From Theory to Practice: An Empirical Study of Software Metrics Usage, Challenges and Outcomes

Ajwad Abrar
Software Engineering
Islamic University of Technology
Dhaka, Bangladesh
ajwadabrar@iut-dhaka.edu

Nafisa Tabassum
Software Engineering
Islamic University of Technology
Dhaka, Bangladesh
nafisatabassum4@iut-dhaka.edu

***Abstract*—In the realm of software development, metrics play a crucial role in assessing and enhancing various aspects of the development life cycle. They provide quantifiable insights into the efficiency, quality, and management of software projects. However, despite their significance, there exists a scarcity of in-depth studies within the software industry that comprehensively explore and identify metrics with substantial benefits. Our research addresses this gap through an interview-based analysis of 15 software practitioners from 13 software development companies. Through this research, we have identified the usage of software metrics, explored the challenges associated with their implementation, and assessed their impact on the software industry.**

***Index Terms*—software metrics; code quality; productivity; reliability; software development life cycle**

I. Introduction

In the ever-evolving landscape of software development, the use of software metrics has become integral to assess and enhance the quality and efficiency of the development processes. Software Metrics provide a measurement for the software and the process of software production [1]. It can also help to get insights into the performance, reliability, and maintainability of software systems.

These metrics play a crucial role in aiding decision-making, improving development practices, and ultimately contributing to the success of software projects. By analyzing key indicators, software practitioners can identify areas for improvement, optimize resource allocation, and mitigate potential risks.

There has been a significant increase in the level of software metrics activity actually taking place in industry [2]. Despite the growing importance of software metrics, there remains a gap in our understanding of their usage and challenges, particularly within the software industries of Bangladesh. While global best practices are available, the unique socio-economic and cultural context of Bangladesh may present distinct challenges and opportunities for the effective implementation of software metrics.

This paper aims to bridge this gap by presenting insights from an empirical study conducted through 13 in-depth interviews involving 11 tech companies in Bangladesh. We identify software metrics that exhibit significant potential for benefiting the software industry in this region. Throughout the interview process, we thoroughly explore the impact of these metrics on software development processes, the challenges encountered during their implementation, the tools they use, and best practices for maximizing their effectiveness.

Our study plays a crucial role in filling the existing research gap concerning software metrics in Bangladesh. There has been a noticeable absence of in-depth investigations into the specific needs and intricacies of the software industry in this region. Through our research, we aim to shed light on these aspects, offering valuable insights that can inform and improve software development practices not only in Bangladesh but also beyond its borders.

Our research is specifically directed towards companies with a primary emphasis on software development, IT services, and consulting. To ensure a targeted investigation, we further refine our focus to those organizations actively implementing software metrics for a duration of at least one year. In this context, the formulation of the problem statement can be framed as follows:

“What are the significant software metrics, how are they being used, what challenges do they pose and which tools are being used for code quality in the software industry?”

In brief, our research makes notable contributions in the following aspects:

(i) We identify the most commonly used software metrics in the Bangladeshi tech industry.

(ii) We explore the challenges faced during the implementation of these metrics in the industry.

(iii) We investigate the impact of software metrics on the tech industry, specifically focusing on how they influence the practice of code quality. Our findings provide insights into prevalent practices and challenges, contributing to a better understanding of software development in Bangladesh.

The rest of the paper is organized as follows: Section II covers the related work of our study, Section III outlines the research methodology used in the study, Section IV presents the analysis of the interview data, Section V offers recommendations, Section VI provides concluding remarks of the study and future scope, Section VII discusses limitations, Section VIII contains the ethics statement.

II. Related Work

Software metrics play a key role in measuring software quality and understanding the characteristics of the source code [3]. But to get the best out of it, it is important to have a good understanding of these metrics. It is important for every step in the Software Development Life Cycle [4]. At each phase of the development life cycle, metrics can identify potential areas of problems that may lead to problems or errors [5]. Finding these areas in the phase they are developed decreases the cost and prevents potential ripple effects from the changes, later in the development life cycle [5]. Software developers can plan the future of project based on the measurement results [4]. A good software metric should be simple, precisely definable, measurable, and objective should be attainable at a reasonable cost [6]. Good metrics should facilitate the development of models that are capable of predicting process or product parameters [7].

Evolution in the domain of software metrics was influenced by changes in the development of software, with increasingly specific metrics being proposed for the measurement of both software products as well as software processes [8]. This is reflected in the appearance of software metric tools, both general and language dependent, stand-alone as well as integrated into IDEs in the form of plugins [8]. So, the usage of software metrics also evolves over time to keep pace with the rapidly growing tech industry.

