On the “Naturalness” of Buggy Code

Baishakhi Ray, Vincent Hellendoorn, Saheel Godhane,
Zhaopeng Tu, Alberto Bacchelli, Prem Devanbu

NSF Grant No. 1414172

الله:halla
من:morf
في:ni
الذين:ohw-esoht
لا:ton
إن:deedni
من:fo
ما:tahw
إلا:tpecxe
أن:taht
ثم:neht
ولا:ton-dna
وما:ton-dna
والله:halla-dna
الأرض:htrae-eht
قال:dias-eh
لهم:meht-rof
قل:yas
ذلك:taht
لكم:uoy-rof
السماوات:snevaeh-eht
على:no
إلى:ot
قالوا:dias-yeht
من:ohw
وما:tahw-dna
إنا:ew-deedni
إذا:nehw
الذي:ohw-eno-eht
الكتاب:koob-eht
ربك:drol-ruoy
شيء:gniht
ولا:ton
هذا:siht
هم:yeht
كل:yreve
يوم:yad
بعد:retfa
وهو:eh-dna
إذ:nehw
أو:ro
عن:morf
لم:ton
كان:si
حتى:litnu
إلا:tub
أولئك:esoht
إنما:ylno
وإذا:nehw-dna
آمنوا:eveileb

Real Programs are Natural!!

```
1  public PubnubPublishTest() {
2          _pubnub = new Pubnub("demo", "demo", false);
3          _pubnub_enc = new Pubnub("demo", "demo", "demo", "demo", false);
4
5          ...
6
7          jsarr.put("This is json array");
8          jsarr.put("One more entry in json array");
9
10         ...
11
12         jsobj.put("msg1", "Hi");
13         jsobj.put("msg2", "Java");
14     }
```

Highly repetitive, predictable,
amenable to large-sample statistical methods

[Hindle Et al., Allamanis et al., Nguyen et al.]

What does it mean when code is “unnatural” ?

Is “unnatural” code more defect-prone?

What is unnatural code?

How are they related to bugs?

Naturalness of Code : *N-Gram Language Model*

*G*iven the previous words, learn the *conditional distribution* of the next word.

n-grams	*frequencies*	*probabilities*
for (int i = 0 …	14	**0.70**
for (int i = start …	5	**0.25**
for (int i = end …	1	**0.05**

Naturalness of Code : *N-Gram Language Model*

Given the previous words, learn the *conditional distribution* of the next word.

n-grams	*frequencies*	*probabilities*
for (int i = 0 …	14	**0.70**
for (int i = start …	5	**0.25**
for (int i = end …	1	**0.05**

Less seen token

Un-natural Code

Naturalness of Code : *N-Gram Language Model*

n-grams	*frequencies*	*probabilities*
for (int i = ***end*** …	1	**0.05**

Unnaturalness is measured by **entropy** (“improbability”)

$$Entropy(t) = - P(t) \log_2 P(t)$$

Higher entropy value = lower probability
= less naturalness

How does “unnatural” code relate to bugs?

Unnatural code

=> Buggy lines of code

for Defect prediction

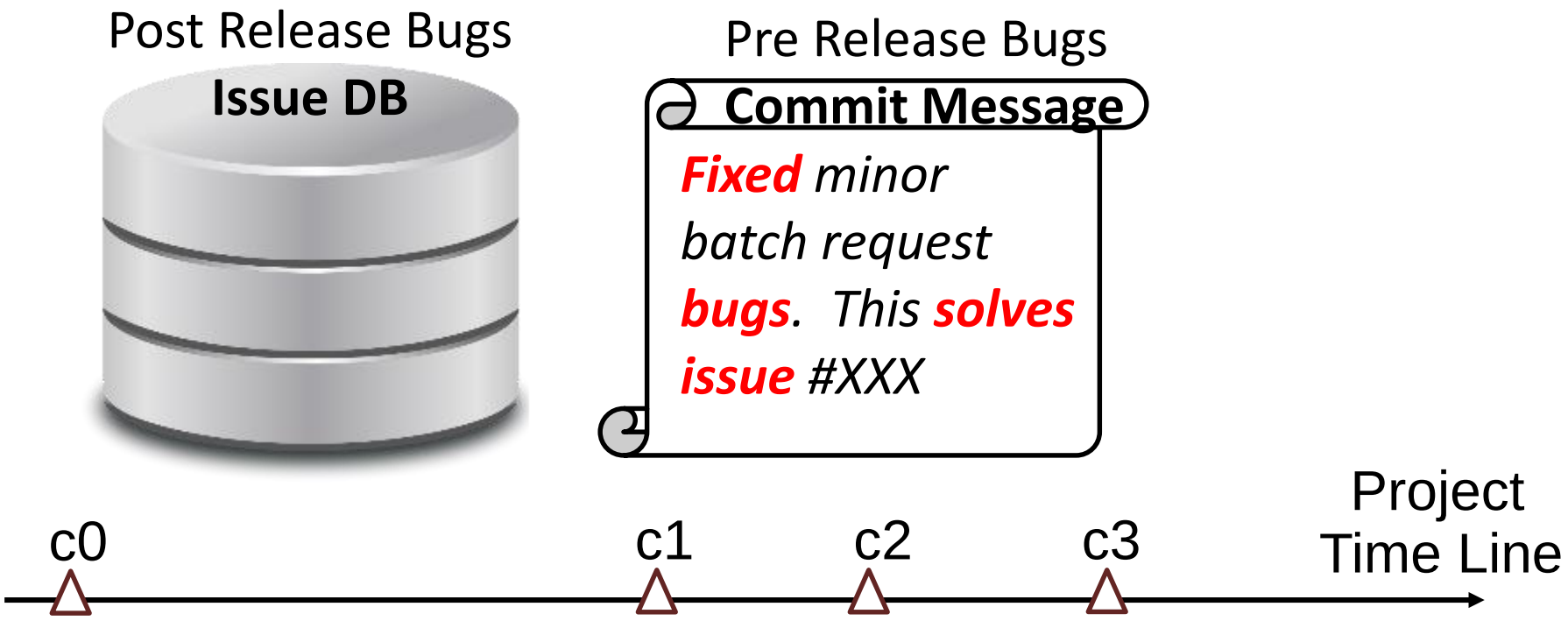
vs. Static analysis tools in defect prediction

