Challenges in Software Metrics Adoption: Insights from Cluj-Napoca's Development Community

Laura Diana Cernău[a], Laura Dioșan[b] and Camelia Șerban[c]
Babeș-Bolyai University, Faculty of Mathematics and Computer Science, 1, Mihail Kogălniceanu, Cluj-Napoca, Romania

Keywords: Software Development, Software Metrics, Questionnaire Survey.

Abstract: Established research directions yield concrete outcomes on the benefits of using software metrics in software development processes, such as notable correlations between software metric values and various quality attributes of software systems or defect prediction. A discrepancy exists between academic proposals and actual practices used in software development, influenced by factors like budget constraints, prioritisation, and misconceptions regarding software metrics' purpose and potential applications. Consequently, this study seeks to document current practices concerning the usage of software metrics, as well as the advantages and challenges associated with their integration into the software development process. This questionnaire is based on a survey of 40 participants occupying various roles in software systems development teams based in Cluj-Napoca, Romania. Most subjects mentioned that improving confidence in metric usage involves better prioritisation and understanding of metric interpretation. On the other hand, the main reasons for participants not using software metrics are lack of awareness and proper prioritisation. Although this study has revealed the existence of various software metrics-related concepts within industry software development processes, it is apparent that their capabilities are not fully understood.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The quality of software products plays an increasingly important role in the software development industry. The rapid pace of innovation and changes in this field require developing robust and scalable systems to keep up with the rapid changes. In (Pargaonkar, 2023), the author describes software quality's crucial role, not for following a set of coding standards but for building a robust and adaptable foundation. Moreover, some of the advantages of code quality mentioned by the author are reducing the number of defects, improving code maintainability and facilitating collaboration among the developers of the same project.

The correlation between software metrics and specific quality attributes is not novel in academic research. Numerous studies have attempted to demonstrate these correlations. For instance, in (Medeiros et al., 2020), the authors conducted an experiment where software metrics were used as features in machine learning algorithms to differentiate between vulnerable and non-vulnerable code segments.

The primary objective of this research paper is to gather relevant empirical data concerning current practices in the software development industry, specifically focusing on the measurement, analysis, and utilisation of software metrics. A similar empirical study was conducted by Eisty et al. (Eisty et al., 2018), which investigated the utilisation of software metrics in software development. In our research, we aim to build upon this work by emphasising the tools employed, examining not only what metrics are used but also why and how they are implemented in practice. Additionally, we seek to explore the respondents' perceptions of these tools, particularly regarding their utility and complexity. This approach allows for a more comprehensive understanding of the practical challenges and attitudes surrounding using software metrics.

This research arises from the need to bridge the mismatch between academic research and industry practices, shedding light on the underutilisation of software metrics in the software development processes. In their work, Ferreira et al. (Ferreira et al., 2021) examine the gap between research and practi-

[a] https://orcid.org/0000-0002-6876-9065
[b] https://orcid.org/0000-0002-6339-1622
[c] https://orcid.org/0000-0002-5741-2597

Cernău, L. D., Dioșan, L. and Șerban, C.
Challenges in Software Metrics Adoption: Insights from Cluj-Napoca's Development Community.
DOI: 10.5220/0013239200003928
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 425-432
ISBN: 978-989-758-742-9; ISSN: 2184-4895
Copyright © 2025 by Paper published under CC license (CC BY-NC-ND 4.0)

tioners in the context of software maintenance. Despite extensive research in this domain, there is a gap between theoretical advancements and practical applications. The authors suggest that this mismatch may stem from insufficient knowledge of software metrics among practitioners and the lack of appropriate tools to use these metrics in real-world projects effectively.