Size estimation is a complicated activity, the results of which must be constantly updated with actual counts throughout the life cycle [9]. Size measures include source lines-of-code, function points, and feature points [9]. Companies fully into Agile practices mostly use specialized tools like JIRA/ Greenhopper to keep track of metrics while others relied heavily on Microsoft Excel [6].

III. Research Methodology

A. Research Method

The research was conducted based on data gathered from interviews with software professionals. The initial objective and focus of the research were to identify the benefits and challenges of metric usage. We also explored the related tools used in the tech industry in Bangladesh. However, after several discussions with professionals, it was identified that most companies are still searching for more appropriate metrics to use within their projects and have yet to establish proper guidelines for metric usage. Additionally, there is variation from one company to another in this regard.

We divided our data collection procedure mainly into three phases. Figure 1 illustrates the entire process. In the first phase, we studied relevant literature to identify the research question. After that, in the second phase, we prepared a questionnaire based on the information gathered from the literature review. The questionnaire was then refined after receiving valid criticism from five industry experts: one principal software engineer, three senior software engineers, and one tech lead. They have an average of around 7 years of experience, with none having less than 4 years. All of them are from reputable tech companies in Bangladesh.

The questionnaire was semi-structured and included multiple-choice questions, checkboxes, and open-ended questions. Semi-structured interviews are a widely used technique in development research [10]. Unlike formal interviews, which follow a rigid format of set questions, semi-structured interviews focus on specific themes but cover them in a conversational style [10].

In the third and concluding phase, we carefully identified individuals to serve as representatives for our survey samples. Following this selection process, we proceeded to conduct comprehensive interviews with each of the meticulously chosen participants. As compared to survey, the interviews offer a deeper insight into the research topic [11]. There is no need for an external expert to prepare the research questions as it is possible to ask probing questions during the interview and the personal touch can make the communication detailed and better as compared to surveys [11]. Despite encountering some challenges, we decided to collect data through online interviews.

B. Data Collection

The research focused on the population of software development projects, particularly web projects. For this study, we considered software development organizations of various sizes, including small, medium, and large-scale entities. The inclusion criteria involved organizations that have undertaken at least one project incorporating software metrics. We used Snowball sampling method to select the potential tech companies in Bangladesh. Snowball sampling is one of the most popular methods of sampling in qualitative research, central to which are the characteristics of networking and referral [12]. The researchers usually start with a small number of initial contacts (seeds), who fit the research criteria and are invited to become participants within the research [12].

TABLE I List of Companies Used for the Study

#	Company Name (Sorted Alphabetically)
1	Bondstein Technologies Limited
2	Brainstation 23
3	Cefalo
4	Enosis Solutions
5	Intelligent Machines
6	Orange Business Development Ltd.
7	Optimizely
8	RedDot Digital Ltd.
9	ShopUp
10	Singularity Limited
11	Synesis IT
12	Technohaven Company Ltd.
13	Therap (BD) Ltd.

We initially selected a total of 20 tech companies in Bangladesh and invited at least one representative from each company with a minimum of three years of industry experience to participate in our interview. Due to time constraints, we

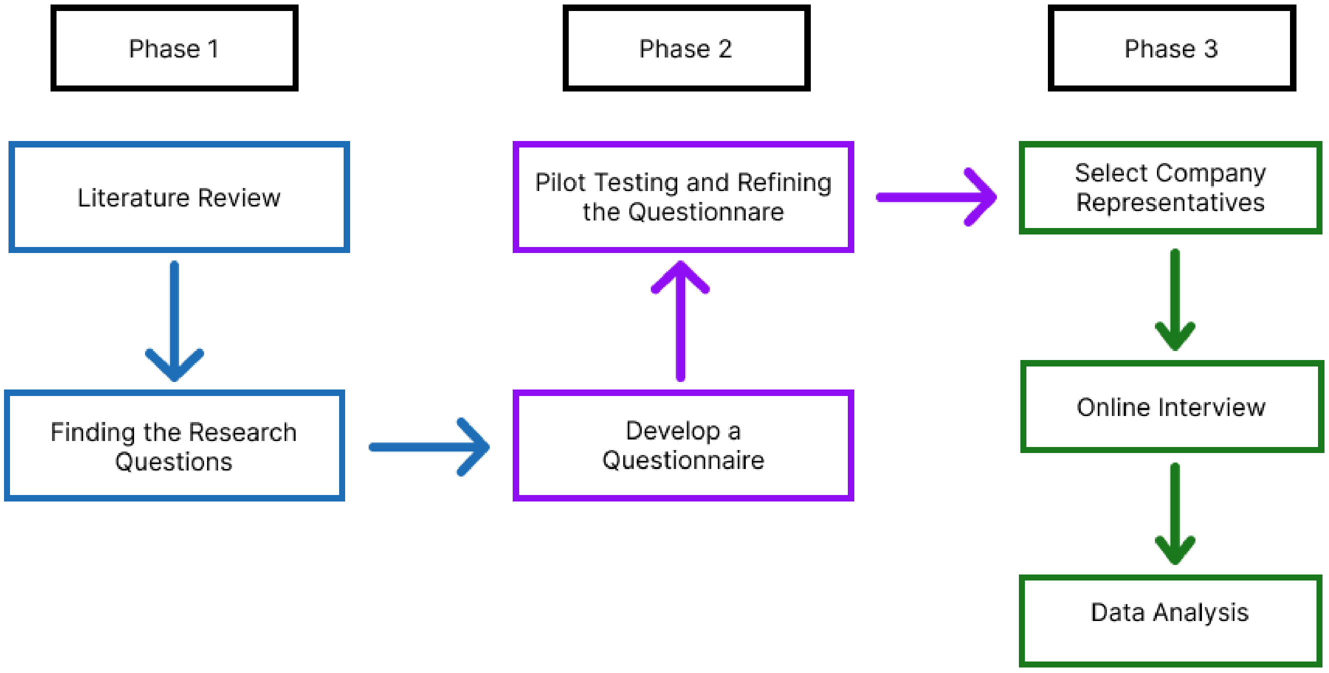
Phase 1
Phase 2
Phase 3
Literature Review
Finding the Research Questions
Develop a Questionnaire
Pilot Testing and Refining the Questionnare
Select Company Representatives
Online Interview
Data Analysis