Methodology

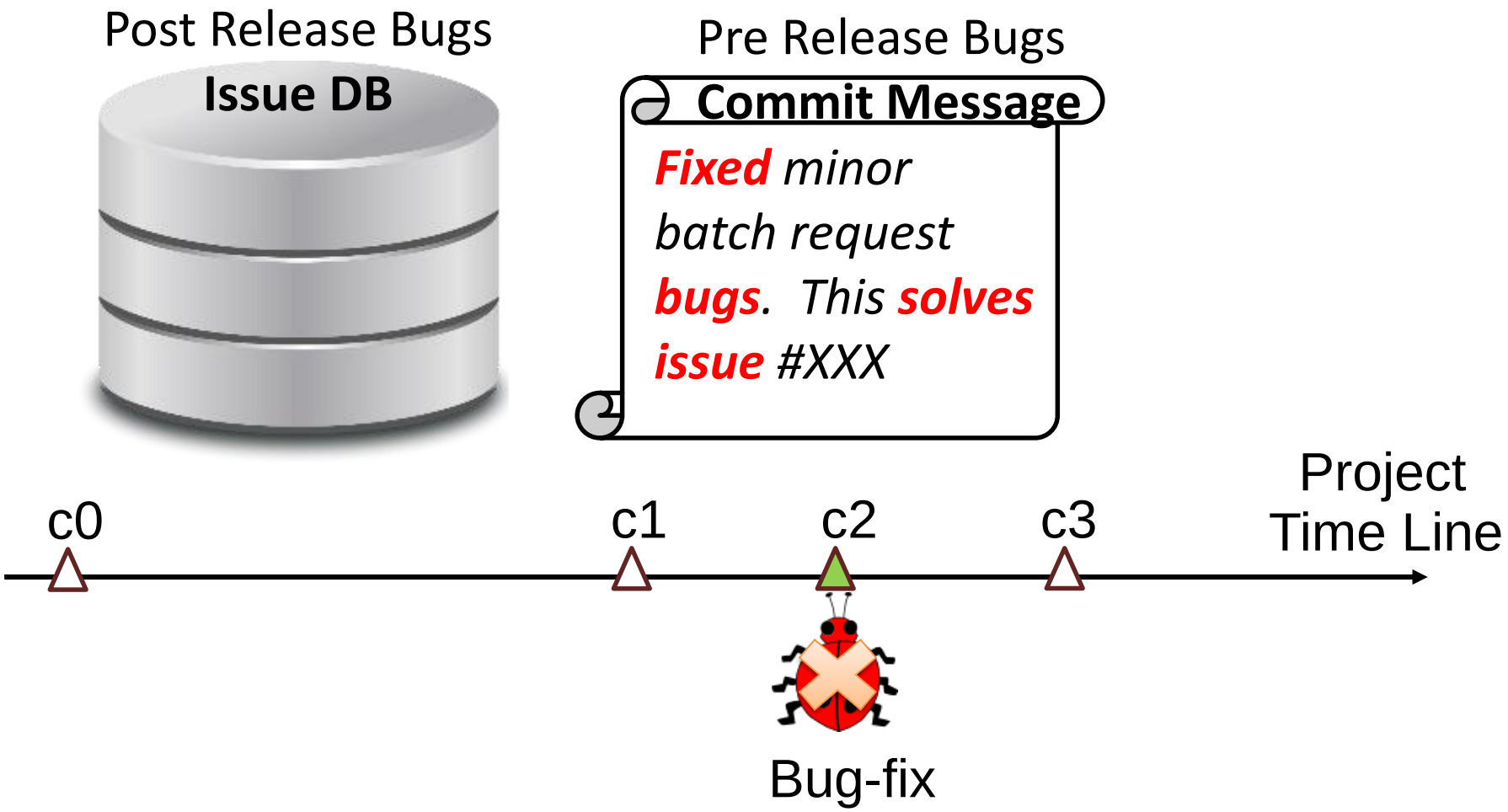
Step 1: Identify buggy lines in each version

