Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 393–398
DOI: 10.15439/2025F2661
ISSN 2300-5963 ACSIS, Vol. 43

Challenges in Evaluating OSS Quality: Results from SLR on Quality Evaluation Tools

M. Aslı Taşgetiren
0009-0001-5369-4080
Hacettepe University
Üniversiteler Mah., 06800 Çankaya/Ankara/Türkiye
Email: aslitasgetiren@cs.hacettepe.edu.tr

Ayça Kolukısa-Tarhan
0000-0003-1466-9605
Hacettepe University
Üniversiteler Mah., 06800 Çankaya/Ankara/Türkiye
Email: atarhan@cs.hacettepe.edu.tr

***Abstract*—Open Source Software (OSS) quality evaluation is essential for ensuring the adoption and effective use of OSS projects across various domains. Most works on OSS quality evaluation have focused on the development of models or frameworks rather than providing practical implementations. These proposals can be challenging to use for inexperienced users, which highlights the need for user-friendly tools. This paper presents preliminary findings from a Systematic Literature Review (SLR) that investigates the characteristics, limitations, and gaps of current Quality Evaluation Tools (QETs) for OSS. Our analysis reveals the diversity of quality models underlying these tools and the absence of standardization, which impedes the comparability and reliability of evaluation results. This work also informs the next steps in improving OSS quality evaluation practices.**

I. Introduction

DEVELOPMENT and the use of open source software (OSS) has become widespread, with more than 100 million public projects available on GitHub alone. However, the abundance of options makes selecting the right OSS challenging. Unique characteristics of OSS, such as full availability of source code, version history, and community-driven and decentralized development, further complicate the selection process. These traits limit the effectiveness of standard software quality evaluation tools, which are typically designed for closed-source software and do not account for OSS-specific characteristics. As a result, tools specifically designed or adapted for OSS are needed to support more accurate, consistent, and relevant evaluations.

Specialized quality evaluation tools are also critical, as quality evaluation directly impacts OSS adoption, maintenance, and contribution decisions. Developers, organizations, and users rely on these evaluations to determine whether an OSS project is reliable, secure, and maintainable enough to integrate into their systems. Without appropriate tools, OSS quality evaluations have a risk of becoming inconsistent, overly subjective, and difficult to compare across projects.

To allow for ease in OSS selection and comparison, various quality models and frameworks have been proposed. A systematic literature review published in 2020 [1] identified 35 primary studies that proposed an OSS quality evaluation model or framework. Another published in 2022 [2] identified 36 primary studies with the same purpose. However, manually applying these models can be challenging for the average user, creating a barrier to their practical adoption. To facilitate the use of these quality models, an easy-to-use and preferably automated method for adopting these quality models is necessary. To this end, some quality evaluation tools have been proposed along with studies about quality models.

In [2] it was discovered that out of 36 primary studies, 16 of them (44%) were supported by a tool. Additionally, [1] revealed that among the 35 primary studies analyzed, 22 referenced the availability of toolsets.

Nevertheless, proposing tools is not enough; the tools must be published and maintained for long-term usage. Only two tools mentioned in the reviewed papers in [1] had active, up-to-date web pages. Only one tool remains accessible in 2025.

In addition to tool availability, aspects such as usability, clear evaluation criteria, and compatibility with evolving OSS practices are essential for real-world adoption. Tools that are difficult to install, bound to outdated standards, or lacking documentation further reduce their utility to practitioners and researchers alike. These challenges highlight the need for a structured investigation into the current state of QETs.

Previous study [2] investigated literature up to 2020 and focused on Quality Evaluation Models or Frameworks (QEMoF) for OSS. It provided valuable insights into OSS quality assessment approaches by investigating various aspects of QEMoF, including their goals, structure, and evaluation criteria. However, it only briefly examined the existence of supporting tools as a sub-question, and did not have a dedicated focus on Quality Evaluation Tools (QETs).

