University of Ibadan
Journal of Science and Logics in ICT Research

Assessing the Effects of Academic and Work Experience Backgrounds on Software Defect Detection Effectiveness

Arowolo Oladimeji ABIOLA and Solomon Olalekan AKINOLA
oladimejiarowolo@yahoo.co.uk solom202@yahoo.co.uk

Department of Computer Science, University of Ibadan, Nigeria

Abstract
Inspection of various software artefacts increases the quality of the end product – the software. The question yet unanswered is "*Does effectiveness of software inspection depend largely on the academic and work experience backgrounds of individual inspectors involved*?" To address this issue, a medium-scale controlled code inspection experiment with 28 final year students from selected Departments in the Faculty of Science and 10 professionals was conducted at University of Ibadan. The experiment was designed to find out the relationship (if exist) between inspectors' academic and work experience backgrounds and their defect detection effectiveness during an industrial code inspection. The results of the study showed that those with computer related background found significantly more defects than those with non computer related background. It is also found out that prior industrial code writing and inspection experiences significantly impact the effectiveness of an inspector. Finally, professionals with prior code inspection experience found significantly more defects than their student counterparts.

Keywords: Code inspection, Academic background experience, work experience, Inspection effectiveness

1. Introduction

Software inspection is a necessary and important tool for software quality assurance. It involves ability to estimate software faults early on during the development process or even before starting a project can be indispensable in minimizing software development time and effort, where accurate software detection model can reduce the efforts needed to detect software errors throughout the software life cycle and minimize the number of modules developed in each activity [1].

Confusing code is any code element in which developers have considerable difficulty reaching its comprehension. These difficulties may result from a plethora of characteristics of the source code and external issues. In this way, there is still a lack of knowledge on the core issues of confusing code [2]. The development products could be specifications, source code, contacts, test plans and test cases, the use of inspections throughout the software lifecycle is an important factor in improving the overall quality of the resulting software. Code reviews have been used to improve code quality since the 1970s. Most practitioners in the field of software have some experience with respect to the technique (Ilenia et al, 2019). More than thirty years since the inception of inspections, researchers have made modifications to the original process with the goal of improving the effectiveness, efficiency, or applicability in various settings.

Prior research indicates that the overall effectiveness of an inspection team depends largely on the effectiveness of the individual inspectors who make up that team. But, the effectiveness of those inspectors varied widely, even when they use the same inspection technique. Much of this variation was attributed to the inherent differences among the inspectors. Therefore, in order to better understand these differences, this work focuses on the variations in academic and work experience backgrounds of inspectors to understand the variation in their effectiveness.

Arowolo Oladimeji ABIOLA and Solomon Olalekan AKINOLA (2022). Assessing the Effects of Academic and Work Experience Backgrounds on Software Defect Detection Effectiveness, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR),* Vol. 8 No. 1, pp. 81 - 90

Most of the existing software inspection research has focused on improving the methods and techniques used for inspections but the effect of software inspectors' academic and professional backgrounds have not been extensively dealt with.

The common observation nowadays is that non-Information technology (IT) experts are seen developing software for commercial purposes in the country. A good example is the result computation software which is being used by many departments in the University of Ibadan. The software was developed by a non-IT professional who happened to be a pharmacist by profession. Before software is taken out to the market-place, inspection for bugs must be carried out on it in order to ascertain its quality.

Ten (10) professional software practitioners were selected from three (3) software houses and participated in the inspection of an industrial code with the goal of identifying as many defects as possible. Second set of the subjects were the final year students from computer science department and non-computer based departments. The specific focus of this study is to compare the performance of these different groups of subjects to determine whether one group is more effective than the other. The results provide insight into the types of knowledge or experience that are beneficial for inspectors.
Although software review (inspection) could be said to be inevitable in order to ensure software quality assurance, arguments exist on whether academic backgrounds have effects on the reviewer's effectiveness or not.