Fig. 1: Research Methodology Process

were able to arrange interviews with 15 software practitioners from 13 tech companies(listed in Table I). Additionally, another 8 tech companies are expected to participate in our study within the next 15 days. Our plan is to include 50 tech companies to our study. For most research enterprises, however, in which the aim is to understand common perceptions and experiences among a group of relatively homogeneous individuals, twelve interviews should suffice [13].
Data were collected over a period of 15 days, from December 21, 2023, to January 5, 2024. In order to ensure the credibility of the organizations, we selected those that are members of one or more of the following professional organizations: Bangladesh Association of Software and Information Services (BASIS) and the American Chamber of Commerce in Bangladesh. In the case of international companies operating in Bangladesh, our consideration was limited to their activities within the country.

IV. Data Analysis

We reached out to software practitioners from 20 companies and received responses from 13. This sample includes organizations of diverse sizes engaged in both service delivery and product-based development. Subsequently, 15 individuals from these 13 companies participated in our interviews. The participating companies' primary areas of focus can be divided into four major categories: government projects, local client projects, outsourcing projects, and in-house products, as referred to in Figure 2. Here, we observe that a significant number of companies are involved in government projects.

Software development life cycle (SDLC) is a method by which the software can be developed in a systematic manner and which will increase the probability of completing the software project within the time deadline and maintaining the quality of the software product as per the standard [14]. We analyzed tools used in various stages of the software development cycle, including Planning, Coding, Testing, Deployment, and Maintenance. In the planning phase, numerous tools are employed by the studied companies, as evident in Figure 3. Jira emerges as the most frequently used tool, followed by MS Excel and Google Sheets. Insights from practitioners reveal that Excel is predominantly utilized during the pre-development phase, whereas Jira finds application across various stages of the Software Development Life Cycle (SDLC).

