浙江大學
ZHEJIANG UNIVERSITY

A Comprehensive Study on Quality Assurance Tools for Java

ISSTA 23

Han Liu, Sen Chen, Ruitao Feng, Chengwei Liu, Kaixuan Li, Zhengzi Xu et al.
East China Normal University, Tianjin University, UNSW, NTU

Motivation: Background

小时候读《三国演义》一直耿耿于怀的一件事，是刘备坐拥五虎将却没能一统天下；今天读完我们要介绍的这篇论文，你会发现即使坐拥五大（静态）代码缺陷检测工具（static bug detector），却现实中1%的bug都检测不出来。究竟原因是为什么呢？请看ISSRE 2023研究论文*Automatic Static Bug Detection for Machine Learning Libraries: Are We There Yet?*

Automatic Static Bug Detection for Machine Learning Libraries: Are We There Yet?

Nima Shiri harzevili*, Jiho Shin*, Junjie Wang†, Song Wang*, Nachiappan Nagappan‡
*York University; †Institute of Software, Chinese Academy of Sciences; ‡META
{nshiri,jihoshin,wangsong}@yorku.ca; junjie@iscas.ac.cn; nachiappan.nagappan@gmail.com

首先要强调的是，这不是第一篇研究代码缺陷检测工具的有效性的论文，不过本文作者主要关注的是在当前最热门的机器学习（ML）相关的代码库中找bug的需求。作者对4个非常热门的ML代码库——Mlpack、MXNet、PyTorch和TensorFlow的代码提交记录进行了分析，从中筛选出410条和bug及修复相关的commit记录，然后想用代码缺陷检测工具来扫描一把，看看到底这些已经被人工发现的bug，有几个能被代码缺陷检测工具的自动化扫描给捕获到。究竟结果

Motivation: Background

- **Motivation.** At a high level, what is the problem are you/the authors are working in and why is it important?

- Quality assurance (QA) tools are receiving more and more attention and are widely used by developers. **(Important)**

- Due to the growing size and complexity of software, developers are facing a particularly difficult situation compared to that of the past few years. **(Difficult)**

- Quality assurance (QA) tools have been widely used due to their low cost, convenience, and ability to find bugs. **(wide)**

Motivation: Problems

- **Motivation.** What is the specific problem considered in this work? This slides narrows down the topic area of the current work.

- Given the wide range of solutions for QA technology, it is still a question of **evaluating QA tools**.
  - Tool Selection
  - Datasets
  - Scanning Rules
  - Time performance

Literature and Existing Efforts

- ICSE'2018. How many of all bugs do we find? A study of static bug detectors.
- Journal of Systems and Software 2020. Some SonarQube issues have a significant but small effect on faults and changes. A large-scale empirical study.
- ISSTA'2018. Evaluating and Integrating Diverse Bug Finders for Effective Program Analysis.
- Et al.

Limitations of Existing Works

- You need to discuss the limitations of these existing approaches at a high level.

- Limitations:
  - Compare tools without considering **scanning rules analysis**.
  - They disagree on the effectiveness of tools due to the **study methodology** and **benchmark dataset**.
  - They do not separately analyze the **role of the warnings**.
  - There is no large-scale study on the analysis of **time performance**.

The Paper's Methodology

1. Select **6 free or open-source tools** from 148 existing Java QA tools.
2. Map the **scanning rules** to the **CWE** and analyze the **coverage** and **granularity** of the scanning rules.
3. Experiment on 5 benchmarks, including 1,425 bugs.
4. Assess these tools' **time performance** on 1,049 projects.