Step 2: Measure entropy of each program line

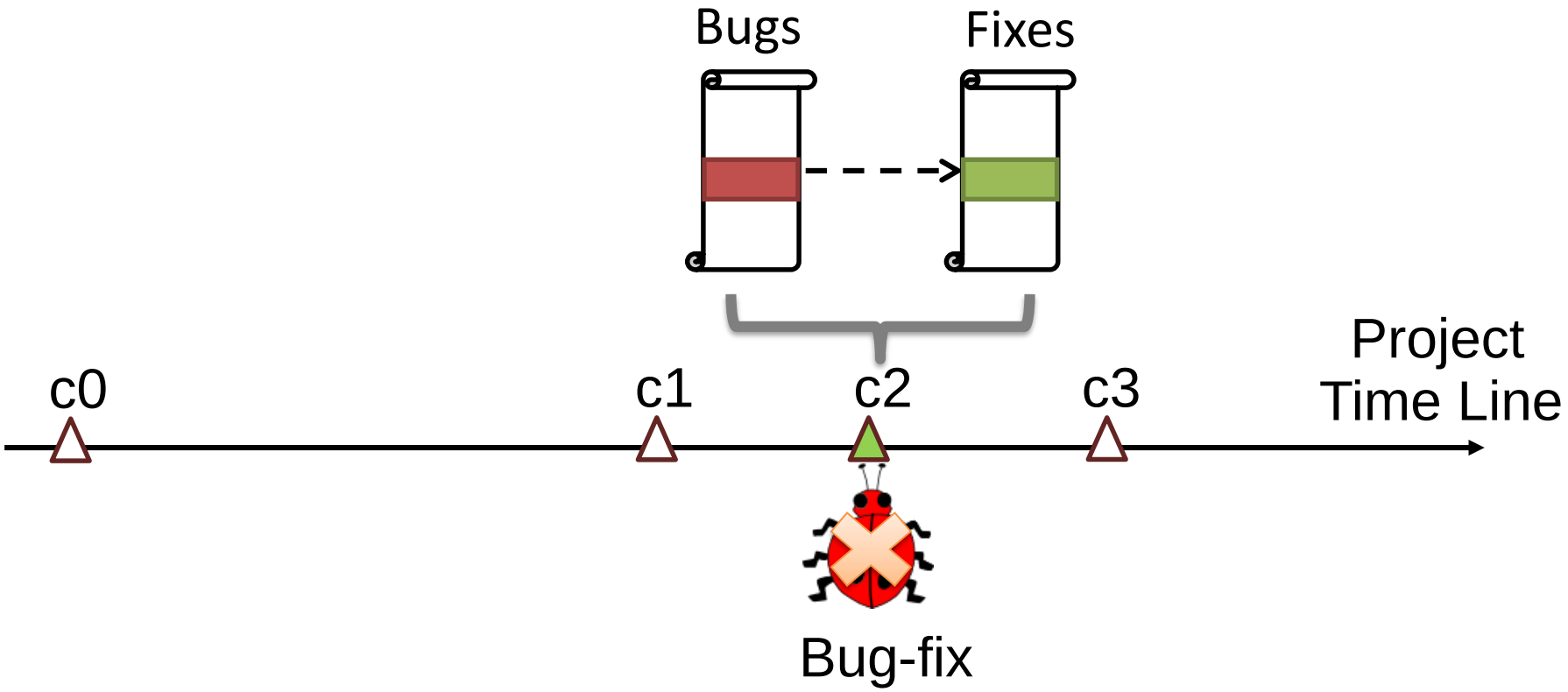
Methodology: Identify Buggy Lines



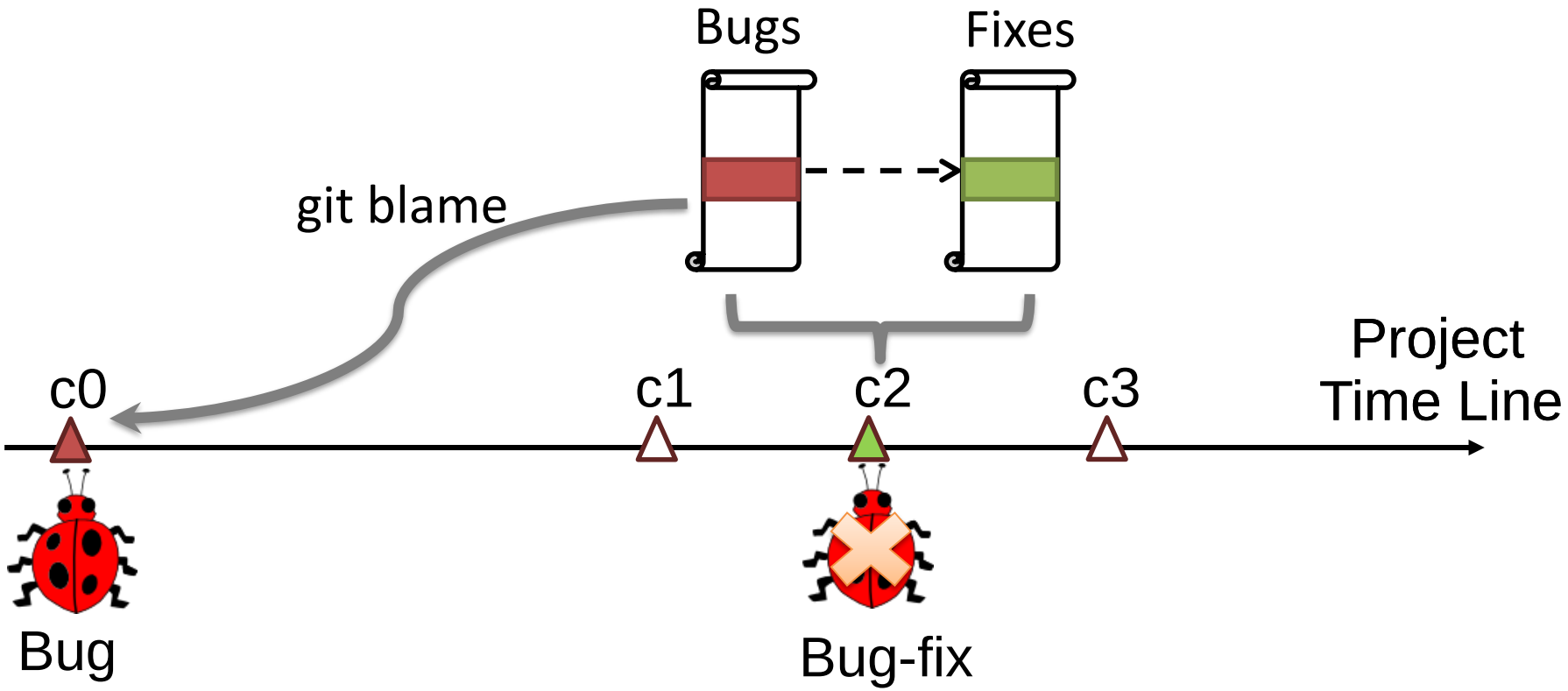
Methodology: Identify Buggy Lines



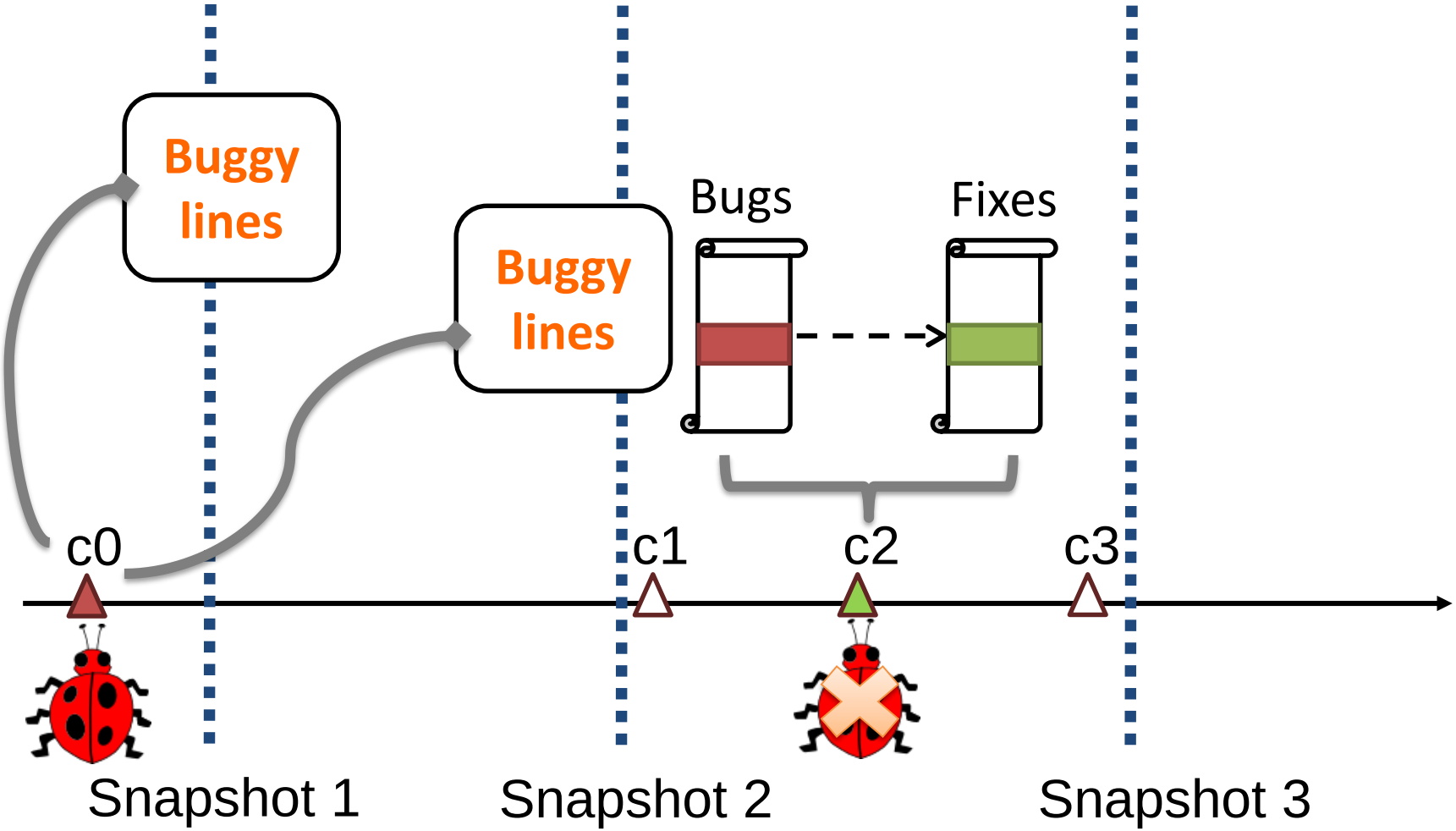
Methodology: Identify Buggy Lines



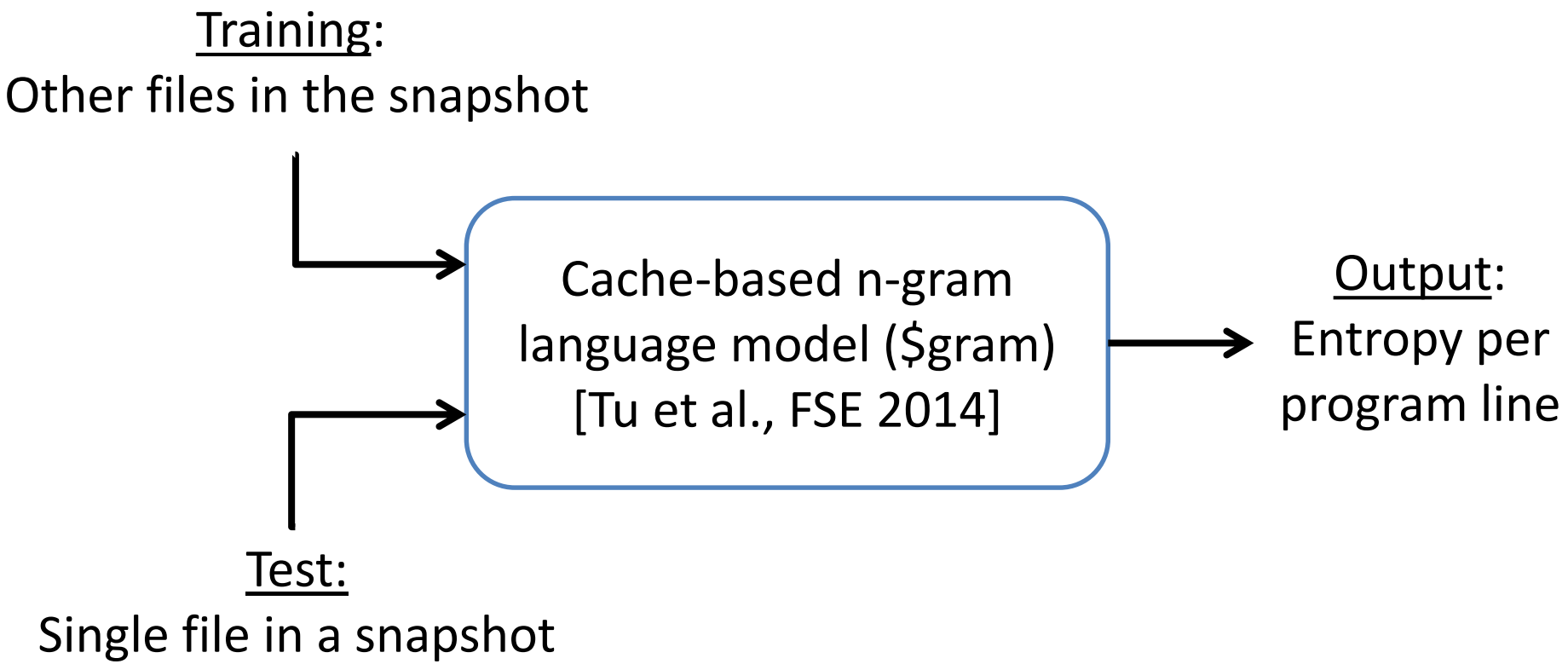
Methodology: Identify Buggy Lines



Methodology: Identify Buggy Lines



Methodology: Measure Entropy



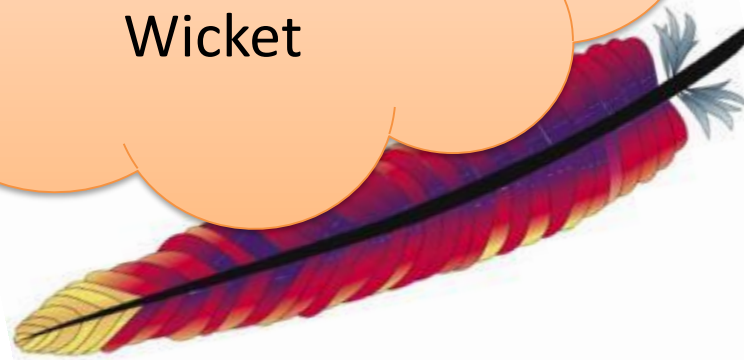
Study Subjects

GitHub

Atmosphere, Presto, Elasticsearch, Netty, Facebook-android-sdk