The major aspect to be considered is the personality of the reviewers, their level of exposure in terms of education and experience. The following research questions were answered in the experiment:

1. Are inspectors who have computer science background more effective during inspection than inspectors with non-computer based background?
2. Does the effectiveness of inspection depend on team size?
3. Does number of inspection meetings – single or multiple – affect the performance of software inspection process?
4. What effect does choice of inspection technique or fault detection method has on inspection performance?
5. Does preparation before inspection meeting has any significant effect on the effectiveness of software inspection process?

1.1 Research Hypotheses

Based on the research questions above, the following hypotheses are tested within the span of this empirical research study:

$H1_o$: There is no significant difference between the effectiveness of computer based inspectors and that of non-computer based inspectors.

$H2_o$: There is no significant difference between the effectiveness of Ad-hoc and Checklist reading techniques.

$H3_o$: Inspections with large team size have longer inspection effort (time), but finds no more defects than smaller teams.

$H4_o$: There is no significant difference between the effectiveness of inspectors with little code writing experience and average code writing experience during inspection.

$H5_o$: There is significant difference between the effectiveness of inspectors with inspection experience and inspectors without inspection experience during inspection

1.2 Paper Outline

The rest of this paper is organized as follows. Section 2 discusses the related works on this line of research while in Section 3; methodology for carrying out the study is discussed. In Section 4, results and their analyses were presented and discussed while the conclusion and recommendation are presented in Section 5.

2.0 Literature Review

Software inspection is as old as programming itself. In principle, code inspection is a transparent process in which reviewers aim to assess the qualities of the software on its technical merits in a timely manner; however, in practice the execution of this process can be affected by a variety of factors, some of which

are external to the technical content of the software itself [3]. At the outset, programmers found out that writing completely accurate programs was too great a problem for the unaided human mind. Hence informal reviews were done. However, as software projects increased in size and ambition, the steps of the review process were gradually written down and improved upon.

Over the years, various metaheuristic algorithms have been hybridized with different kinds of prediction models to obtain an optimized weight that can be fed to the prediction model in an attempt to achieve better prediction accuracy. These endeavours have resulted in the development of hybrid methods that have attained better performance when compared to standard prediction approaches [4, 5].

Although researchers realize the importance of understanding the impacts of an inspector's background and experience, little previous research has focused specifically on identifying the characteristics that make an inspector particularly effective. In a paper that lays out a research program focused on understanding technical reviews and how to improve them. Baum *et al.* [6] highlight the importance of code review has an effective quality assurance technique for decades. In the last years, industrial code review practices were observed to converge towards "change- based/modern code review", but with a lot of variation in the details of the processes.

Inspection is a static verification and validation process in which a software system is reviewed to find defects [7]. Prior work has shown that formal code inspections tend to improve the quality of delivered software. However, the formal code inspection process mandates strict review criteria (e.g. in-person meetings and reviewer checklists) to ensure a base level of review quality, while the modern, lightweight code reviewing process does not. Practitioners do not always use the systematic probably because they do not buy the idea of eliminating the general, identical nature of responsibilities given to reviewers and replacing it with narrowing individual reviewers with the responsibility of finding specific faults that may not be present in the artefact. Whatever the case may be, before any method may be used, it may be necessary to consider the nature and size of the material to be inspected as well as the inspection history available as it affects the materials to be inspected.

RI Hussein *et al.* [8] subdivided defects into two broad types.

1. Omission: Some information is left to understand and the following errors are included in omission.
    - Missing functionality
    - Missing performance
    - Missing interface
    - Missing environment

2. Commission: some information that is irrelevant, ambiguous, or not correct. Following errors are included in the commission.
    - Ambiguous information
    - Inconsistent information
    - Incorrect or extra information
    - Wrong selection

To evaluate the frequency of the error, defect report forms and reviewer defect report forms can be used.