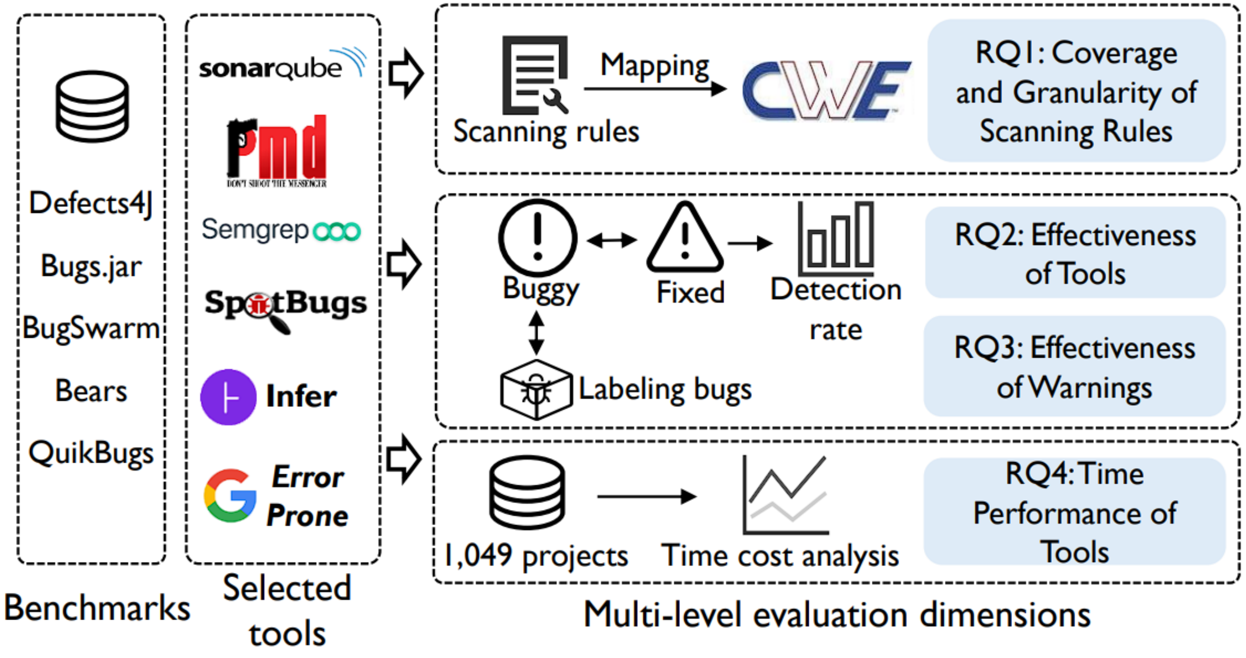

Figure 1: Overview of our study

The Paper's Methodology

Map the **scanning rules** to the **CWE**

- 418 CWE weaknesses, 40 CWE categories
- map each scanning rule to the CWE category
- map the scanning rule to CWE weakness.

