ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

UNDERSTANDING THE CONSISTENCY OF STACK OVERFLOW CODE: A CAUTIONARY SUGGESTION

Mohammed Lawal Toro
[1]Nigeria Police Academy, Wudil Kano. Nigeria molatoro@hotmail.com

ABSTRACT— Community Question and Answer platforms (CQA) use the power of online groups to solve or gain information about problems. Since these websites contain valuable information, high-quality data must be given on these pages, so that consumers may trust the data. Given the widespread usage of data in all parts of contemporary society, this is particularly important for software creation. Stack Overflow is CQA 's leading programmer platform, with a community of over 10 million contributors.

Although research supports Stack Overflow's popularity, doubts have been posted about the consistency of the answers provided to stack overflow queries. Application fragments have been examined often found in these answers; nevertheless, the accuracy of these objects remains uncertain. It may present a challenge for the software development world, as data suggests that samples from Stack Overflow are widely included in both open source and commercial applications. By assessing the consistency of code snippets on Stack Overload, this work fills the void.

I have discussed various facets of the quality of code excerpts, including usability and accordance with programming standards, readability, efficiency, and protection. Outcomes indicate variability in the consistency of Stack Overflow code snippets for the various dimensions; but generally, quality problems were not necessarily dangerous in Stack Overflow snippets. Vigilance with anyone that duplicate fragments of Stack Overflow code is urged.

Keywords— Stack overflow, Programming lines of code, Consistency, performance, Security

I.INTRODUCTION

Public Question and Answer Platforms (CQA) promote the usage of community control (i.e., public communities) to address concerns [1]. Platforms include Stack Overload or Yahoo! Answers provides a website for those who take to the internet to resolve queries or to search relevant details. These forums, in particular, support software professionals finding knowledge because many other people may have encountered a similar issue, and therefore a related query might already have been raised and has provoked a suitable answer [2]. On the other side, these forums can encourage new problems to be built, so experts to offer their unique knowledge, helping them to address a challenge so earn the confidence of their colleagues in the group. Though valuable information is hosting these websites, the information presented on these channels must be of a high standard so that consumers may trust the data. It supports a study policy to provide quality control in this kind [3].

Nonetheless, even the terminologies used to describe these sites are incoherent because this work field is still evolving. Srba and Bielikova find that there are several words used to apply to CQA sites. Of starters, though, this paper refers to the styles of group question and answers sites such as CQA, Q&A, media Q&A, and discussion forums are usually classified as such [4].

Because of its customer involvement, Stack Overload has been noted as a popular forum, indicating that the usage of gamification strategies directly adds to its popularity. On Stack Overload, however, there are concerns about the accuracy of the responses [5]. Furthermore, considering that developers use much of the code snippets in Stack Overflow posts during creation, assessing the consistency of such objects is critical. In particular, though mutual monitoring by the Stack Overflow group will help to detect and enhance code snippet errors on this forum and users should use code snippets effectively by adjusting them to their unique problem/task [6]. But addressing this research project's goal requires defining code snippet quality. It is achieved by reviewing work and assessing the accuracy of the snippet code in comparison to well-established and well-understood software standard metrics. It has contributed to our evaluation of code stability and compliance with programming standards, readability, consistency, and protection. Because of our quality definition, which consists of multiple dimensions, code snippets are

ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

extracted from Stack Overflow and analysed against those criteria. We then provide tests at various granularity rates representing all breaches (or errors), piece relevant breaches, and contextual analysis evaluating the consequences for the occurrence of violations, as well as attempts by the Stack Overflow group to resolve such [7]. The findings of this research are a review of the breaches of the Stack Overflow application snippets against specific consistency criteria, as well as the forms of violations that are visible in contributors' code. They further discuss the framework for how the software development environment, given the support that Stack Overflow offers, should establish a strategy to improve product quality [8]. appear in an Abstract. The introduction should contain a succinct description of the issue being reported, its historical antecedents, and the study objective.

II. LITERATURE REVIEW

To accomplish the aim of this study, the literature on Stack Overflow Application Consistency is examined with a special focus on interpreting works that have assessed Code.

