Comparing the Effectiveness and Efficiency of Inspections and Walkthroughs

Nimra Arif
FAST School of Computing
National University of Computer and Emerging Sciences
Lahore, Pakistan
nimraarif311@gmail.com

Ali Afzal Malik
FAST School of Computing
National University of Computer and Emerging Sciences
Lahore, Pakistan
ali.afzal@nu.edu.pk

2024 5th International Conference on Advancements in Computational Sciences (ICACS) | 979-8-3503-9478-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICACS60934.2024.10473293

***Abstract*—The success of a software project depends heavily on the quality of its work products and deliverables. Quality of software can be judged by using a number of quality assurance tools like peer reviews and testing. This study focuses on two popular team-based peer review techniques called inspections and walkthroughs. A controlled experiment is designed and conducted to compare the defect detection effectiveness and efficiency of these two peer review techniques. Three different types of software artifacts – software requirements specification (SRS), code, and test cases – of multiple real-life industry projects are subjected to both types of peer reviews. A comparison of experiment results indicates that inspections were more effective (at least 1.54 times) than walkthroughs in detecting defects in all three artifacts while requiring only a slightly longer timeframe (1.13 - 1.31 times more hours). Therefore, an additional time investment in the case of employing inspections would be paid back by an improvement in the quality of software via an increase in the effectiveness of defect detection.**

Keywords—code, comparative study, defect detection, effectiveness, efficiency, inspections, peer reviews, software quality, software requirements specification (SRS), test cases, walkthroughs

I. INTRODUCTION

In order to succeed, a software project must satisfy all of its stakeholders. Stakeholder satisfaction, in turn, depends on how well their requirements are met. A good quality product meets its various explicit and implicit technical and managerial requirements. Making sure that these requirements have been satisfied prior to delivery requires a plethora of quality assurance tools.

Peer reviews are one such tool. Unlike testing, peer reviews can be performed during both upstream and downstream software development life cycle (SDLC) activities. This is primarily because peer reviews do not require the presence or execution of code. All sorts of software artifacts, work products, and deliverables like software requirements specification (SRS), design models, code, and test cases can be subjected to a peer review.

Conducting peer reviews prior to testing has multiple benefits. One of the most important of these benefits is the reduction in software development cost which occurs due to the early detection of defects. A defect detected in the early stages are much less costly to fix as compared to the one detected later in the SDLC. [1]

Realization of this utility of peer reviews in cutting software development cost has led to the introduction of a variety of peer review techniques. These techniques differ in their process, degree of rigor, and participants. This study focuses on and compares (with respect to effectiveness and efficiency) two of the most used peer review techniques i.e. inspections and walkthroughs. Since multiple processes correspond to these names in the literature, the processes we have used for these peer reviews are the ones presented by Galin [2]. Both techniques use group-based meetings to identify defects in software artifacts. Walkthroughs, however, have a less formal and shorter process. Inspections, on the other hand, have a longer, more thorough, and more formal process requiring explicit planning, correction and rework, follow-up meetings, and more comprehensive documentation (i.e. reports).

A controlled experiment involving real-life projects is designed and conducted in an industrial setting to answer the following two research questions (RQs):

a) **RQ1:** Which of the two peer reviews – Inspections or Walkthroughs – is more effective in detecting defects in different software artifacts?

b) **RQ2:** Which of the two peer reviews – Inspections or Walkthroughs – is more efficient in detecting defects in different software artifacts?

The next section of this paper summarizes related past work in this area. The methodology used in this study is described in the third section. The fourth section presents the main results of our experiment. Section five discusses the salient aspects of these results and presents a high-level comparison of inspections and walkthroughs with respect to their efficiency and effectiveness. Threats to the validity of the results of this study and mitigations to these threats are discussed in section six. The final section of this paper presents the main conclusions and some avenues for future work in this area.

II. RELATED WORK

Jayatilake et al. conducted a study [3] to understand the software inspection process in Sri Lankan firms and the

979-8-3503-9478-8/24/$31.00 ©2024 IEEE

impact of this process on product quality. Interviews and surveys were used to collect data. It was found that many Sri Lankan firms use software inspections to improve product quality despite facing obstacles such as implementation costs.

