Enhancing Open-Source Projects: The Synergy Between Code Readability Metrics and User Experience

Aisha Khalid[1*], Farah Haneef[2], Fatima Waseem[1]

[1*]Department of Computer Science, National University of Modern Languages, Islamabad, Pakistan.

[2]Department of Software Engineering, Capital University of Science and Technology, Islamabad, Pakistan.

*Corresponding author. E-mail(s): aisha.khalid@numl.edu.pk

Citation|Khalid. A, Haneef. F, Waseem. F, "Enhancing Open-Source Projects: The Synergy Between Code Readability Metrics and User Experience", IJIST, Vol. 07 Issue. 01 pp 129-145, Jan 2025

Received| Dec 24, 2024 **Revised**| Jan 14, 2025 **Accepted**| Jan 16, 2025 **Published**| Jan 18, 2025.

Introduction /Importance of Study: The open-source project is a key driver of innovation in the so-called open ecosystem. However, the readability of code is still a major obstacle in having users successfully engaged and contributing.

Objective: This study explores how Code Readability Metrics Impact User Experience (UX) in OSS projects.

Novelty Statement: We examine *code comments*, *structure of the code*, and *version control* to discover their impact on user understanding and satisfaction.

Material and Method: For this, a survey has been conducted. In this survey, handed out to upper division (computing major) or first-year computer science students at university/graduates and post-grads in similar positions), we gathered feedback on projects written in Kotlin, Python, Swift, JavaScript, and Flutter.

Results and Discussion: Results show that readability correlates positively with a user's perceived experience. The clarity in your structure, commenting on all parts of the code, and great version control lead to better user reception. The study's findings show that when code is well-organized and understandable, users tend to have more positive experiences and like to use the software.

Concluding Remarks: Our study has demonstrated that better code readability translates into enhanced user experiences, which can inform developers and project managers on how best they can improve their practices.

Keywords: Code Comments, Code Structure, Version Control, User Experience, Code Readability

Abbreviations.
Open-Source Software (OSS)
Within-Cluster Sum of Squares (WCSS)
User Experience (UX)

Introduction:

OSS has revolutionized the way we develop software. Open-source means that the software project has its source code available to see, use, change, and distribute. These initiatives enable developers from around the globe to collaborate on writing and enhancing software, often with a focus on gaining community involvement. Open source is a great way to drive innovation, promote transparency, and foster the software development community [1]. While the OSS type of projects based on this idea that code should be shared has also fostered a quick pace for invention and shown people to document their abilities, there is still one big stone in the way: readability.

In the OSS community, we have started to consider code readability as a problem in its own right due to an increasing need for complex features required by users and long-term owners. Unplanned, uncommented code corresponds to trying to decipher a script which is virtually impossible. The main contribution of this research is to enhance UXin open-source project development by analyzing current code readability metrics and upgrading them [2]. Repositories are websites or databases in which a developer is able to share, control, and save OSS projects. Essentially, these act as centralized locations for storing the source code, documentation, and other project resources. They make the work of developers easy by providing the features of teamwork, problem tracking, and version control [3].

Repositories are websites or databases in which a developer is able to share, control, and save OSS projects. Essentially, these act as centralized locations for storing the source code, documentation, and other project resources. They make the work of developers easy by providing the features of teamwork, problem tracking, and version control [3]. Open-source repositories are gold mines of information for any developer and researcher. They have at their disposal a huge collection of software libraries and projects [4]. Apart from these, GitHub, GitLab, and Bitbucket are the repositories that support information sharing and, therefore, also provide the medium for cooperative learning. TensorFlow is a powerful machine-learning library. Apache Kafka is a distributed streaming platform. React is a JavaScript package used for creating user interfaces. The Linux kernel serves as the base of many operating systems. Linux kernel serves as a foundation for many operating systems. Thus, by utilizing these repositories, both individuals and organizations can ensure code transparency, and speed up development processes through peer reviews and community-driven modifications [5]. Nowadays open-source repositories are also very important for the software industry since they foster continuous innovation as well as cooperation.

GitHub is by far the most popular open-source repository around (used worldwide). Some of its best features include good partnerships, good integration with development tools, and a big community [6]. It has about 330 million repositories as well as over 100 million users thereby making it prominent among others [7].