The quality of code on stack overflow:

Stack Flood is CQA's biggest programming network for over 10 million individuals, including around 16 million inquiries starting in 2019. Albeit none may debate the utility of Stack Flood and related discussions to programming improvement professionals, concerns have been posted about the consistency of the reactions given to worries from specialists [9]. For instance, in the mission to comprehend the nature of Stack Flood substance, Ginsca and Popescu explored the connection between Stack Flood client profiles and reaction quality, discovering relationships between a finished client profile and the nature of the appropriate response. Jin et al. concentrated on how the need to 'win' notoriety prizes can impact the nature of the reaction, as clients can in some cases be headed to give answers without thinking about the suggestions for quality. Anand and Ravichandran perceived a need to recognize the appropriate response content from the noticeable quality of clients, as the motivator plot that Stack Flood utilizes frequently conflicts with consistent reactions. For instance, clients on occasion may attempt to address questions identifying with well-known or secure points, which thus prompts an expansion in their notoriety [10]. Additionally, studies such as Treude et al. Considering the previous issues, helping the software development group with Stack Overflow consistency validations fill this void is essential for analysis work. Furthermore, considering that developers use much of the code snippets in Stack Overflow posts during creation, assessing the consistency of such objects is critical. Some investigations focused on specific aspects of code snippets, such as Squire and Funkhouser's work, which recommended a 1:3 code-to-text ratio in Stack Overflow responses [11]. These authors' findings suggest that a considerable number of code snippets are anticipated in the answers provided on this portal. Yang et al. analyzed Stack Overflow's application compatibility level, discovering that just one percent (1 percent) of derived Java code could be compiled successfully. These authors showed that introducing class structures and semicolons to the system in incomplete fragments might collect an additional two percent (totalling 3 percent) of content. While these papers investigate some attributes of Stack Overflow code (i.e., security or usability), a comprehensive assessment of code quality is required in Stack Overflow posts [12].

Indeed, a variety of studies have recently documented the usage of Stack Overflow code snippets in open-source projects and examined their effect on application quality. In terms of technology continuity over time, Ahmad and Ó Cinnéide analyzed GitHub projects which have repeated Java code snippets. They observed that 42 per cent of the project groups displayed diminished unity, much of which did not restore the demonstrated solidarity until incorporating the snippet Stack Overflow application. Abdalka-reem et al. assessed project efficiency by classifying code changes as either bug fixing or non-bug fixing. Ragkhitwetsagul et al. [13] surveyed Stack Overflow users who found that significant issues associated with stack overflow responses involve outdated solutions and unstable code. It was verified by the discovery of Java fragments used in open-source initiatives, 66 per cent of which were deemed obsolete implementations and more than 5 per cent deemed unstable. Campos et al. compiled and evaluated Stack Overflow JavaScript fragments in terms of code law

ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

breaches (using ESLinter), with the most common forms of design breaches (82.9 per cent of violations). A limited amount of code fragments correlated with breaches have been identified to be included in projects under GitHub. In terms of technological debt (the work needed to correct system inefficiencies), Nikolaidis et al. calculated consistency, finding that the Java application fragments were generally correlated with an overall lower technological debt level than the project code [14]. Numerous analyses likewise took a gander at tests of Stack Flood code for highlights certainly connected with item consistency. Treude and Robillard evaluated the obvious substance of Java code parts on Stack Flood; An audit of code passages overlooked the reaction text and the test remarks. These were acquainted with individuals on GitHub, who concluded that less than half of the code tests were viewed as clear as crystal [15]. Since the piece isn't simple, the ill-advised utilization of a Stack Flood scrap in a product advancement venture is probably going to influence the consistency of the task code. This could likewise be proper to survey the inquiry and answers gave the code piece (counting client remarks) for it to act naturally illustrative. Wu et al. Dissected whether bits of Stack Flood code was remembered for open-source programs, finding that 44 percent of documents containing Stack Flood passages have been changed before getting included all through the program. This might be a pointer that the underlying piece loses application consistency [16]. An engineer overview indicated that 32 percent of respondents re-executed code scraps as opposed to reusing them in their unique state inferable from the conviction that these code bits might be of bad quality. Inferable from fluctuating translations and inclinations, this can be trying to portray while assessing conceivable substance ideas about code (or programming). Although the investigations, as referenced above, apply all the more extensively to the information, a few examinations explicitly address the consistency of data. Jones and Bonsignor look at various parts of programming quality, for example, specialized quality, nature of the procedure, and nature of utilization, adding up to 121 quality properties. Since the exploration focuses on programming quality as opposed to code quality, these authors talk about significant innovative quality-related variables that might hold any importance with this theory [17]. The ISO250103 determination depicts the consistency of the product item as giving specialized reasonableness, toughness, the effectiveness of the activity, openness, solidness, adaptability, viability, and convey ability, with a scope of sub-attributes for each. Lu et al. allude to the consistency of programming as clarity and security, using assets to assess programming issues, for example, possible glitches and application remarks. These creators are additionally examining quality against easygoing and experienced clients [18].