The lack of standardised project verification through software metrics is a notable shortcoming in today's software development landscape. To address this, we undertook a survey focusing on the following research questions:

- **RQ1.** *Which aspects do developers focus on when integrating software metrics into the development process of software systems?*
- **RQ2.** *To what extent do the existing tools that measure software metrics fulfil the requirements in the software industry?*
- **RQ3.** *What factors need to be addressed for software metrics to be recognised as indicators of software quality and project success?*

Our study involved a diverse group of 40 participants, all actively contributing to various aspects of software development, therefore bringing their unique perspectives and experiences to the table. This group included software developers with different technical expertise (backend developers, frontend developers, software engineers, machine learning engineers, and QA engineers), as well as individuals in leadership roles (team leaders and community managers) and product management (business analysts and product managers). This diverse representation ensures a comprehensive understanding of the industry's perspective on software metrics.

By analysing the responses to these research questions, we aimed to focus on the discrepancies between scientific articles that demonstrate that software metrics can be employed to verify and guarantee certain qualities of software systems and the actual practices within the software development industry. This mismatch is not just a theoretical concern but a pressing issue that needs to be addressed for the industry to progress. This paper could provide some novelty by examining the current state of practice in using software metrics in the software development industry.

The remainder of this paper is organised as follows. Section 2 provides information on related studies relevant to the topics explored in this study. Section 3 elaborates on the case study design and the dissemination of the survey, while Section 4 analyses the results obtained through this study. Section 5 presents some possible threats to validity and how they were mitigated. Finally, Section 6 concludes the work that was done in this study and provides directions for future work.

2 RELATED WORK

A considerable number of articles in the field of software engineering research discuss software metrics. The primary studies conducted on software metrics examine their applicability in detecting software defects and explore the relationship between software metrics and various quality attributes of a software system.

An example is the study by (Sultana et al., 2021), in which the authors used software metrics as features in a machine learning algorithm to predict vulnerabilities in four open-source Java projects. Similarly, in (Alqadi and Maletic, 2020), the authors ran experiments on defect prediction using cognitive complexity slice-based metrics. Their analysis of ten datasets concluded that 94% of the metrics under investigation demonstrate statistical significance concerning their relation to defects.

Similarly, software metrics can be used to assess the level of quality in a system. More precisely, in (Apel et al., 2019), the authors stress the need to establish methods to evaluate the quality of microservice architectures. Their proposal involves a solution designed to compute a set of system metrics and to investigate its potential to assess software quality according to the ISO 25010 standard.

Regarding industry surveys related to software quality and metrics, a notable example is represented by article (Sas and Avgeriou, 2020). The authors addressed the topic of trade-offs between run-time and design-time quality attributes from the perspective of embedded software system engineers. The article is based on an exploratory case study comprising interviews and a focus group, during which participants expressed their preferences for various qualities and discussed the potential features of a tool for managing these trade-offs. One important finding this study provided is that practitioners may overlook the implications of the trade-offs due to the need for more tools for monitoring run-time qualities.

Another relevant example for this context is (Haindl and Plösch, 2022), where the authors conducted an online survey on 61 value-oriented metrics and their relevance to practitioners. One of the goals of this article was to assess the practical relevance of a set of software metrics for practitioners and possible ways to measure them. The study contains answers from 40 participants holding diverse roles within soft-

ware development teams, and one of the main contributions of this study is a list of metrics validated by industry practitioners.

3 METHODOLOGY OVERVIEW

3.1 Survey Design

The present study is based on a three-part questionnaire, all formulated in English. The first part was mandatory for all participants and consisted of the following questions:

- **Q1.** What is your main role in the development process?
- **Q2.** How many years of experience do you have in software development?
- **Q3.** Have you utilised software metrics in any of your software development projects?

Questions *Q1* and *Q2* are open-ended questions, whereas *Q3* is a closed-ended question. All three questions had predefined answers, and for *Q1* the participants could provide answers outside of the predefined options.