The most frequently used detection methods, which are ad hoc and checklist rely on non-systematic techniques. Ad hoc fault detection method requires that all reviewers use non – systematic techniques and are given the same responsibilities. Checklist is similar to Ad hoc, but here, individual reviewers are given a checklist. The checklist contains items, which are used to extract vital lessons from previous inspections in a particular environment or application. These items may state characteristics faults or ask questions that will aid reviewer's responsibilities and recommend ways reviewers may follow to find faults. In the scenario method, each reviewer employs different, systematic techniques to locate different, specific types of faults. Popular methods used by practitioners are the, Ad hoc and checklist methods where responsibilities are general and identical. Code defects are often considered as key indicators of software quality degradation. If code defects are not systematically removed from a program, its continuous degradation may lead to either major maintenance effort or the complete

redesign of the system. For several reasons, software developers introduce defects in their code as soon as they start to learn programming [9].

3. The Experiments

3.1 Subjects

Twenty eight (28) final year students from seven Departments in the University of Ibadan and ten (10) professional software practitioners were also involved throughout the experiments; the Departments are Computer Science, Physics, Electrical/Electronics, Geography, Economics, Chemistry, Mathematics, and Mechanical Engineering. The final year students from Computer Based Backgrounds have undergone (CSC232, Structured programming), (CSC231 Scientific Programming) while those from other department have not done any Scientific Programming course. In the process of conducting the experiment, the reviewers were divided into two groups based on the focus of this research. The first group of reviewers are those with Computer Based Backgrounds and second are those with non- Computer Based Backgrounds. Each of the groups is divided into four team sizes (1, 2, 3 and 4); for each group of reviewers, two variables were measured (effectiveness and effort). Therefore, this work adopted 2 x 2 x 4 factorial experimental design. The first 2 indicates the main factors measured – Computer Based Backgrounds and non-Computer Based Backgrounds, the middle term 2 indicate sub-factors (dependent variables) measured. The last 4 indicates that 4 replications (team sizes) were measured.

3.2 Experimental Settings

The artefact inspected was Students Registration Software (SRS), an industrial Java code of 450-line of code that accepts students biodata, course registration, mark scored and generate transcript for each student. The program uses all the functionalities of frmlog, showlog, JFrame, connection to db etc. The program accepts student's bio data, courses to be taken in that year and carried over courses, the marks were entered against each courses. The code was compiled successfully and implemented okay by the researcher before it was finally seeded with 45 bugs – 18 Logical, and 27 syntactic/semantic errors. The program performs all the operations on the input data and reports the output results of the computation if there were no errors. If there were errors in form of operational condition not being fulfilled for any of the operations, the program reports appropriate error log for that operation.

3.3 Experiment Design

The experimental design adopted for this research is an independent, two-group between-subject design. It is one in which participant were randomly assigned to the code artefact to be inspected for errors. In this design, six independent variables are measured on each inspector.

3.4 Variables

The experiment manipulated six independent variables: the number of reviewers per team (1, 2, 3, or 4 reviewers), educational background, educational degree industrial experience, code writing experience and inspection experience. Two dependent variables were measured in the experiments: inspection effort in terms of time spent (in minutes) on inspecting the code artefacts, estimated defect detection ratio measured as a ratio of the total true defects detected by the reviewers to the total seeded defect in the inspection artefact.

3.5 Experimental Instrumentation

The designed instruments for this experiment are the Experimental code, Preparation Forms and Defect Collection Meeting Forms. The Experimental Code serves as the artifact reviewed by the reviewers in the inspection processes. Preparation Forms were filled during preparation phase by the reviewers. The Experimental Code and Preparation forms were both given to the reviewers to inspect individually as preparations for a maximum of 70 minutes during which the artefact was reviewed, and the line number of each issue ("suspected defect") as well as the description of the defects suspected. Most importantly, the reviewers recorded their Identity Numbers and their names on the forms.