Apache

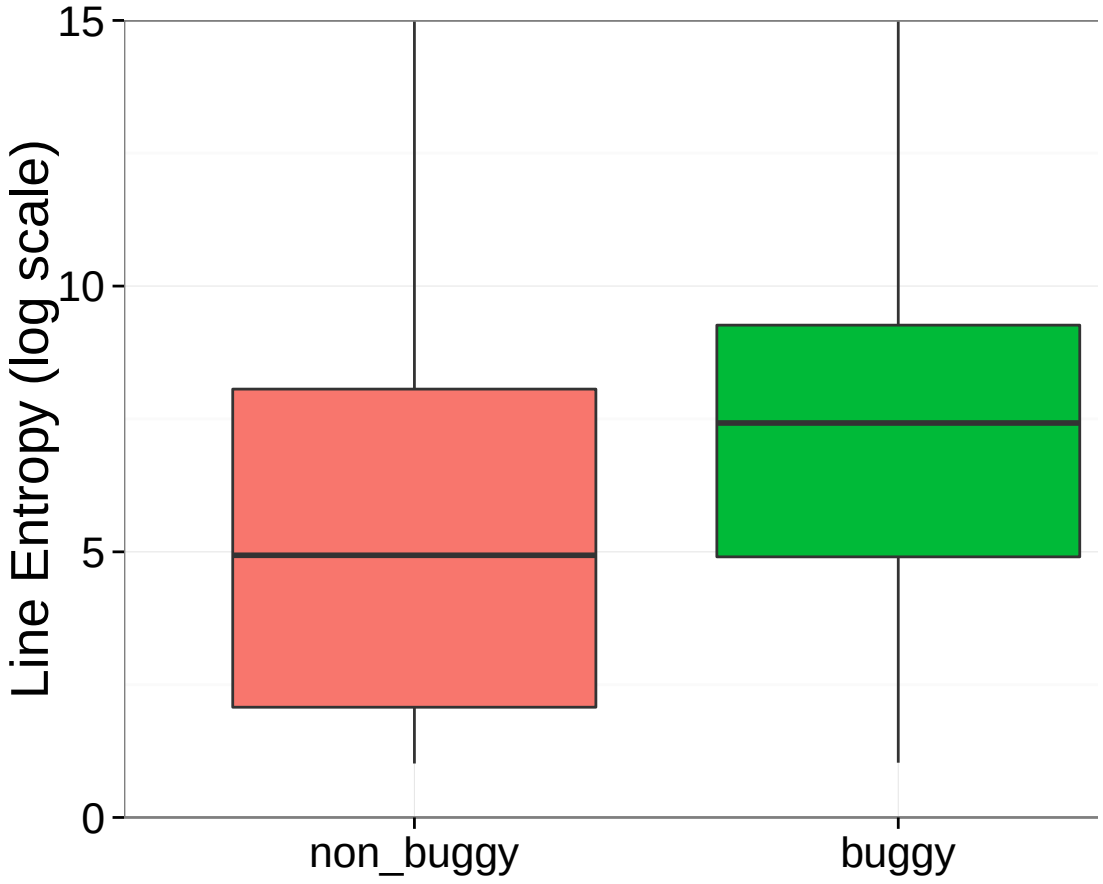
Derby, Lucene, OpenJPA, Qpid, Wicket



10 projects, 120 versions, 113K Files, 35M Lines, 7K bugs

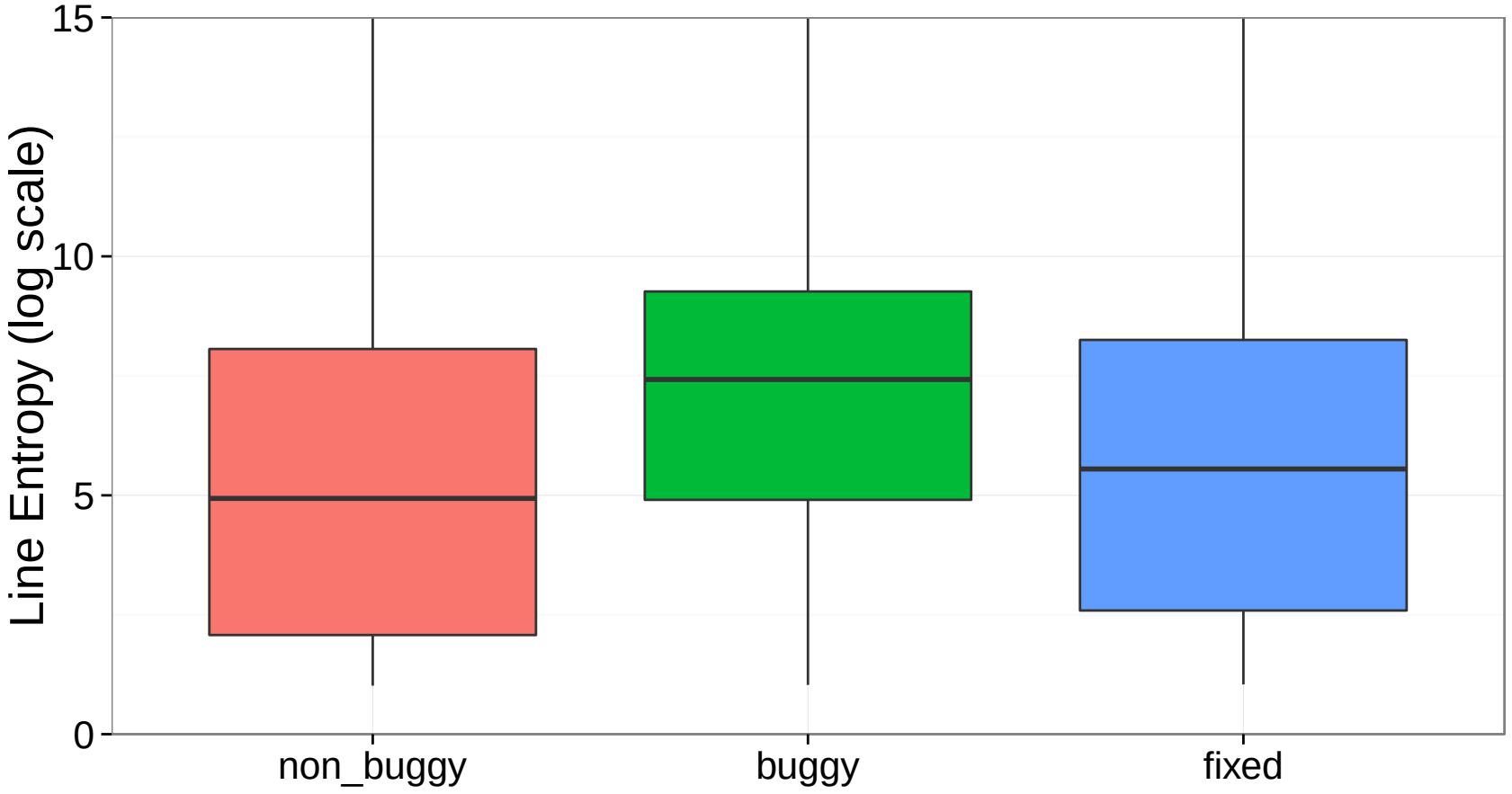

RESULTS

RQ1. Are buggy lines more “unnatural" than non-buggy lines?



The difference is more for low latency & less scattered bugs

RQ2: Do buggy lines become more “natural" after bug-fixes?



RQ2. Do buggy lines become more “natural" after bug-fixes?

Netty: incorrect method call

```
if (isTerminated()) {
