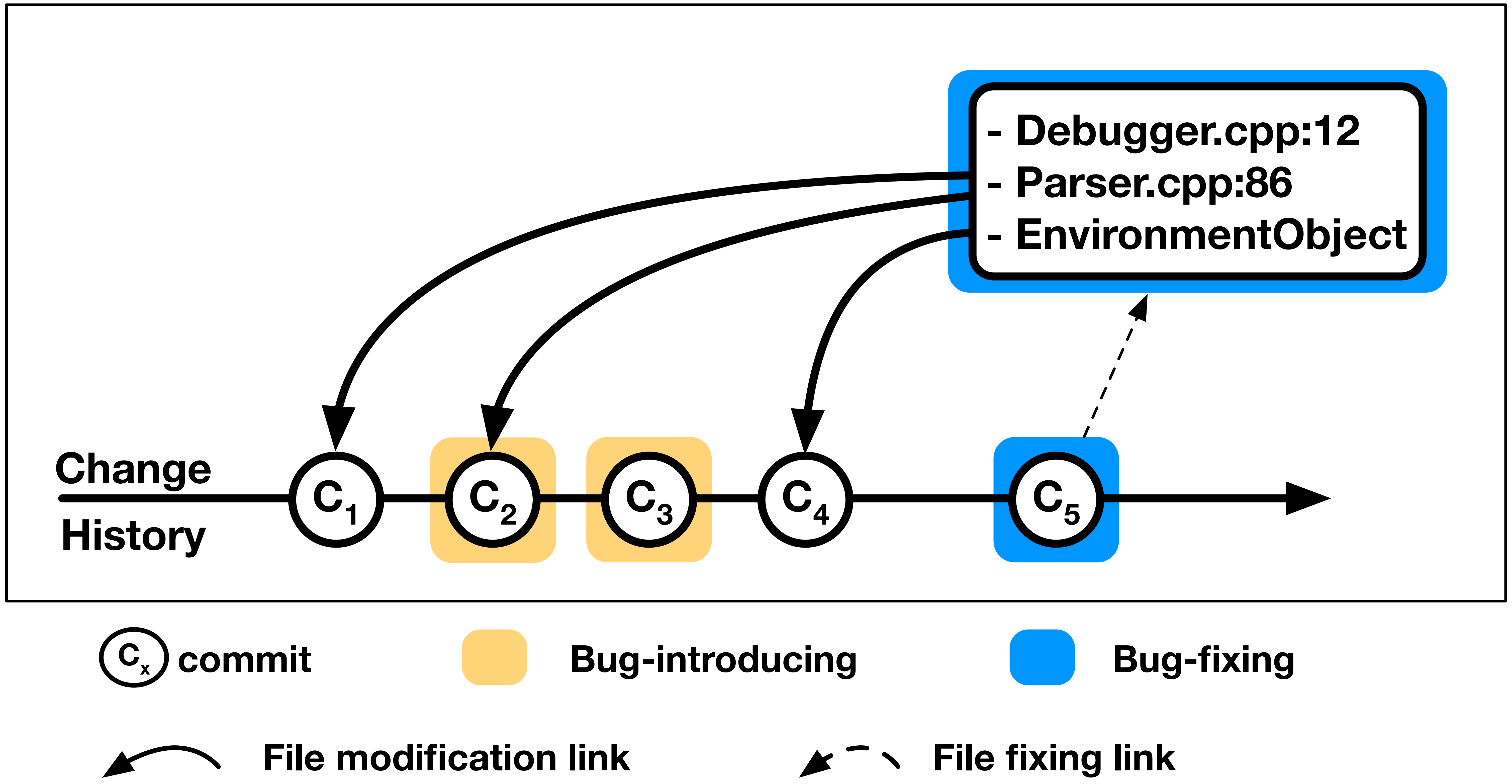
Which change properties may lead to problems? We can investigate which properties of a change correlate with inducing fixes, for instance, changes made on a specific day or by a specific group of developers.

How error-prone is my product? We can assign a *metric* to the product—on average, how likely is it that a change induces a later fix?

How can I filter out problematic changes? When extracting the architecture via co-changes from a version archive, there is no need to consider fix-inducing changes, as they get undone later.

Can I improve guidance along related changes? When using co-changes to guide programmers along related changes, we would like to avoid fix-inducing changes in our suggestions.

This paper describes our first experiences with fix-inducing changes. We discuss how to extract data from version and bug archives (Section 2), and how we link bug reports to changes (Section 3). In Section 4, we describe how to identify and locate fix-inducing changes. Section 5 shows the results of our investigation of the MOZILLA and ECLIPSE: It turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied. Sections 6 and 7 close with related and future work.

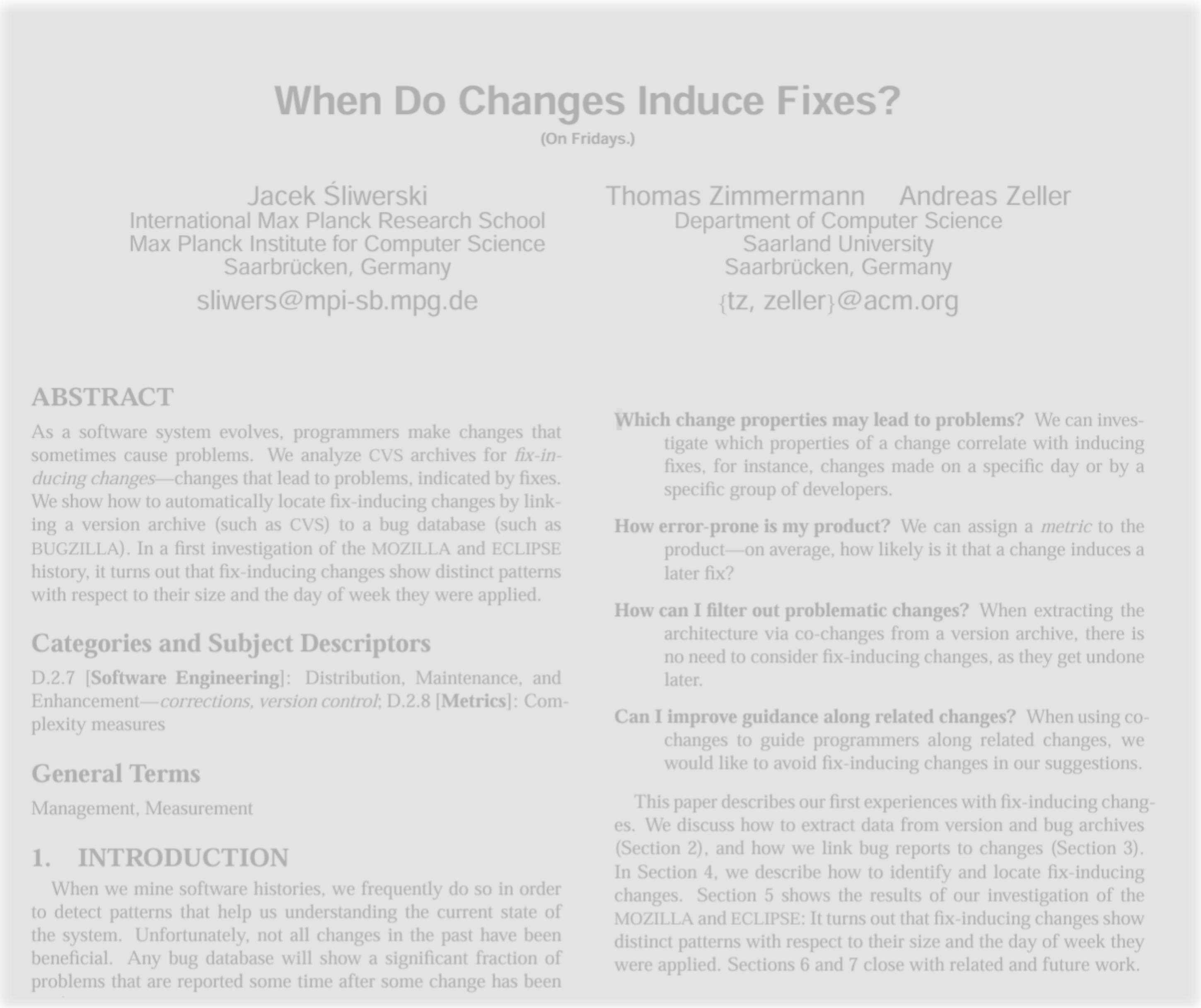


SZZ

All changes are fix related — Tangled commits

The same files caused the bug — Ghost commits

Lack of developer-labeled data



When Do Changes Induce Fixes?

(On Fridays.)

Jacek Śliwerski
International Max Planck Research School
Max Planck Institute for Computer Science
Saarbrücken, Germany
sliwers@mpi-sb.mpg.de

Thomas Zimmermann Andreas Zeller
Department of Computer Science
Saarland University
Saarbrücken, Germany
{tz, zeller}@acm.org

ABSTRACT

As a software system evolves, programmers make changes that sometimes cause problems. We analyze CVS archives for *fix-inducing changes*—changes that lead to problems, indicated by fixes. We show how to automatically locate fix-inducing changes by linking a version archive (such as CVS) to a bug database (such as BUGZILLA). In a first investigation of the MOZILLA and ECLIPSE history, it turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections, version control*; D.2.8 [**Metrics**]: Complexity measures

General Terms

Management, Measurement

1. INTRODUCTION

When we mine software histories, we frequently do so in order to detect patterns that help us understanding the current state of the system. Unfortunately, not all changes in the past have been beneficial. Any bug database will show a significant fraction of problems that are reported some time after some change has been