As for Stack Overflow, Rahman et al. say that address comments may be used as a reference for assessing the consistency of code snippets. We conclude that fragments of imperfect performing code will produce further feedback. In reality, a discussion has echoed disputes over the texture of Stack Overflow questions and answers. Throughout the light of these studies, it is evident that the software development community recognizes different components of code quality [19].

III. METHODOLOGY

In keeping with the open nature of our overarching question, what is the quality of code snippets on Stack Overflow in the answers? To our analyzes, we use an exploratory style, beginning with a detailed quantitative analysis before carrying out a more in-depth qualitative study. A recent review discusses consequences for security failures and the actions of the Stack Overflow group to maintain an understanding of possible application consistency vulnerabilities by users. We give more information on the two (quantitative and qualitative) methods of research below.

Criteria for standard code excerpts:

Code quality is hard to describe [20], and as such, this analysis must identify what is meant by code snippet quality. It is evident when analyzing the literature that the standard of code comprises several dimensions. The quality of application and code snippet content are evaluated both individually and about each other to define a collection of code quality attributes in

ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

software quality assessments [21]. When trying to identify the quality of code snippets, it can be essential to consider the quality of software at a higher level, as this topic is defined and understood more widely. However, when evaluating the consistency of the application fragment, it is presumed that such excerpts would be copied and pasted onto a broader project (i.e., they are a subset of technology). Therefore, a literature analysis was performed to understand the different dimensions of the quality of code snippets. Notice also that it is believed that fragments of Stack Overflow code are meant to be right (i.e., not deliberately wrong) [22]. Initially, considering the idea of applications, Jones and Bonsignor take a gander at programming quality from a monetary perspective and afterwards recognize seven programming quality classifications: specialized/auxiliary quality, strategy ease, nature of utilization, simplicity of activity, visual quality, quality guidelines, and sound quality. The more exact and fitting meaning of the nature of the product is ISO25010,4, which determines the nature of the product item regarding eight highlights – specialized reasonableness, strength, execution consistency, availability, assurance, security, support, and compactness. Similarly, Spinellis and Zou et al. recommend the six quality characteristics of the program as determined in the before ISO/IEC9126 standard [23]: security, steadiness, openness, execution, support, and convey ability; insurance included as security and efficiency included as yield. The likely danger of programming changes and code-delivered mistakes (specialized obligation) is additionally now observed as vital to the consistency of programming. The two components of code quality are associated with innovative obligation, including shortcomings identified with mechanical reasonableness, sturdiness, the productivity of activity, availability, insurance, adaptability, support, and convenience. Such mistakes may bring about costs related to the absence of potential, duty, and promise to fix them. Notwithstanding, Kottom recognizes five measurements that are basic for estimating the nature of code, particularly in programming testing. Those are: application inclusion how regularly application is tried; security–guaranteeing code mitigates potential weaknesses; consistency and productivity perceiving how long a framework takes to execute various undertakings; structure and multifaceted nature utilizing a reliable format and remarking to guarantee that all engineers may interpret code; lastly, Lines of Code (LOC) [24].

Since huge numbers of the quality guidelines do exclude explicit and direct estimation markers for the consistency of code, endeavours have likewise been made to incorporate practical quality models. First of all, the Viability File (MI) has been acquainted with incorporate a proportion of the practicality of source code. The SIG Upkeep Model is said to determine the MI's weaknesses by looking past the use of a norm and remember data for specific programming inadequate support capacities. The Delta Support Model (DMM) expands the SIG Maintainability Model, which gives submit level upkeep estimations. Quamoco is frequently said to decrease the contrast among effectiveness and computation attributes. A large number to incorporate handy choices incorporate Squale, CAST is looking to finish up the innovative obligation, SQALE, and Columbus QM [25].