  - terminationFuture.setSuccess(null);   // entropy = 5.96
  + terminationFuture.trySuccess(null);   // entropy = 1.34
}
```

Entropy dropped after bugfix : 4.63

Lucene: missing conditional check

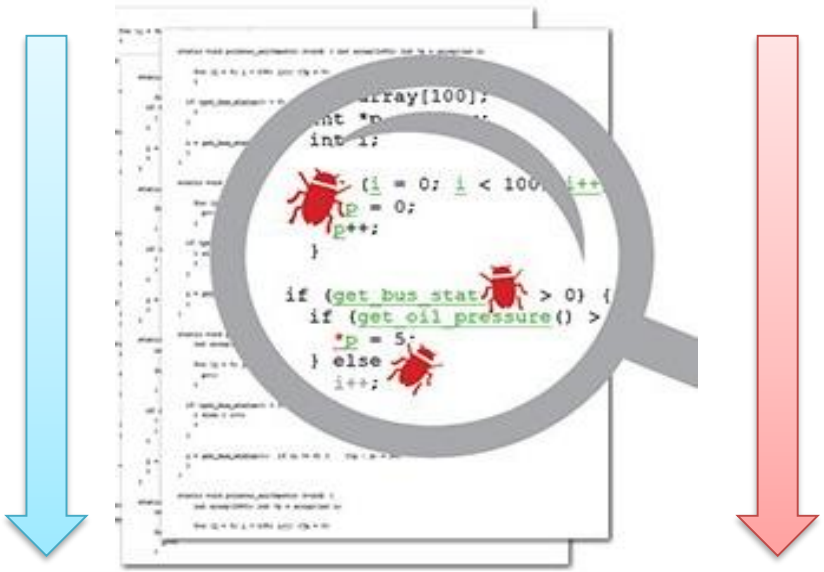
```
if (!directory.exists())
    - directory.mkdir();              // entropy = 9.21
   + if (!directory.mkdir())         // entropy = 5.34
        + throw new IOException (…)
```

Entropy dropped after bugfix : 3.87

Buggy lines have higher entropies than non-buggy lines; entropy drops after bug-fixes.