The second part was dedicated to those participants who responded with 'Yes' to question **Q3**, more exactly to those who used software metrics in the projects they worked on in the industry. The following questions were part of this section:

- **Q4.** What type of project did you use software metrics for?
- **Q5.** What industry sector did the project operate in?
- **Q6.** Which programming languages were primarily used in the project?
- **Q7.** What development methodologies were predominantly used in the project?
- **Q8.** What specific software metrics did you employ in your project? (select all that apply)
- **Q9.** How did you collect and analyse the software metrics data?
- **Q10.** What were the main objectives or goals behind using software metrics in your project?
- **Q11.** Did you encounter any challenges or difficulties when using software metrics in your project? If so, please elaborate.
- **Q12.** How did the use of software metrics impact the project outcomes?
- **Q13.** How did the use of software metrics impact the decision-making processes?
- **Q14.** Do you think there is any correlation between bugs and software metrics?
- **Q15.** Do you think there are any misconceptions about software metrics that contribute to their underutilisation?
- **Q16.** Do you have any additional comments or feedback you would like to share?

All the questions in this section are open-ended. Questions *Q4*, *Q5*, *Q6*, *Q7* and *Q8* come with predefined answers, though participants are free to provide responses beyond those options. Questions *Q9*, *Q10*, *Q11*, *Q12*, *Q13*, *Q14*, *Q15* and *Q16* are designed to allow free-text response.

Finally, the third part was completed by those participants who responded with 'No' to question **Q3**, which had the following list of questions:

- **Q17.** What were the primary reasons for not using software metrics in your projects?
- **Q18.** What resources do you think are needed to support metrics evaluation in projects?
- **Q19.** What resources or support would you need to feel more confident in using software metrics in your projects?
- **Q20.** Do you have any additional comments or feedback you would like to share?

All the questions in this section are open-ended. Question *Q17* offers predefined answers, but participants may also provide responses outside of these options. Questions *Q18*, *Q19* and *Q20* are intended for free-text responses.

3.2 Survey Dissemination

The target population for this survey includes professors, PhD students, and professionals from the technology sector based in Cluj-Napoca, Romania. Cluj-Napoca was selected as the focus area due to its status as a prominent hub for the technology industry in Romania. This distinction arises from the city's strong academic presence, which contributes to a high concentration of students, a thriving startup culture, and a rapidly expanding IT&C market (Lesniak, 2024), (Council, 2024).

Regarding the distribution of this survey, we used a public link so that anyone with the link could complete it. We recruited participants by personally inviting professionals knowledgeable in the software metrics domain and distributing it in our networks of acquaintances and colleagues.

The participants in this study comprise individuals associated with distinct companies, including some who are concurrently pursuing doctoral studies at the

university while also working as software developers. In designing our study, we opted not to collect information regarding the companies where respondents are employed. This decision was made to avoid introducing biases that could arise from variations in company type, which might skew the results and lead to conclusions that are influenced by organisational context rather than reflecting the broader trends in the use of software metrics. By excluding this variable, we aim to ensure that our findings remain more generalisable and focused on the respondents' personal experiences with software metrics, independent of their corporate environments.

4 RESULTS AND ANALYSIS

4.1 Background Information of Subjects

All participants in this survey are from the same geographical area, the city of Cluj-Napoca, Romania, and their responses were provided in English.

The participants in this study are members of software product development teams, each with different roles. As can be seen, the roles of the participants are diverse, including those focused on technical development, management, and leadership. The vast majority is represented by roles primarily involved in technical implementation (*Software Engineer*, *Backend Engineer*, *QA Engineer*, *Fullstack Developer*, *AI Engineer*, *ML Engineer*, *Frontend Engineer*), collectively constitute 77.5% of the distribution. On the other hand, Technical Leaders, accounting for 15% of the total, shoulder responsibilities that span both the technical and leadership domains. Additionally, there are roles dedicated to leadership and management, containing 7.5% of the distribution.

Figure 1 illustrates the range of experience among the participants, showcasing a diverse spectrum that includes both novice developers and individuals holding senior positions. Moreover, some of these participants, in addition to holding positions as software developers, are also doctoral students at the university.

4.2 Findings and Discussions

Among those surveyed, 70% indicated they had used software metrics in at least one project they had worked on, while the remaining 30% reported they had not utilised them up to this point.