Another study, conducted by Dorin [4], addressed the significance of coding practices during inspections and reviews. The author emphasized the need for agile development teams to adopt effective coding techniques that can enhance the overall efficiency and effectiveness of these processes. By examining the agile software development context, Dorin shed light on the role of code quality in ensuring successful inspections and reviews.

Sherif and Kelly [5] introduced the software inspection process as a better formal alternate method than technical walkthroughs and described two types of reviews: peer reviews and formal inspections. The latter ensures software quality and engages developers and other team members in meetings to detect defects. The study conducted experiments at JPL, comparing the walkthrough and inspection processes based on the Defects/KLOC metric.

Dooley [6] emphasized the importance of high-quality code in software development and highlighted the negative consequences associated with code defects. Various types of inspections and reviews, including formal inspections, walkthroughs, and code reviews, were explored, elucidating their individual advantages and limitations. Drawing on existing literature, Dooley examined different approaches and best practices for effectively conducting inspections and reviews. A systematic and structured approach was advocated, involving multiple stakeholders and the utilization of checklists or guidelines to ensure a thorough examination of the code. This paper concluded that inspections and reviews hold a vital role in identifying and rectifying code defects, enhancing software quality, and ultimately resulting in more reliable and maintainable software systems. The author stressed the need for organizations to prioritize code inspections and reviews as an integral part of their software development process to achieve higher code quality and reduce post-release issues. Overall, this paper serves as a valuable resource for practitioners and researchers seeking to understand the importance of coding inspections and reviews in software development, while also providing practical insights for implementing effective review practices.

Gollamudi, Waller, and Williams [7] conducted a study to investigate the effectiveness of two commonly used software inspection techniques – formal inspection and peer review – in identifying software defects. The study compared the number of defects identified by each technique in a controlled experiment using two software modules reviewed by experienced reviewers. The results showed that formal inspection was more effective in detecting defects than peer review, identifying 28% more defects and requiring less time and resources. While formal inspection is the preferred for detecting software defects, peer review may still be more effective in some cases, such as when reviewing complex code. Overall, this study provided valuable insights into the effectiveness of formal inspection and peer review for software development teams looking to improve their quality assurance processes.

A 2018 study by Misra and Singh [8] compared the effectiveness and efficiency of formal inspections and peer review techniques in identifying software defects. Both methods were effective with formal inspections slightly better at detecting defects. Peer reviews were more efficient in terms of time and effort. The authors recommended choosing the method based on the software development team's specific needs and project requirements.

In 2019, Chris and Mark [9] conducted a comprehensive investigation into the relationship between review rates and defect removal effectiveness in software development. Utilizing data from the Personal Software Process (PSP), the research affirmed that the quality of software is intricately linked to the quality of work conducted throughout the software development life cycle. The findings underscored the significance of employing effective quality control mechanisms, particularly in the form of design and code reviews. The study revealed that the review rate significantly influenced defect removal effectiveness, emphasizing the importance of maintaining a pace within recommended limits to ensure optimal results. Interestingly, the research suggested that beyond a certain threshold (200 LOC/hour), a more deliberate review pace did not substantially enhance performance. The study's [9] statistical analyses provided valuable insights, acknowledging the correlation between review rates and defect identification while recognizing the complexity of causation. Authors concluded that disciplined processes adhering to recommended practices could notably enhance software development performance.

In the research paper [10] authored by Asad, Sidra, and Dr. Mamoona, they undertook a survey to investigate existing approaches to the software inspection process. Commencing with an exploration of the historical context of traditional software inspection processes, the study progressively advanced toward formal software inspection methodologies. The research identified Fagan's methodology as a foundational framework for formally based inspection approaches. Subsequently, a comparative analysis was conducted on well-known software inspection methodologies and models, aiming to enhance overall software quality. The examination also encompassed tools designed to automate the inspection process, supporting various activities such as document handling, code inspection, meetings, and checklist management. The outcomes of this comparative analysis, serve as the groundwork for future endeavors, intending to compile literature data and identify methodologies capable of improving software quality attributes across all phases of the software development lifecycle.