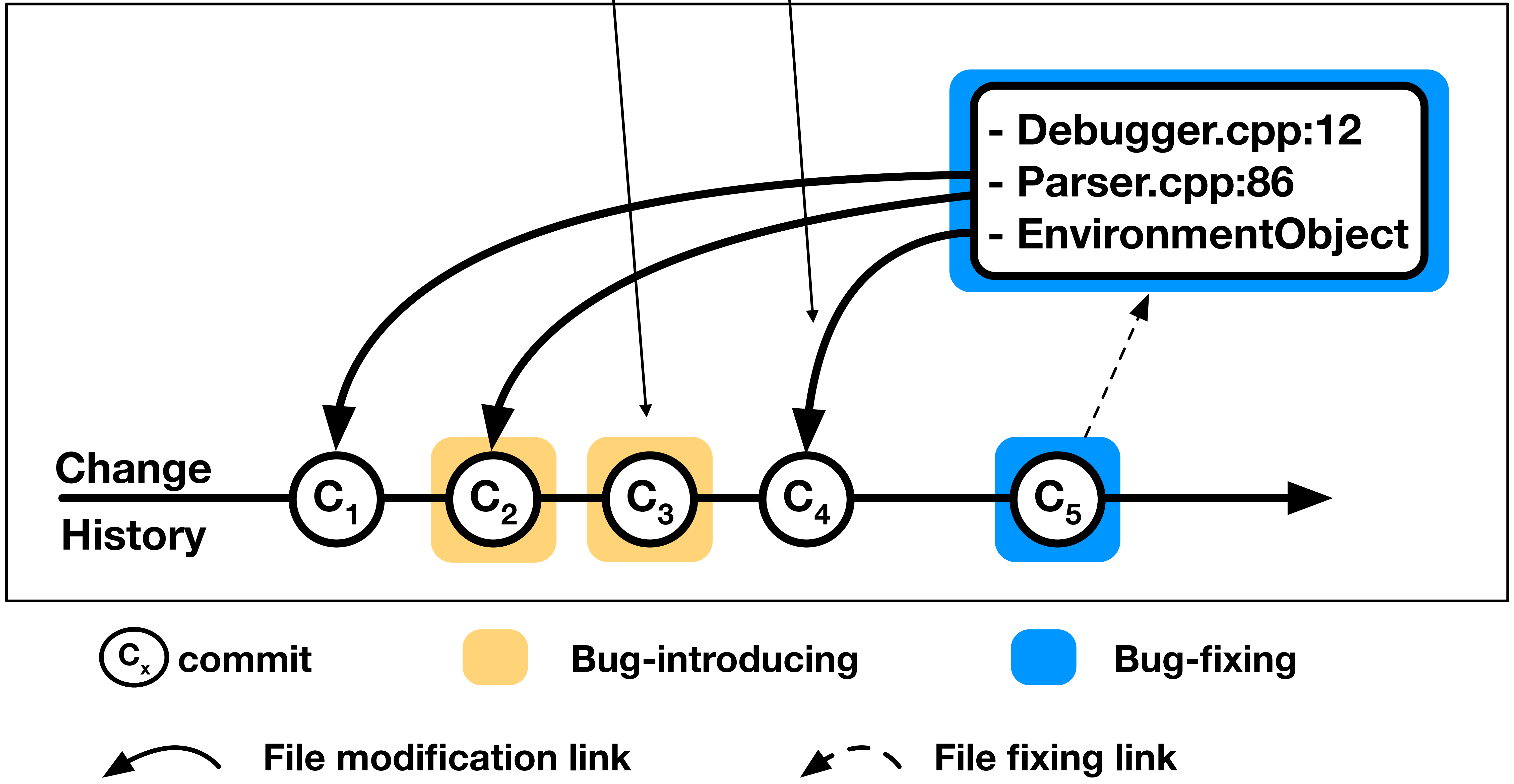
Which change properties may lead to problems? We can investigate which properties of a change correlate with inducing fixes, for instance, changes made on a specific day or by a specific group of developers.

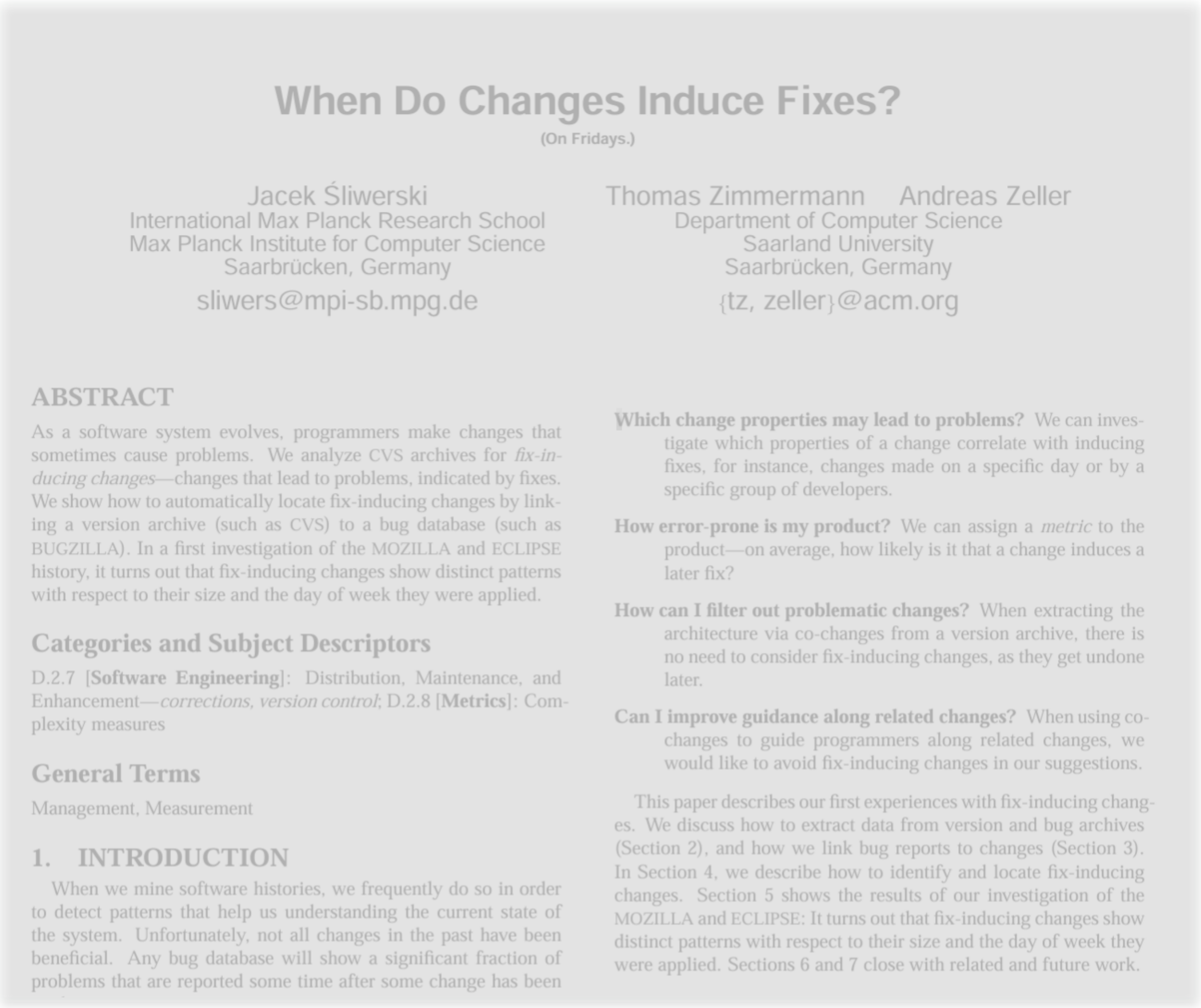
How error-prone is my product? We can assign a *metric* to the product—on average, how likely is it that a change induces a later fix?

How can I filter out problematic changes? When extracting the architecture via co-changes from a version archive, there is no need to consider fix-inducing changes, as they get undone later.

Can I improve guidance along related changes? When using co-changes to guide programmers along related changes, we would like to avoid fix-inducing changes in our suggestions.

This paper describes our first experiences with fix-inducing changes. We discuss how to extract data from version and bug archives (Section 2), and how we link bug reports to changes (Section 3). In Section 4, we describe how to identify and locate fix-inducing changes. Section 5 shows the results of our investigation of the MOZILLA and ECLIPSE: It turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied. Sections 6 and 7 close with related and future work.





When Do Changes Induce Fixes?

(On Fridays.)

Jacek Śliwerski
International Max Planck Research School
Max Planck Institute for Computer Science
Saarbrücken, Germany
sliwers@mpi-sb.mpg.de

Thomas Zimmermann Andreas Zeller
Department of Computer Science
Saarland University
Saarbrücken, Germany
{tz, zeller}@acm.org

ABSTRACT

As a software system evolves, programmers make changes that sometimes cause problems. We analyze CVS archives for *fix-inducing changes*—changes that lead to problems, indicated by fixes. We show how to automatically locate fix-inducing changes by linking a version archive (such as CVS) to a bug database (such as BUGZILLA). In a first investigation of the MOZILLA and ECLIPSE history, it turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*corrections, version control*; D.2.8 [**Metrics**]: Complexity measures

General Terms

Management, Measurement

1. INTRODUCTION

When we mine software histories, we frequently do so in order to detect patterns that help us understanding the current state of the system. Unfortunately, not all changes in the past have been beneficial. Any bug database will show a significant fraction of problems that are reported some time after some change has been

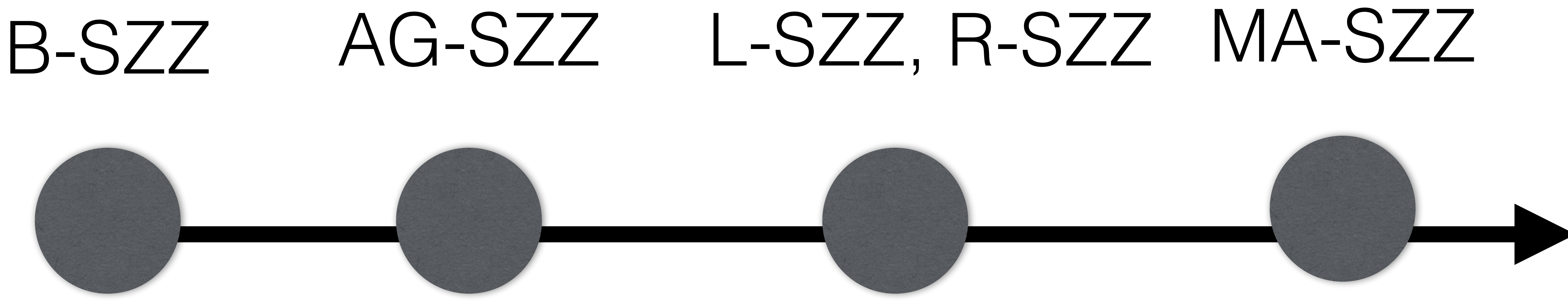
Which change properties may lead to problems? We can investigate which properties of a change correlate with inducing fixes, for instance, changes made on a specific day or by a specific group of developers.

How error-prone is my product? We can assign a *metric* to the product—on average, how likely is it that a change induces a later fix?