An hour after the preparation phases were completed, the collection meetings were held. The meeting Forms were filled in at the Defect Collection Meeting. When completed, they gave the time during which the meetings were held, line number and a description of the defect. The team's identity numbers were recorded on the defect collection meeting form to identify which team has which form.

3.6 Conducting the Experiments

The reviewers were broadly grouped into two. Each group was then distributed into teams of varying sizes from 1 to 4. The first group were the Computer Based Background Inspectors while the second group are the Non - Computer Based Background Inspectors. The reviewers not minding their initial experiences were giving proper trainings on some trivial aspects of the experimental artifacts, such as the algorithms for codes during the first weekend meeting. These were done to ensure they understand the inspection artifacts very well. The experiments were closely monitored and organized by the researcher. In order to minimize errors in the experiments, participants were randomly reassigned to teams for each experiment.

During preparations, reviewers analyze the codes in order to find defects. All suspected defects were recorded on the Preparation Forms given them. The experiments placed no time limit on preparations but an average of 70 minutes (1.17 hours) was generally observed by the reviewers for the inspections. During the defect collection meetings, one of the reviewers in each team was selected as the reader as well as the recorder and the moderator. This reviewer paraphrases the code. During this activity, reviewers may bring up any issues found during preparation or discuss new ones. All issues raised were thus recorded in the defects collection forms by the recorder. Before the commencement of the defect collection meetings, the preparation forms were collected by the researcher in order that the reviewers do not mistakenly add to their preparation forms any issues that were not found until collection. Also, there was no time limit placed on defect collection meetings but an average of 47 minutes (0.78 hours) was generally observed by the reviewers.

3.7 Threats to Validity

The question of validity draws attention to how far a measure really measures the concept that it purports to measure (Christoph et al 2021). Therefore in this experiment, we considered two important treats that may affect the validity of the research in the domain of code inspection.

3.7.1 Threats to Internal Validity

Threats to internal validity are influences that can affect the dependent variable without the researcher's knowledge. We considered 3 such influences:
(1) Selection effects, (2) Maturation effects, and (3) Instrumentation effects.

Selection effects are due to natural variation in human performance. For example, if one-person inspection is done only by highly experienced people, their average skill can be mistaken for a difference in the effectiveness of the treatments. We limited this effect by randomly assigning team members for each inspection, this way individual difference was spread across all treatments.

Maturation effects result from the participants' skills improving with experience. Randomly assigning the inspectors and doing the review within the same period of time checked this effect.

Instrumentation effects are caused by the code to be inspected, by difference in data collection forms, or by other experimental materials. In this study, this was very negligible or did not take place at all since all the groups inspected the code artifact within the same period of time. Again, one set of data collection forms was used for the entire group.

3.7.2 Threats to External Validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice. We considered three sources of such threats: (1) experimental scale, (2) subject generalizability,

and (3) subject and artefacts representativeness. Experimental scale is a threat when the experimental setting or the materials are not representative of individual practice; this experiment was carried out on industrial live software project in reputable software industry in Nigeria.

A threat to subject generalizability may exist when the subject population is not drawn from the industrial population; again this experiment was carried out with industrial software professionals. Threats regarding subject and artefact representativeness arise when the subjects and artefact population is not representative of the industrial population. Same issue is attributed to this threat.

4. Results

This study has one major research question and one secondary research question. The major research question is: "Are inspectors who have a degree in computer science more effective during inspection than inspectors with non-computer science degrees?" The secondary research question is: "Do other variables (The number of reviewers per team, Educational Background, Educational Degree, Industrial Experience, Code Writing Experience and Experience with inspection) impact the effectiveness of an inspector?" Prior to conducting T-test and an ANOVA, the first step is to perform a data reduction exercise to ensure that the six secondary variables are all independent. If any of the variables are not independent, then they should be removed prior to conducting the ANOVA to increase the power of the analysis.