To the best of our knowledge, no prior study has directly compared (using the same set of projects and artifacts) the effectiveness and efficiency of the two peer review techniques – walkthroughs and inspections – presented in Galin [2]. This study aims to fill this research gap by conducting a controlled experiment.

III. RESEARCH METHODOLOGY

Fig.1. provides a pictorial summary of the research methodology adopted by this study. The following subsections briefly describe each step of this methodology.

A. Group Formation

The experiment was conducted in a software development organization which has offices in the UK, USA, and Pakistan. This organization is certified in SugarCRM and Salesforce and is experienced in using lean and agile methodologies in projects related to CRM optimization, integration, deployment, testing, and maintenance/support. It also has expertise in developing mobile apps, web apps, and AI-related apps. For this experiment, two groups – Group A and Group B – were created within this organization.

Both groups consisted of five members. A conscious effort was made to make the composition of both groups as similar as possible. For this purpose, factors such as qualification, experience, domain expertise, gender, and role were considered and balanced.

Each group was composed of four male members and one female member. The highest qualification of all members of both groups was a bachelor's degree in the field of information technology. Each group had one lead, two senior members, and two junior members (with designation Associate).

Group A was asked to use the inspection process to evaluate the quality of different software artifacts of different projects while Group B was asked to use the walkthrough process for the same artifacts of the same projects.

B. Project/Artifact Selection

As shown in TABLE I, six different real-life projects of varying sizes and domains were selected.