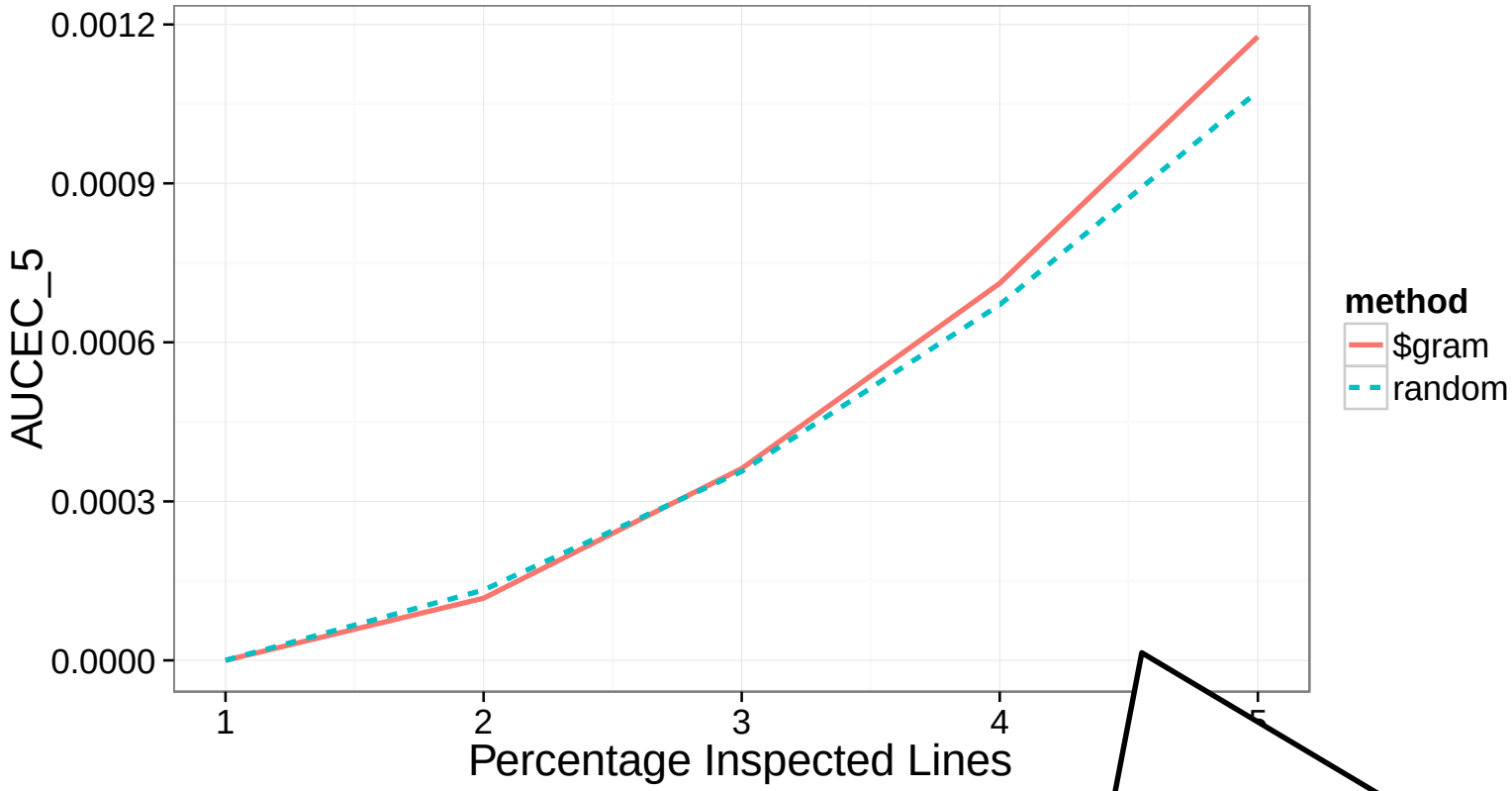
RQ3. Is unnaturalness useful for defect prediction?

Baseline:
Order lines randomly



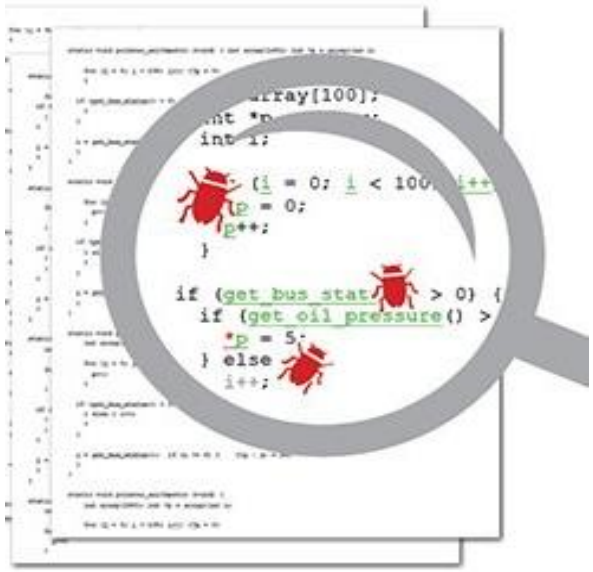
Order lines by decreasing entropy

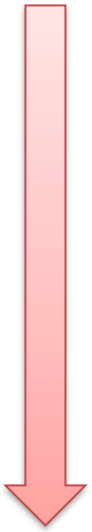
RQ3. Is unnaturalness useful for defect prediction?



AUCEC = Area under the cost-effectiveness curve.
Cost is inspection effort and payoff is number of bugs found

RQ3. Is unnaturalness useful for defect prediction?





Order lines by decreasing entropy

Problem: some line types are more entropic than others (eg., import statement vs. for loop)

RQ3. Is unnaturalness useful for defect prediction?