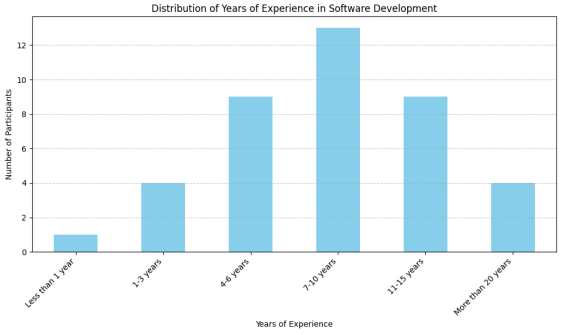

Figure 1: The distribution of the seniority of the participants.

4.2.1 Discussion on Software Metrics Adopters

In the context of question **Q4** regarding project size, the following dimensions were defined:

- Small-scale project (e.g., individual software components, minor feature enhancements)
- Medium-scale project (e.g., standalone software applications, moderate system integrations)
- Large-scale project (e.g., enterprise-level systems, complex software development initiatives)

According to the answers to this question, software metrics have been predominantly used in medium-sized (42%) and large-sized (42%) projects and less so in small-sized ones (16%). This distribution is also highlighted in Figure 2. It is important to note that this question allowed multiple responses, so participants selected all the types of projects in which they had used software metrics.

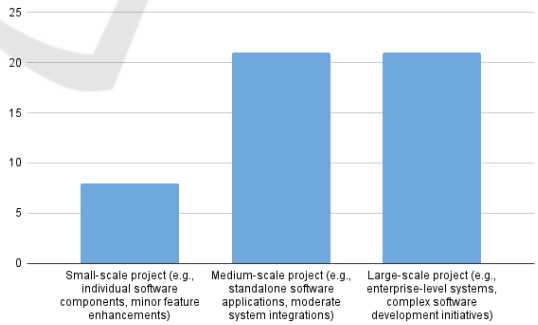

Figure 2: The distribution of project types in which participants used software metrics.

Moreover, Table 1 shows the frequency of the software metrics used, divided by project sizes. Given that the inquiry regarding project size was structured as a multiple-choice question, we calculated the usage of metrics by considering the selected project sizes and tallying the metrics associated with each size category. The first observation is related to the

size of the projects on which these metrics were used. It can be seen that the greater preponderance is on large-sized projects, but at a short distance, there are also medium-sized ones. On the other hand, a clear difference can be seen between these and small-sized projects, where the frequency of using metrics is much lower.

The scope of question **Q9** was to gather insights on how the software metrics are collected and analysed on the projects. According to the answers to this question, the metrics are computed using automated tools, and they are analysed at the code review phase (by implementing CI/CD pipelines based on their values): "*All the ecosystems provide tooling (free or commercial) for measuring these. Some results were aggregated periodically, some were aggregated on each pull request. Some were more permissive, some were restrictive enough, so that code that violated the policy was not merged.*".

Questions **Q10**, **Q12**, and **Q13** delve into the rationale behind using software metrics, their effects on project outcomes, and their influence on decision-making processes. Overall, they aim to assess the value and effectiveness of employing software metrics in software development processes. Therefore, the subjects stated they used software metrics mainly to improve code quality, increase maintainability and testability, detect bugs early in the development process, and identify performance improvements. More precisely, they stated that when monitoring test coverage, they made sure the majority of the source code was covered by tests, which resulted in fewer defects and better overall product quality. Another aspect mentioned was that the continuous deployment practice was enabled by having maintainable code: "*Decreasing accidental complexity and high testability increased the maintainability of the project in such a way that very complex requirements didn't introduce regressions in already delivered functionality.*". Finally, one answer stated that by using software metrics, they were able to identify the problematic areas in their code: "*After starting to use software metrics, finding the problematic areas became a lot easier, you could look into the tools and identify the part of code or resource that was taking longer or had a high error rate. Also reducing the amount of duplicate code and keeping it at a high quality helped developers to code faster and reuse a lot more code. Overall there was a improvement in performance on average above 40% but in some small parts the improvements in performance were 10000% better because a lot of bugs and bad code were impacting the performance although the application was doing what it was supposed to.*".