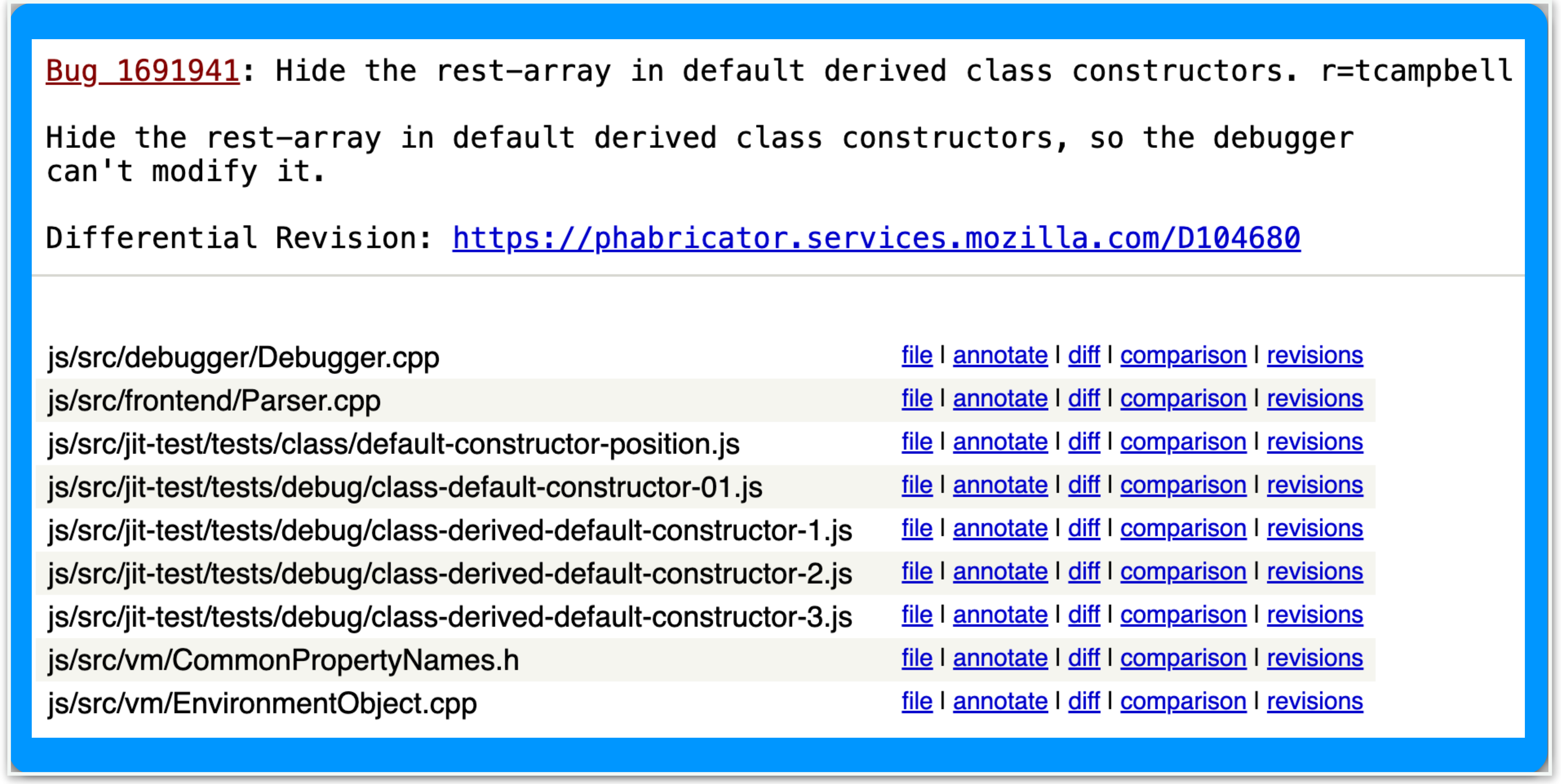
How can I filter out problematic changes? When extracting the architecture via co-changes from a version archive, there is no need to consider fix-inducing changes, as they get undone later.

Can I improve guidance along related changes? When using co-changes to guide programmers along related changes, we would like to avoid fix-inducing changes in our suggestions.

This paper describes our first experiences with fix-inducing changes. We discuss how to extract data from version and bug archives (Section 2), and how we link bug reports to changes (Section 3). In Section 4, we describe how to identify and locate fix-inducing changes. Section 5 shows the results of our investigation of the MOZILLA and ECLIPSE: It turns out that fix-inducing changes show distinct patterns with respect to their size and the day of week they were applied. Sections 6 and 7 close with related and future work.



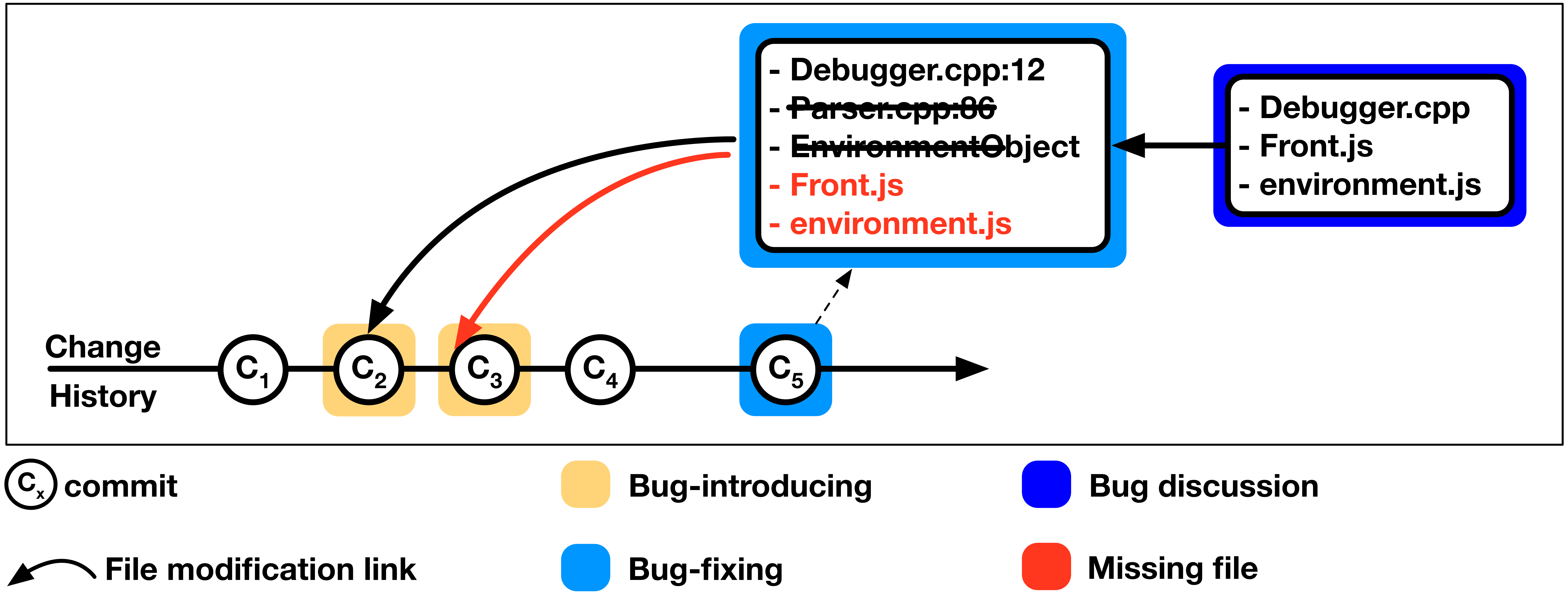
What if we get relevant files?

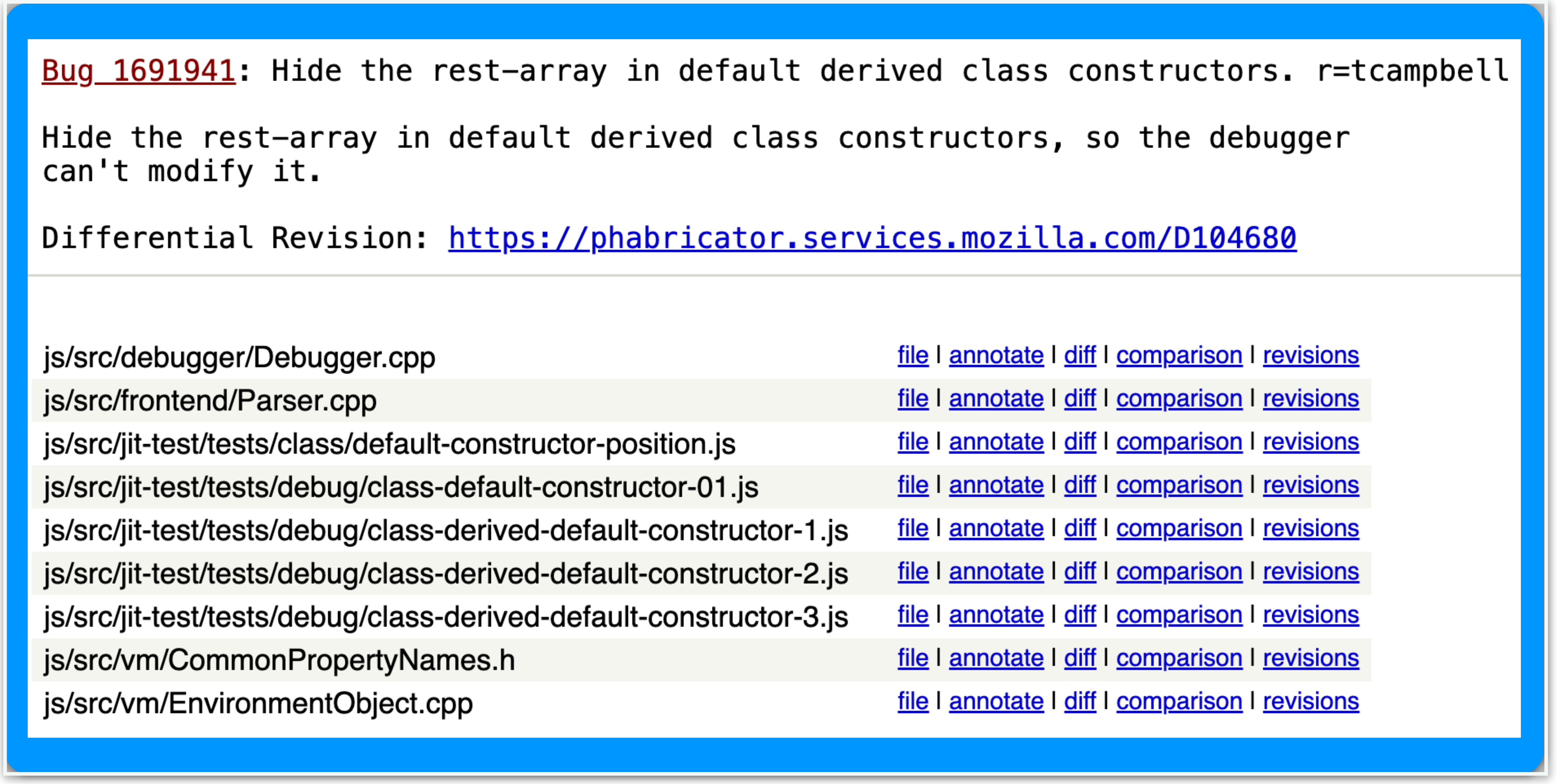


```
Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger
can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680
```

js/src/debugger/Debugger.cpp	file | annotate | diff | comparison | revisions
js/src/frontend/Parser.cpp	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/class/default-constructor-position.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file | annotate | diff | comparison | revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file | annotate | diff | comparison | revisions
js/src/vm/CommonPropertyNames.h	file | annotate | diff | comparison | revisions
js/src/vm/EnvironmentObject.cpp	file | annotate | diff | comparison | revisions

What if we get relevant files?



