ICSE2023

Can static analysis tools find more defects?

A qualitative study of design rule violations found by code review

Empirical Software Engineering, Volume 28, Issue 1 (2023)

Sahar Mehrpour (smehrpou@gmu.edu)
Thomas LaToza (tlatoza@gmu.edu)

Developer Experience Design Laboratory

GEORGE MASON UNIVERSITY

What is a defect?

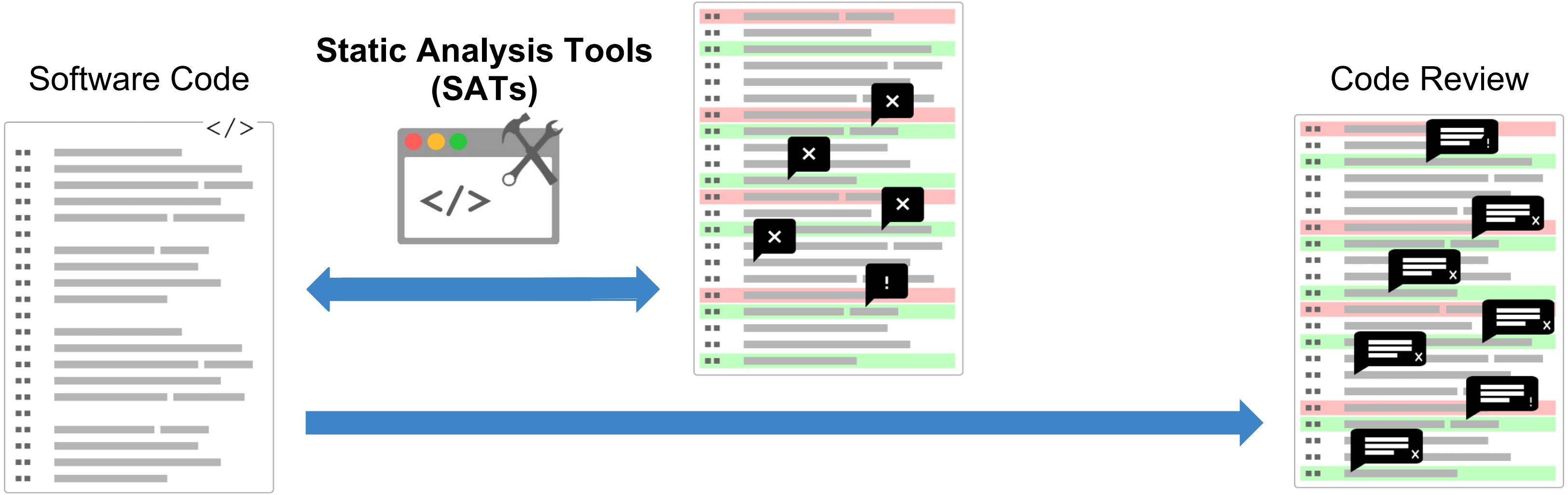
- Code which may lead to a **system failure**. [1]
  - "Event is thrown before registration."
- Code that reduces the **quality** of the codebase. [2]
  - "Don't use print, instead use loggers."

✓ Any code that a code reviewer thinks is wrong. [3]

  - "New code misses test cases."
  - "Don't keep commented out code."

[1] Laitenberger O (1998) *Studying the effects of code inspection and structural testing on software quality*. International symposium on software reliability engineering (ISSER), pp 237–246.
[2] Gilb T, Graham D, Finzi S (1993) *Software inspection*. Addison-Wesley.
[3] Mäntylä MV, Lassenius C (2009) *What types of defects are really discovered in code reviews?* Transactions on Software Engineering 35(3):430–448.

Software tools can help in identifying defects.



Which questions are answered?

Prior studies examined the ability of **individual** SATs to detect known defects.

What is the **potential** of different **types** of SATs to find defects?

Our research questions:

- **How many** defects can be potentially be **detected** by SATs?
- What **features** should be added to SATs to detect **more** defects?