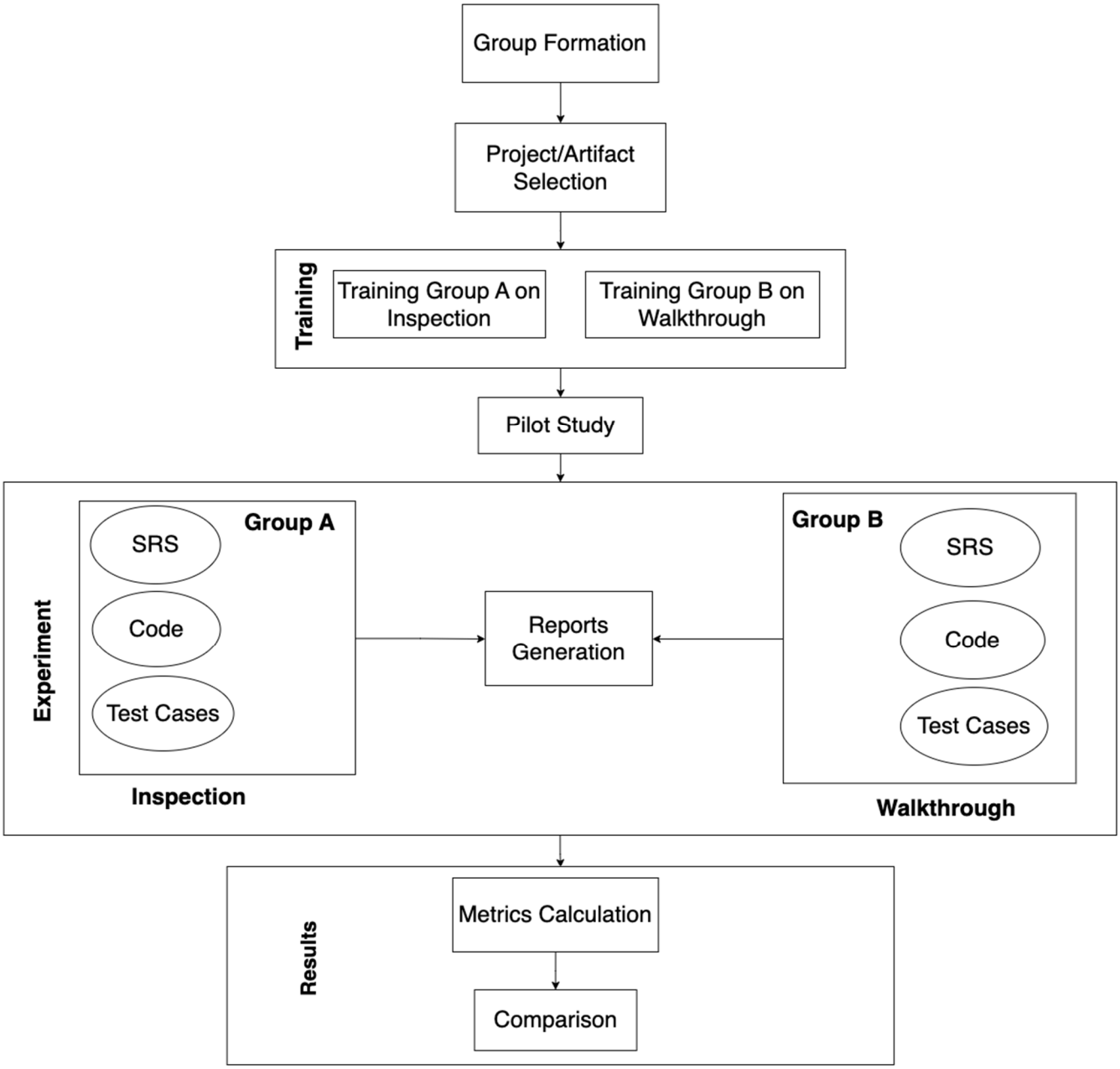

Fig. 1. Research Methodology

TABLE I. PROJECTS SELECTED

Project	Domain	Language	Module Size (LOC)
P0	Healthcare	WordPress	909
P1	Education	JavaScript	400
P2	Marketing	PHP	1537
P3	Sales	PHP	560
P4	Banking	JavaScript	910
P5	Sales	PHP	660

One of these (i.e. P0) was used in the pilot study while the rest of the five (i.e. P1 – P5), were used in the actual experiment. Due to the limited availability of group members, only one specific module was selected from each project. The size of this module, in terms of lines of code (LOC), is shown in the last column of TABLE I. Three main artifacts of this module – Software Requirements Specification (SRS), Code, and Test Cases – were reviewed via inspections (by Group A) and walkthroughs (by Group B).

C. Training

Both groups were trained in their respective peer review process. Group A received training on the inspection process while Group B received training on the walkthrough process. Both groups were provided training by the same individual i.e. one of the co-authors of this study. Checklists created to review each of the three artifacts were provided to both groups. Templates to be used for documenting the assessments during reviews were also shared with the groups.

D. Pilot Study

To ensure that the experiment was set up correctly and to identify any potential issues that might arise, both teams first conducted a pilot study on a separate single project (i.e. P0) before carrying out the actual full-fledged experiment. This pilot study also enabled the teams to become more familiar with their respective review process and to obtain clarifications if needed.

E. Experiment

In this stage of the research methodology, the three selected artifacts of the five projects (P1 – P5) were reviewed by the groups following their assigned process. Review meetings were held and important details (including number and severity) of defects found in the artifacts were documented in reports. The time spent in various stages of the review processes was also recorded by the groups.

F. Results

Once Group A and Group B had finished reviewing the three artifacts for all five projects and produced the required inspection/walkthrough reports, the raw data from these reports was compiled for further processing. For each artifact of each project, four metrics adapted from [1] were calculated.

The first two metrics – Average Defects (AD) and Average Standardized Defects (ASD) – are related to effectiveness (EFT). AD is a simple average of defects per artifact unit while ASD is a weighted average of defects per artifact unit. A simple 1 – 5 defect severity scale [1] is used for weighing defects where 1 represents a minor defect while 5 represents a critical defect. The artifact unit varies with the type of artifact. For SRS, a page is used as the artifact unit. For code, a line of code (LOC) is used as the artifact unit. For test cases, a test case is used as the artifact unit.