4.2.2 Discussion on Non-Adopters of Software Metrics

Among the 40 subjects, 12 answered that they have not used software metrics in the development of software projects so far. The main reasons why they said they did not use them are represented in Figure 3. "*Lack of awareness or knowledge about software metrics*" holds the highest percentage as the most voted reason for not using software metrics with a percentage of 33,3%. The remaining reasons can be divided into three distinctive sections as follows:

- when prioritising resources in software system development, software metrics do not take precedence ("*Other priorities taking precedence over software metrics implementation*" and "*Time constraints or resource limitations*") - 33,3%
- challenges related to measuring and interpreting software metrics ("*Difficulty in implementing or interpreting software metrics*" and "*Unavailable tools*") - 18,5%
- the discouragement and recommendations against the use of software metrics ("*Organisational culture or policies discouraging the use of software metrics*" and "*Perception that software metrics are not valuable or relevant to projects*") - 14,8%

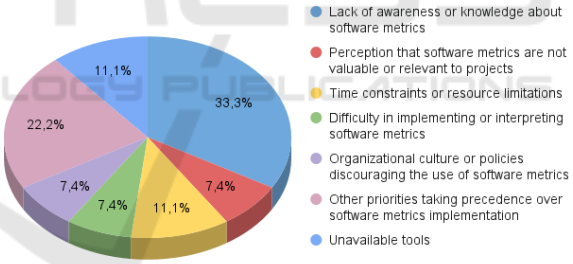

Figure 3: The reasons for not using software metrics in software development.

Through questions "*Q18. What resources do you think are needed to support metrics evaluation in projects?*" and "*Q19. What resources or support would you need to feel more confident in using software metrics in your projects?*" we intended to identify possible solutions for increasing the usage of software metrics. The necessary resources and solutions varied. However, the common themes mentioned are related to having more time to invest in this endeavour (time which results from better prioritisation), a better understanding of software metrics and proper tools for measuring software metrics. Some emphasised the importance of team leaders encouraging the use of software metrics, while others emphasised the need for comprehensive training programs or hands-

Table 1: Frequency distribution of software metrics used in various projects.

Software Metric	Small-scale project	Medium-scale project	Large-scale project	Total usage
Code coverage	7	19	19	45
Performance metrics	8	17	19	44
Code duplication	6	14	15	35
Technical debt	4	14	13	31
Cyclomatic complexity	4	11	13	28
Lines of Code (LOC)	4	8	10	22
Usability metrics	4	5	8	17
Coupling metrics	4	7	5	16
Maintainability index	2	5	6	13
Bug density	1	4	5	10
Cohesion metrics	3	4	3	10
Defect removal efficiency	1	2	3	6
Code churn	1	1	2	4

on tutorials. Additionally, an answer mentioned better development and support for effective metrics interpretation, along with suggestions for dedicated teams to analyse metrics and propose solutions.

4.2.3 RQ1. Which Aspects Do Developers Focus on when Integrating Software Metrics into the Development Process of Software Systems?

Our initial assumption that software metrics are not widely used in the software development industry is not supported by the results of this survey, where 70% of the participants use software metrics to some extent in their projects. However, what can be observed from the responses to question **Q9** is that predominantly more general tools are used, which do not solely aim at measuring and interpreting metrics. Further research should focus on the software development process to identify the most commonly employed software metrics and the specific contexts in which they are applied. Furthermore, according to the responses to **Q13**, the interpretation of these metrics is mainly used to identify parts of the source code that need improvement: "*Some of the architectural decisions were based on software metrics. Performance metrics revealed issues in which data was accessed, therefore it informed us about data access patterns that helped us shape the design of the internal models and our API. The continuous attempt to improve testability informed us about how we should structure our tests and how we should design our domain boundaries to keep coupling to a minimum.*". Another comprehensive answer worth mentioning was: "*The use of software metrics provided valuable data-driven insights that facilitated informed decision-making processes, leading to more effective resource allocation, risk management, and project planning.*". This research question let us to observe that software metrics are predominantly utilised for source code improvements and increasing code coverage by tests.

4.2.4 RQ2. To What Extent Do the Existing Tools that Measure Software Metrics Fulfil the Requirements in the Software Industry?