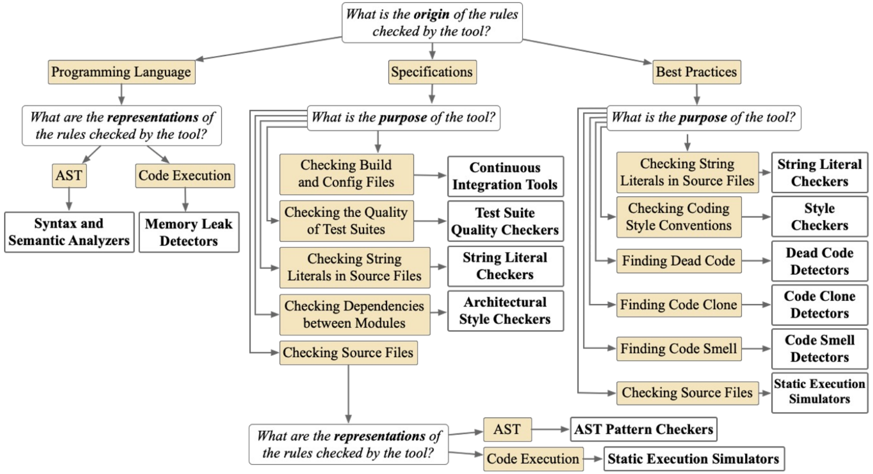
Our methodology

- We collected1323 review comments in pull requests of public GitHub repositories.
- We examined the defects found in the review comments, and mapped them to violations of rules.
- Some rules can be checked by SATs.
- We matched the characteristics of SATs and rules in our dataset.

Our SAT taxonomy

We created a taxonomy of SATs with 12 classes.

- AST Pattern Checkers
  - Check rules expressed as AST patterns
- Style Checkers
  - Check rules related to code style conventions
- Code Clone Detectors
- Dead Code Detectors
- …



*What is the **origin** of the rules checked by the tool?*

Programming Language

*What are the **representations** of the rules checked by the tool?*

AST → **Syntax and Semantic Analyzers**

Code Execution → **Memory Leak Detectors**

Specifications

*What is the **purpose** of the tool?*

Checking Build and Config Files → **Continuous Integration Tools**

Checking the Quality of Test Suites → **Test Suite Quality Checkers**

Checking String Literals in Source Files → **String Literal Checkers**

Checking Dependencies between Modules → **Architectural Style Checkers**

Checking Source Files

*What are the **representations** of the rules checked by the tool?*

AST → **AST Pattern Checkers**

Code Execution → **Static Execution Simulators**

Best Practices

*What is the **purpose** of the tool?*

Checking String Literals in Source Files → **String Literal Checkers**

Checking Coding Style Conventions → **Style Checkers**

Finding Dead Code → **Dead Code Detectors**

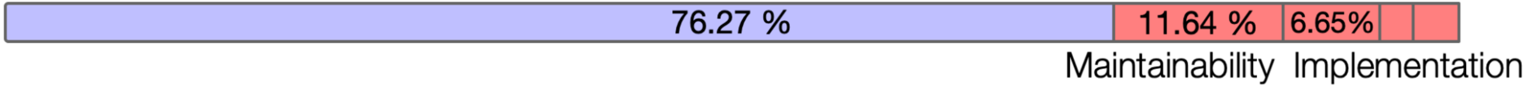
Finding Code Clone → **Code Clone Detectors**

Finding Code Smell → **Code Smell Detectors**

Checking Source Files → **Static Execution Simulators**

How many defects are potentially detectable by SATs?

Potentially detectable by SATs **Not detectable**



- **76%** of defects found in code review could be potentially detected by SATs.
- Half of them by **AST Pattern Checkers** and **Style Checkers.**
- Defects not detectable by SATs are mainly **Maintainability** defects and **Implementation** defects.

What features in SATs are necessary to detect more defects?

- Tools should be extended to support configuring and checking **project-specific** rules.
  - "Instead of status code, return 'success' or readable error message.”
- **AST Pattern Checkers** should support configuring and checking **crosscutting** rules through executing multiple AST queries.
  - "If a method exists both for a class and its parents, then it can override, so it doesn't require repeated annotations.”
- **Style Checkers** may require **complex parsers** or apply **dynamic analysis** to check more complex rules.
  - "Order method definitions according to the order they are called.”

What we learned?

- We studied the **potential of SATs** to detect more defects.
- We found that SATs have a great potential to **detect more defects** (76% of defects found in code review).
- There are many **project-specific defects**.
- **AST Pattern Checkers** and **Style Checkers** are the most applicable SATs (more than 50%).
- Some defects require **human judgement** to be detected. Tools using information like identifiers, comments, or developers' knowledge might detect them.
  - "Add a comment for a *complex* block explaining the procedure."

Can static analysis tools find more defects?

A qualitative study of design rule violations found by code review

Sahar Mehrpour (smehrpou@gmu.edu)

Thomas LaToza (tlatoza@gmu.edu)

ICSE2023

- We studied the **potential of SATs** to detect more defects.
- SATs have great potentials to **detect more defects** (76% of defects found in code review).
- There are many **project-specific defects**.
- **AST Pattern Checkers** and **Style Checkers** are the most applicable SATs.
- Some defects require **human judgement** to be detected.

Developer Experience Design Laboratory

GEORGE MASON UNIVERSITY