The graph in Figure 1 shows that out of 45 bugs seeded into the artifact, computer based background inspectors with four member team detected highest number of bugs in the artefact, they detected 73.33% of the bugs. While the non - computer based background counterpart with a team of four also detected 28.88% of the bugs. The highest defect detection by non-computer based background inspectors is from team member one and they detected 42.22% of the seeded defects.

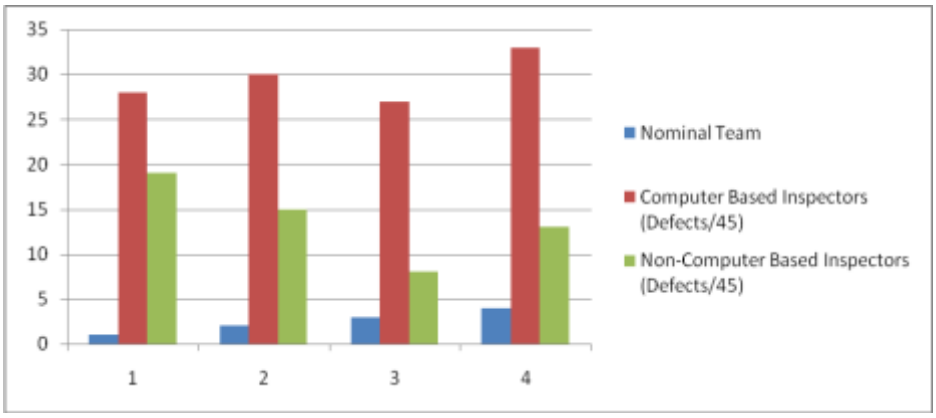

Figure 1. Graph of defect detection by computer based Background and Non-Computer based Background Inspectors.

Table 1: Defect detection between Computer based background and Non-Computer based background Inspectors

	N	Mean	Std. Deviation	t	df	p
Computer based	4	29.5000	2.64575	5.962	6	0.001
Non-Computer based	4	13.7500	4.57347			

Table 1 shows that Computer Based Background (CBB) inspectors uncover more defects than the Non-Computer Based Background (NCBB) inspector with mean defect detection of 29.5000 and 13.7500 respectively. This means that the null hypothesis did not hold based on the result. Statistical test results from Table 1 reveals that there is significant difference in the effectiveness of Computer Based Background Inspectors than Non-Computer Based Background Inspectors ($p = 0.001$). This result may be attributed to the fact that the computer based inspectors had gone through series of courses in code writing, inspection and have acquired skills during their one year industrial attachment.

4.1 Analysis of Code Writing Experience

Part of the vital data obtained from the inspectors was their code writing experience. The inspectors' experience with code writing varies based on the number of years they have been writing codes. The variable value 3 is for inspectors having inspection experience above 3 years and they are revered to as high experience while those having less than 3 years of experience are revered to as low experience. The hypothesis for code writing experience states that there is no significant difference between the effectiveness of inspectors with low code writing experience and those with high code writing experience during inspection.

Table 3. Effect of Code Writing Experience

	Mean	Std. Deviation	t	df	p
Low Experience	25.0	12.47	-0.91	18	0.4
High Experience	29.4	8.64			

Results from the analysis in Table 3 shows that the mean value for low experience inspectors is 25.000 and inspectors with high experience is 29.3571. The high experiences inspectors are more effective in defect detection than low experience inspector which means there experience have impact in their effectiveness. From analysis using independent t-test, the result shows that there is no significant different in the effectiveness of low and high experience inspectors ($p = 0.377$). The null hypothesis holds for this results and this can be attributed to small size of the code.

4.2 Analysis of Inspection Experience

The software industries site surveyed that was carried out in Lagos state Nigeria shows that many of the software houses have different set of people for their inspection work which make the software product to be free from bugs that can prevent the software from efficient operation. Considering the third hypothesis which states that, there is no significant difference between the effectiveness of inspectors with low inspection experience and high inspection experience.