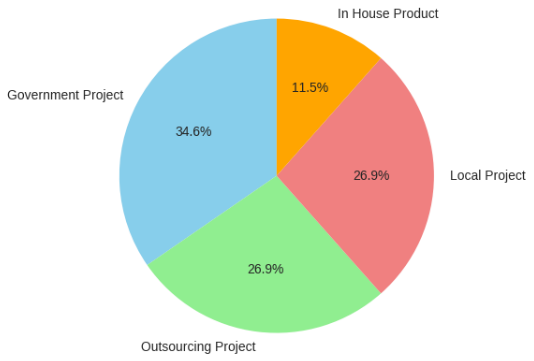

Fig. 2: Visualization of Companies' Areas of Focus

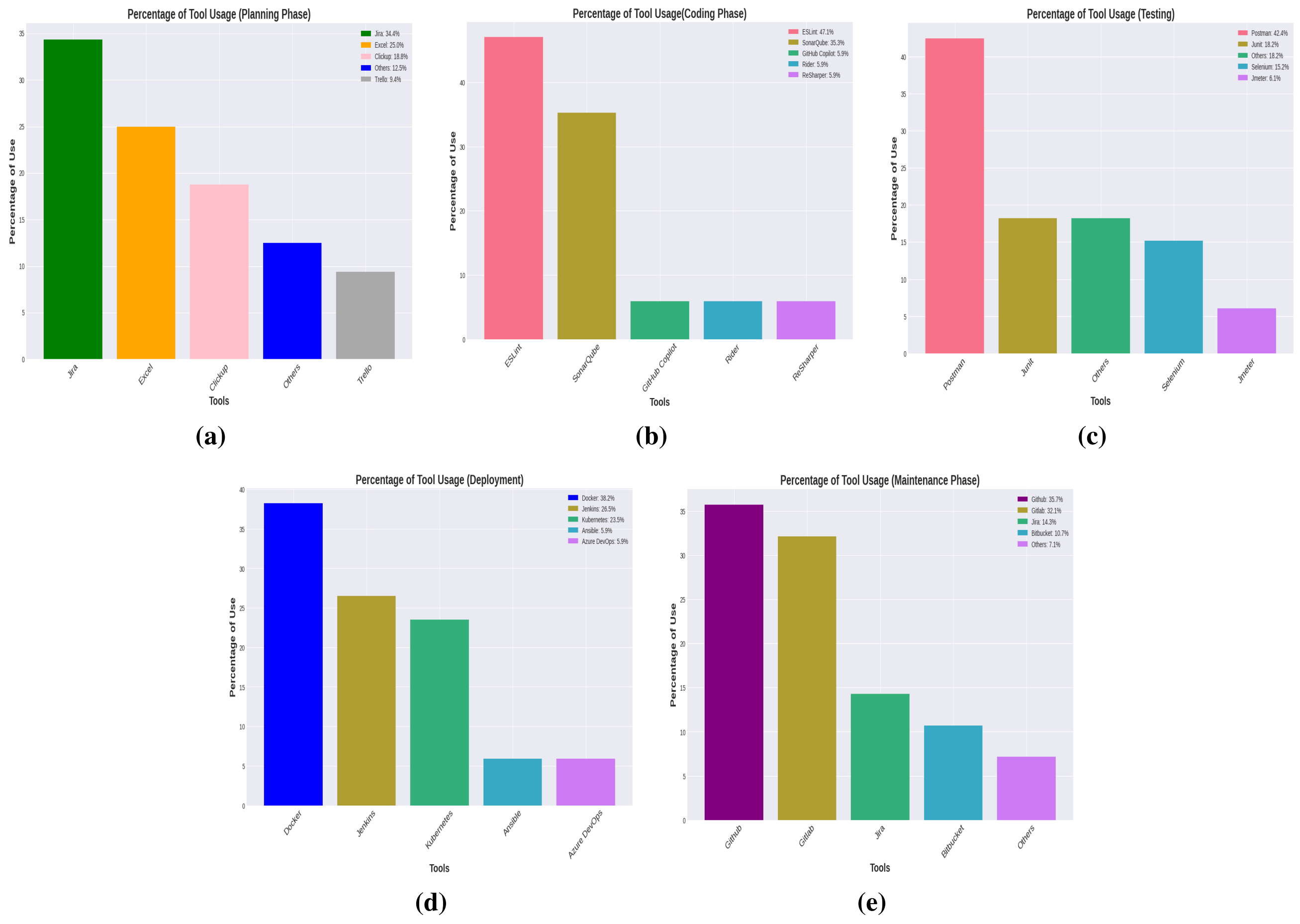

Fig. 3: Tools Usage in Different SDLC Phases - **(a)** Planning **(b)** Coding **(c)** Testing **(d)** Deployment **(e)** Maintenance

Maintaining code quality is crucial for effective software development. In the coding phase, software practitioners employ various tools, as illustrated in Figure 3. ESLint stands out as the most widely used tool, with 47.1% of users, closely followed by SonarQube at 35.3%. Additionally, a few companies utilize tools such as Github Copilot, Rider, and Resharper.
Testing is conducted rigorously in the studied companies, as revealed through interviews with their practitioners. In the testing phase, companies execute various types of testing, employing specific tools tailored to different testing needs. Notably, for API testing, Postman is the preferred tool, JUnit is utilized for unit testing, Selenium for acceptance testing, and JMeter for load testing, as illustrated in Figure 3.

During the deployment phase, companies use various tools, as shown in Figure 3. Docker is the most popular choice (32.8%), followed by Jenkins (26.5%). Kubernetes is also commonly used, while Azure DevOps and Ansible are employed by a limited number of companies for deployment purposes.

In the maintenance phase, companies utilize a variety of tools, as evident in Figure 3. Github takes the lead as the primary choice for version control, embraced by 35.7% of companies. Following closely, Gitlab holds the second position with a substantial 32.1% usage, while Jira trails at 14.3%. Interestingly, only a minority of companies opt for Bitbucket and other tools for their maintenance tasks.