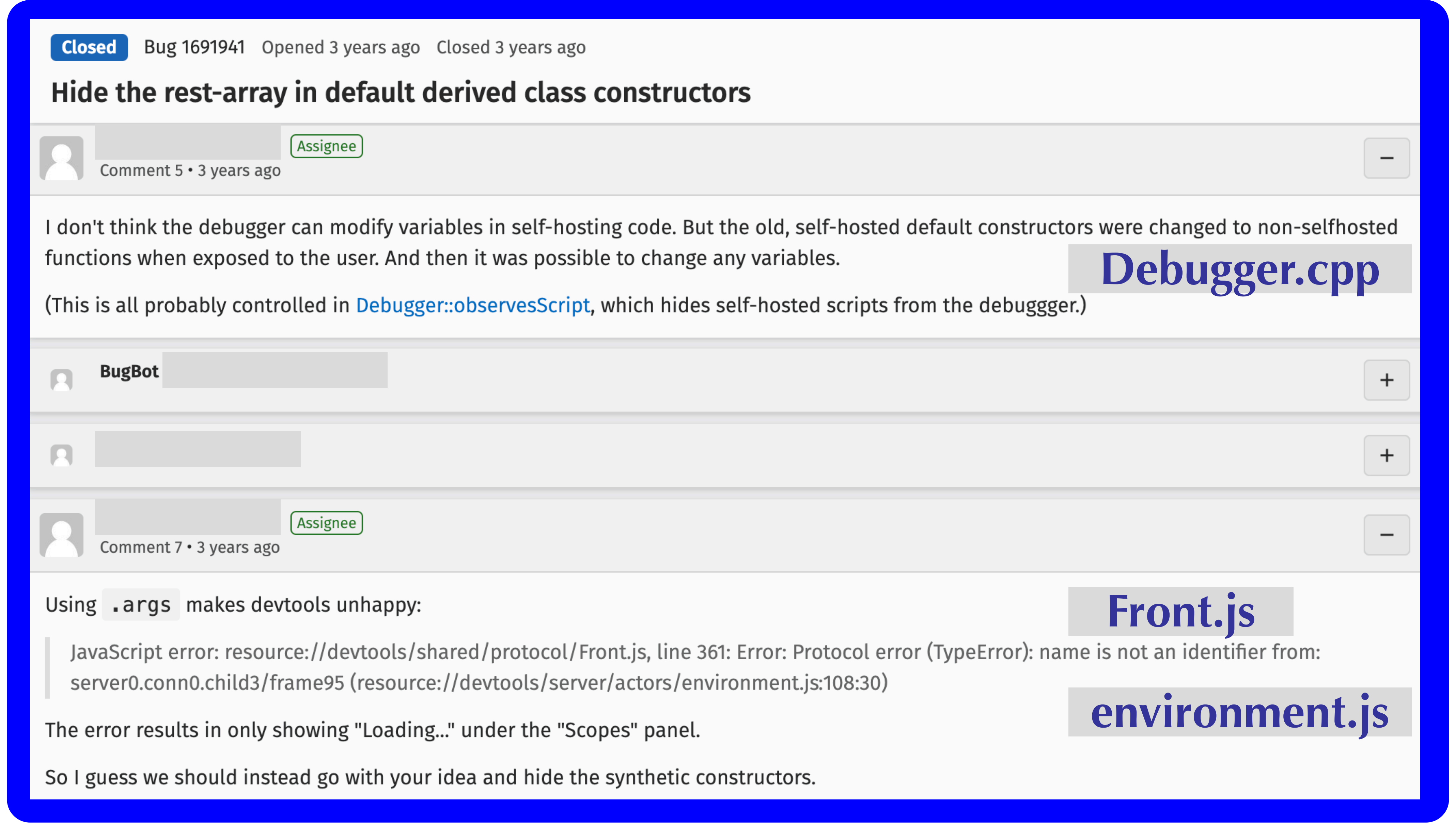


Bug 1691941: Hide the rest-array in default derived class constructors. r=tcampbell

Hide the rest-array in default derived class constructors, so the debugger can't modify it.

Differential Revision: https://phabricator.services.mozilla.com/D104680

js/src/debugger/Debugger.cpp	file I annotate I diff I comparison I revisions
js/src/frontend/Parser.cpp	file I annotate I diff I comparison I revisions
js/src/jit-test/tests/class/default-constructor-position.js	file I annotate I diff I comparison I revisions
js/src/jit-test/tests/debug/class-default-constructor-01.js	file I annotate I diff I comparison I revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-1.js	file I annotate I diff I comparison I revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-2.js	file I annotate I diff I comparison I revisions
js/src/jit-test/tests/debug/class-derived-default-constructor-3.js	file I annotate I diff I comparison I revisions
js/src/vm/CommonPropertyNames.h	file I annotate I diff I comparison I revisions
js/src/vm/EnvironmentObject.cpp	file I annotate I diff I comparison I revisions



Closed Bug 1691941 Opened 3 years ago Closed 3 years ago

Hide the rest-array in default derived class constructors

[redacted] Assignee
Comment 5 • 3 years ago

I don't think the debugger can modify variables in self-hosting code. But the old, self-hosted default constructors were changed to non-selfhosted functions when exposed to the user. And then it was possible to change any variables.

(This is all probably controlled in Debugger::observesScript, which hides self-hosted scripts from the debuggger.)

BugBot [redacted]

[redacted]

[redacted] Assignee
Comment 7 • 3 years ago

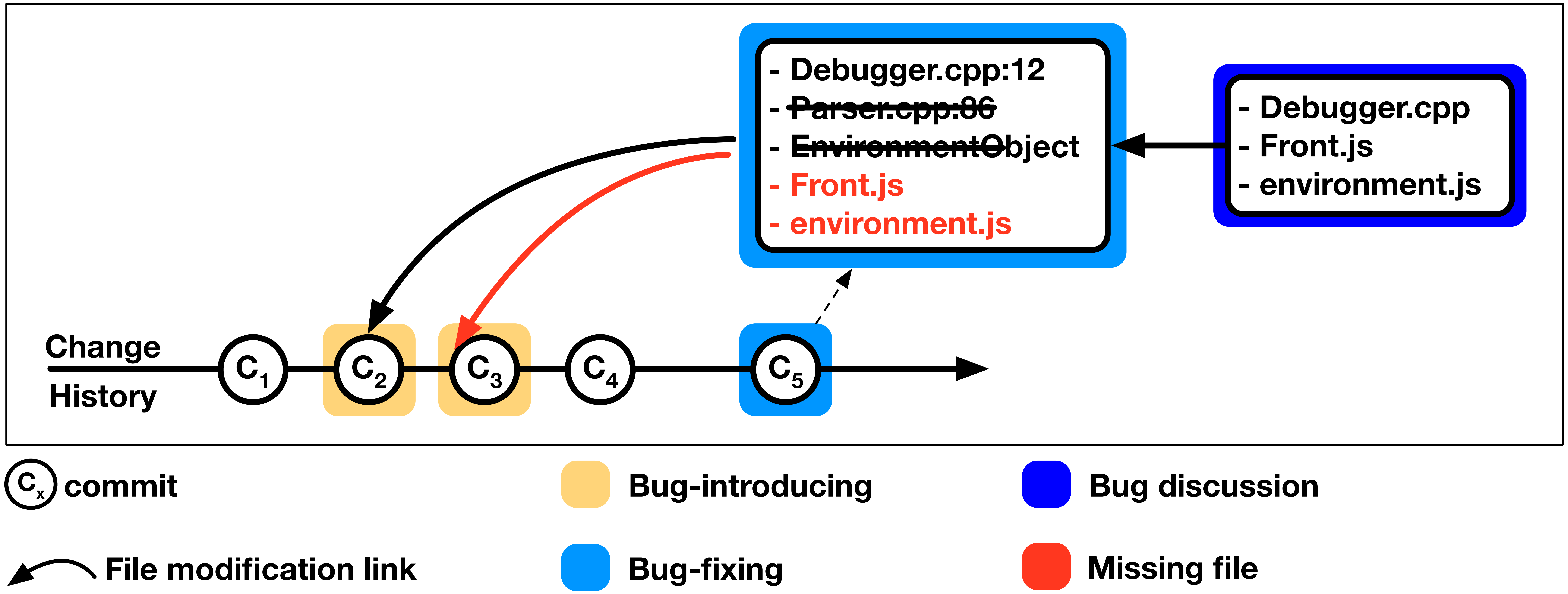
Using `.args` makes devtools unhappy:

> JavaScript error: resource://devtools/shared/protocol/Front.js, line 361: Error: Protocol error (TypeError): name is not an identifier from: server0.conn0.child3/frame95 (resource://devtools/server/actors/environment.js:108:30)

The error results in only showing "Loading..." under the "Scopes" panel.

So I guess we should instead go with your idea and hide the synthetic constructors.

What if we get relevant files?



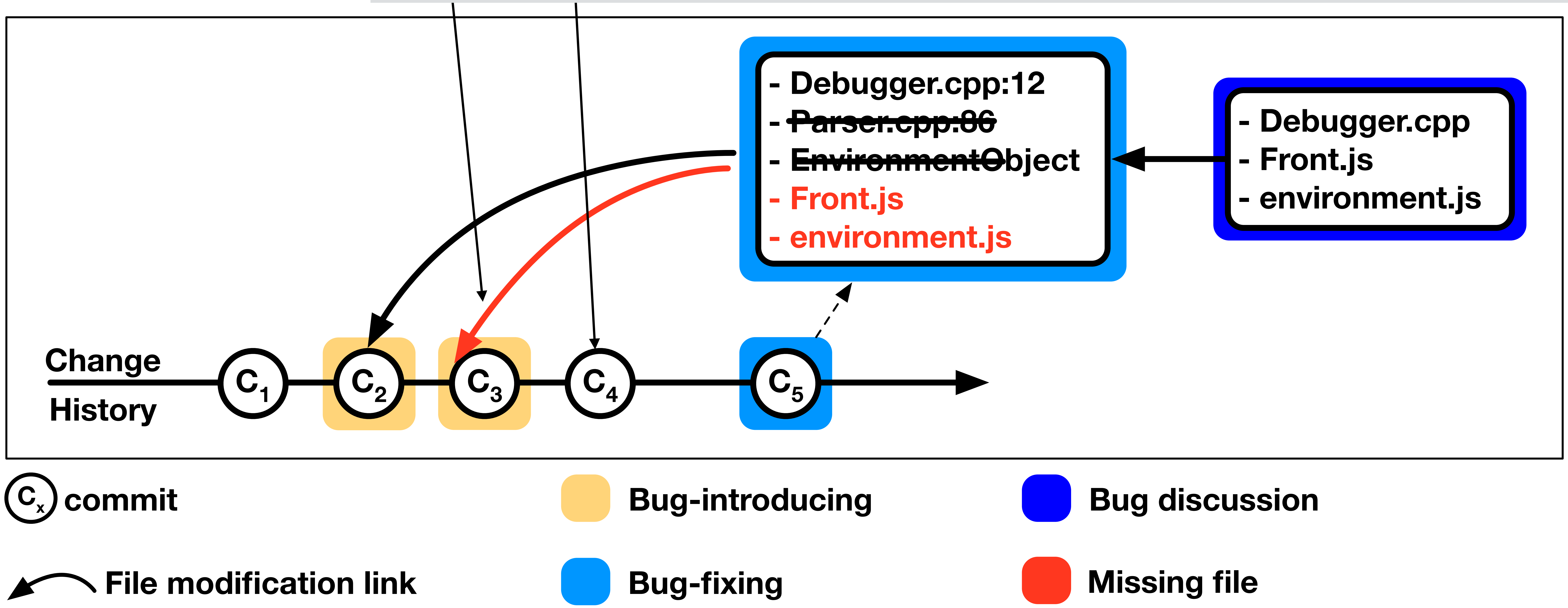
Remove Noisy files

Add external files

Tangled commits

Ghost commits

What if we get relevant files?