Table 4 Experience with inspection

	N	Mean	Std. Deviation	t	df	p
Low experience	8	25.5	8.93	-0.83	20	0.4
High experience	14	29.2	10.75			

Result from the analysis in Table 4 shows that there is no significant difference in the effectiveness of reviewers with low inspection experience and that of reviewers with high inspection experience, since the significance level ($p = 0.419$) is higher than the allowed error probability level (α) of 0.05.

The null hypothesis holds for student inspectors and can be attributed to time factors (insufficient time) in training and in inspection. But while comparing the students' performance and the performance of professionals from the industry, the null hypothesis did not hold, from Table 5, the analysis on students' data and the practitioners from the industry was compared and the result shows that the professional reviewers are more effective than the students

in terms of inspection experience, the value of the ANOVA F-test is 41.778. The result shows that there is a significant difference in the effectiveness of student inspectors and that of professionals. The level of significance gave $p < 0.005$; the result is attributed to their years of experience in the software industry.

Table 5. Analysis of result of Industrial Practitioners and Student's Inspectors

	N	df	F	p	power
Non-Computer based	4	2	41.78	<0.005	40.7500
Computer based	4	2			
Professional	4	2			

4.3 Discussion of Results

The academic background of inspectors has a significant effect on their performance during inspection, the software industry comprises of graduates from computing related and non computing related background. The graduates from computing related background outperforms the non-computing related graduates. 60% of individuals employed in computer industry do not have a computing – related education, but the result from the analysis has shown that there is a significant difference in the effectiveness computer related inspectors compare to non computer related inspectors ($p = 0.001$) from independent t test in Table 3.

The code writing experience of the inspectors was also analysed and there in no significant difference in the effectiveness of inspectors with little experience and that of inspector with average code writing experience, the t test value ($p = 0.377$) confirmed the result.

The results of the practitioners in the industry was compare with that of the student inspectors and the result shows that there is a significant difference in the effectiveness of code writing experience of the practitioners compare to that of student inspectors. The level of significance is ($p = 0.000$) and this is attributed to their year of experience in the industry. The result shows that years of experience in the industry improve the performance of inspectors irrespective of their academic backgrounds.

Two inspection techniques that were employed in this study is Ad hoc and Checklist technique and the result of the analysis reveals that there is no significant difference in the effectiveness of Ad hoc over Checklist reading techniques.

The relative improvement of a method over another is a measure of how that method outperforms the other [10]. Checklist reading technique performs better than Ad hoc and the inspectors that employed these techniques also proves that inspectors with computing related background outperforms those that did not have computing related background.

5 Conclusions / Future Direction of Works.

Software inspection is an essential constituent of software quality assurance process, yet significant controversies beset the most efficient effective review method used. In this study,

i. The necessity of holding software inspection meetings was questioned against the traditional belief that supports its importance.
ii. The role of collaborative, distributive tool in software inspection was demonstrated.

It is hereby concluded from this study that:

i. Inspections with large teams have higher inspection costs (effort in teams of time expended) but find no more defects than smaller teams.
ii. As far as defect detection performance is concerned, meeting based review methods are not considerably better than the meeting-less based methods.
iii. The level of significance in the effectiveness of Computer based background (CBB) inspectors and Non-Computer based background (NCBB) inspectors is significant ($p = 0.001$). Computer based background (CBB) inspector uncover more defects than the Non-Computer Based Background (NCBB) inspectors.

In addition to gaining a better understanding of the inspection process, the results of this work will also provides guidance to inspection planners in selection and training of inspection team members.

However, future research will enhance this work by examining the effects of a priori professional and cognitive experiences of reviewers on.