Given the strong connection between QEMoFs and QETs, we build upon this prior work by conducting a new Systematic Literature Review (SLR) that explicitly focuses on QETs for OSS. Our work expands on [2] by:

1) Introducing new research questions specifically targeting QET characteristics.
2) Extending the investigation coverage beyond 2020 to include more recent developments.
3) Modifying the search strategy to specifically identify studies that propose practical, user-accessible tools rather than only conceptual models.

This study was supported by Scientific and Technological Research Council of Turkey (TUBITAK) under the Grant Number 746360. The authors thank to TUBITAK for their supports.

To that end, in comparison to 13,146 studies initially retrieved in [2], our updated search retrieved 15,036 additional works. After applying inclusion and exclusion criteria, we selected 19 primary studies on QETs, which are published between 2006 and November 2024, for detailed analysis.

This paper presents initial findings from this SLR on QETs for OSS, highlighting key challenges, limitations, and gaps in existing tools. We discuss the lack of maintenance, accessibility, and standardization among the QETs and outline a vision for future research towards developing standardized, user-friendly tools to improve OSS quality evaluation practices.

II. Research Design

Goal of this research was to comprehensively examine and characterize existing quality evaluation tools for OSS by their content, structure and practical use. The research design followed the SLR methodology [3] and took advantage of PICOC (population, intervention, comparison, outcomes and Context) template [4].

Population: OSS quality evaluation

Intervention: OSS quality evaluation tools

Comparison: Characteristics of OSS quality evaluation tools

Outcomes: Basic characteristics and structure, degree of guidance provided, basic characteristics in evaluation, development method, and validation method of the proposals.

Context: Academia (scientific literature)

A. Research Questions

Given the strong connection between QEMoFs and QETs, some questions were taken from [2] and were adapted to be more in line with our focus on tools. Aim of this adaptation was to ensure that our study builds upon prior research while addressing the gap in practical tool implementations.

B. Search String

To make search results comparable with those of the SLR done for QEMoFs in 2020, the search string in [2] was taken as a basis. In order to focus the search on QETs in this SLR, the phrase *(tool OR framework OR application)* was added to the search string in [2] to obtain the final search string. This adjustment ensured that the search captured studies proposing implemented and potentially usable tools, rather than solely conceptual models. The final search string was:

> **(tool OR framework OR application)** AND
> ("quality") AND
> ("evaluation model" OR "assessment model" OR "measurement model" OR "evaluation framework" OR "assessment framework" OR "measurement framework") AND
> ("OSS" OR "FOSS" OR "FLOSS" OR "open source software")

This query was executed across seven major scientific databases: IEEE Xplore, ACM Digital Library, ScienceDirect, Scopus, Springer, Web of Science, and Google Scholar.

C. Inclusion and Exclusion Criteria

As a result of the updated search string, our SLR retrieved 15,036 additional studies that were published between 2020 and 2024, beyond those analyzed in [2]. While the study count is relatively high for a four year period, most of the results came from Google Scholar which is prone to result duplication.

1) Exclusion Criteria (EC): The following criteria were applied to remove irrelevant or duplicate studies:

EC1: Duplicate studies

EC2: Studies that were not published in English

EC3: Studies that are not accessible in full text

EC4: Secondary studies (e.g., SLR, meta-analyses)

EC5: Books and theses

EC6: Studies that discuss OSS quality evaluation without proposing, applying, or describing a specific QET

EC7: Studies on OSS quality evaluation that do not reference a known quality model or standard

EC6 aims to exclude purely conceptual or theoretical discussions that lack reference to any practical tool or implementation. EC7 was introduced to distinguish dedicated OSS quality evaluation tools from generic tools that only incidentally evaluate software quality (e.g., security scanners, static analysis tools).

2) Inclusion Criteria (IC): A study was included if:

IC1: It explicitly focused on OSS quality evaluation.

IC2: It proposed a tool specifically designed for evaluating OSS quality.

D. Data Extraction

After screening, 19 primary studies on QETs were selected, these primary studies are listed in a supplementary material (see Data Availability section). To ensure a systematic review process, data extraction was guided by the predefined research questions (RQs) which are introduced and answered in Section III. A structured spreadsheet was used to record the extracted data, with each sub-question represented as a column. For each study, relevant information was manually extracted from the full text.

