U F M G

lab·soft
software engineering laboratory

A systematic literature review of code smell detection tools for JavaScript systems

Saymon Souza

08/11/2024

Code Smells

- A code smell is any symptom in the source code of a program that possibly indicates a problem.

    - While this concept originally, first introduced by Fowler, applied to code that broke the principles of object-oriented programming, it soon expanded to include more languages and programming paradigms.

JavaScript

- JavaScript is an interpreted and dynamically typed programming language that stands out year after year in popularity surveys due to its ability to adapt to different contexts.

    - Due to this flexibility, various challenges arise in maintaining the quality of software built with this language.

Automated detection tools

- More and more automated tools are emerging to reduce the manual effort needed to prevent and correct code smells.

    - For the JavaScript language, for example, tools like JSNose, JSHint, and JSLint are part of the technological toolkit used by development teams around the world.

Goal

- Although the number of automated code smell detection tools for JavaScript continues to grow, there is still a need for a comprehensive study that analyzes the key characteristics of each tool.
    - This paper aims to conduct a systematic literature review (SLR) to fill this gap in the literature.

Research questions

RQ1: What is the publication landscape of code smell detection tools in JavaScript since the release year of the language?

RQ2: What code smell detection tools were published so far and what are their main features?

Systematic literature review

- The systematic literature review process used was based on the guidelines proposed by Kitchenham[1].
    - This process involves three main stages: planning, conducting and reporting.

[1] Barbara Kitchenham and Stuart Charters. 2007. **Guidelines for performing systematic literature reviews in software engineering**. TechnicalReport EBSE-2007-01. Keele University and University of Durham.

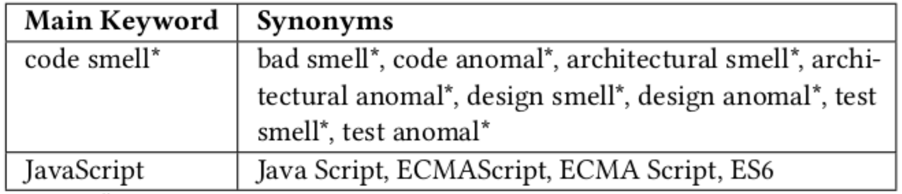
Systematic literature review

- The search string was created taking into consideration some variations for identifying the code smell expression, as well as JavaScript.
    - It was also considered that, for this work, it should include a term to filter for detection tools.

Systematic literature review

- This string was used to query primary studies in the Scopus database based on their metadata.

Table 2. List of keywords that constitute our search string



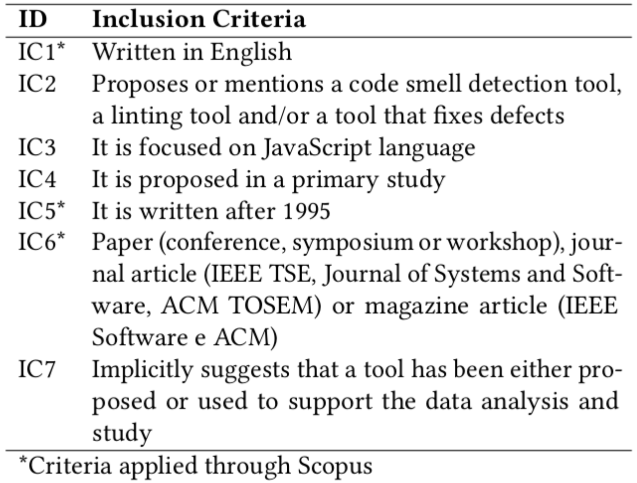
Main Keyword	Synonyms
code smell*	bad smell*, code anomal*, architectural smell*, architectural anomal*, design smell*, design anomal*, test smell*, test anomal*
JavaScript	Java Script, ECMAScript, ECMA Script, ES6

Note: "*" is a wildcard character

Systematic literature review

- At first 360 primary studies were found, but most of it would not fit this work, so the following filtering criteria were used:

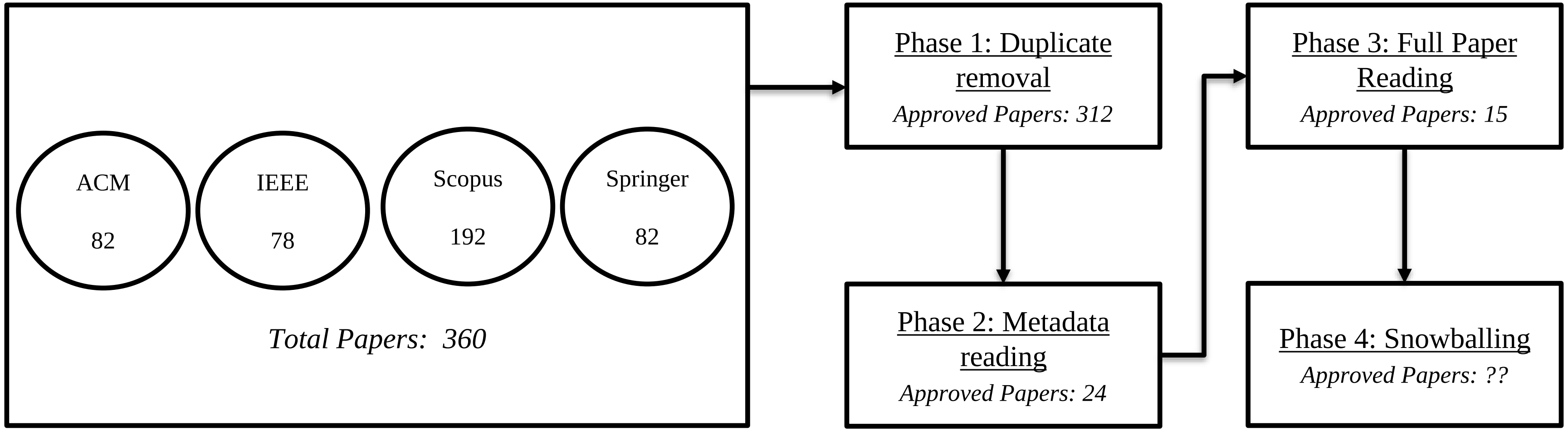
Table 3. Filtering Criteria



ID	Inclusion Criteria
IC1*	Written in English
IC2	Proposes or mentions a code smell detection tool, a linting tool and/or a tool that fixes defects
IC3	It is focused on JavaScript language
IC4	It is proposed in a primary study
IC5*	It is written after 1995
IC6*	Paper (conference, symposium or workshop), journal article (IEEE TSE, Journal of Systems and Software, ACM TOSEM) or magazine article (IEEE Software e ACM)
IC7	Implicitly suggests that a tool has been either proposed or used to support the data analysis and study

*Criteria applied through Scopus

Systematic literature review



Systematic literature review

- With these criteria, a manual checking on the primary studies resulted in **15 remaining papers**, all of them proposing a tool that could evaluate a JavaScript system and check for one or more code smells.
    - We ran our final search string on August 13, 2024.

Research question #1

- What is the publication landscape of code smell detection tools in JavaScript since the release year of the language?
    - **Trending upwards:** There has been a recent expansion in the literature regarding the study of code smell detection tools in JavaScript.

RQ1 – Frequency of paper publication by year

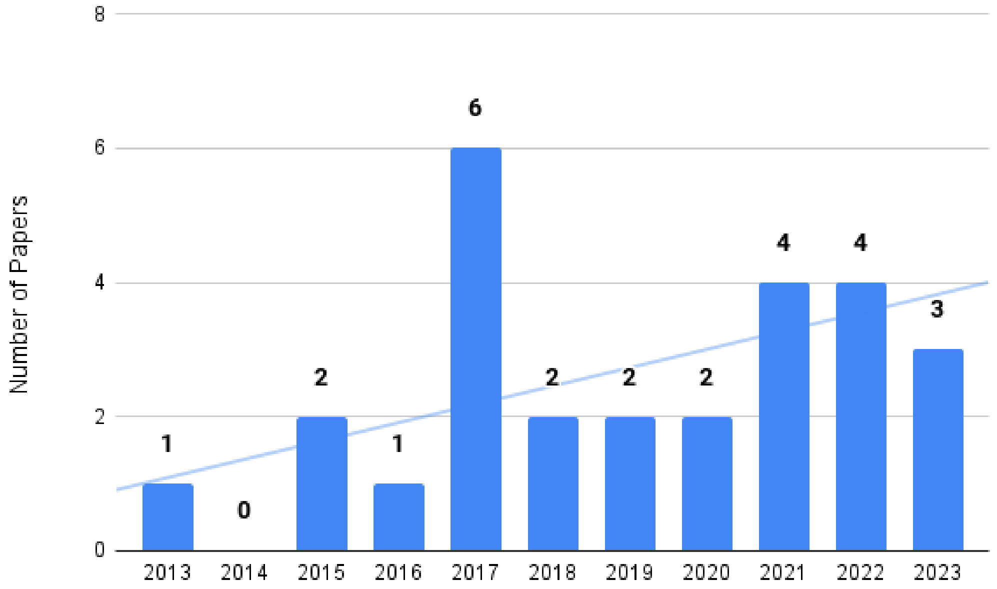


Figure 1: Frequency of paper publication by year

RQ1 – Trends

- It is noticeable that since 2015, the number of publications has significantly increased, with the peak occurring in 2017, with 6 papers published.
    - The same trending can be seen in conferences and journals.