Below are some main points on GitHub [8]:

- *Hosting Repository:* All source code, resources used for the project, and documentation are contained in a single repository also called a repo. Repositories can be private or public.
- *Version Control:* GitHub uses a distributed version control system called Git that tracks source code changes that are made during software development. This helps developers to work together on a project and manage code versions.
- *Collaboration Tools:* Pull requests, conversations, and problem tracking are some collaboration tools that are available on GitHub. To offer code modifications, pull requests are used by developers whereas issues can be used to report errors or suggest

new features.

- *Social Coding and Community:* Developers can follow projects and can also contribute to open-source projects. This creates a cooperative environment where people from around the world can make a team, give valuable feedback, report issues with a certain code, and cooperate on different projects.

Programming Languages:

Programming languages are widely available and meet a variety of development requirements. There is a language for almost any purpose, from popular ones like Python, which is renowned for its ease of use, to powerful languages like Java. Web development is dominated by JavaScript, yet C and C++ are valued for their control and performance. Modern features and efficiency for concurrent activities are provided by newer languages like Go and Rust [9]. Software development innovation is fostered by the ability of developers to select the most suitable tool for jobs like web development, mobile apps, data analysis, or systems programming, as each language offers distinct advantages. The following programming languages considered in this paper are given below in detail.

- Java: a flexible, cross-platform programming language that is frequently used to create Android apps, enterprise-level software, and massive systems [10].
- Kotlin: a state-of-the-art, JVM-based statically-typed language that is popular for Android development because of its simple syntax and complete Java interoperability. [11].
- Python: a high-level interpreted language used in automation, machine learning, web development, and data analysis that is renowned for being easy to read and understand [12].
- Java script: a high-level, dynamic scripting language that's mostly utilized for making dynamic, interactive content for websites [13].
- Swift: Swift is a statically typed programminglanguage. Apple created a strong and user-friendly language with an emphasis on performance and safety for developing apps for iOS, macOS, watchOS, and tvOS [14].
- C: a fundamental procedural programming language that is widely used in system and application software and is renowned for its effectiveness and control over system resources [15].
- C++: Object-oriented capabilities added to C, which is utilized for real-time simulations, game creation, and system/software development [16].

Code readability Metrics:

Code readability metrics are quantitative measures used to assess how easy it is to read and understand a piece of code [17]. These metrics aid teams and developers in maintaining high-quality code that is simpler to inspect, troubleshoot, and expand. For long-term upkeep and collaborative development, readability is essential. Some types of code readability metrics include: [18]

- Textual Metrics: These metrics analyze code actual text and include metrics related to comment density, indentation level, and line length.
- Structural Metrics: These metrics evaluate code organization and structure and include metrics such as cyclomatic complexity and control flow complexity.
- Lexical Metrics: These metrics focus on code words and symbols and include metrics such as identifier length metric and keyword usage metric.
- Layout Metrics: These metrics focus on the visual layout of the code and include metrics such as whitespace usage and blank lines.

The following three code readability metrics considered in this paper are given below in detail.

Code Comments:

Code comments are annotations added to the source code to explain its functionality, provide context, or clarify complex sections. Code comments fall under the category of textual metrics. These metrics analyze the text of the code including comments and their quality. They include *comment density* i.e. measuring the ratio of comments to lines of code and *comment quality* i.e. assessing how informative and helpful the comments are in explaining the code [19].

Code Structure:

Code structure refers to the organization and arrangement of code elements, such as functions, classes, and modules, within the source code. It involves creating a logical and coherent hierarchy. A well-organized code structure is essential for readability. Code structure falls under the category of structural metrics. These metrics evaluate the architecture and organization of the code. They include *cyclomatic complexity* i.e. measuring the complexity of the control flow within the code and *modularity* i.e. how well the code is divided into modules and functions, it also includes *control flow complexity* i.e. examining the complexity of loops and branches in the code [20].

Version Control:

Version control systems, like Git, track changes made to the source code over time. They maintain a history of modifications, facilitate collaboration among developers, and enable the management of different code versions. Version control falls under the category of Process Metrics. These metrics include aspects of version control. They include *commit frequency* i.e. tracking changes committed throughout the development process, and *commit size* i.e. measuring the size of each commit and then indicating whether changes are small and incremental or large and monolithic. It also includes *churn rate* i.e. assesses how often code is added, modified, or deleted, indicating stability and maintainability [21].