We categorized metrics into five main categories: size, test, productivity, structural, and reliability (as listed in Table II).

Our investigation revealed that companies utilize various metrics within each category. Notably, for the size category, function points and module count are the most prevalent, each used by 33.33% of software engineers. Additionally, metrics such as lines of code, number of classes, and cyclomatic complexity are commonly employed for size estimation. An interesting finding from our study is that many companies determine their project size primarily through requirement analysis rather than relying solely on the aforementioned metrics.

From our interviews with software developers, it's clear that most companies in Bangladesh highly value test metrics for ensuring software quality. About 60% focus on test coverage, and 53.33% use pass/fail rates. Many companies, at 46.67%, consider both test execution time and test case execution status important. Defect density and test efficiency ratio are also widely utilized. These findings (Table II) emphasize the

TABLE II Metrics Distribution and Usage

No.	Category	Metric	Usage (%)
1	Size	Function Points	33.33%
		Modules Count	33.33%
		Lines of Code	13.33%
		Num. of Classes	13.33%
		Cyclomatic Complexity	6.67%
2	Test	Test Coverage	60.00%
		Pass/Fail Rate	53.33%
		Test Execution Time	46.67%
		Test Case Status	46.67%
		Defect Density	33.33%
		Test Efficiency Ratio	20.00%
3	Productivity	Burndown Rate	33.33%
		Velocity	20.00%
		Cycle Time	20.00%
4	Structural	Coupling Between Objects	66.67%
		Lack of Cohesion	33.33%
		Instability	13.33%
		Fan-in and Fan-out	6.67%
5	Reliability	Uptime	60.00%
		Latency	46.67%
		Throughput	40.00%

dedication of companies to provide high-quality products.

Within the productivity category, our study identified three primary metrics utilized by tech companies. The burndown rate, though unconventional, takes the lead with 33.33% adoption, while both velocity and cycle time are employed by 20.00% each. Surprisingly, our findings indicate that a significant portion of companies still do not rely on established metrics to evaluate developer productivity. Instead, many continue to prioritize manual processes in their assessment practices.

In the structural metric category, our study reveals the prominence of four key metrics, as outlined in Table II. Notably, 66.67% of companies prioritize the coupling between objects, reflecting a collective effort towards maintaining decoupled code. Following closely, lack of cohesion emerges as the second most utilized metric, employed by 33.33% of the companies. While instability, fan-in, and fan-out are also in use, their adoption is observed among a smaller number of companies.

In the reliability category, our investigation identified three primary metrics. Uptime stands out as the most widely adopted metric, utilized by 60.00% of the companies, indicating a strong emphasis on continuous system availability. Latency follows closely, with 46.67% of companies incorporating it into their reliability assessments, emphasizing the importance of responsive systems. Throughput is employed by 40.00% of the companies, highlighting a focus on the efficiency and capacity of the systems under scrutiny. These results show that companies use various important metrics to check how reliable their systems are.

During discussions with software practitioners, a common theme emerged: they primarily use software metrics to address code smells. Subsequently, we explored the types of code smells that tech companies typically handle in our analysis.

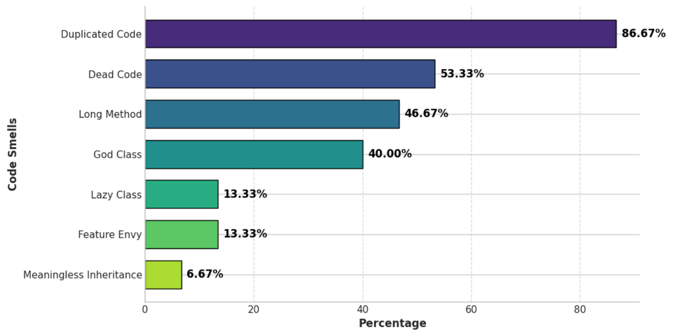

Fig. 4: Percentage of Handled Code Smells

Code smells are indicators of poor coding and design choices that can cause problems during software maintenance and evolution [15].

In the horizontal bar chart shown in Figure 4, various code smells routinely addressed by tech companies are highlighted. Notably, a substantial 86.67% of companies prioritize handling or removing code duplication. Dead code or unused code is another prevalent concern, addressed by 53.33% of the companies. Additionally, 46.67% of companies focus on effectively managing long methods, while 40.00% deal with the challenges posed by god classes. A few companies also address issues like lazy class, feature envy, and meaningless inheritance. This analysis underscores the determined efforts of companies to ensure their code remains free from undesirable code smells.