RQ1 – Frequency in conferences

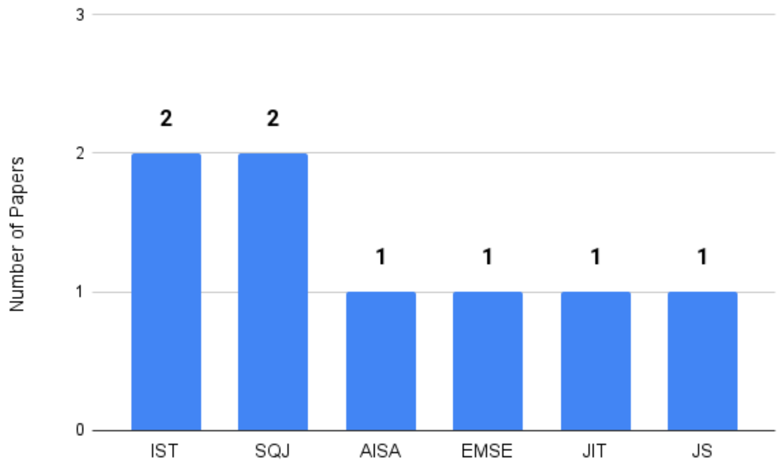


Figure 2: . Frequency of the topic discussion in conferences

RQ1 – Frequency in journals

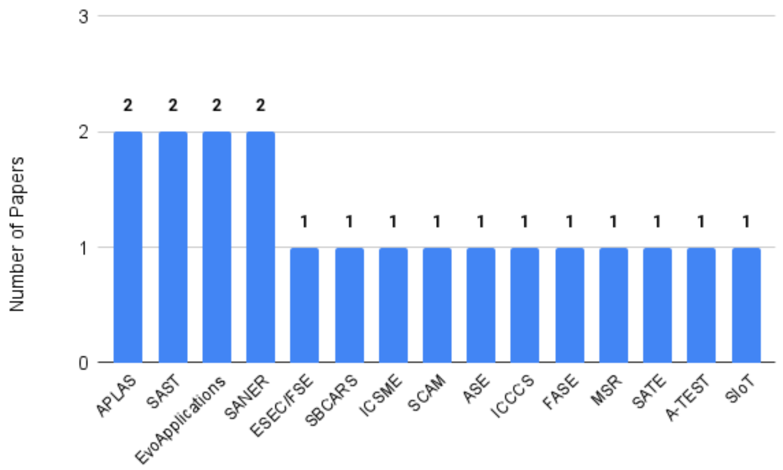


Figure 3: Frequency of the topic discussion in journals

RQ1 – Summary

- There is considerable interest in the literature for studying code smell detection tools in JavaScript systems.
  - From 2015 to 2023 alone, over 25 studies were published proposing and analyzing such tools.
  - These studies were published in major conferences and journals, reinforcing the relevance of the topic and the need for a systematic literature review to compile a set of already studied tools.

Research question #2

- What code smell detection tools were published so far and what are their main features?
  - The number of tools capable of detecting code smells in JavaScript code is significant already.