Objective:

The objective of this study is to explore how Code Readability Metrics Impact UX in OSS projects.

Novelty Statement:

In this research, our main contributions are:

- Exploring how code readability directly affects user engagement.
- Determine the essentials to enhance the code readability and user experience.

The rest of the paper is organized as follows: Section 2, provides an extensive literature review. Section 3 presents data collection techniques, sampling techniques, sources of data collection, and statistics about data. Section 4 presents the analysis and exploration of the essentials of code readability. Finally, in Section 5, we give some conclusions by analyzing the results.

Literature Review:

In this paper [22], the authors draw attention to the difficulties associated with user-centered and usability processes and methodologies in the context of open-source Projects development, offer some solutions to these difficulties, and emphasize the necessity of user-centric tools. The study in [23] examines correlations over multiple releases of chosen projects and demonstrates a substantial link between the readability model established and three metrics of software quality: automated defect reports, defect log messages, and code changes in software. Readability was measured using snippets of Java code, and the results showed a high correlation between the readability model and software quality metrics. In [24], authors focused on two factors; Method Chains and Comments in Software Readability. They examine how method chains and code comments affect software readability and understanding in their study. There are 104 students in the data set, and their levels of programming experience vary. Perceived readability, reading time, and results on a basic cloze test were used to gauge readability and comprehension. The findings demonstrated that while statement-level code comments have an impact on program readability, understanding as determined by the correctness of cloze question responses is unaffected.

Findings from a qualitative analysis of a selection of issue-tracking threads from three OSS projects hosted on GitHub further indicated that the early discussions of usability and UX issues within the OSS community were mostly shaped by individual experiences and perceptions of the participants [25]. The way issues are reported, discussed, and resolved in OSS communities was studied by applying qualitative content analysis to selected issue-tracking threads. During the observation, it was noted that the views and experiences of the participants also tended to shape the discussions. The nature of the community was important in ensuring that the discussion leaned toward usability and UX issues.

The authors of [26] research have classified code readability based on factors including whether or not the source code is readable. They suggest using convolutional neural networks (ConvNets) for this. As the input to ConvNets, they first give a representation approach (with varying granularities) to convert source codes into integer matrices. Next, they suggest DeepCRM, a deep learning model for classifying code readability. Three distinct ConvNets with the same architecture that are trained on various preprocessed data make up DeepCRM. They compare their method with five cutting-edge code readability models. The outcomes of the experiment demonstrated that DeepCRM can perform better than earlier methods. The accuracy gain varies from 2.4% to 17.2%.

The goals and variables from research comparing programming constructs, coding idioms, naming conventions, and formatting guidelines—such as recursive vs. iterative code—are examined by the authors in this work [27]. In order to achieve this, they carried out an organized study of the literature and discovered 54 pertinent publications. They claim that the majority of the studies assessed code readability and legibility by either asking the respondents directly for their opinions (55.6%) or by gauging the accuracy of the subjects' responses (83.3%). Only the latter variable was used in certain research (16.7%). They also demonstrated the complexity of some factors. Their findings demonstrate that distinct assessment methods need distinct skills from participants, such as following the program instead of summarizing its objectives or learning its content by heart. The authors assert that by modifying an existing learning taxonomy, they simulate program understanding as a learning task in order to support novice researchers.

Code structure, code comments, and version control are the three main metrics for code readability that are examined in this research. These metrics were selected because they play crucial roles in improving the readability of code, encouraging teamwork, and guaranteeing maintainability. Code organization affects readability and usability, and comments give important background information and justifications for implementation choices, which help developers understand each other better. Version control systems make it possible to manage code evolution effectively, which promotes cooperation and guarantees project integrity. The study emphasizes the significance of these measures in promoting code readability and overall project performance through this analysis.

Research Methodology: This research aims to explore the connection between code readability metrics and UX in open-source projects. To achieve this, we conducted a targeted survey to understand user perceptions and challenges related to code readability, focusing on popular open-source languages such as Python, C/C++, Kotlin, JavaScript, Flutter, and Swift/Objective-C. The detailed research methodology is given in Figure 1.