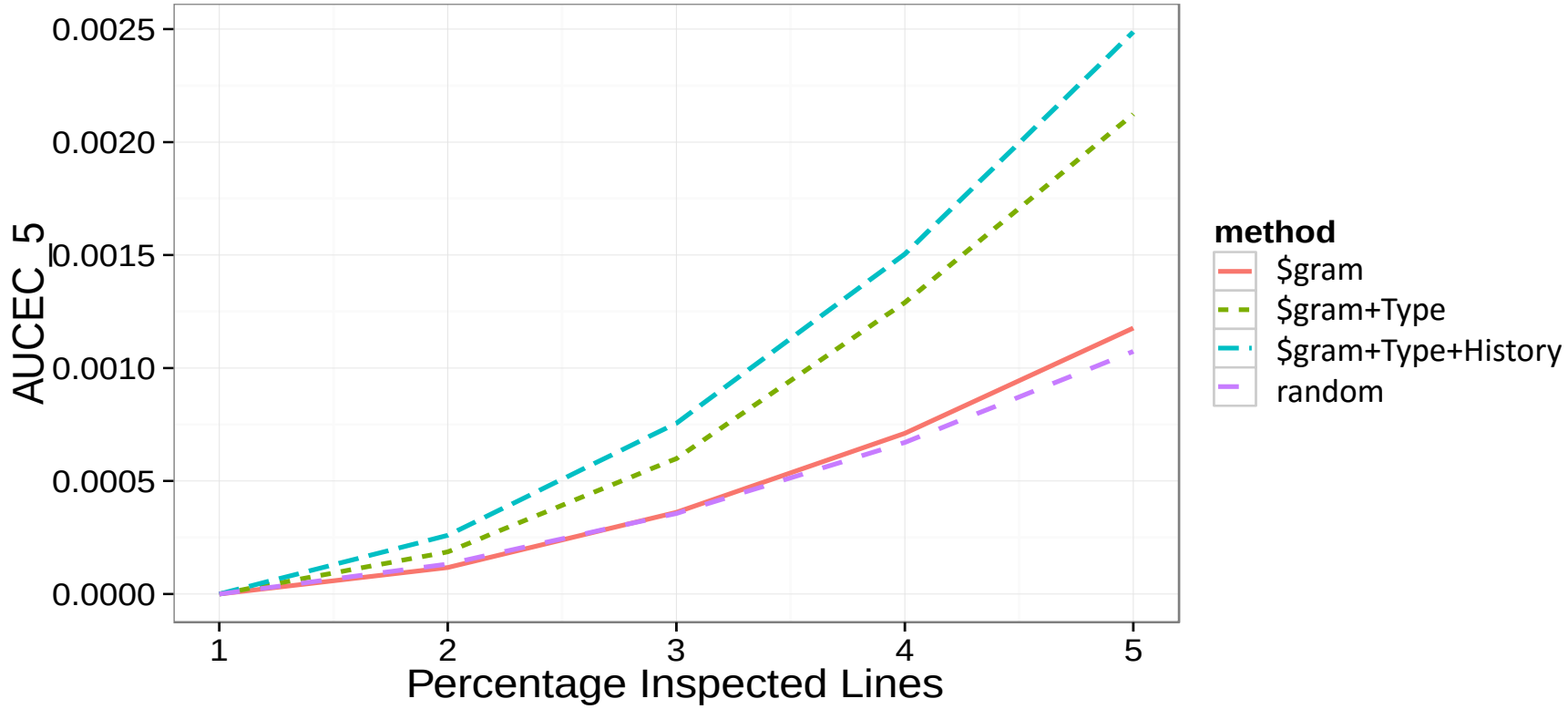
Problem: some line types are more entropic than others (eg., import statement vs. for loop)

Solution: can we leverage the line types?

$gram+Type: how much a line's entropy deviates from the mean entropy of its own line-type (normalized Z-score)

$gram+Type+History: how much a line-type was buggy in the past.

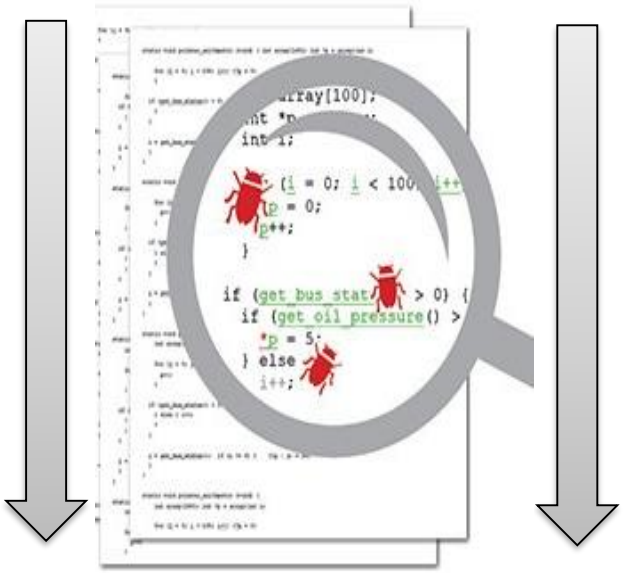
RQ3. Is unnaturalness useful for defect prediction?



Entropy can be used to guide bug-finding efforts at **line level**

RQ4. How does unnaturalness perform against static bug finding technique?