With Stack Flood giving less educated designers a wellspring of coding help, it is conceivable that code bit deformities can proliferate when these are accidentally reused. More experienced specialists can frequently settle on compromise decisions between long haul application consistency and transient advantages in creating quickened refreshes (e.g., because of buyer request or the need to boost profitability), which may add to specialized obligation amassing [26]. All things considered, while innovative reasonableness, life span, execution power, ease of use, unwavering quality, similarity, practicality, and convenience contemplations can help survey code scrap exactness, these boundaries are proposed to decide a whole programming venture or bundled programming, which means they are not commonly pertinent to the little bits of code on Stack Flood. In all actuality, attempting to use this far-reaching set of measurements may in any case not be down to earth (e.g., it is illogical to ascertain the viability of Stack Flood application pieces, as an application on this stage as often as possible tends to a solitary client's question or concern [27]. For this research, six possible snippet quality measurements of code were initially considered despite the proof in the general works on software quality above. Those were: reliability, stability, readability/maintenance,

ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

accessibility, compatibility, and efficiency. Readability was chosen over support because ISO25010 defined maintainability as consisting of modularity, reusability, analysability, modifiability, and testability. While such elements are more commonly useful for assessing the consistency of applications, they may be inadequate for testing code snippets [28].

Table 1

Dimensions of the content of the snippet code:

Quality Dimension	Descriptions
Reliability and Software Enforcement Codes (RQ1)	Code excerpts should not be incomplete, misleading, or vulnerable to errors in runtime, ensuring they will be able to compile (given necessary syntactic adjustments) and have no glitches or errors. Software samples will meet with commonly agreed programming laws. It can be calculated by attempting to compile and conduct code fragments, catching the particular mistake or violation
Simple to interpret (RQ2)	Code fragments will adopt standard Java readability guidelines to ensure code can be readily interpreted and preserved in the future. This can be calculated by testing whether code samples are consistent with specific rules and whether they are clarified.
Performance (RQ3)	Code excerpts will recognize the success of the suggested solutions or their output. The code fragment, for example, has answered the query in a way that saves time or growing the number of time measures.
Security (RQ4)	Software fragments will consider the security limitations of potential implementations so that protection is not affected. For example, using interpolated strings rather than prepared sentences for input into the database will be considered a security error.

IV. DATA ANALYSIS & RESULTS

In Fig.1, we represent the general result trends for violations detected by the PMD, Check style, and Find Bugs software (excluding certain abuses caused by the class declaration wrapper or particular identification file name code snippet during pre-processing). The findings in Fig.1 demonstrate that the code snippets generating errors and the code snippets not causing errors have very close characteristics.