moz://a

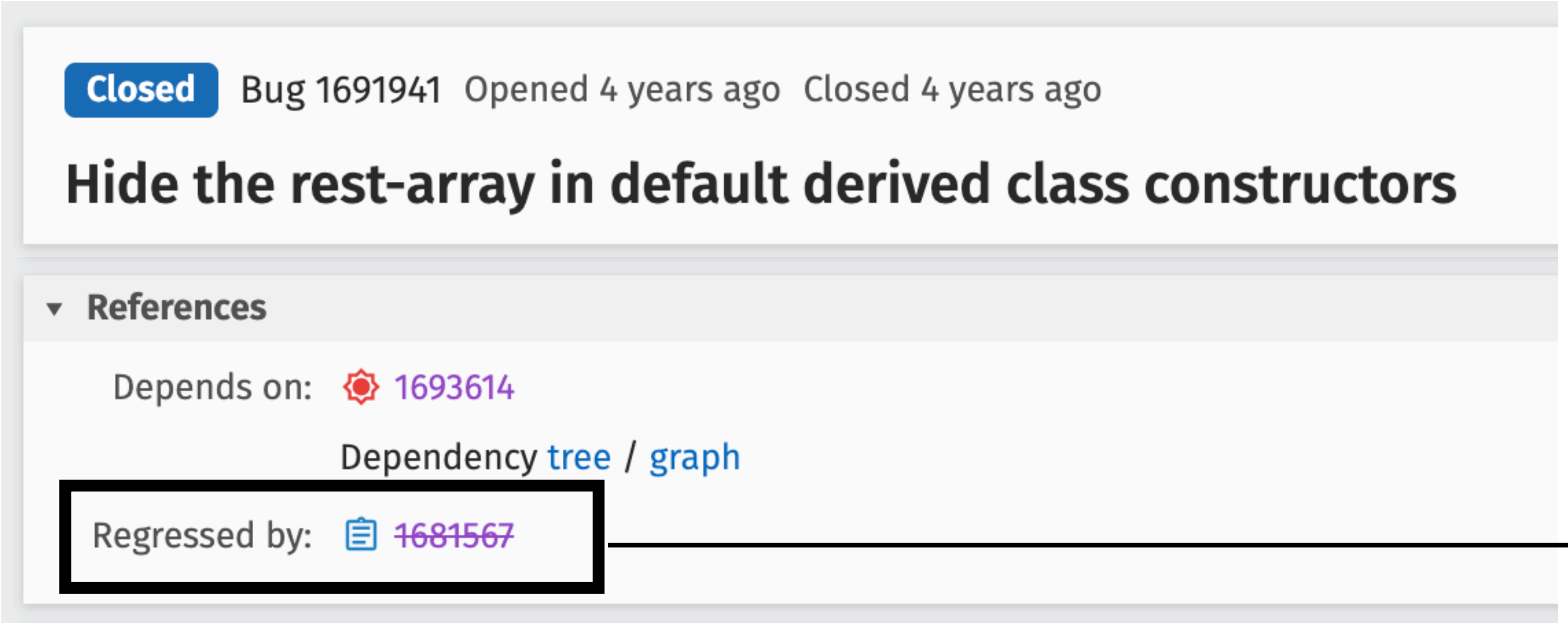
(1) Mine Mozilla codebase
of 251, 601 files