TABLE III Challenges in Software Metrics Implementation

Sl. No	Challenges	Usage (%)
1	Skepticism about the Effectiveness of Metrics	46.67%
2	Insufficient Training on Metric Usage	40.00%
3	Perception of Increased Workload	40.00%
4	Budget Constraints	33.33%
5	Deadline Pressures	33.33%

From the survey, we have come across a number of challenges (listed in Table III) faced in the tech industry while using the software metrics. About 46.67% of the respondents doubt how useful metrics really are, indicating they might not fully grasp their role in project success. Additionally, 40.00% feel they lack the right training to use metrics effectively, showing a need for better guidance on how to use and understand them. Another 40.00% think using metrics adds more work, possibly making it harder to fit into their usual tasks. One-third of the respondents pointed to budget constraints while another one-third feel pressure from project deadlines, causing them to prioritize immediate tasks over spending time on metrics.

As many participants pointed out meeting deadlines as a challenge in implementing software metrics(listed in Table III), our analysis explores the influence of software metrics on meeting these deadlines (listed in Figure 5). According to the study, the figure reveals that 42.86% of software practitioners believe they consistently meet deadlines when

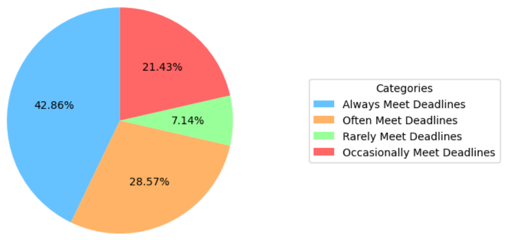

Fig. 5: Influence of Software Metrics on Meeting Project Deadlines

utilizing software metrics. Additionally, 28.57% of participants can often meet their deadlines, 21.43% occasionally meet their deadlines, and a small portion, 7.14%, rarely meet their deadlines while incorporating software metrics.

Despite encountering challenges as outlined in Table III, our investigation reveals positive sentiments among participants regarding the implementation of software metrics (listed in Figure 6). The majority of practitioners (86.67%) express a belief in the positive impact of software metrics on code maintainability and error detection. Additionally, team productivity is reported to benefit from software metrics implementation, with 53.33% acknowledging a positive impact. Furthermore, 40.00% of software practitioners believe that development speed also experiences positive effects through the utilization of software metrics. These findings highlight the utility of implementing software metrics in addressing these critical aspects.

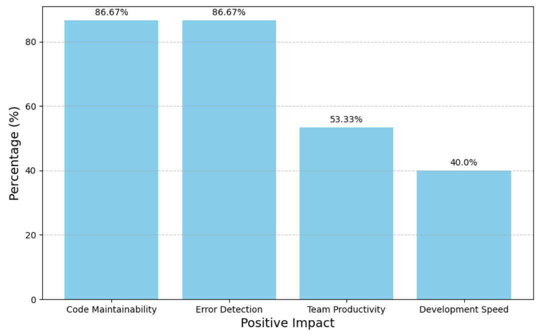

Fig. 6: Positive Impact of Software Metrics Usage

Code quality is a key issue in software development [16]. We conducted an analysis to assess the impact of software metrics implementation on code quality, with the results presented in Table IV. The findings reveal that code quality is substantially improved, as indicated by 53.33% of the participants. Additionally, 40.00% of the respondents perceive an improvement to some extent, while a minority of 6.67% believe there is no noticeable change.

Effective collaboration among team members is a crucial factor for the successful implementation of software projects. Our study analysis indicates that in 57.14% of cases, the implementation of software metrics significantly improved the efficiency of collaboration. Additionally, in 42.86% of cases, there was an observable improvement to some extent. This highlights the substantial role that software metrics play in enhancing collaboration efficiency among team members, as evidenced by our study.

TABLE IV Impact of Metrics on Code Quality

Perception	Percentage
Substantially improved	53.33%
Improved to some extent	40.00%
No noticeable change	6.67%

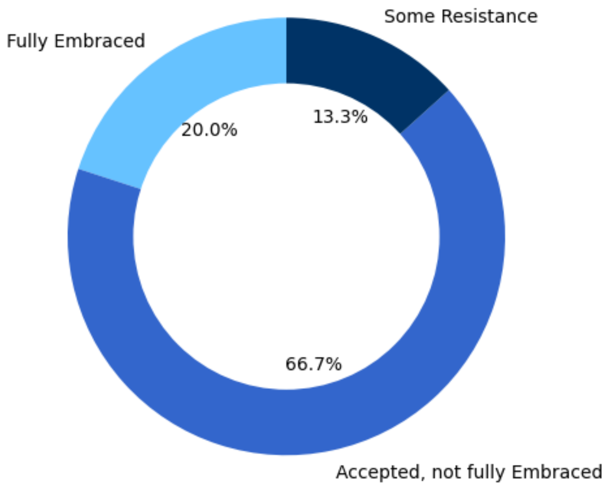

Fig. 7: Organizational Acceptance of Software Metrics

The findings from our study shed light on the prevailing acceptance of software metrics culture among tech companies in Bangladesh. A notable 66.7% of these companies have acknowledged and integrated software metrics practices but haven't fully embraced them, as illustrated in Figure 7. A smaller yet significant proportion, comprising 20.0% of the companies, has fully embraced the software metrics culture. However, 13.3% of the companies still face some resistance towards the complete adoption of these practices.