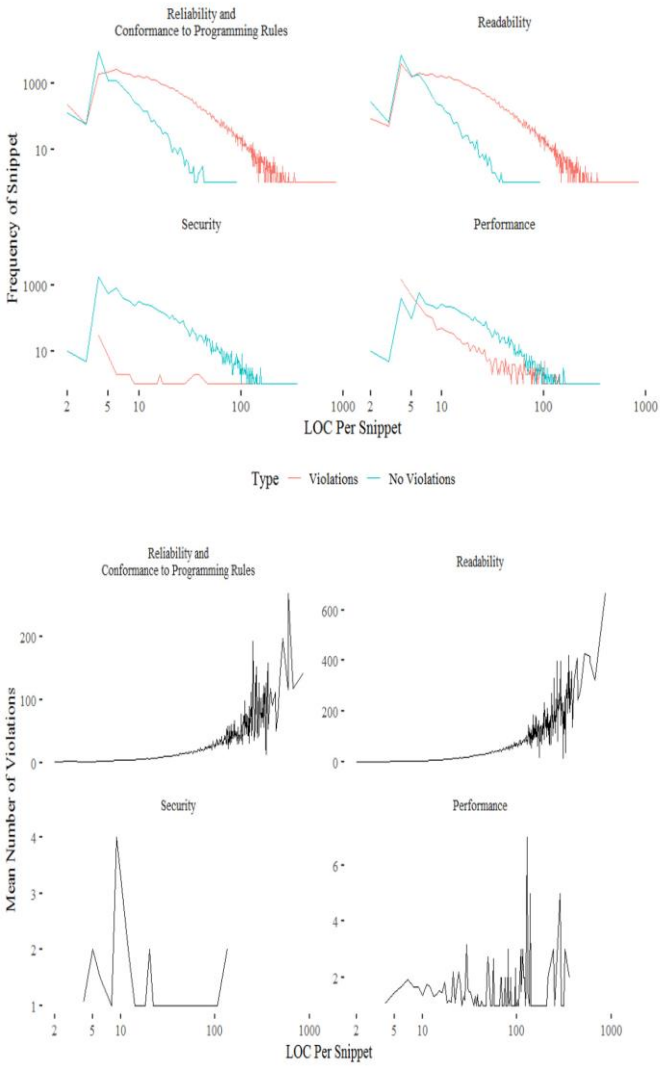

Figure 1(a) and (b)

ISSN: 2708-7123 | Volume-02, Issue Number-01 | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

Many of the code snippets that do not create reliability and compliance with programming rules errors comprise between four and 10 LOC, while the rest of code snippets yield reliability. In contrast, following programming rules, errors are more than four LOCs [29]. The resulting trend is also convergent for compliance breaches, as most code snippets that do not yield compliance errors and performance errors are larger than four LOCs. For readability and security errors, the majority of both code snippets that do not produce errors and code snippets that cause errors consist of more than four LOCs. Fig.1.b also indicates that longer code fragments are more prone to result in mistakes and a more significant number of errors, in particular for stability and consistency with programming standards and errors in readability [29].

Breaking down the first 151,954 code parts, 50,717 were tried for ease of use and congruity with PMD programming laws; however, just 50,472 were analyzed for lucidness utilizing Check structure because of blunders. Utilizing Discover Bugs, another 8,010 code pieces have been tried for consistency and unwavering quality. This pipeline is laid out and shows the measure of unpassable and uncompilable documents distinguished.

Of the 186 infringement of programming rules regarded important for code bits, the 20 percent most significant infringement (38 infringement) were resolved for code scraps. Since the PMD technique gives every infringement a need, the classifications of breaks were characterized arranged by significance (beginning with the most noteworthy need infringement) trailed by the quantity of infringement. The primary offence was Quit Tossing Crude Exemption Types, which is equivalent to a first concern since using an uncommon special case structure will deliver the wellspring of a mistake uncertain and, at last, add to investigating issues.

The penetrate is comparatively material to all code parts and Java records because of the steady need to toss a special case since dealing with mistakes is a necessary part of a program's structure. All things considered, this break was not seen as across the board in tests of Stack Flood code (multiple times or 0.12 percent). Explicit related security and understanding with programming rules surrender requested by significance (criticality), demonstrating specific occurrences, some of which are particularly unmistakable [30].

Measurements in this examination uncover that a majority of infringement of programming norms usefulness and implementation (88.32 percent) needs three ("suggested change"). Conversely, simply 0.38 percent are needed one infringement ("change important"). Around 33% of the code scraps broke down had intelligibility infringement somewhere in the range of one and five, with proficiency and insurance code pieces announcing less than one infringement overall. These outcomes show that the consistency of pieces of Stack Flood code varies depending on the perspective to be broke down [31].

V. DISCUSSION

Questions and answers stages like Stack Flood are now fundamental to how web engineers address their data holes while creating applications. In this manner, the interest and push to comprehend the idea of the data offered on these entries are that. We set out to clarify the consistency of code portions that are normally introduced in Stack Flood postings. To do as such, we operationalize code scrap quality along with the elements of dependability and consistency with programming rules, comprehensibility, execution, and security, given that pieces of Stack Flood code regularly focus on the inquiry or worry of a particular client.

We at that point produced four examination questions (RQ1–RQ4) and, in the past section, detailed related discoveries. Here we dissect these discoveries and talk about their results in the accompanying four sections, concentrating on our investigation of the infringement of the Stack Flood code pieces against the standard boundaries estimated, just as the types of infringement that are clear in the code scraps given by supporters [32].