The data extraction was performed by the first author, and ambiguous cases were reviewed and discussed with the second author to promote consistency and reduce bias. The RQs were designed to capture key characteristics of each QET, including:

- Basic characteristics (RQ-1)
- Structural attributes (RQ-2)
- Guidance and usability (RQ-3)
- Evaluation methods (RQ-4)
- Development details (RQ-5)
- Current accessibility and maintenance status (RQ-6)

This structured approach ensured that findings are comparable across different QETs and can be synthesized effectively.

III. Results

This section presents the preliminary findings of our study. The results are structured based on the defined RQs and cover aspects that were specified in the previous section.

The following subsections systematically address each RQ, summarizing the key insights extracted from the reviewed studies. An overview of the results is provided in table format, along with full references to primary studies, on Zenodo (see Data Availability section).

The findings highlight recurring issues such as lack of standardization, limited usability, and sustainability concerns; reinforcing the need for more consistent, well-maintained, and user-friendly QETs.

A. RQ-1: What are the basic characteristics of the QET proposed for OSS?

Data were extracted from each primary study to determine the main purpose of the QET and assess the use of AI techniques in its evaluation logic. Understanding these is essential for distinguishing between broadly applicable tools and those tailored to specific evaluation goals or contexts.

1) RQ-1.1: What is the main purpose of the QET in the study?: Each QET was categorized based on its primary evaluation focus. Tools were labeled as either:

- G: General-purpose i.e., designed to evaluate overall OSS quality, or
- S: Domain-specific i.e., targeting one or more focused quality attributes such as maintainability or security.

The findings indicate that out of the 19 tools analyzed, 13 (68%) of the QETs were designed for general-purpose evaluation, while 6 (32%) focused on a specific quality aspect. This distribution suggests a trend toward designing general-purpose evaluation approaches, although domain-specific tools remain relevant for targeted quality evaluations.

2) RQ-1.2: Does the QET utilize AI?: For AI usage, we checked if the tool explicitly described the integration of AI-based techniques (e.g., machine learning, natural language processing (NLP)) in its implementation. Identifying the potential use of AI integration with this sub-question helps to asses how modern and scalable these tools are, and highlights opportunities for future research.

Out of 19 QETs, only one utilized AI methods. The work used NLP to extract data from community channels. While there are other works that talked about incorporating AI to their tools in the future, they have not released a version that contains it yet. This suggests that although the application of AI in OSS quality evaluation is recognized as promising, its actual implementation in the published QETs remains rare.

B. RQ-2: How is the QET for OSS structured?

1) RQ-2.1: Which quality model is the QET based on?: We analyzed if the QETs referred to formal software quality models—such as ISO/IEC 25000, ISO/IEC 9126, or Dromey—or used custom or OSS-specific models such as OpenBRR, QSOS, or OSMM.

This sub-question was included to assess if the QETs are grounded in standardized definitions of software quality, which supports consistency and comparability across different tools and evaluation contexts. When tools rely on outdated or informal models, their results may not align with current quality standards or be usable beyond their original context. Thus, this analysis helps determine how closely existing QETs align with widely accepted software quality frameworks and whether there is a need for standardization or modernization.

Results indicate that 47% of the QETs used ISO/IEC 9126 [5] as at least one of their base models; however, ISO/IEC 9126 has been replaced with ISO/IEC 25010 [6] and new models should take this standard as the basis. Only 16% of the tools were based on the newer ISO/IEC 25010 standard. Several tools referenced OSS-specific quality models like QSOS, OpenBRR, and OSMM, which were designed to address OSS evaluation challenges but may no longer align with current international standards. This suggests a lag in the adoption of up-to-date quality models within the QETs.

2) RQ-2.2: From what aspects can OSS be evaluated by the QET?: The QETs were classified based on the quality aspects they evaluated, including product quality, quality-in-use, community-related factors, a combination of these, or a single specific attribute.