RQ2 – Number of tools per primary study citations

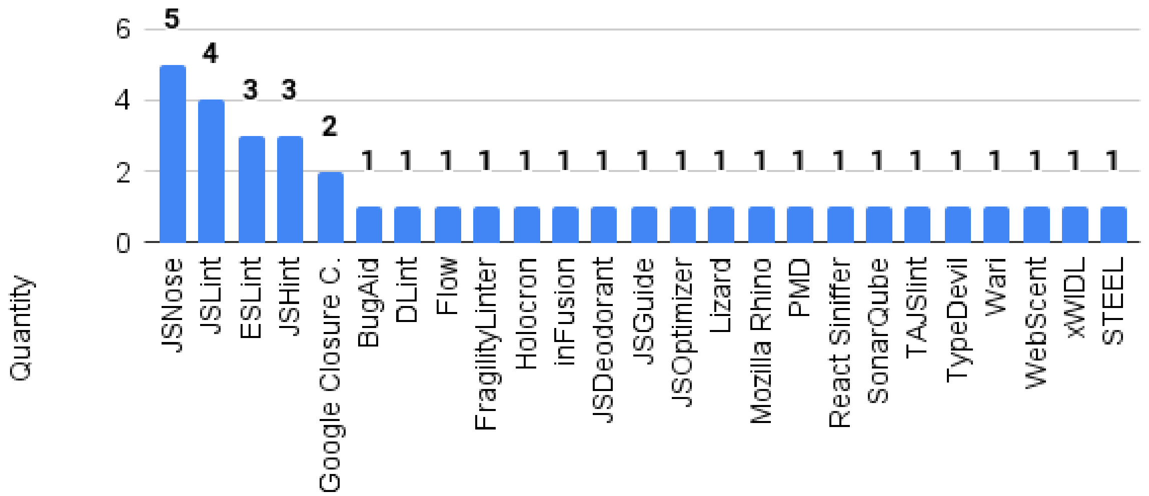


Figure 4: List of the tools cited in the PSs with its amount of citations

RQ2 – Summary

- This number is only expected to increase as JavaScript becomes increasingly used in the Web, being present on more and more devices both desktop and mobile.

    - Another conclusion that can be drawn from the data obtained is that the variety of tools available shows that it is possible to find the best tool for each type of use.

List of code smells detection tools

Tool name	Plug-in	Detected Bad Smells	Detection Technique	Free For Use	Implementation Language	Guide	Gui
Lizard	Standalone, API	Cyclomatic complexity, Duplicated code	Static	Yes	Python	Yes	No
JSNose	Standalone	Long method, Long parameter list, Refused bequest and 10 others	AST + metrics	Yes	Java	No	No
JSLint	Standalone, API	Cyclomatic complexity, Duplicated code	AST + metrics	Yes	JavaScript	Yes	Hybrid
Google Closure Compiler	Standalone, API	Dead code, Tempory field	AST + metrics	Yes	Java	Yes	No
ESLint	Standalone, API	Duplicated code, Long parameter list, Long Method/Function and ?? others	AST + metrics	Yes	JavaScript	Yes	No

Coming soon

- Perform a full comprehensive comparison of the tools, evaluating them based on characteristics such as detected code smells, detection techniques, release year, and more.

Threats to validity

- **Construct validity:** For the systematic literature review, we used an extensive search string that included some of the most common terms related to code smells in JavaScript.
    - One potential issue with this approach is the possibility that the search string might not cover a sufficient number of primary studies.
    - To mitigate this possible threat, we created the search string iteratively and in pairs and through multiple tests in real web search engines.

Threats to validity

- **Internal Validity:** A potential threat is related to the incorrect export and tabulation of data reported by the Scopus web search engine.
    - <u>To mitigate this threat,</u> we performed the data collection and tabulation in pairs.

Threats to validity

- **External Validity:** We decided to focus our work on tools specifically designed for software built using JavaScript rather than so-called language-agnostic tools.
    - This decision may have significantly limited our results, reducing the number of tools identified.
    - To mitigate this threat, we used the guidelines of a systematic literature review, which describe that industrial tools not published in articles may be removed from the list of study objects

Evaluating code smell detection between LLMs and tools *

* tentative name

Goal

- We know that LLMs are capable of detecting code smells in Java code.[2] But how do they compare to tools specifically designed for this type of detection?

    - To answer this question, I plan to compare the performance of ChatGPT with known tools in the literature in detecting code smells in Java.

[2] Silva, L. L., Silva, J., Montandon, J. E., Andrade, M., & Valente, M. T. **Detecting Code Smells using ChatGPT: Initial Insights**. ESEIW/ESEM Emerging Results, Vision and Reflection Papers Tracks, 2024.

Research questions

- To address the central question of the paper, several research questions (RQs) were formulated:
    - How does the effectiveness of ChatGPT in detecting code smells compare to the state of art tools?
    - Which code smells does ChatGPT find more challenging to detect compared to the state of art tools?

Dataset

- The chosen dataset is:
  - Santana, A., Figueiredo, E., Alves Pereira, J. et al. **An exploratory evaluation of code smell agglomerations.** Software Qual J (2024). https://doi.org/10.1007/s11219-024-09680-6

Thank you!

Questions?