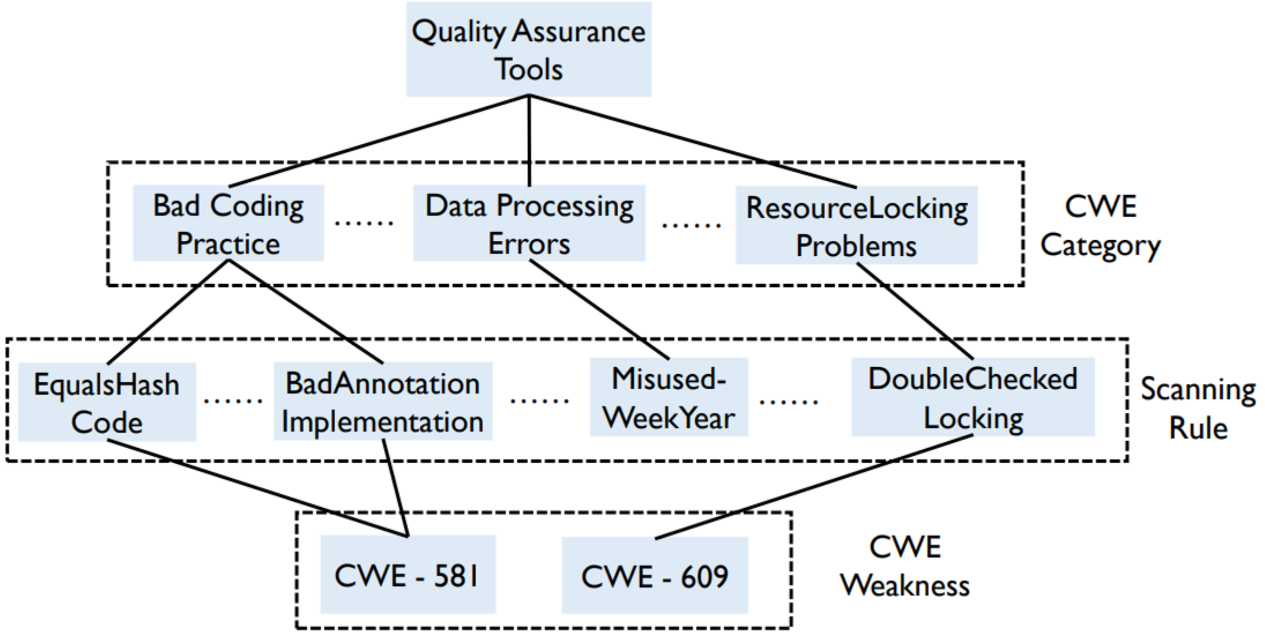

Figure 2: The mapping structure of CWEs

The Paper's Contributions

Answer following research questions.

1. **Coverage** and **Granularity** of Scanning Rules (RQ1).
   - coverage of scanning rules for different bugs
   - granularity gap of scanning rules in different tools
2. Effectiveness of Tools (RQ2).
   - Tools' **detection rates** for detecting bugs in different benchmarks
3. Effectiveness of Warnings (RQ3).
   - the gap between the warnings and the real source of bugs
4. Time Performance of Tools (RQ4).

The Paper's Contributions

Coverage and Granularity **of Scanning Rules** (RQ1).

Table 3: Rules mapping results

Tools	# mapped CWE categories	# mapped CWE weaknesses
SonarQube	28	41
SpotBugs	22	33
PMD	19	12
Error Prone	24	21
Infer	19	28
Semgrep	18	22

- The scanning rule **coverage** of QA tools needs to be **improved**.
- Each tool has its own **specific focus point**. Users can combine the features of the tools with their own needs in practice (e.g., select Infer for point issue detection).

The Paper's Contributions

Coverage and **Granularity of Scanning Rules** (RQ1).

- Number of rules mapped to CWE. (**right column**)

- The granularity of Infer, Semgrep, and SpotBugs' rules is finer than that of SonarQube, Error Prone, and PMD.

- This is achieved by detailing the result of their **sub-detectors**. But the gap between tools and CWE is still quite wide since only **a small part** of the rules are successfully mapped to CWE weaknesses.