This sub-question was included to evaluate how comprehensively each tool assesses OSS quality. OSS differs from proprietary software not only in its technical traits, but also in how it is developed, used, and maintained by its community. As such, a meaningful OSS quality evaluation should ideally include technical, user-facing, and community aspects. By analyzing which aspects each QET covers, we can evaluate if a tool is too narrowly focused or provides a more comprehensive assessment aligned with the unique nature of OSS.

The analysis revealed that 84% of the QETs addressed product quality (e.g., maintainability, reliability), 58% considered quality-in-use (e.g., usability, user satisfaction) and 74% incorporates community-related metrics (e.g., maintenance capacity, sustainability). Seven tools specifically used a mix of each of these quality aspects and two tools focused only on a single attribute such as security. These results indicate that while most tools emphasize code-level characteristics, many also acknowledge the importance of user experience and community health, which are both critical to the success and adoption of OSS projects.

C. RQ-3: What degree of guidance is provided for the evaluation of OSS by the QET?

This question examined how clearly the QETs explain their evaluation procedures and whether they demonstrated the tool's application through practical examples. Clear guidance is essential for consistent use and broader adoption; without it, even technically sound tools may remain unused.

1) RQ-3.1: Is the evaluation procedure of the QET adequately described in the study?: To assess the transparency of the evaluation process, each study was analyzed to determine whether it provided an adequate (yes), partial, or no description of how the QET could be used to assess OSS. A well-documented evaluation process increases the likelihood that a tool can be adopted by practitioners or extended by other researchers.

Less than half (47%) of the QETs were determined to include a detailed evaluation procedure, while 32% provided partial descriptions, and 21% lacked a guideline. These findings indicate that the majority of tools do not offer sufficient instructions, which can make them harder to apply in practice and reduce the consistency and reproducibility of their results.

2) RQ-3.2: Is a demonstration of the evaluation using the QET provided in the study?: Demonstration of the tool in practice is critical for illustrating how the tool functions, validating its usefulness, and helping users understand expected outcomes.

The analysis revealed that 47% of the studies included some practical demonstration. The remaining 53% lacked detailed examples of how the tool was applied, which limits users' ability to assess its effectiveness and real-world applicability.

D. RQ-4: What are the basic characteristics of the QET for evaluating OSS?

1) RQ-4.1: Is the license type of the OSS product used as an evaluation criterion?: Licensing type is a very important characteristic in OSS, as it can potentially restrict or influence the licensing of other software that incorporates it. Seven (37%) of the QETs explicitly incorporated license type as a factor in quality evaluation, while the rest did not consider licensing constraints in the assessment process.

2) RQ-4.2: Does the QET support subjective or objective evaluation?: The studies were analyzed to determine whether the QETs relied on subjective assessments (e.g., taking user opinions into consideration) or objective assessments (e.g., metric-based analysis). Objective assessments tend to support consistency and automation, while subjective ones can capture nuanced user perspectives that are difficult to quantify.

The results indicate that 53% of the QETs combined subjective and objective approaches, while 16% supported entirely objective evaluation and 32% relied on subjective assessments. These results suggest that while many tools aim for a balanced approach, a significant portion still depends heavily on subjective input. Subjective input is risky as it can limit standardization or automation in large-scale OSS evaluations; however, some tools knowingly allow subjective input to include diverse user perspectives.

3) RQ-4.3: Does the QET support qualitative or quantitative evaluation?: This sub-question assessed the QET's reliance on quantitative methods (e.g., numeric metrics, automated analysis) or incorporation of qualitative insights (e.g., user interpretations, expert reviews). The evaluation approach influences how scalable, replicable, and interpretable the tool is in practice.

The analysis reveals that 89% of the QETs primarily relied on quantitative evaluation, using methods such as automated code analysis and structured data collection. Remaining two tools combined both approaches for a more comprehensive evaluation. The predominance of quantitative evaluation aligns with the goal of automation but may limit the capture of context-dependent or user-centered insights.