The next two metrics – Defect Detection Efficiency (DDE) and Standardized Defect Detection Efficiency (SDDE) – are related to efficiency (EFC). DDE is a ratio of hours spent in review and the total defects found while SDDE is a ratio of hours spent in review and the total standardized defects (i.e. defected weighted by severity). It must be noted that, due to the way these two metrics have been defined, higher values of DDE and SDDE indicate lower efficiency.

Equations (1) – (4) below show the formulas for all four metrics.

$$AD = Total\ Defects / Artifact\ Unit \quad (1)$$

$$ASD = Total\ Standardized\ Defects / Artifact\ Unit \quad (2)$$

$$DDE = Hours / Total\ Defects \quad (3)$$

$$SDDE = Hours / Total\ Standardized\ Defects \quad (4)$$

As is evident from their descriptions and formulas, the two metrics considering standardized defects i.e. ASD and SDDE, by virtue of weighing defects by their severity level, provide a more accurate picture of effectiveness and efficiency.

IV. EXPERIMENT RESULTS

TABLE II to TABLE IV summarize the results of the experiment for all three artifacts of all five projects. Average values of all four metrics (calculated using data from all five projects) have been shown in these tables for both inspections (performed by Group A) and walkthroughs (performed by Group B).

TABLE II shows the results of reviewing SRS via inspections and walkthroughs. These results reveal that, in the case of SRS, on average the inspection process is 1.08 times more effective at detecting defects and 1.54 times more effective at detecting standardized defects. However, as far as requirements-related defect detection efficiency is concerned, the walkthrough process appears to be better. On average, the inspection process took 1.27 times more hours to detect a single defect and 1.29 times more hours to detect a single standardized defect.

TABLE II. RESULTS OF REVIEWING SRS

Metric		Inspection (Group A)	Walkthrough (Group B)
EFT	Avg AD per page	1.056	0.982
	Avg ASD per page	6.054	3.93
EFC	Avg DDE	0.836	0.656
	Avg SDDE	0.136	0.1056

TABLE III shows the results of reviewing code via inspections and walkthroughs. These results reveal that, in the case of programming language code, on average the inspection process is 1.24 times more effective at detecting defects and 1.81 times more effective at detecting standardized defects. However, as far as code-related defect detection efficiency is concerned, the walkthrough process appears to be better. On average, the inspection process took 4.75 times more hours to detect a single defect and 1.13 times more hours to detect a single standardized defect.

TABLE III. RESULTS OF REVIEWING CODE

Metric		Inspection (Group A)	Walkthrough (Group B)
EFT	Avg AD per LOC	0.058	0.0466
	Avg ASD per LOC	0.322	0.1782
EFC	Avg DDE	0.78	0.164
	Avg SDDE	0.148	0.1308

TABLE IV shows the results of reviewing test cases via inspections and walkthroughs. These results reveal that, in the case of test cases, on average the inspection process is 2.40 times more effective at detecting defects and 2.44 times more effective at detecting standardized defects.

However, as far as test cases related defect detection efficiency is concerned, the walkthrough process appears to be better. On average, the inspection process took 1.24 times more hours to detect a single defect and 1.31 times more hours to detect a single standardized defect.

TABLE IV. RESULTS OF REVIEWING TEST CASES

Metric		Inspection (Group A)	Walkthrough (Group B)
EFT	Avg AD per TC	0.637	0.265
	Avg ASD per TC	3.932	1.61
EFC	Avg DDE	1.076	0.87
	Avg SDDE	0.164	0.125

V. DISCUSSION

Fig. 2 to Fig. 5 show the side-by-side comparison of these two peer review processes with respect to both effectiveness and efficiency. As shown in Fig. 2 and 3, inspections are better than walkthroughs in detecting both non-standardized and standardized defects in all three artifacts (SRS, Code, and Test Cases) considered in this study. These figures reveal that the inspection process was at least 1.54 times more effective than the walkthrough process in detecting standardized defects. This superiority of effectiveness of inspections was the most pronounced (i.e. 2.44 times) in the case of test cases.