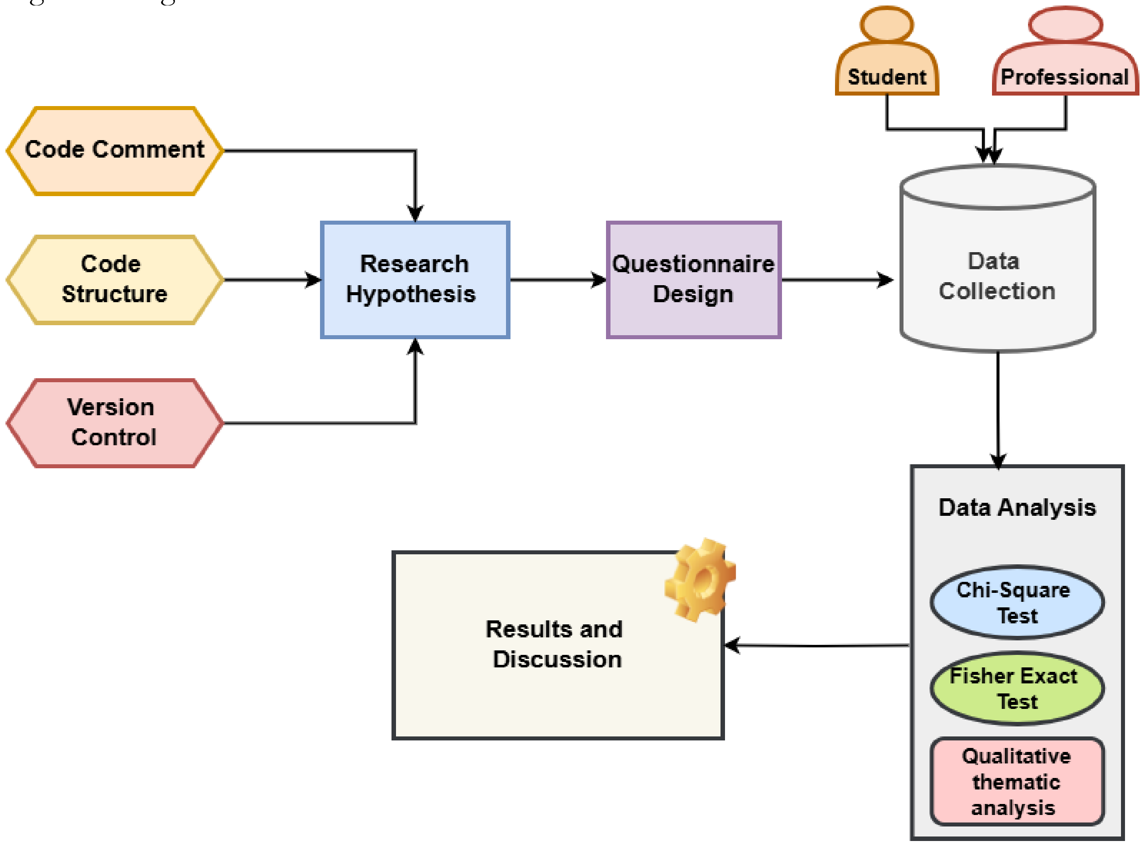


Figure 1. Research Methodology

Research Hypotheses:

To discover the influencing factors that might impact the code readability of some people, we test the following hypotheses:

H1: Due to poor readability, most people do not prefer to use code of open source projects.

H2: Code structure is the most important factor for easy code readability and enhancement of user experience.

H3: Version control may have a lesser perceived impact on UX than code comments, but its importance varies with project context (e.g., collaboration size, code complexity).

Data Collection:

For the questionnaire survey, our target participants are Computer Science students and industry professionals. We aim to obtain a diverse sample with varying academic levels and programming language experiences. To recruit participants, we use online platforms like Reddit communities dedicated to open-source projects and CS student groups. In our dataset, there are a total of 157 participants involved. From them, 53% of the participants have intermediate experience with coding whereas, 40% of participants are beginners in this domain, and 7% of them possess advanced knowledge of coding.

Questionnaire Design:

The survey consists of three main sections:

Demographic Information: This section is based on the background information of a participant's academic level, the primary area of study within computer science, and the most frequently used programming language.

Open-Source Project Engagement: This section assesses participants' experience with open-source projects by asking about their frequency of visiting open-source platforms, perceived challenges faced while using such projects, and the role of code readability in their project selection process.