4) RQ-4.4: What quality attributes are used for evaluation by the QET?: The inclusion of different attributes reflects how the tool defines "quality" and which aspects of OSS it prioritizes. The most frequently addressed quality attributes were maintainability (68% of the tools), reliability (58%), efficiency (47%), and security (42%).

These results suggest that most QETs focused on code-centric attributes commonly emphasized in traditional software quality models. This focus was likely due to relative ease of automatically extracting code-related metrics. However, for tools specialized in evaluating OSS, it is important to consider OSS-specific attributes. RQ-4.5 will show that while there are tools that consider community activity metrics, which are unique to OSS, code related metrics still play a more prominent role.

5) RQ-4.5: What software metrics are used for evaluation by the QET?: To further understand how the QETs measure and apply quality criteria, we investigated the specific software metrics used in evaluation. These metrics serve as the foundation for generating evidence-based quality assessments. Across the reviewed studies, commonly used metrics included:

- Code quality indicators (e.g., complexity, code smells),
- Community activity metrics (e.g., issue tracking, pull request volume),
- Bug tracking statistics,
- Support responsiveness and service quality.

Details of these distributions can be seen more clearly in the summary table (see Data Availability section).

6) RQ-4.6: What aggregation methods are used for evaluation by the QET?: Metric values gathered using various techniques tend to be heterogeneous in nature, making combining and comparing them difficult. To combine different metric values into a single score, aggregation methods are used. Weighted arithmetic mean was the most used aggregation method among the studies (63%), with some using other techniques such as overall sum or other mathematical equations.

The widespread use of weighted mean indicates a preference for simple and interpretable methods of combining quality metrics. At the same time, the presence of alternative aggregation methods, such as summation or custom formulas, reflects a lack of standardization in how overall quality is calculated. This variation can reduce the transparency and comparability of results across different QETs.

7) RQ-4.7: Does the QET support evaluation at a single point in time or throughout the evolution of OSS?: Evaluating quality over the course of a project's development is particularly relevant for OSS, where code and community activity evolve continuously. Tools that support analysis throughout evolution can help detect trends, regressions, or improvements, and are valuable for maintainers making long-term decisions.

The results indicate that 79% of the QETs performed single-point evaluations, while 3 tools (16%) allowed tracking quality over time, enabling historical analysis of OSS quality.

A study made on OSS project size growth patterns [7] observed that many OSS repositories reach approximately 75% of their final code size within the first 25% of commits, which

might suggest that a snapshot taken after this period could capture much of the system's core. However, there is no guarantee that the snapshot will be taken at the right time, potentially causing significant early changes to be missed. Additionally, one-time evaluations risk overlooking later community-driven refinements or regressions.

8) RQ-4.8: How are data collected for OSS evaluation in the QET?: Data collection methods were categorized as manual (e.g., form entries, expert-provided ratings etc.) and automated (e.g., metric extraction from repositories, codes, or APIs). The method of data collection directly affects a tool's usability, scalability, and potential for continuous evaluation. Automated tools typically require less user effort and support more consistent and repeatable assessments. However, not all the data can be retrieved automatically, which might limit the data to be taken into consideration in fully automated tools.

The findings showed that 47% of the QETs relied on manual input, 47% used fully automated data retrieval, and one did not specify it. This even split suggests that while many tools aim to automate evaluation, a significant number still depends on user-provided data; which can limit scalability and introduce subjectivity, but also allow for the inclusion of qualitative or context-specific information that automated methods may not be able to access.

9) RQ-4.9: What is the required skill level of users who evaluate OSS using the QET?: Usability and accessibility are critical for encouraging widespread adoption of the tools, especially in OSS communities, where users may include non-experts.

The analysis showed that 58% of the QETs required a medium to high level of expertise, such as familiarity with software quality models, metric interpretation, or manual configuration. This suggests that many tools are geared toward expert users and may pose a barrier for broader adoption, particularly among smaller or less formal OSS projects.