Fig. 4 and Fig. 5 reveal that walkthroughs are more efficient in identifying both non-standardized and standardized defects in all three artifacts (SRS, Code, and Test Cases) considered in this study. A closer look at Fig. 5, however, indicates that this difference in efficiency is not much in the case of standardized defects. Inspections require only a slightly longer timeframe (1.13 - 1.31 times more hours) to identify standardized defects. These findings answer the two research questions – RQ1 and RQ2 – posed at the start of this research.

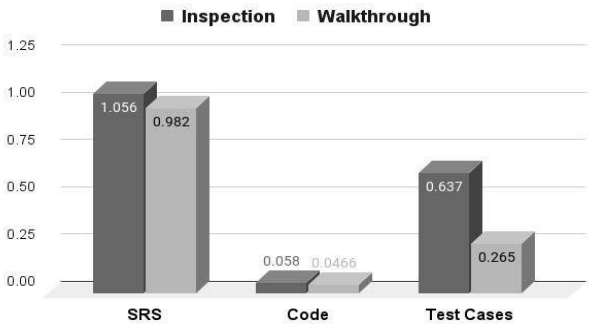

Fig. 2. Comparison of Avg AD per unit of artifact

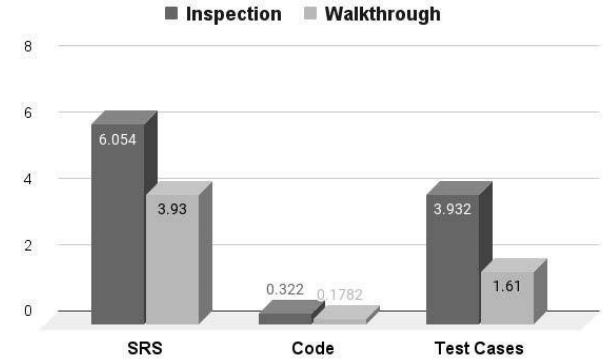

Fig. 3. Comparison of Avg ASD per unit of artifact

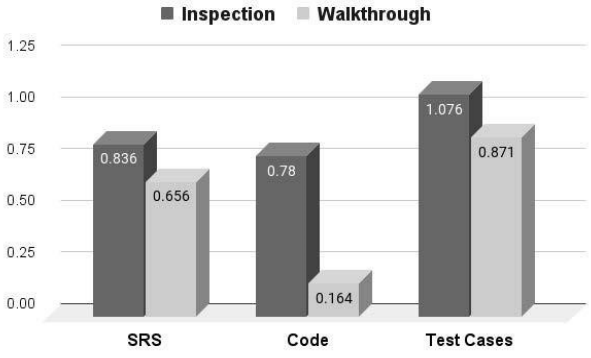

Fig. 4. Comparison of Avg DDE per unit of artifact

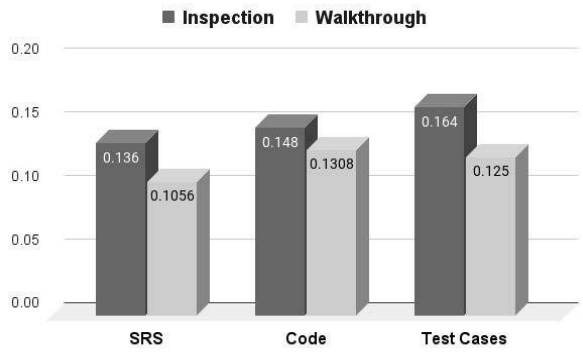

Fig. 5. Comparison of Avg SDDE per unit of artifact

VI. Threats to Validity

Threats to internal validity of this research were mitigated using a number of measures. First and foremost, the same set of projects and the same set of artifacts were used for both inspections and walkthroughs. Secondly, both groups (A and B) were composed in such a way that they were as similar as possible with respect to the experience, qualification, gender, domain expertise, and role of group members. Last, but not the least, training was provided to both groups by the same person i.e. one of the co-authors of this paper.