VI. CONCLUSION

In this investigation, I set out to address the focal inquiry: What is the consistency of code bits in Stack Flood reactions? I saw that even though reviews have raised worries about the consistency of the substance introduced in Stack Flood reports, scarcely any endeavours have been made to decide the nature

ISSN: 2708-7123 | Volume-**02**, Issue Number-**01** | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

of code pieces on this site. This is awful, as exploration has exhibited that engineers use this discussion widely during application creation to take care of issues. To fix this void, I built up a goal and embraced an exploratory strategy for assessing the substance of Stack Flood code bits. Characterizing application quality bit was essential to tending to our general question. Code piece quality as depicted in four measurements by our evaluation of the work's code quality body: unwavering quality and understanding with programming laws, lucidness, execution, and insurance. Suitable techniques were surveyed and picked for our investigates, and pieces of Stack Flood code were dissected in contrast with singular estimations.

My examination was planned for deciding the quantity of value infringement per scrap, just as the types of such infringement for every one of the four quality elements of the bit language. All things considered, 4.8 solidness and consent to programming rules mishandles and 10.5 meaningfulness penetrates. All things considered, however, there were simply 0.5 breaks of wellbeing in application tests, with much less penetrates of assurance found. Huge numbers of the infringement of similarity and congruity with programming norms happened in the "suggested improvement" run, even though infringement of limit read in the blank area were more incessant. Infringement of security was for the most part new field and unused field classes, with the changeable static field bunch speaking to the greater part of infringement of insurance. I found that more extended parts of Stack Flood code had more incredible toughness and consistency with programming laws and intelligibility manhandles. All things considered, the gathering has given little consideration to these portions and may therefore not be broadly reproduced. Our discoveries indicated that sections of Stack Flood code that were longer had less precision penetrates, recommending that more significance was offered to these arrangements by the arbitrators offering responses.

At last, however alert is required, the examples from Stack Flood broke down in this investigation were not of the alarmingly awful norm. Such sections, thusly, scored far higher on consistency and assurance than ease of use and congruity with programming rules and meaningfulness.

REFERENCES

[1] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, R. Ali, Crowdsourcing: a taxonomy and systematic mapping study, Comput. Sci. Rev. 17 (2015) 43–69.

[2] J. San Pedro, A. Karatzoglou, Question recommendation for collaborative question answering systems with RankSLDA, in Proceedings of the 8th ACM Conference on Recommender Systems, ACM, 2014, pp.193–200.

[3] I. Srba, M. Bielikova, A comprehensive survey and classification of approaches for community question answering, ACM Trans. Web 10(3) (2000) 18.

[4] P.C. Rigby, M.P. Robillard, Discovering essential code elements in informal documentation, in Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 2013, pp.832–841.

[5] C. Shah, S. Oh, J.S. Oh, Research agenda for social Q&A, Library Inf. Sci. Res. 31(4) (2009) 205–209.

[6] Y. Jin, X. Yang, R.G. Kula, E. Choi, K. Inoue, H. Iida, Quick trigger on stack overflow: a study of gamification-influenced member tendencies, in: Proceed-ings of the 12th Working Conference on Mining Software Repositories, IEEE Press, 2015, pp.434–437.

[7] A.L. Ginsca, A. Popescu, User profiling for answer quality assessment in Q&A communities, in: Proceedings of the 2013 Workshop on Data-Driven User Behavioral Modelling and Mining from Social Media, ACM, 2013, pp.25–28.

[8] H. Cavusoglu, Z. Li, K.-W. Huang, Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community, in: Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing, ACM, 2015, pp.171–174.

[9] C. Jones, O. Bonsignour, The Economics of Software Quality, Addison-Wesley Professional, 2011. L. Ponzanelli, A. Bacchelli, M. Lanza, Leveraging crowd knowledge for software comprehension and development, in: Proceedings of the 17th European Conference on Software Maintenance and Reengineering, CSMR, IEEE, 2013, pp.57–66.

[10] T. Kamiya, S. Kusumoto, K. Inoue, CCFinder: a multilinguistic token-based code clone detection system for large scale source code, IEEE Trans. Softw. Eng. 28(7) (2002) 654–670.

