Q0.

Survey on Static Analysis Tools: Evaluating the Effectiveness of Code Review Historical Data for Rule Recommendations

We appreciate your participation in this survey, as we aim to assess the effectiveness of utilizing code review historical data for recommending rules in static analysis tools. Your valuable insights and feedback will contribute to enhancing the quality and efficiency of static analysis processes. Please take a few moments to share your experiences and opinions regarding this topic.

We would like to assure you that all the data collected through this survey will be used solely for academic research purposes. Your responses will be treated with the utmost confidentiality and will only be analyzed in an aggregated and anonymized form.

If you have any concerns or questions regarding the data collection process, please feel free to contact us. Thank you for your cooperation and valuable contribution to our academic research.

Q1. How many years of experience do you have in software development?

- ○ Less than 1 year
- ○ 1-3 years
- ○ 3-5 years
- ○ 5-10 years
- ◉ More than 10 years

Q22. Which programming language(s) do you primarily use in your work or development projects? (Select all that apply)

This question was not displayed to the respondent.

Q19. How familiar are you with the practice of code review?

- ◉ Very familiar, I regularly participate in code reviews
- ○ Familiar, I have some experience with code reviews
- ○ Somewhat familiar, I have limited experience with code reviews
- ○ Not familiar, I have little to no experience with code reviews

Q2. Have you ever used static analysis tools in your project to check code quality and identify potential issues?

- ◉ Yes
- ○ No

Q3. If you have used static analysis tools, please choose or list the names or platforms of the tools you have used.

- ☑ SonarQube

- [ ] ESLint
- [x] PMD
- [x] Checkstyle
- [x] FindBugs
- [x] Others spotbugs

Q3. How frequently do you use static analysis tools in your daily development work?

- ( ) Every day
- ( ) Every week
- ( ) Every month
- (•) Others every commit

Q4. How effective do you think static analysis tools are in improving code quality and identifying potential issues?

- ( ) Not effective at all
- ( ) Slightly effective
- ( ) Moderately effective
- (•) Very effective
- ( ) Extremely effective

Q5. When using static analysis tools, which aspect do you prioritize the most?

- ( ) Identifying potential errors or vulnerabilities
- (•) Checking code conventions and style
- ( ) Performance optimization and resource management
- ( ) Others (Please specify)

Q6. Have you encountered any of the following issues when using static analysis tools? (Select multiple options if applicable)

- [ ] Too many false positives (incorrect warnings)
- [ ] Hard to understand reports or results
- [ ] Significant performance impact
- [ ] Integration and deployment difficulties
- [ ] Other (please specify)
- [x] Haven't encountered any issues

Q7. Do you write or refactor code based on the rule set provided by static analysis tools?

- (•) Yes
- ( ) No

Q8. If you have used the rule set provided by static analysis tools, please rate the following aspects:

Completeness and applicability of the rules	1
Clarity and ease of understanding of the rules	2
Existence of too many redundant rules	3
Existence of critical rules missing	4
Other (please specify)	5

Q9. Have you customized the rule set of static analysis tools to meet specific project or team requirements?

- (●) Yes, I frequently customize the rule set
- ( ) Yes, I occasionally customize the rule set
- ( ) No, I have never customized the rule set

Q10. In your opinion, what is the most challenging aspect of customizing the rule set for static analysis tools?

- (●) Complexity of rule writing and configuration
- ( ) Determining the scope of rule set applicable to the project
- ( ) Addressing false positives and false negatives
- ( ) Understanding the impact of rules on code quality
- ( ) Resource and time constraints
- ( ) Other (please specify)

Q11. Do you think utilizing code review historical data for recommending static rules is feasible? Why?

- [x] Yes, historical data can provide insights into common code issues and patterns
- [ ] Yes, it can help identify recurring issues and improve rule recommendations
- [ ] No, historical data may not accurately represent current code quality
- [ ] Others

Q12. Do you think utilizing code review for recommending static rules would be helpful for the usage of static analysis tools?

- (●) Yes, it can provide more accurate and targeted static analysis results
- ( ) No, it doesn't have apparent benefits in practical usage
- ( ) Others (Please explain why)

Q21. Are you a developer or a user of pinot project?

This question was not displayed to the respondent.

Q20. When participating in code reviews, which role do you primarily take on in pinot project?

- ◯ Developer (who commit the changes in reviews)
- ◯ Reviewer
- ◉ Both
- ◯ N/A

Q14. Among the following 30 rules, half of them consist of the top 15 rules that were selected based on reviewer comments on **apache/pinot (https://github.com/apache/pinot)** repository and received the most attention. The other half consists of randomly selected rules from the remaining set. Now, please rate the importance of the following rules on a scale of 1 to 5. (The order of the rules presented below is random.)

Classes should not be empty	4
Inappropriate "Collection" calls should not be made	4
Two branches in a conditional structure should not have exactly the same implementation	5
Null should not be returned from a "Boolean" method	1
The value returned from a stream read should be checked	3
Loops should not be infinite	5
Return values from functions without side effects should not be ignored	3
Redundant casts should not be used	4
Mutable fields should not be "public static"	5
Boolean literals should not be redundant	4
Constant names should comply with a naming convention	5
Track uses of "FIXME" tags	4
Standard outputs should not be used directly to log anything	4
Sections of code should not be commented out	4
Abstract classes without fields should be converted to interfaces	2
JUnit assertTrue/assertFalse should be simplified to the corresponding dedicated assertion	4
Classes extending java.lang.Thread should override the "run" method	5
Unicode Grapheme Clusters should be avoided inside regex character classes	5
Regexes containing characters subject to normalization should use the CANON_EQ flag	5
Private fields only used as local variables in methods should become local variables	2

Type parameter names should comply with a naming convention	5
Raw byte values should not be used in bitwise operations in combination with shifts	2
"URL.hashCode" and "URL.equals" should be avoided	4
Delivering code in production with debug features activated is security-sensitive	3
Ints and longs should not be shifted by zero or more than their number of bits-1	4
Methods and field names should not be the same or differ only by capitalization	2
"@Deprecated" code marked for removal should never be used	4
Getters and setters should access the expected fields	4
Nested "enum"s should not be declared static	1
Regular expressions should not overflow the stack	4

Q16.

Below are the 15 rules selected based on reviewer comments for pinot project.

1. Classes should not be empty Inappropriate.
2. "Collection" calls should not be made.
3. Two branches in a conditional structure should not have exactly the same implementation.
4. Null should not be returned from a "Boolean" method.
5. The value returned from a stream read should be checked.
6. Loops should not be infinite.
7. Return values from functions without side effects should not be ignored.
8. Redundant casts should not be used.
9. Mutable fields should not be "public static".
10. Boolean literals should not be redundant.
11. Constant names should comply with a naming convention.
12. Track uses of "FIXME" tags.
13. Standard outputs should not be used directly to log anything.
14. Sections of code should not be commented out.
15. Abstract classes without fields should be converted to interfaces.

Do the results align with your expectations?

- (●) Yes
- ( ) No (please explain why)

Q13. If you have any additional comments or suggestions, please provide them below.

Location Data

Location: (40.4163, -3.6934)

Source: GeoIP Estimation