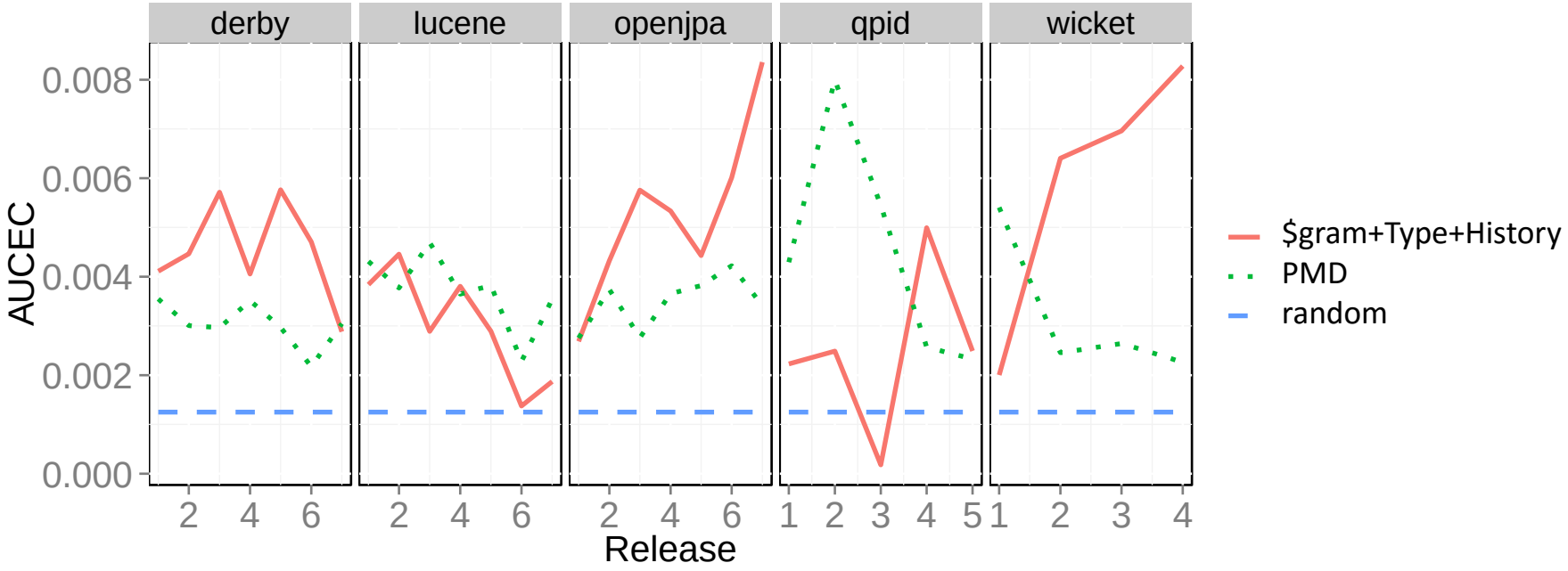
decreasing entropy



decreasing warning level of Static Bug Finder

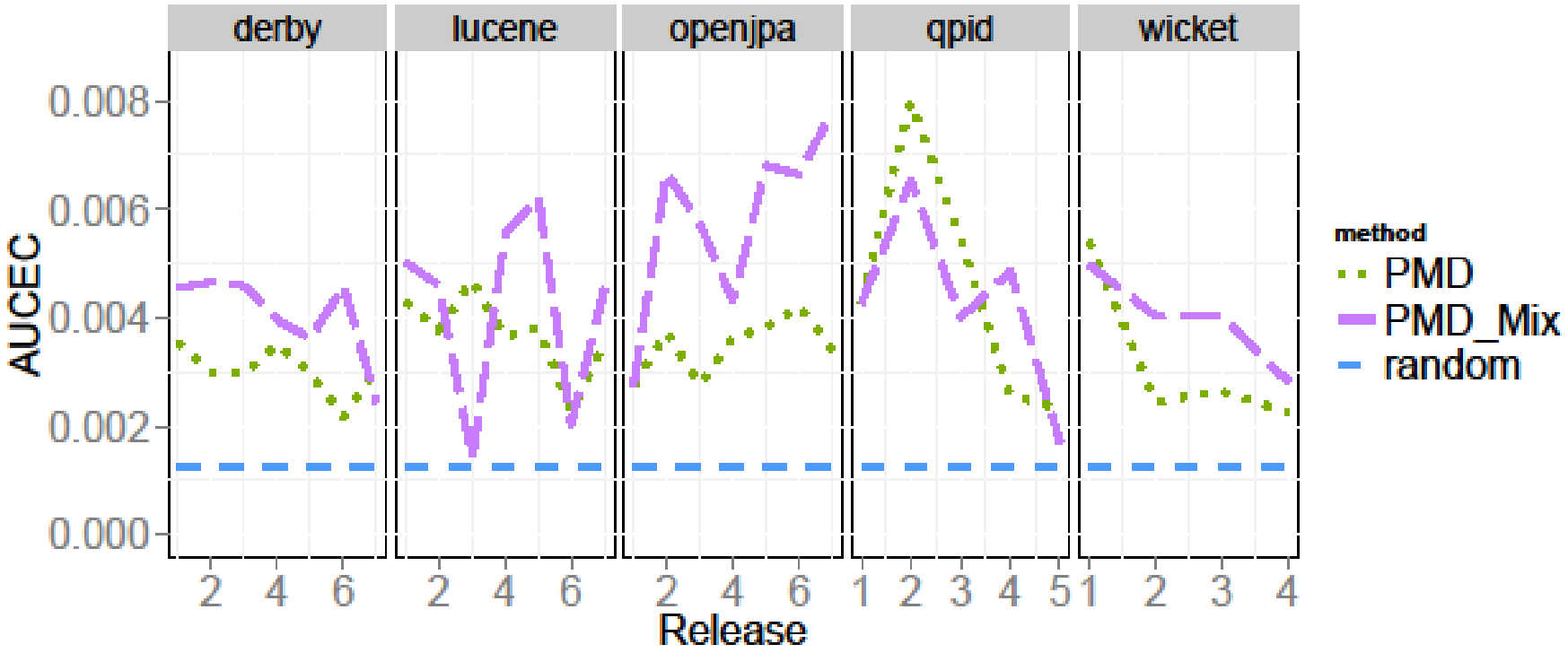
Compared against two popular static bug finding tools: FindBugs and PMD

RQ4. How does unnaturalness perform against static bug finding technique?



Entropy achieves comparable performance to commonly used Static Bug Finders in defect prediction.

RQ5. Does unnaturalness boost inspection effort on Static Bug Finder's warnings?



Ordering Static Bug Finder's warnings by priority and entropy significantly improves SBF performance.

Summary

Unnatural code

=> Buggy lines of code ✓

for Defect prediction at line granularity ✓

vs. Static analysis tools ✓

Implications: Search-based bug repair may benefit for both fault-localization and searching for fixes

Acknowledgement

National Science Foundation
(Grant 1414172)





Questions?

Unnatural code

=> Buggy lines of code ✓

for Defect prediction at line granularity ✓

vs. Static analysis tools ✓

Implications: Search-based bug repair may benefit for both fault-localization and searching for fixes

Naturalness of Code: *Cache Language Model*

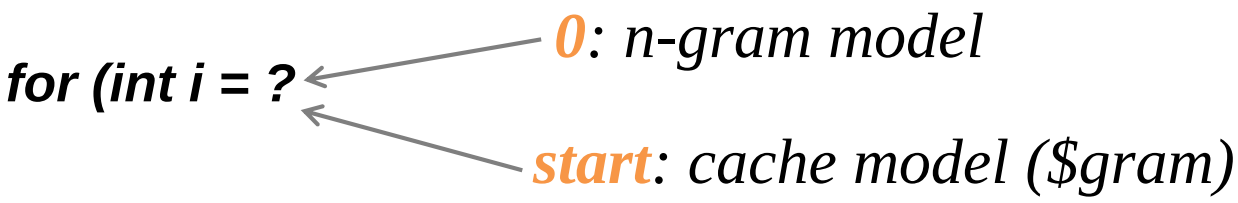
Global Context

```
for ( int i = 0; i < 10; ++ i )
…
for ( int i = 0; i < 10; ++ i )
…
for ( int i = 0; i < 10; ++ i )
…
```

Local Context

```
for ( int i = start; i < end; ++ i )
…
for ( int i = start; i < end; ++ i )
…
```

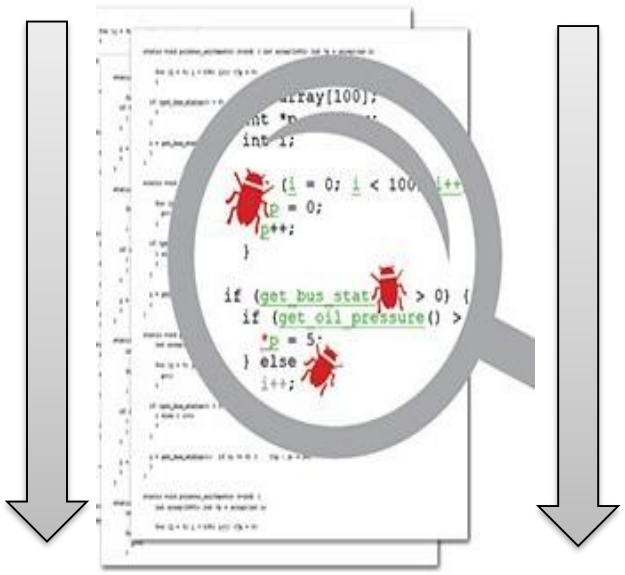
An additional cache component that memorizes the n-grams in the locality to capture [Tu et al. FSE.14]



$gram is used measure entropy of each program statement

RQ4. How does “unnaturalness” perform against static bug finding technique?

decreasing entropy



decreasing warning level of SBF

There might be unwarned lines!!

Assign 0 value to unwarned lines

Some lines will have same warning level!!

Add a random tie-breaker!!

Summary

- Unnatural code is more likely to be buggy.
- Can be effective for defect prediction, even at line granularity.
- A simple way to complement static bug finding techniques.
- Search-based bug-fixing methods may benefit from using entropy both for fault-localization and searching for fixes.

Check it out @ http://odd-code.github.io