<250K files

<13 languages

<25M LOC

Labeled by developers



Closed Bug 1691941 Opened 4 years ago Closed 4 years ago

Hide the rest-array in default derived class constructors

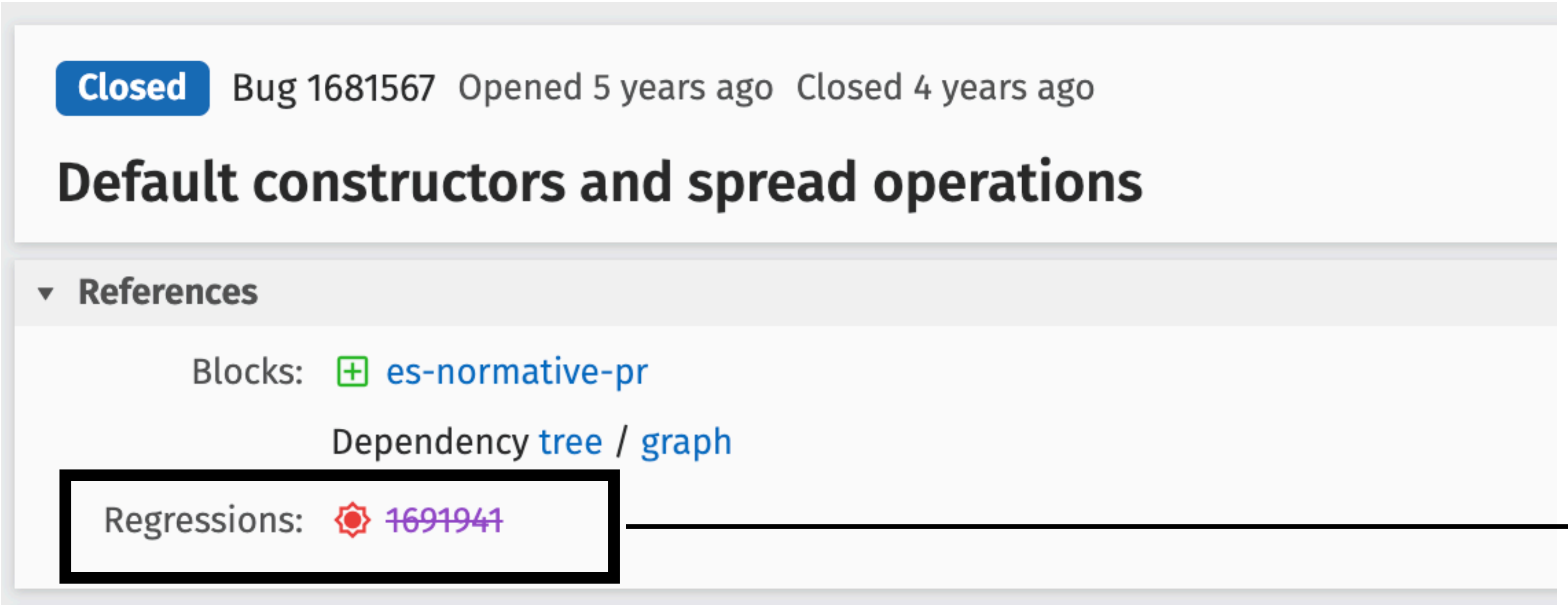
References

Depends on: 1693614

Dependency tree / graph

Regressed by: ~~1681567~~

🟨 **Bug-introducing**



Closed Bug 1681567 Opened 5 years ago Closed 4 years ago

Default constructors and spread operations

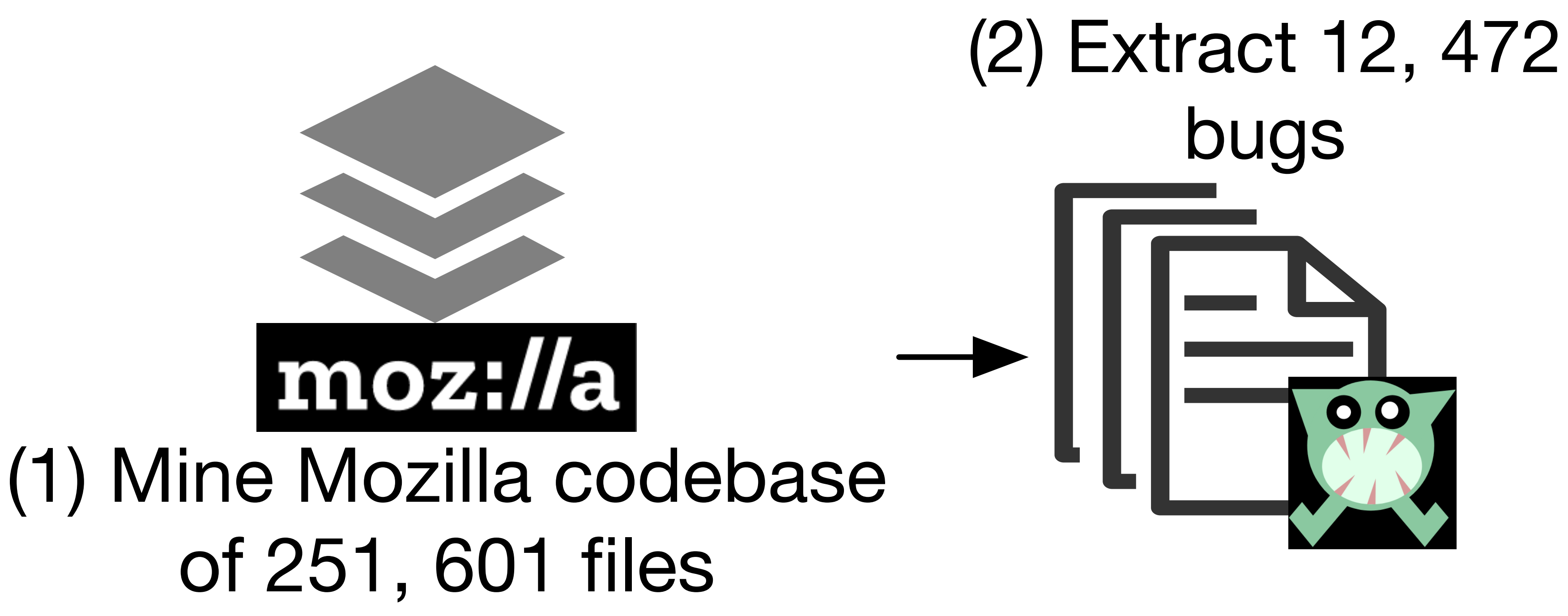
References

Blocks: es-normative-pr

Dependency tree / graph

Regressions: ~~1691941~~

🟦 **Bug-fixing**



(1) Mine Mozilla codebase of 251, 601 files

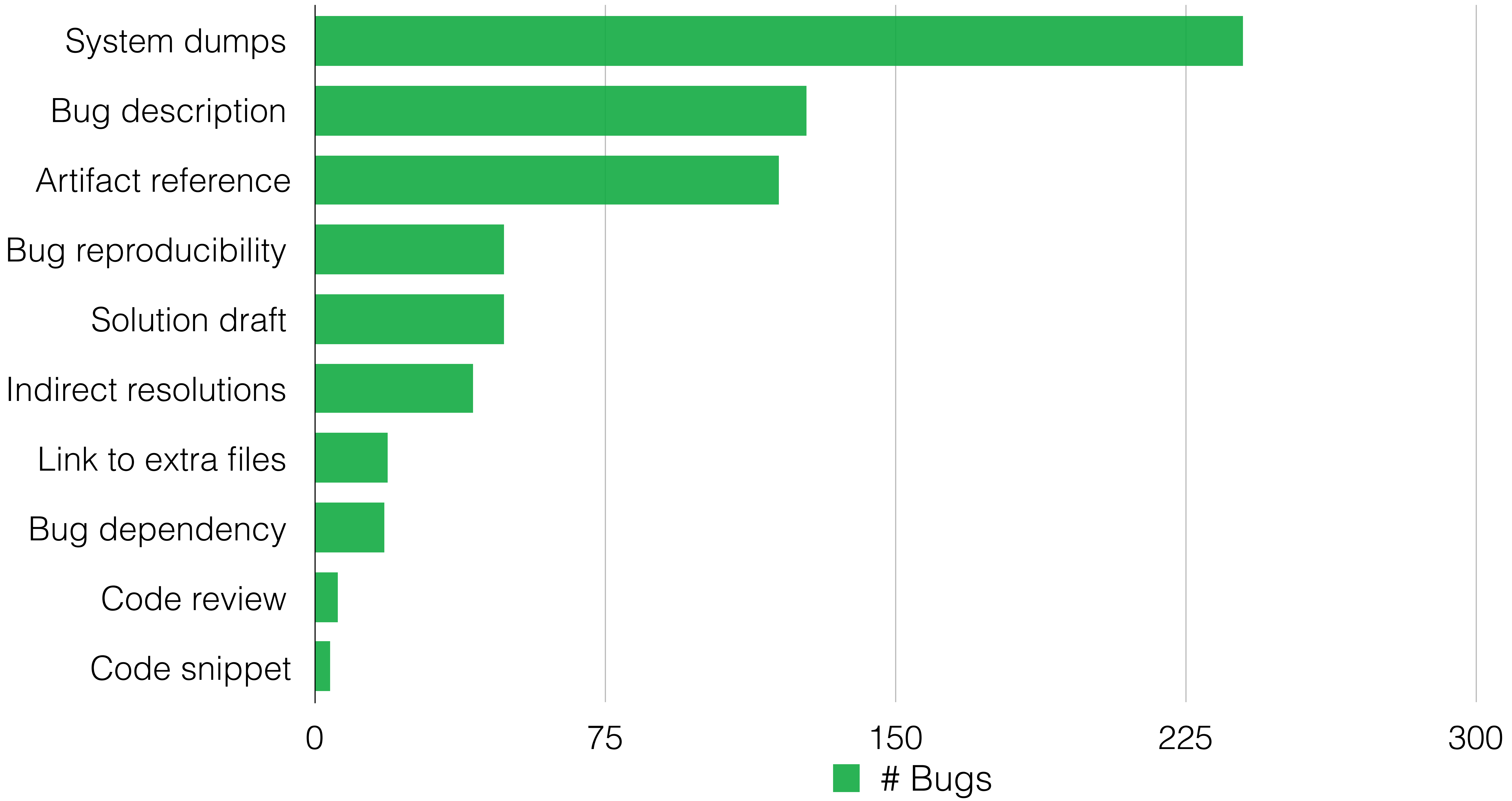
(2) Extract 12, 472 bugs

RQ1: Why do developers discuss files?

RQ2: How often are these files part of bug-related commits?

RQ3/4: What is the impact on tangled and ghost commits?

RQ1: Why do developers discuss files?



RQ2: How often are these files part of bug-related commits?

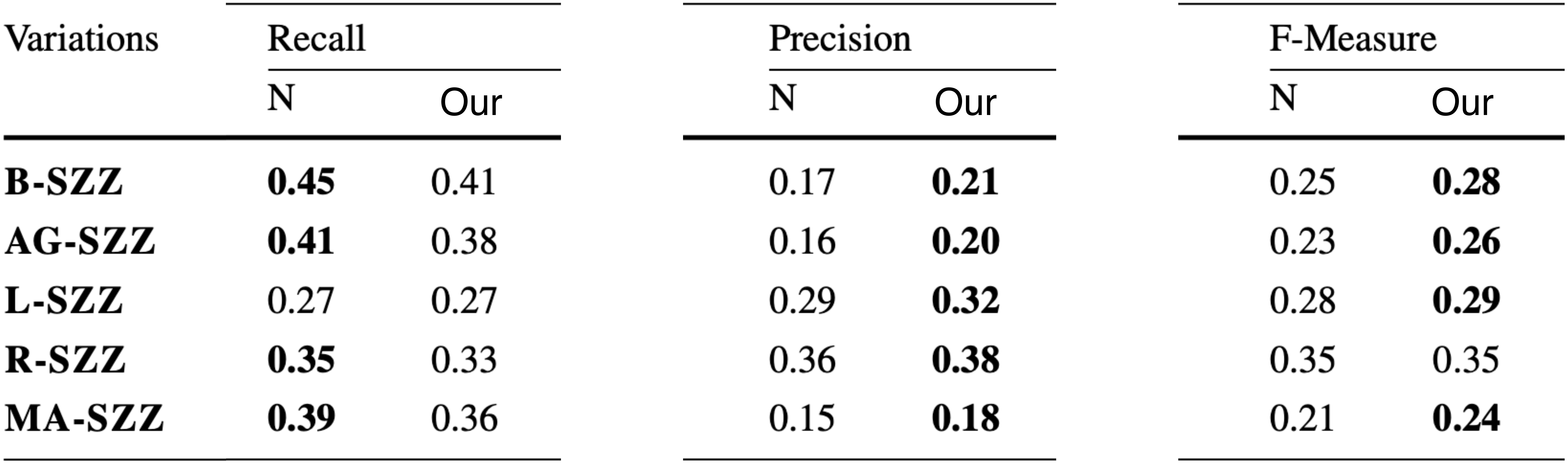
74% of bug reports mention at least one file

71% of those are part of bug-related commits

RQ3: Impact on tangled commits

% of actual bug-introducing commits that SZZ finds

% of SZZ-flagged commits that are truly bug-introducing



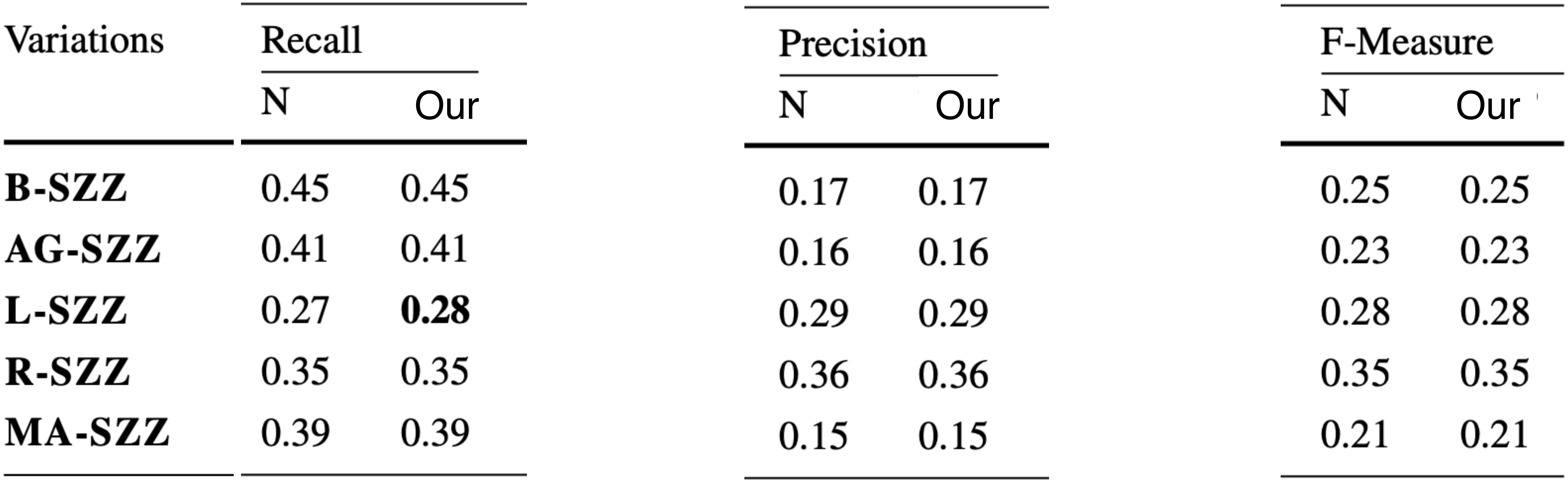
Variations	Recall		Precision		F-Measure	
	N	Our	N	Our	N	Our
B-SZZ	**0.45**	0.41	0.17	**0.21**	0.25	**0.28**
AG-SZZ	**0.41**	0.38	0.16	**0.20**	0.23	**0.26**
L-SZZ	0.27	0.27	0.29	**0.32**	0.28	**0.29**
R-SZZ	**0.35**	0.33	0.36	**0.38**	0.35	0.35
MA-SZZ	**0.39**	0.36	0.15	**0.18**	0.21	**0.24**

Removing noisy files is productive

RQ4: Impact on ghost commits

% of actual bug-introducing commits that SZZ finds

% of SZZ-flagged commits that are truly bug-introducing



Variations	Recall		Precision		F-Measure	
	N	Our	N	Our	N	Our
B-SZZ	0.45	0.45	0.17	0.17	0.25	0.25
AG-SZZ	0.41	0.41	0.16	0.16	0.23	0.23
L-SZZ	0.27	**0.28**	0.29	0.29	0.28	0.28
R-SZZ	0.35	0.35	0.36	0.36	0.35	0.35
MA-SZZ	0.39	0.39	0.15	0.15	0.21	0.21