The positive correlation between the number of employees in a company and the success rate of software projects with the consistent use of software engineering metrics is depicted in Figure 8. The bold regression line and a correlation coefficient of 0.66 affirm a moderately strong positive relationship (listed in Table V). As employee count increases, there is a tendency for a higher success rate, suggesting a positive impact of team size on project outcomes.

TABLE V Employee Correlation

Relation	Value	Category
Employees vs. Project Success	0.66	Strong Positive
Employees vs. Acceptance	0.37	Weak Positive

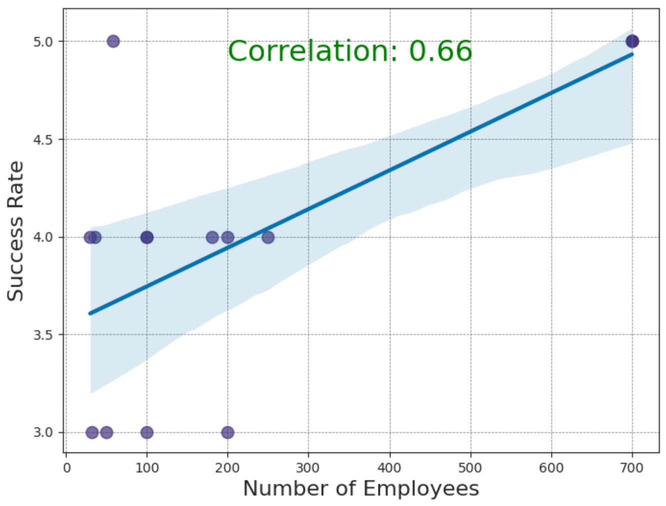

Fig. 8: Employee Success Correlation

Also, we have analyzed the correlation coefficient value between the employee size and the well-acceptance of the use of software metrics as a cultural practice within an organization. The value we got is 0.37 (listed in Table V), which indicates a weak positive correlation. It means there is a mild tendency for larger employee sizes to be associated with a slightly higher level of acceptance of software metrics as a cultural practice within organizations.

V. Recommendations

Through an examination of interviewees' experiences and the challenges they've encountered, certain factors emerge as key recommendations for enhancing overall project success through the utilization of software metrics. Our analysis has identified specific factors that should be incorporated as crucial recommendations.

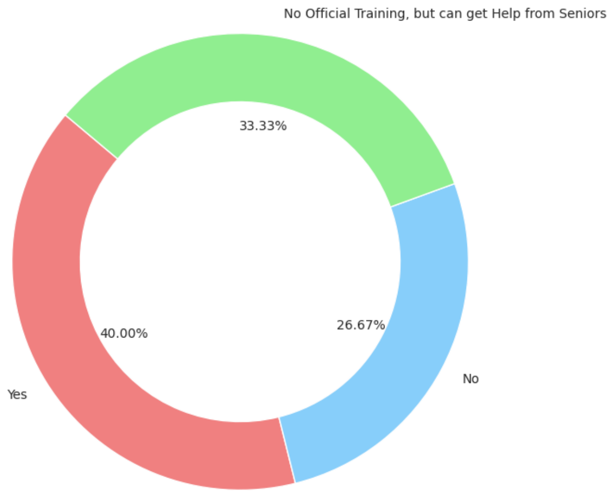

Fig. 9: Employee Training Chart

Findings from our study (listed in Figure 9) reveal that only 40% of software practitioners receive formal training on software metrics provided by their companies. Additionally, 33.33% of practitioners lack official training but seek assistance from senior colleagues when needed. Alarmingly, one-third of the participants in our study reported receiving no training in software metrics. As a robust recommendation, it is imperative for companies to prioritize comprehensive training on software metrics for practitioners, as this will significantly contribute to the overall benefit of the organization.

VI. Conclusion and Future Work

In this paper, we explore how companies in Bangladesh utilize software metrics, the challenges they encounter, and the subsequent impact on code quality. With the rapid growth of the tech industry in Bangladesh, we recommend that companies prioritize the development of high-quality products to stay competitive in the evolving tech landscape. The paper explores the practical challenges and real-world implications associated with the adoption of these metrics. Additionally, we analyze the most commonly used tools across the five phases of the Software Development Life Cycle (SDLC): Planning, Coding, Testing, Deployment, and Maintenance.