In their answers to **Q11**, related to the challenges encountered when setting up the environment for software metrics usage, a high number of subjects answered that the initial setup of the tools might be an elaborate process. One particular answer to this question caught our attention because it summarised the challenges and shortcomings of using software metrics in industry projects: "*All these metrics are informative at best, so their contribution to software quality should be taken as such. Testability - high coverage does not equal actual coverage of business requirements and various negative paths. So lack of code coverage needs to be interpreted as an indicator that some things need closer attention and need to be fleshed out as actual requirements. Complexity - this is a highly debatable topic because there is a subjective element that drives the perception of complexity. At the moment, I am not aware of metrics that can actually evaluate complexity in a way that it can measure business impact, so this is always a difficult subject. Maintainability and deliverability - this is a subject that is very dependent on the skill level and experience of the team that delivers the software. Often times lack of experience can influence certain architectural and complexity metrics and the no amount of technical magic or tooling can change that, except training.*". Therefore, due to the comprehensive responses to this question, we observe that existing tools do not provide all the functionalities developers require.

4.2.5 RQ3 What Factors Need to Be Addressed for Software Metrics to Be Recognised as Indicators of Software Quality and Project Success?

The responses to question **Q14** were favourable, with participants stating that they perceive a correlation between the occurrence of defects and the values of software metrics. In addition to correlations, participants also mentioned aspects related to code complexity and duplication, as well as test coverage and effective testing. A response that best summarises these aspects is: "*In some cases. High cyclomatic complexity and low test coverage can give very good hints about volatility and lack of robustness, which are major source of bugs, when the complexity is on the business side. But when the complexity is in UI / UX, or when the bugs are mostly on the UI, then some of these traditional metrics might not apply (e.g CSS maintainability should probably be measured in a different way)*". However, the answers to question **Q15** revealed a complex landscape of misconceptions about software metrics. Many answers pointed out that the root cause of these misconceptions is the lack of awareness of software metrics, and because of this, many think that they are challenging to set up or are time-consuming. Contrary to these answers, there was a particular one that differed from the rest: "*I think it's the other way around. Some of these metrics do not paint an accurate picture about the reality and many teams try to reduce complexity just to satisfy the numbers. Sometimes the business requirements are complex, so complexity is high and that's just the reality and trying to reduce complexity metrics tends to increase accidental complexity and decrease maintainability.*". This answer highlights the intricate nature of the problem, where the business logic is so complex that adding software metrics on top might result in added complexity. Furthermore, questions **Q18** and **Q19** (dedicated to those subjects who did not use software metrics) showed that the lack of awareness about the software metrics and their capabilities, was the main reason for their underutilisation, alongside better prioritisation. Moreover, they pointed out that leadership should encourage the usage of metrics, and there should be more hands-on programs on software metrics and their utilisation in software development.

5 THREATS TO VALIDITY

5.1 Internal Validity

The validity of this study may be internally threatened due to the varying levels of knowledge among the participants; therefore, they may have different understandings of the questions. To mitigate this threat, we took great care in formulating the questions as clearly as possible, and when necessary, the question options were accompanied by relevant annotations. Our survey had two possible paths: for participants who used software metrics and those who did not. Another problem might be related to the fact that we attempted to make the questions more general to encourage participants to complete this survey. Some more specific questions could have provided more in-depth insights in this case. However, the clarity of our questions and the two-path survey design have significantly reduced this potential threat to internal validity.

This study is subject to response and non-response bias. In the first case, factors like the wording of questions could have influenced the responses, especially in self-reported metrics use or project details. To mitigate this, the questions were designed to be neutral, and we assured the participants that the answers were anonymous. Secondly, to minimise the non-response bias, we distributed the survey widely, and follow-up reminders were sent to encourage more participation.