[11] F. Chen, S. Kim, Crowd debugging, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, 2015, pp.320–332.

[12] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Routledge, 2013.

ISSN: 2708-7123 | Volume-**02**, Issue Number-**01** | March-2021
LC INTERNATIONAL JOURNAL OF STEM
Web: www.lcjstem.com | DOI: https://doi.org/10.47150

W. Cunningham, The WyCash portfolio management system, ACM SIGPLAN OOPS Messenger 4(2) (1993) 29–30.

[13] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, X. Zhang, Which non-functional requirements do developers focus on? An empirical study on stack overflow using topic analysis, in: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, MSR, IEEE, 2015, pp.446–449.

[14] D. Westen, R. Rosenthal, Quantifying construct validity: two simple measures, J. Pers. Soc. Psychol. 84(3) (2003) 608.

[15] D. Spinellis, Code Quality: The Open Source Perspective, Adobe Press, 2006.

[16] M. diBiase, A. Rastogi, M. Bruntink, A. van Deursen, The delta maintainability model: measuring maintainability of fine-grained code changes, in: 2019 IEEE/ACM International Conference on Technical Debt, TechDebt, IEEE, 2019, pp.113–122.

[17] H. Cavusoglu, Z. Li, K.-W. Huang, Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community, in: Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing, ACM, 2015, pp.171–174.

[18] D. Yang, A. Hussain, C.V. Lopes, From query to usable code: an analysis of stack overflow code snippets, in: Proceedings of the 13th International Conference on Mining Software Repositories, ACM, 2016, pp.391–402.

[19] S. Subramanian, R. Holmes, Making sense of online code snippets, in: 2013 10th IEEE Working Conference on Mining Software Repositories, MSR, IEEE, 2013, pp.85–88.

[20] T. Bakota, P. Heged˝us, P. Körtvélyesi, R. Ferenc, T. Gyimóthy, A probabilistic software quality model, in: 2011 27th IEEE International Conference on Software Maintenance, ICSM, IEEE, 2011, pp.243–252.

[21] S. Baltes, R. Kiefer, S. Diehl, Attribution required: stack overflow code snippets in GitHub projects, in: Proceedings of the 39th International Conference on Software Engineering Companion, IEEE Press, 2017, pp.161–163.

[22] V. Bauer, J. Eckhardt, B. Hauptmann, M. Klimek, An exploratory study on reuse at Google, in: Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices, ACM, 2014, pp.14–23.

[23] U. Campos, G. Smethurst, J.P. Moraes, R. Bonifácio, G. Pinto, Mining rule violations in JavaScript code snippets, in: MSR, IEEE Press, 2019, pp.195–199.

[24] H. Cavusoglu, Z. Li, K.-W. Huang, Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community, in: Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing, ACM, 2015, pp.171–174.

[25] F. Chen, S. Kim, Crowd debugging, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, 2015, pp.320–332. J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Routledge, 2013. W. Cunningham, The WyCash portfolio management system, ACM SIGPLAN OOPS Messenger 4(2) (1993) 29–30.

[26] S. Ercan, Q. Stokkink, A. Bacchelli, Predicting answering times on stack overflow, in: Proceedings of the 12th Working Conference on Mining Software Repositories, IEEE Press, 2015, pp.442–445.

[27] N.A. Ernst, S. Bellomo, I. Ozkaya, R.L. Nord, I. Gorton, Measure it? Manage it? Ignore it? Software practitioners and technical debt, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, 2015, pp.50–60.

[28] A.L. Ginsca, A. Popescu, User profiling for answer quality assessment in Q&A communities, in: Proceedings of the 2013 Workshop on Data-Driven User Behavioral Modelling and Mining from Social Media, ACM, 2013, pp.25–28.

[29] M. Squire, C. Funkhouser, “A bit of code”: how the stack overflow community creates quality postings, in: 2014 47th Hawaii International Conference on System Sciences, HICSS, IEEE, 2014, pp.1425–1434

[30] D. Westen, R. Rosenthal, Quantifying construct validity: two simple measures, J. Pers. Soc. Psychol. 84(3) (2003) 608.

[31] H. Suri, Purposeful sampling in qualitative research synthesis, Qual. Res. J. 11(2) (2011) 63.