It should be noted that in cases where the required skill level was not explicitly stated, it was inferred based on the tool's interface, configuration requirements, technical documentation and results from RQ-3.

10) RQ-4.10: How are the results of the evaluation provided to the user in the QET?: The format of the results significantly influences the clarity, practical usefulness, and availability of quality assessments—especially for non-expert users or decision-makers.

The analysis showed that the QETs presented their results using the formats listed below. Some used more than one format to show their results; these were counted in the percentages:

- Indexed values in range (e.g., 0-1, 0-100) in 53%,
- Ordinal scales (e.g., ranked scores) in 42%,
- Nominal scales (e.g., "good," "fair," "poor") in 11%.

These findings indicate a tendency toward numeric outputs, which are well-suited for integration into automated systems or dashboards but may reduce usefulness for stakeholders needing more descriptive evaluations.

E. RQ-5: What programming tools and frameworks were used while developing the QET?

The technologies used to develop the QETs were analyzed, including programming languages, frameworks, and external libraries. This information is important for assessing the maintainability, accessibility, and extensibility of the tools. It can also provide guidance for future QET developers.

Most papers did not specify the technologies used. To address this, the source code of the tools—when available—was examined to manually extract the necessary information.

Most commonly used technologies were found to be Python, Java and JavaScript. However, due to limited reporting and missing source code links, a complete analysis could not be performed for all the QETs.

F. RQ-6: What is the current status of the QETs for OSS?

Evaluating the current state (i.e., maintained, discontinued, or still in progress) of each tool is essential to understand its sustainability, potential for long-term use, and relevance to practitioners.

A five-year threshold was used to assess maintenance status, i.e., the tools that had no updates within the last five years were classified as no longer actively maintained. According to this criterion, only six of the tools (32%) were found to be actively maintained. Out of the six active tools, only one was published before 2019. Additionally, out of these, only four tools have had updates within the last five years.

For tools that were discontinued, an archived version was searched on platforms such as Wayback Machine and GitHub. This process aimed to determine whether older versions of the tools were still accessible for reference, analysis or use. The results indicate that 63% of the discontinued tools had an archived version available, while an archived version could not be found for the remaining 37%.

These results highlight a significant sustainability issue: Many QETs are not maintained over time, and in several cases, their disappearance makes it difficult for others to study, reuse, or build on them. This poses a barrier to both reproducibility and practical adoption of the QETs for OSS evaluation.

IV. Discussion

This SLR on Quality Evaluation Tools for OSS reveals several practical and methodological challenges that limit the tools' effectiveness, adoption, and long-term utility.

A key issue is the short lifespan and limited maintenance of many QETs; several tools are either no longer accessible or have not been updated in recent years. This result might be influenced by our focus on academic literature, where prototypes might not be sustained beyond a single study. However, to support reliable and replicable evaluation in real-world OSS contexts, QETs must remain accessible over time. Even if they are not maintained, at a minimum, tools should be archived and publicly available to enable reproducibility and research. Ideally, QETs should also be hosted in open-source repositories, like the OSS projects they assess. Without this,

both practitioners and researchers face significant barriers in applying existing tools or advancing the field.

A second major concern is the lack of standardization across the tools. The QETs are based on a wide range of quality models, ranging from outdated standards (e.g., ISO/IEC 9126) to OSS-specific frameworks such as QSOS or OpenBRR. This diversity, while reflective of differing needs, leads to inconsistent and non-comparable evaluation results. The absence of a shared framework or terminology makes it difficult to compare evaluations across tools or projects, limiting the potential for creating universal benchmarks or guidelines. This puts users in a position where they might need to evaluate the QETs before they can use the QET to evaluate OSS quality.

Finally, usability remains an issue. While some tools were aimed to be user-friendly, many require medium to high levels of technical expertise to use, limiting accessibility for non-expert users. This usability gap limits the widespread adoption of the QETs. Future tools should prioritize intuitive interfaces, quality model templates, clear documentation, and automated data collection to lower the barrier for inexperienced users.