5.2 External Validity

External validity concerns allude to the extent to which this study's findings can be extended and applied to comparable contexts. In this case, there are two generalisation perspectives, one for the participants and the second for the projects mentioned in the study. Concerning the subjects, the data gathered for this study comes from a broad spectrum of software developers with different specialisations and backgrounds working at various companies. The seniority of the participants ranges from early-stage developers (less than one year) to senior engineers with more than twenty years of experience. One possible concern is that most participants have more than seven years of experience (65%). However, looking at this from a different angle is advantageous because we have data from subjects with more experience. On the other hand, one concern related to the generalisation based on the types of projects the participants applied software metrics for is that most participants selected medium-sized and large-sized projects as their target. Nevertheless, this is understandable, as on smaller-scale projects, there is a higher likelihood that soft-

ware metrics may not be employed due to budget or time constraints.

6 CONCLUSIONS AND FUTURE WORK

In contrast to one of our initial assumptions, findings from this survey revealed that 70% of respondents have either utilised or are currently employing software metrics within software development projects. However, respondents also acknowledged that there are still misconceptions due to the lack of awareness, and even if they used them, they still indicate reservations about their utility. Furthermore, insights from the participants who did not employ software metrics revealed a need for better knowledge and training on software metrics, not only for software engineers but also for other stakeholders involved in the software development processes. The underutilisation of software metrics usage in projects is primarily attributed to the lack of prioritisation and resources. The causes can be traced back to the management and leadership roles needing more insight into leveraging software metrics for evaluating software product quality.

In terms of future work, our research aims to delve deeper into the usage of software metrics in the software development process. As highlighted in the discussions regarding the survey results, there needs to be more clarity on the specific reasons for using certain tools, as they tend to be general-purpose. Moving forward, one of our objectives is to understand better the software metrics being utilised and the specific use cases in which they are applied. We plan to conduct comprehensive interviews with both software engineers who have and have not used software metrics, as well as representatives from non-technical roles within software development teams. This approach will provide a more nuanced understanding of the current landscape and help identify areas for improvement and further investigation.

7 DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used *Grammarly* in order to check and correct the written text in terms of grammar and expression. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

REFERENCES

Alqadi, B. S. and Maletic, J. I. (2020). Slice-based cognitive complexity metrics for defect prediction. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 411–422.

Apel, S., Hertrampf, F., and Späthe, S. (2019). Towards a metrics-based software quality rating for a microservice architecture. In Lüke, K.-H., Eichler, G., Erfurth, C., and Fahrnberger, G., editors, *Innovations for Community Services*, pages 205–220, Cham. Springer International Publishing.

Council, S. C. (2024). Cluj-Napoca: the "Silicon Valley" of Eastern Europe. https://www.smartcitiescouncil.com/article/cluj-napoca-silicon-valley-eastern-europe?_-_im-qxHGakUl=17128991389544781572. Accessed: October 23, 2024.

Eisty, N. U., Thiruvathukal, G. K., and Carver, J. C. (2018). A survey of software metric use in research software development. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 212–222.

Ferreira, M., Bigonha, M., and Ferreira, K. A. M. (2021). On The Gap Between Software Maintenance Theory and Practitioners' Approaches . In *2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, pages 41–48, Los Alamitos, CA, USA. IEEE Computer Society.

Haindl, P. and Plösch, R. (2022). Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software*, 16(2):167–184.

Lesniak, O. (2024). Software development in romania: a market overview. https://https://www.n-ix.com/software-development-romania-market-overview/. Accessed: October 23, 2024.

Medeiros, N., Ivaki, N., Costa, P., and Vieira, M. (2020). Vulnerable code detection using software metrics and machine learning. *IEEE Access*, 8:219174–219198.

Pargaonkar, S. (2023). Cultivating software excellence: The intersection of code quality and dynamic analysis in contemporary software development within the field of software quality engineering. *International Journal of Science and Research (IJSR)*, 12(9):10–13.

Sas, D. and Avgeriou, P. (2020). Quality attribute trade-offs in the embedded systems industry: an exploratory case study. *Software Quality Journal*, 28.

Sultana, K. Z., Anu, V., and Chong, T.-Y. (2021). Using software metrics for predicting vulnerable classes and methods in java projects: A machine learning approach. *Journal of Software: Evolution and Process*, 33(3):e2303. e2303 smr.2303.