Table 4: Top 5 CWE categories and CWE weaknesses in rules

	CWE category	CWE weakness
SonarQube	Bad Coding Practices (288, 52.2%)	CWE-476 (5)
	Error Conditions, Return Values, Status Codes (43, 7.8%)	CWE-546 (2)
	Expression Issues (36, 6.5%)	CWE-396 (2)
	Permission Issues (28, 5.1%)	CWE-477 (2)
	Data Processing Errors (25, 4.5%)	CWE-595 (2)
SpotBugs	Bad Coding Practices (187, 41.3%)	CWE-476 (9)
	Concurrency Issues (42, 9.3%)	CWE-125 (5)
	Data Processing Errors (41, 9.1%)	CWE-908 (5)
	Permission Issues (38, 8.4%)	CWE-1024 (4)
	API/Function Errors (20, 4.4%)	CWE-248 (4)
PMD	Bad Coding Practices (44, 37.0%)	CWE-252 (2)
	Complexity Issues (18, 15.1%)	CWE-609 (1)
	Error Conditions, Return Values, Status Codes (13,10.9%)	CWE-1339 (1)
	Permission Issues (7, 5.9%)	CWE-1051 (1)
	Data Processing Errors (6, 5.0%)	CWE-570 (1)
Error Prone	Bad Coding Practices (155, 38.3%)	CWE-570 (4)
	Data Processing Errors (73, 18.0%)	CWE-1024 (4)
	Error Conditions, Return Values, Status Codes (31, 7.7%)	CWE-595 (3)
	API/Function Errors (21, 5.2%)	CWE-805 (2)
	String Errors (15, 3.7%)	CWE-581 (2)
Infer	Pointer Issues (28, 23.3%)	CWE-476 (22)
	Resource Management Errors (23, 19.2%)	CWE-124 (7)
	Complexity Issues (12, 10.0%)	CWE-502 (6)
	Resource Locking Problems (11, 9.2%)	CWE-825 (4)
	Memory Buffer Errors (11, 9.2%)	CWE-413 (4)
Semgrep	Data Neutralization Issues (42, 25.6%)	CWE-611 (16)
	Data Processing Errors (25, 15.2%)	CWE-319 (16)
	Cryptographic Issues (23, 14.0%)	CWE-502 (8)
	Information Management Errors (18, 11.0%)	CWE-89 (7)
	Resource Management Errors (16, 9.8%)	CWE-94 (6)

The Paper's Contributions

Effectiveness of Tools (RQ2).

- Experiment shows that the QA tools can not detect bugs as expected.

- The main reasons for the missing detection of bugs are:
  - Insufficient scanning rule coverage,
  - Neglect of highly specific scenarios
  - **Inability** to handle **logical** or algorithmic errors
  - which are not the target of most QA tools.

Table 5: Tool detection results

Tools	# Bugs Detected in Different Benchmarks					
	Defects4J	Bugs.jar	BugSwarm	Bears	QuixBugs	Total
SonarQube	68 (8.1%)	40 (10.8%)	21 (19.4%)	3 (4.2%)	4 (10.0%)	136 (9.5%)
SpotBugs	44 (5.3%)	18 (4.9%)	21 (19.4%)	0 (0.0%)	3 (7.5%)	86 (6.0%)
PMD	90 (10.8%)	30 (8.1%)	22 (20.4%)	3 (4.2%)	1 (2.5%)	146 (10.2%)
Error Prone	61 (7.3%)	40 (10.8%)	5 (4.6%)	2 (2.8%)	1 (2.5%)	109 (7.6%)
Infer	10 (1.2%)	4 (1.1%)	4 (3.7%)	1 (1.4%)	0 (0.0%)	19 (1.3%)
Semgrep	0 (0.0%)	0 (0.0%)	4 (3.7%)	0 (0.0%)	0 (0.0%)	4 (0.3%)
Total Bugs	835	371	108	71	40	1,425

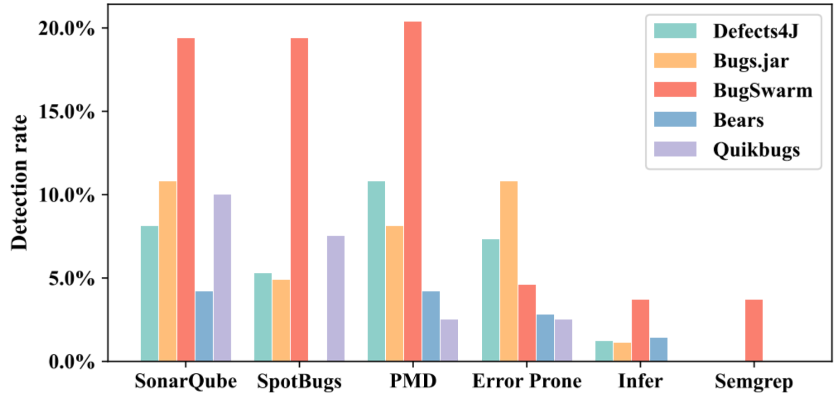

Figure 3: Detection results in different benchmarks

The Paper's Contributions

Effectiveness of Warnings (RQ3).