5.1 Recommendations

Based on the results obtained in this experiment, the following recommendations are made:

- That software practitioner in Nigeria should endeavour to imbibe code inspection culture in their software development projects. We are now in the IT age and software is the vehicle driving the IT. Nigeria is also wakening up to the global challenge of IT revolution with several machineries put in place to ensure that viable software are produced locally for global consumption. Quality is however needed to be built into this software. Inspection has been identified as a process achieving this quality. That academic background of inspectors' impacts their effectiveness in inspection process; the experimental result reveals that Computer Based Background (CBB) inspectors perform better than Non-Computer Based Background (NCBB) inspectors. It is recommended that 65% of individuals employed in computer industry should have computing related – education. Lethbridge *et al*., [11] reported that 60% of individuals employed in computer industry do not have a computing –related education.
- The matter of software review meeting has been a controversial one, on whether a formal software inspection meeting should be done or not. It is recommended that non – meeting based methods should be used during software review.
- That the inspection team should not be larger than two, to avoid incurring too much cost (time) in the inspection process.
- Checklist reading technique is highly recommended for the reviewers in the process as this will provide an aid to them while carrying out the inspection process on software artefacts.

Acknowledgement

We would like to thank the employee of IETECH Computer Ilupeju Lagos, SYSTEM Spec, Obalende Lagos for their participation in the study. We would also like to thank the final year students from the selected departments in the University of Ibadan for their participation in inspection of the artefact.

References

[1] Hamza Turabieh, Majdi Mafarja, Xiaodong Li (2019) Iterated feature selection algorithms with layered recurrent neural network for software fault prediction. *Journal of Expert Systems with Applications*, Vol. 122 (2019), pp. 27-42.

[2] R de Mello, JA da Costa, B de Oliveira (2021), Decoding Confusing Code: Social Representations among Developers, *13th International Conf. 2021 - ieeexplore.ieee.org.*

[3] Baysal, O., Kononenko, O., Holmes, R., Godfrey, M.W. (2016) Investigating technical and non-technical factors influencing modern code review. *Empir. Softw. Eng*. 21(3), 932–959

[4] Quang-Thanh Bui (2019) Metaheuristic algorithms in optimizing neural network: a comparative study for forest fire susceptibility mapping in *dak nong, Vietnam Geomatics, Natural Hazards and Risk*, 10 (1) (2019), pp. 136-150.

[5] Varun Kumar Ojha, Ajith Abraham, Václav Snášel (2017) Metaheuristic design of feedforward neural networks: A review of two decades of research, *Engineering Applications of Artificial Intelligence*, 60 (2017), pp. 97-116.

[6] Baum, T., Leßmann, H., Schneider, K.: The choice of code review process: a survey on the state of the practice. In: Felderer, M., Méndez Fernández, D., Turhan, B., Kalinowski, M., Sarro, F., Winkler, D. (eds.) PROFES 2017. LNCS, vol. 10611, pp. 111–127. *Springer, Cham* (2017). https://doi.org/10.1007/978-3-319-69926-4_9

[7] Ebad, S. (2017) Inspection reading techniques applied to software artifacts - A systematic review. *Comput. Syst. Sci. Eng.* 32(3), 213–226 (2017).

[8] RI Hussein, NSR Najjar, N Pirzada… Software Requirement Inspection And Defect Detection Techniques,- 湖南大学学报 (自然科学版 …, 2021 - johuns.net.

[9] Oliveira, R., Estácio, B., Garcia, A., Marczak, S., Prikladnicki, R., Kalinowski, M., Lucena, C. (2016) Identifying code smells with collaborative practices: a controlled experiment. In: *X Brazilian Symposium on Software Components, Architectures and Reuse* (SBCARS), pp. 61–70. IEEE 2016.

[10] Ilenia Fronza, Arto Hellas, Petri Ihantola & Tommi Mikkonen (2019) Code Reviews, Software Inspections, and Code Walkthroughs: *Systematic Mapping Study of Research Topics Conference paper*.

[11] Lethbridge, T.C., Diaz-Herrera, J., LeBlanc Jr., R.J., and Thompson, J.B. (2004) "Improving software practice through education: Challenges and future trends". *In Proceedings of 29th International Conference on Software Engineering (Future of Software Engineering Track).*