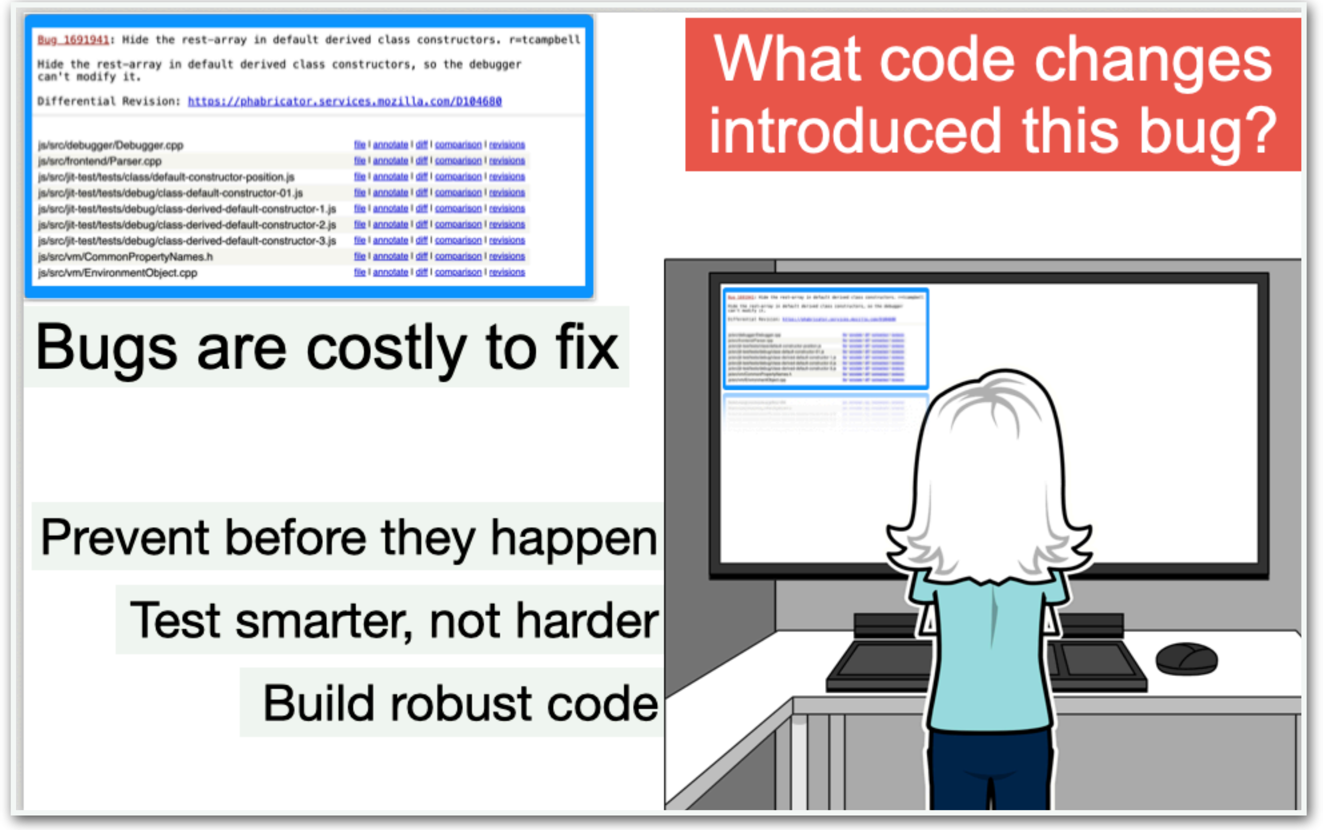
Adding external files did not help

Takeaways

- Not all the mentioned files help resolve the bugs
- Removing noisy files is productive but not adding files
- 12,472 bugs (the links established by Mozilla developers)

Future work

- Filtering files from bug discussions and other artifacts like Emails, Wikis, and Slack
- Pinpoint what changes, in addition to where
- Case study with developers on its usefulness



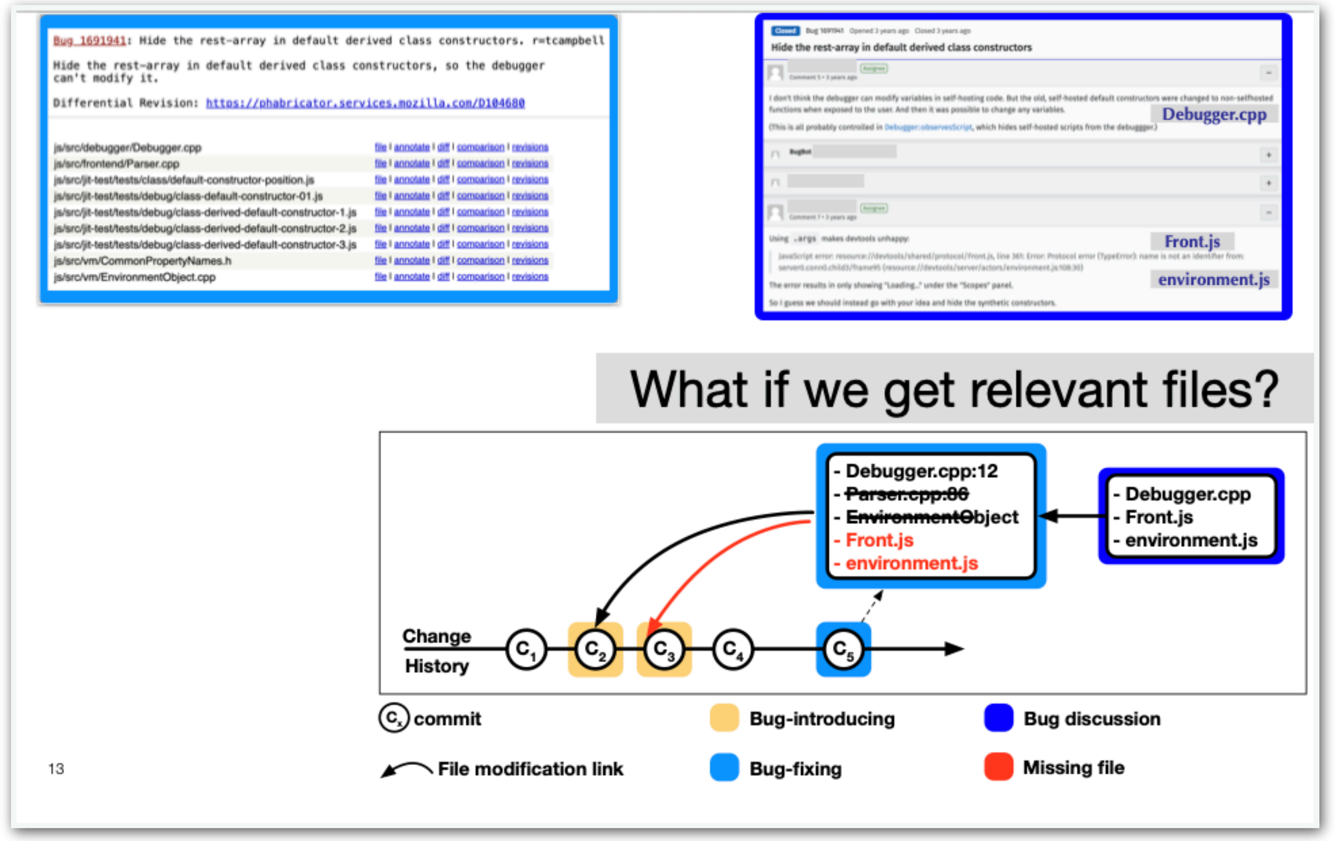
What code changes introduced this bug?

Bugs are costly to fix

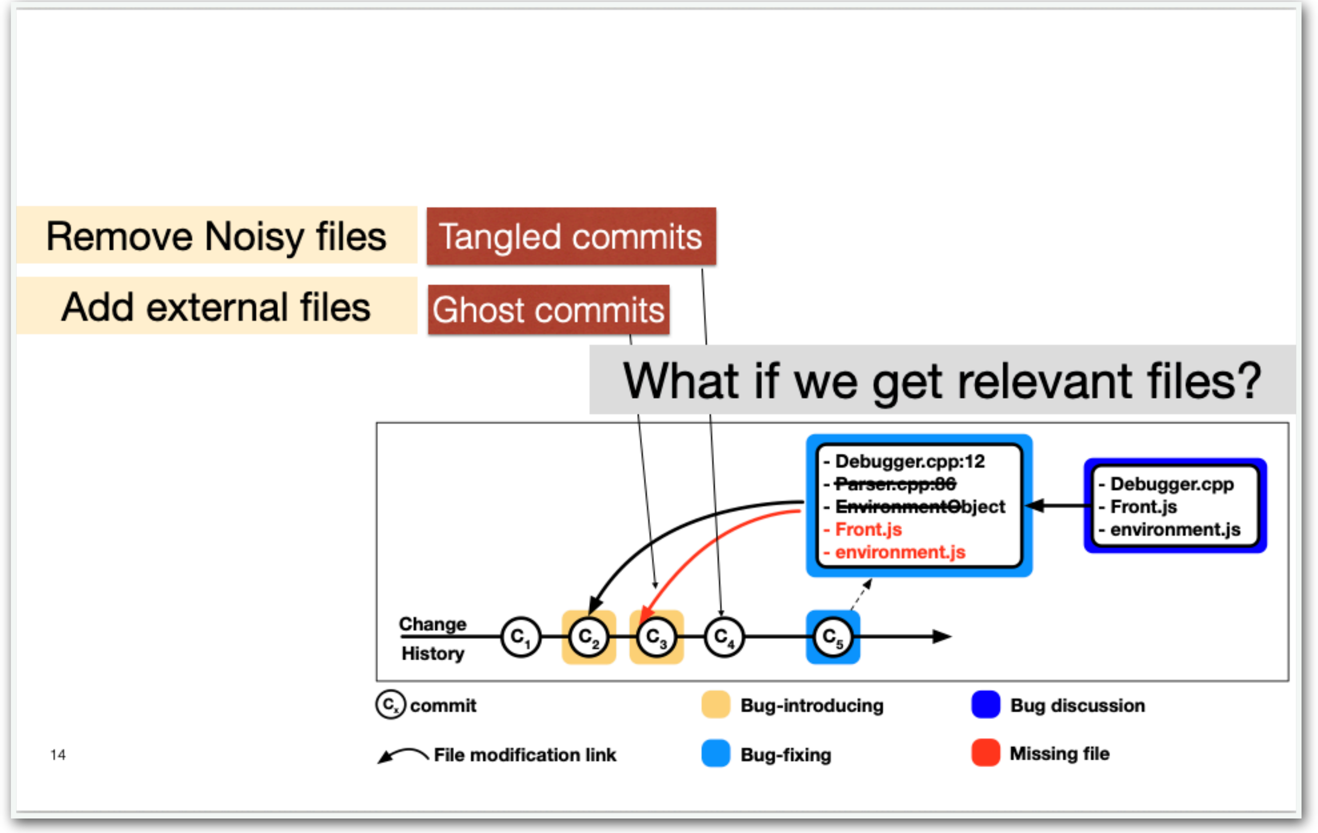
Prevent before they happen

Test smarter, not harder

Build robust code



What if we get relevant files?