The study identifies prevalent code smells effectively managed by tech companies in Bangladesh. Furthermore, it establishes a correlation between the company's size and the influence of software metrics, as evidenced by the research findings.

Due to time constraints, data was collected from 15 software practitioners representing 13 tech companies. Our objective is to expand this study to include data from 50 tech companies, aiming for a more comprehensive and in-depth overview.

VII. Limitations

A limitation of our study is that we were able to conduct interviews with 15 software practitioners from 13 tech companies in Bangladesh due to time constraints. This constraint was due to time limitations during the research process. Even though the sample size is not so large, the collected data is detailed and gives deep insights into the themes we focused on. We carefully chose our participants and conducted detailed interviews, which adds strength to our findings.

VIII. Ethics Statement

All participating software practitioners in the interviews were explicitly informed about the study's purpose, and their consent was obtained for the inclusion of their interviews in our research. No personal information was collected from the participants, and their involvement in the study was entirely voluntary. This ethical approach ensures the confidentiality and willingness of contributors in our research.

References

[1] T. Honglei, S. Wei, and Z. Yanan, "The research on software metrics and software complexity metrics," in *2009 International Forum on Computer Science-Technology and Applications*, vol. 1. IEEE, 2009, pp. 131–136.

[2] N. E. Fenton and M. Neil, “Software metrics: successes, failures and new directions,” *Journal of Systems and Software*, vol. 47, no. 2-3, pp. 149–157, 1999.

[3] M. Y. Mhawish and M. Gupta, “Predicting code smells and analysis of predictions: Using machine learning techniques and software metrics,” *Journal of Computer Science and Technology*, vol. 35, pp. 1428–1445, 2020.

[4] R. Samli, Z. B. G. Aydın, and U. O. Yücel, “Measurement in software engineering: The importance of software metrics,” in *Applications and Approaches to Object-Oriented Software Design: Emerging Research and Opportunities*. IGI Global, 2020, pp. 166–182.

[5] L. Rosenberg, T. Hammer, and J. Shaw, “Software metrics and reliability,” in *9th international symposium on software reliability engineering*, 1998.

[6] K. J. Padmini, H. D. Bandara, and I. Perera, “Use of software metrics in agile software development process,” in *2015 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 2015, pp. 312–317.

[7] R. Bhattacharya, “Importance of metrics in software development life cycle,” *ZENITH International Journal of Multidisciplinary Research*, vol. 2, no. 8, pp. 205–219, 2012.

[8] A.-J. Molnar, A. Neamţu, and S. Motogna, “Evaluation of software product quality metrics,” in *Evaluation of Novel Approaches to Software Engineering: 14th International Conference, ENASE 2019, Heraklion, Crete, Greece, May 4–5, 2019, Revised Selected Papers 14*. Springer, 2020, pp. 163–187.

[9] G. Singh, D. Singh, and V. Singh, “A study of software metrics,” *IJCEM International Journal of Computational Engineering & Management*, vol. 11, no. 2011, pp. 22–27, 2011.

[10] K. Raworth, C. Sweetman, S. Narayan, J. Rowlands, and A. Hopkins, *Conducting semi-structured interviews*. Oxfam, 2012.

[11] N. Jain, “Survey versus interviews: Comparing data collection tools for exploratory research,” *The Qualitative Report*, vol. 26, no. 2, pp. 541–554, 2021.

[12] C. Parker, S. Scott, and A. Geddes, “Snowball sampling,” *SAGE research methods foundations*, 2019.

[13] G. Guest, A. Bunce, and L. Johnson, “How many interviews are enough? an experiment with data saturation and variability,” *Field methods*, vol. 18, no. 1, pp. 59–82, 2006.

[14] A. Mishra and D. Dubey, “A comparative study of different software development life cycle models in different scenarios,” *International Journal of Advance research in computer science and management studies*, vol. 1, no. 5, 2013.

[15] A. Yamashita and L. Moonen, “To what extent can maintenance problems be predicted by code smell detection?–an empirical study,” *Information and Software Technology*, vol. 55, no. 12, pp. 2223–2242, 2013.

[16] J. Börstler, H. Störrle, D. Toll, J. Van Assema, R. Duran, S. Hooshangi, J. Jeuring, H. Keuning, C. Kleiner, and B. MacKellar, “” i know it when i see it” perceptions of code quality: Iticse’17 working group report,” in *Proceedings of the 2017 iticse conference on working group reports*, 2018, pp. 70–85.