- Gap between the warnings and the real bugs to study the **precision**.

- Compared with the real reasons for bugs, only **a few warnings are effective**, and most of them refer to Bad Coding Practice. Although some of the warnings are bug-sensitive, they are **not actually the real cause** of the bugs.

- While some warnings reported by the tool are **not indicative of actual bugs**, a portion (26%) of these warnings (e.g., null point dereference) **do provide valuable insights** for identifying bugs within certain contexts. Nevertheless, a significant percentage of these warnings (74%), such as **useless parentheses**, prove to be of limited usefulness in pinpointing specific bugs.

The Paper's Contributions

Time Performance of Tools (RQ4).

- For big projects, **Infer** is much slower due to its **complex analysis**.
- **Semgrep** is barely influenced by the size of projects due to its unique paralleled processing of **splitted project components**.

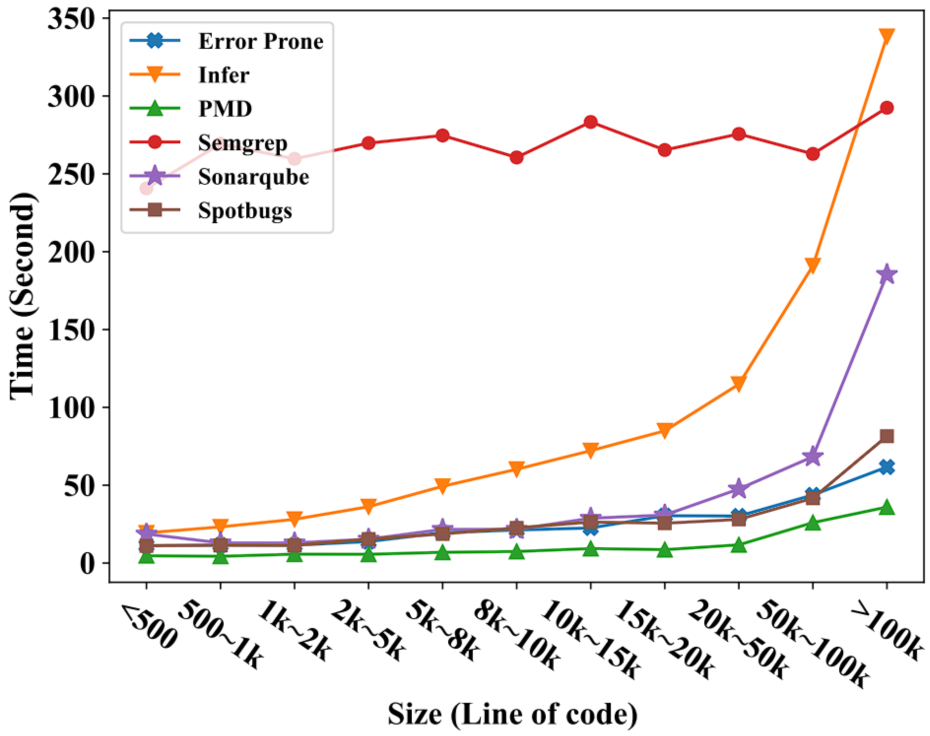

Figure 4: Tool execution time with project size

Summary

- A comprehensive study on 6 Java QA tools in multi-level dimensions.
- To better understand the coverage and granularity of the scanning rules of the tools, we mapped a total of 1,813 rules to CWE.
  - A benchmark experiment to reveal the effectiveness of tools.
- A large-scale experiment to analyze the time performance.
- Unveils many useful findings,
  - Comparison of the **coverage** and **granularity** of scanning rules
  - Detection rate and reasons for missed bugs
  - The role of warnings in bug detection
  - Execution time
  - Reasons for the difference between tools.