Taken together, these findings highlight the need for more sustainable, standardized, and accessible Quality Evaluation Tools for OSS. Addressing these issues would not only improve the reliability and comparability of OSS evaluations, but would also promote widespread adoption and continuous quality improvement in OSS ecosystems.

V. Threats to Validity

This study, like any other SLR, is subject to several potential threats to validity. One potential threat is the completeness of our search strategy. While we carefully designed our search string to capture relevant QETs for OSS, some studies might have been missed due to variations in terminology or indexing issues in databases. Additionally, the digital libraries we used may not cover all relevant papers, particularly those from lesser-known conferences or non-indexed journals.

The scope of the dataset also presents limitations. Our findings are based on a finite set of primary studies, which may not fully represent all existing QETs for OSS. Some tools may exist only as industry applications without academic documentation, leading to a possible bias toward research-focused tools rather than those used in practice. Additionally; EC7, as stated in Section II.C, further limits the scope. Per EC7, this review focuses on QETs that reference known quality models or standards. That is, tools that evaluate OSS quality using non-standard methods (including some that might be widely used in practice) may not be represented. This choice was made to prioritize comparability and structure, but limits generalization of the results to all tools.

It should also be noted that some aspects of the analysis involved interpretive judgment, especially when tool characteristics (e.g., required skill level in RQ-4.9 or maintenance status in RQ-6) were not explicitly stated in the studies. In such cases, we inferred answers based on indicators like documentation detail, user interface design, or repository activity. For example, the five-year threshold in RQ-6 is an arbitrary cutoff and may not fully reflect a tool's usability, especially if it has remained functional without frequent updates.

Lastly, the study does not assess the real-world effectiveness of the identified QETs through hands-on user testing. While we analyzed their documentation and reported functionality, actual usability, reliability, adoption, and performance of the tools remain unverified. Future studies could address this by running case studies or user surveys to validate our findings.

VI. Conclusion

This study provides preliminary insights into the states of Quality Evaluation Tools for Open Source Software, identifying key challenges in tool sustainability, standardization, and usability. While OSS quality evaluation models are well-studied, practical implementation is still an issue due to outdated tools, inconsistencies, and complexity.

Moving forward, future research should focus on improving the longevity and accessibility of the QETs, ensuring that tools remain accessible even if they are not actively maintained. Additionally, establishing standardized frameworks and meta-models could help make evaluation results more comparable across different tools. Finally, incorporating automation and AI-driven approaches, which are not yet widely implemented in the QETs for OSS, has the potential to improve scalability and reduce the manual effort required for quality assessment.

Addressing these gaps will help create more sustainable, standardized, and user-friendly QETs, ultimately contributing to more effective evaluation and selection of OSS products, greater trust in OSS systems, and broader adoption of quality assessment practices.

Data Availability

The complete summary table of RQ results and the full list of analyzed primary studies are openly available on Zenodo at https://doi.org/10.5281/zenodo.15869799

References

[1] V. Lenarduzzi, D. Taibi, D. Tosi, L. Lavazza, and S. Morasca, "Open Source Software Evaluation, Selection, and Adoption: a Systematic Literature Review," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2020. doi: 10.1109/SEAA51224.2020.00076 pp. 437–444.

[2] N. Yılmaz and A. Kolukısa Tarhan, "Quality evaluation models or frameworks for open source software: A systematic literature review," *Journal of Software: Evolution and Process*, vol. 34, no. 6, p. e2458, Jun. 2022. doi: 10.1002/smr.2458 Publisher: John Wiley & Sons, Ltd.

[3] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Technical report, ver. 2.3 ebse technical report. ebse, Tech. Rep., 2007.

[4] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén *et al.*, *Experimentation in software engineering*. Springer, 2012, vol. 236.

[5] "ISO/IEC 9126-1:2001." [Online]. Available: https://www.iso.org/standard/22749.html

[6] "ISO 25010." [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[7] K. Szymański and M. Ochodek, "On the applicability of the pareto principle to source-code growth in open source projects," in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2023. doi: 10.15439/2023F5221 pp. 781–789.