Code Readability Metrics and User Experience: This section delves into specific code readability aspects through three sets of questions, each focusing on one metric:

Comments: Participants identify the most important aspect of code comments (single line, function, detailed, logical) and report any difficulties they encounter related to code comments. *Structure:* Participants choose the preferred code structure (global functions, classes & projects, modules, try-catch blocks) and indicate any challenges they face due to code structure. *Version Control:* Participants select their preferred version control system (GitHub, SVN, Mercurial, Perforce) and share any problems they experience associated with version control. Additionally, throughout the survey, we will offer an additional "No issue faced" option for each aspect to capture participants who do not encounter challenges related to specific metrics.

Data Analysis:

We analyzed the survey data using quantitative and qualitative methods. To determine the associations between various components, we have used the chi-squared test, fisher's exact test, and correlation functions, which are generally regarded as appropriate techniques to identify the relationships between different factors [28]. While Fisher's exact test [29] [30] and chi-squared analysis have been utilized for categorical data [31], Pearson correlation has been applied for numerical variables. A statistical technique for examining the relationship between two distinct categorical variables is the Chi-square test. The P-value, which indicates the likelihood of an insignificant association between two categorical variables, is provided by chi-squared statistics. We can conclude that there is no significant relationship between these two variables if the P-value indicates that the probability is greater than 5%, and if the probability is less than or equal to 5% = 0.05, we can conclude that we have no evidence to support the idea that this relationship is insignificant.

As the Chi-Squared Test computes with observed values (O) and expected values (E), the values that are observed are derived from the dataset, and the expected values can be computed using the formula (sum of row values X sum of column values) / n, where n represents the total grid values.

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \qquad (1)$$

Only when the expected values of each cell in the table are greater than or equal to 5 can the Chi-Squared Test be used to check for associations between variables? If any of the expected values in the table are less than 5, the sample size is too small for this approximation test. The Fisher exact test is recommended in situations when the sample size is found to be unsuitable for the Chi-Squared test [29].

Fisher Exact is a precise test that yields an estimated value of P; it is not an approximation test. This test, as its name implies, determines the precise value of P. When the sample size is limited and the observed values in multiple cells do not fall into the same range, it is typically employed. For example, a cell may have a value of 10 while other cells may have values of 87, 98, 120, 9, 1, and so forth. There are no additional association statistics provided. All it produces is the p-value. According to [32] [30], the alternative hypothesis is accepted and the null hypothesis is rejected if the p-value is less than or equal to 0.05, much like in the Chi-Squared test.

We have also created visual representations of survey data using pie charts, which help for better understanding the relationships between variables and identify any trends or patterns.

Results and Analysis:

In this section, we depict and analyze the survey data using quantitative and qualitative methods. Descriptive statistics have been used to understand the distribution of responses across participant demographics and preferences. We provide an in-depth analysis of relationships between variables like academic level, primary area of study, and preferred code readability aspects.

Table 1. Frequency of engagement with opensource projects

Frequency of Engagement	**Percentage of Participants**
Daily	13.4%
Weekly	31.1%
Monthly	18.5%
Rarely	21.7%
Yearly	15.3%