To mitigate the threats to external validity of this study, several steps were taken. Firstly, five different projects belonging to different domains and implemented in different languages were selected. Moreover, three different types of artifacts representing three different phases of the software development lifecycle were subjected to peer review. Furthermore, multiple metrics were used to measure both effectiveness and efficiency.

VII. Conclusions and Future Work

To the best of our knowledge, this research is the first attempt in directly comparing the effectiveness and efficiency of inspections and walkthroughs. Three different artifacts – SRS, Code, and Test Cases – of five different real-life projects were subjected to both inspections and walkthroughs conducted by comparable groups composed of practitioners. Inspections were found to be more effective in detecting defects in all three artifacts. Walkthroughs, however, were found to be slightly more efficient in detecting defects in all three artifacts.

These findings indicate that spending more time on inspections pays off by improving software quality through better defect detection effectiveness. These findings also imply that walkthroughs may be more suitable for non-critical software applications (e.g. mobile games) which must be delivered within a short time frame to capture a bigger share of the market. Inspections, on the other hand, may be used in projects which are business, safety, or mission critical.

This work may be extended in several directions. One possibility worth exploring involves broadening and expanding the scope of this experiment by peer reviewing a larger number of real-life projects. Projects representing an even greater variety of domains, implementation technologies, and platforms (e.g. desktop, mobile, etc.) may be used. More software development organizations may also be engaged for this purpose. Another worthwhile endeavor involves expanding the set of work products subjected to inspections and walkthroughs, including artifacts such as project plans, analysis and design models, operations, and installation manuals.

ACKNOWLEDGMENTS

We would like to thank the organization which allowed us to conduct our experiment and gave us access to the required project artifacts. We would also like to express our gratitude to the employees of this organization who participated in the group-based peer reviews.

REFERENCES

[1] R. S. Pressman and B. R. Maxim, "Software Engineering: A Practitioner's Approach," New York: McGraw-Hill Education, 2014.

[2] D. Galin, "Software quality assurance from theory to implementation", Pearson, 2016.

[3] S. M. D. J. T. Jayatilake, S. K. K. M. De Silva, U. T. Settinayake, S. A. S. Yapa, J. M. D. A. M. M. S. Jayamanne, A. G. A. M. Ruwanthika, and C. D. Manawadu, "Role of software inspections in the Sri Lankan software development industry," 2013 8th International Conference on Computer Science & Education, 2013.

[4] M. Dorin, "Coding for inspections and reviews," in Proceedings of the 19th International Conference on Agile Software Development: Companion, 2018, pp. 32-33, doi: 10.1145/3205760.3205786.

[5] Y. S. Sherif and J. C. Kelly, "Improving software quality through formal inspections," Microelectronics Reliability, vol. 32, no. 3, pp. 423-431, 1992. doi: 10.1016/0026-2714(92)90072-S.

[6] J. Dooley, "Walkthroughs, code reviews, and inspections," Software Development and Professional Practice, pp. 209–220, 2011.

[7] N. Gollamudi, J. T. Waller and L. Williams, "Comparing the Effectiveness of Inspection and Peer Review in Detecting Software Defects," 2019 IEEE/ACM International Conference on Software Engineering (ICSE), 2019, pp. 673-684, doi: 10.1109/ICSE.2019.00075.

[8] S. Misra and D. Singh, "A Comparative Study of Inspection and Peer Review for Software Quality Assurance," 2016 International Conference on Computational Techniques in Information and Communication

[9] C. F. Kemerer and M. C. Paulk, "The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data," IEEE Transactions on Software Engineering, vol. 35, no. 4, pp. 534–550, Jul. 2009, doi: https://doi.org/10.1109/tse.2009.27.

[10] Masood Qazi, S. Shahzadi, and M. Humayun, "A Comparative Study of Software Inspection Techniques for Quality Perspective," International Journal of Modern Education and Computer Science, vol. 8, no. 10, pp. 9–16, Oct. 2016, doi: https://doi.org/10.5815/ijmecs.2016.10.