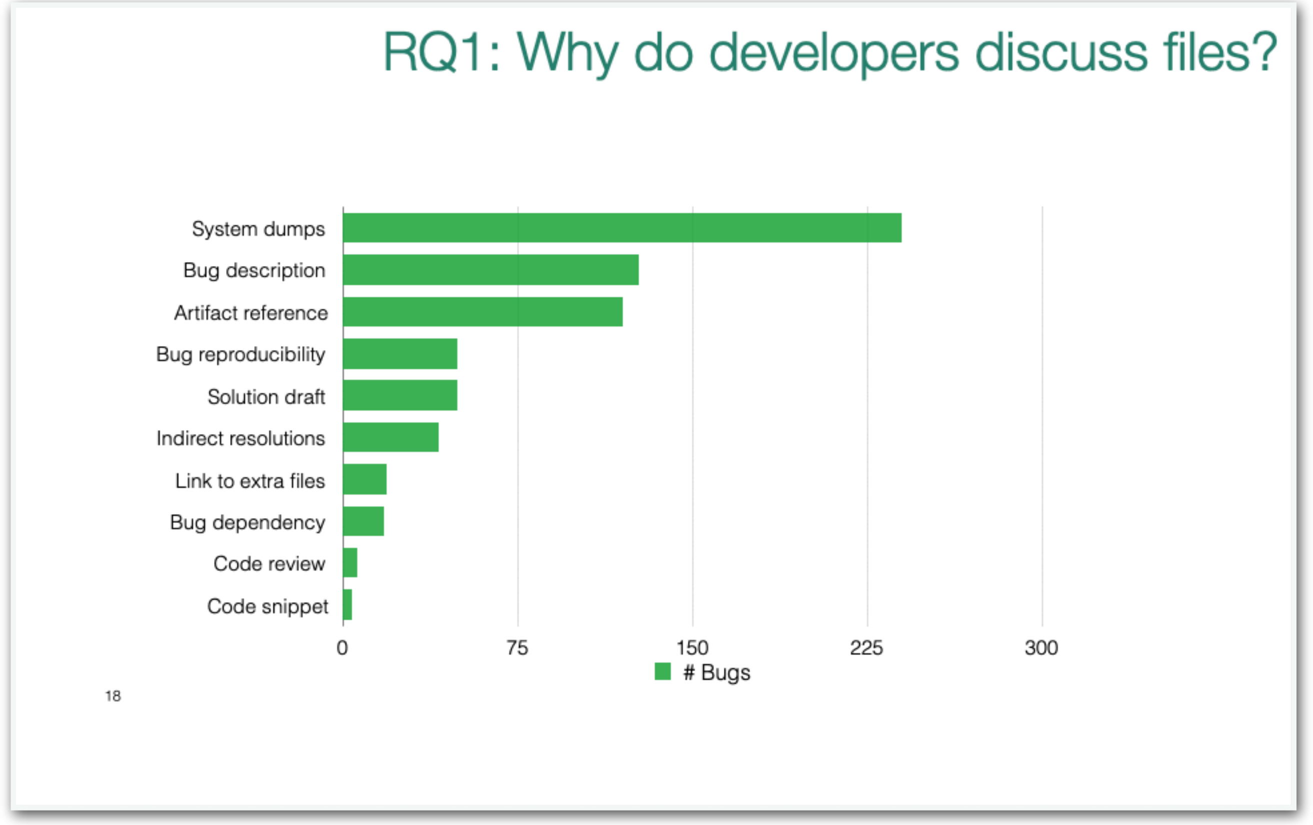
Remove Noisy files

Add external files

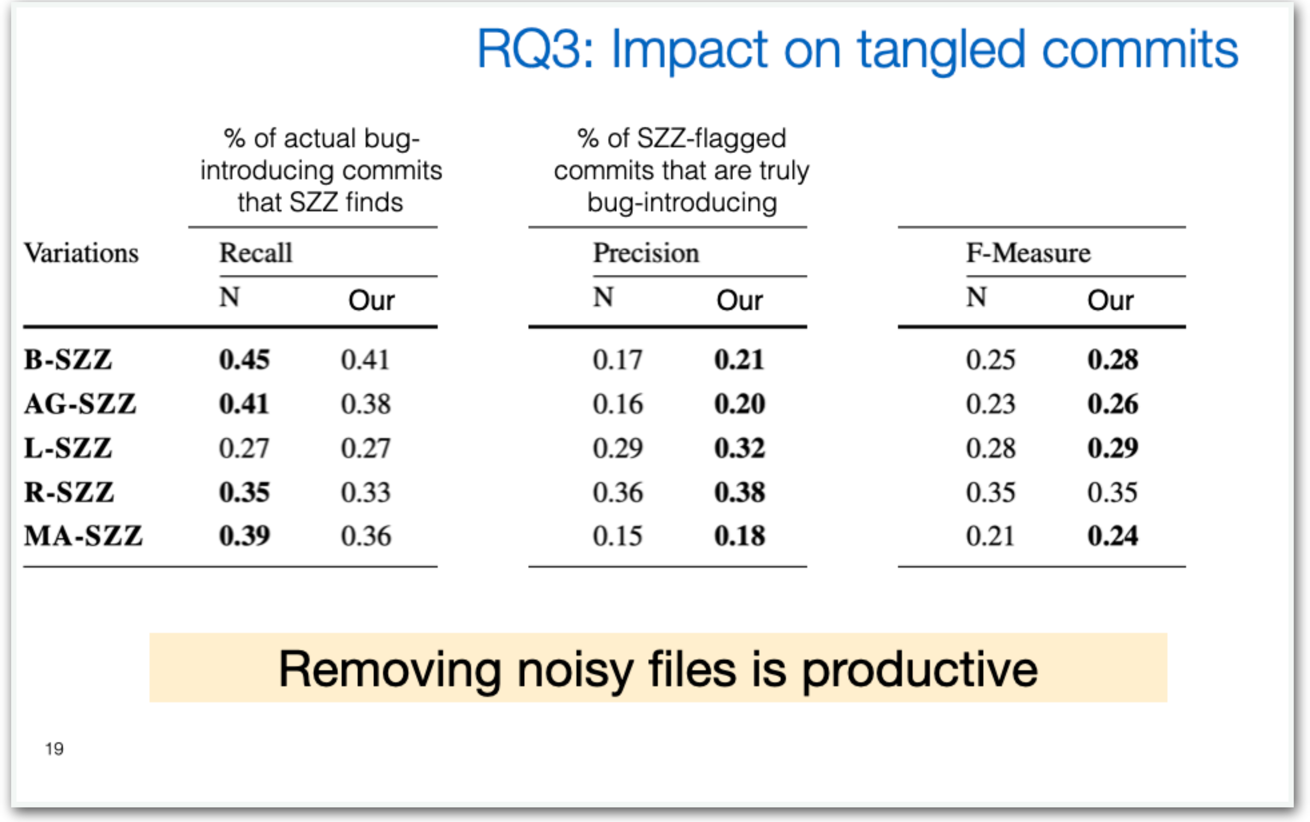
Tangled commits

Ghost commits

What if we get relevant files?



RQ1: Why do developers discuss files?

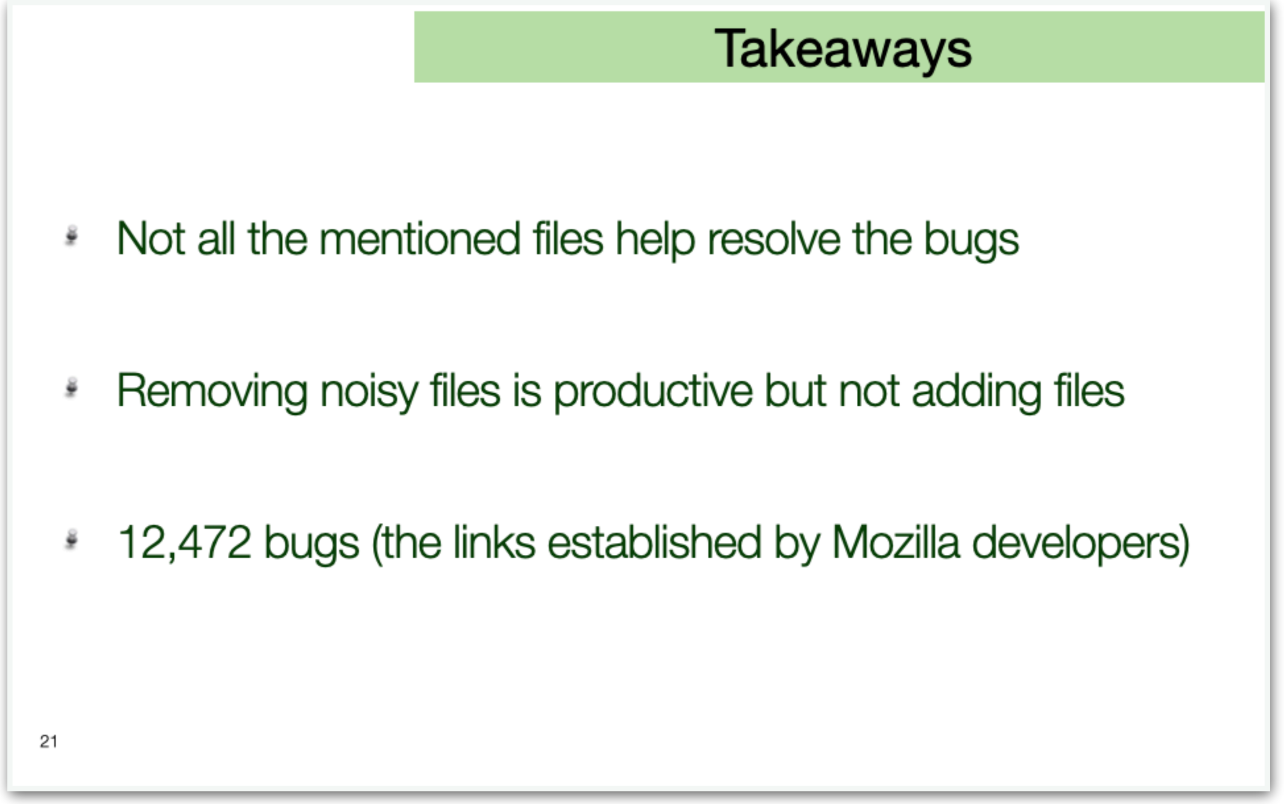


RQ3: Impact on tangled commits

	% of actual bug-introducing commits that SZZ finds		% of SZZ-flagged commits that are truly bug-introducing			
Variations	Recall		Precision		F-Measure	
	N	Our	N	Our	N	Our
B-SZZ	**0.45**	0.41	0.17	**0.21**	0.25	**0.28**
AG-SZZ	**0.41**	0.38	0.16	**0.20**	0.23	**0.26**
L-SZZ	0.27	0.27	0.29	**0.32**	0.28	**0.29**
R-SZZ	**0.35**	0.33	0.36	**0.38**	0.35	0.35
MA-SZZ	**0.39**	0.36	0.15	**0.18**	0.21	**0.24**

Removing noisy files is productive

19



Takeaways

- Not all the mentioned files help resolve the bugs
- Removing noisy files is productive but not adding files
- 12,472 bugs (the links established by Mozilla developers)

21

My profile

Replication Package

Paper