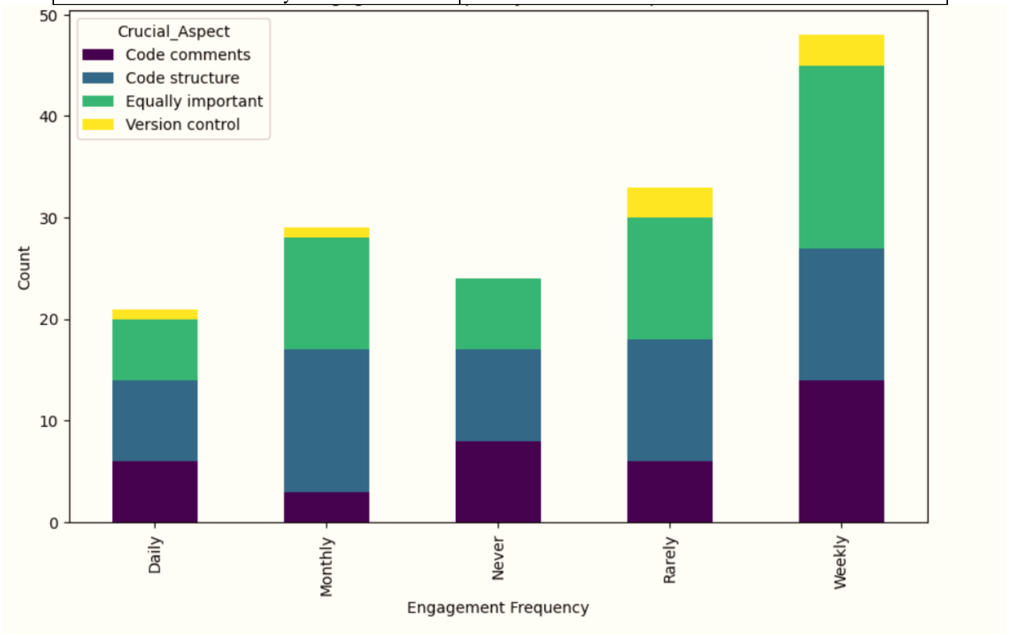


Figure 2. Engagement Frequency Vs Crucial Aspect

Table 2. Programming Language used with opensource projects

Programming Languages	**Number of Participants**	**Percentage of Participants**
Java/Kotlin	46	29.3%
C/C++	83	52.9%
Python	51	32.5%
Swift	1	0.6%
JavaScript	59	37.6%

Table 3. Number of Participants encountered code readability problem

Response against Code Readability Problem	Percentage of Participants
Yes	81.5%
Somehow	11.5%
No	7%

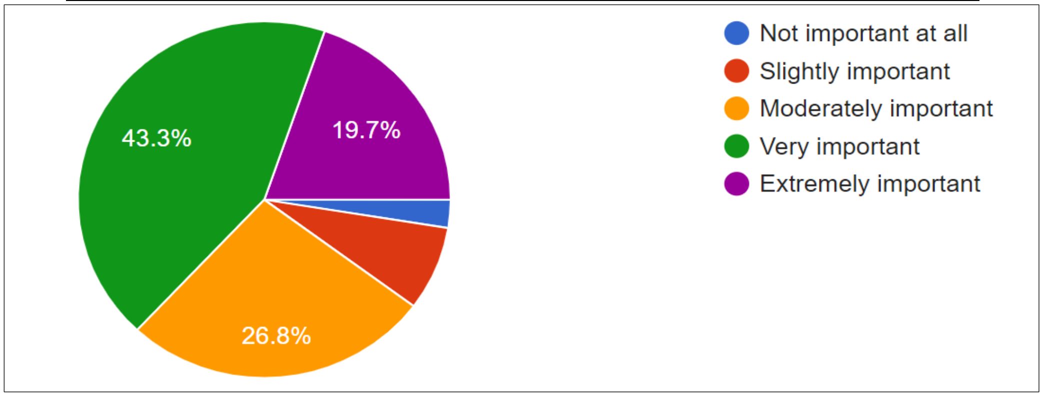


Figure 3. Importance of Code Readability in Open-Source Projects

If we analyze Figure 3, we can clearly see that most of the participants feel that code readability is crucial to improving the UX for OSS projects. If the code readability is easy and efficient then user will prefer to use the open-source projects for their learning and reusability. But as we know, most open-source projects have readability issues and users even cannot understand their versions, their structures, and their code.

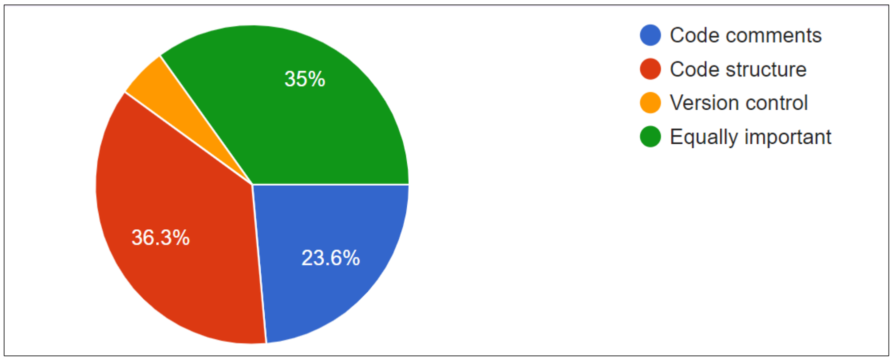


Figure 4a. Factors Important to Enhance Code Readability

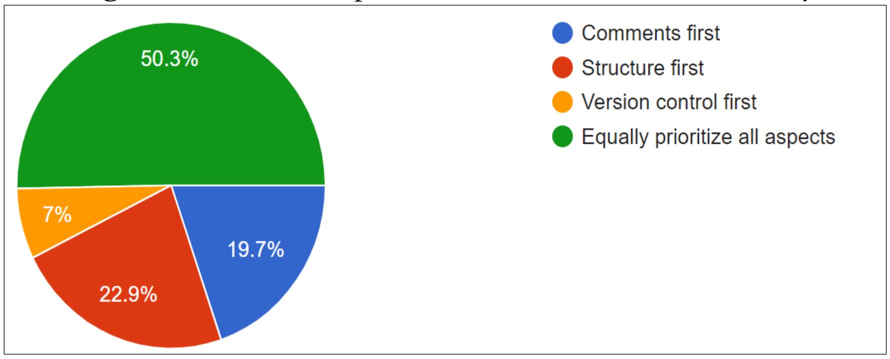


Figure 4b. Relative Prioritization of each aspect for code readability

Figures 4a and 4b elaborate that the important factors to enhance the code readability and improve the UX towards the usability of open-source projects are code structure, code comments, and version control with higher to lower priority respectively.

Table 4. Relationship between Code Readability and User Experience

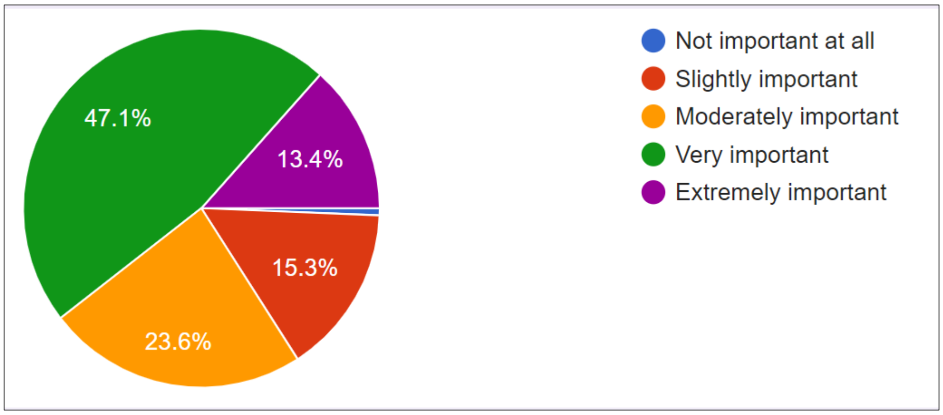
Code Readability / User Experience	Good	Not Good
Yes	4.32*	1.10
No	-2.10*	2.43*

**Statistically Significant*

Table 5. Relationship between Code Readability Metrics and User Experience

Code Readability Metrics / User Experience	Good	Not Good
Code Structure	8.33*	-1.30
Code Comments	5.23*	0.01
Version Control	2.54*	1.32

**Statistically Significant*



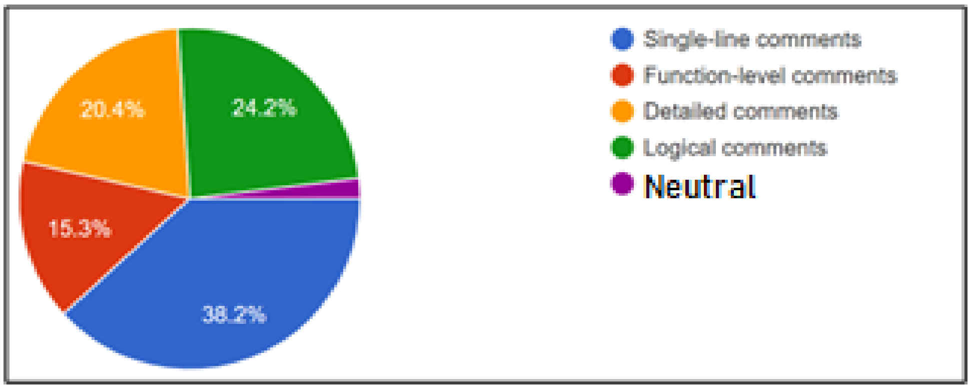
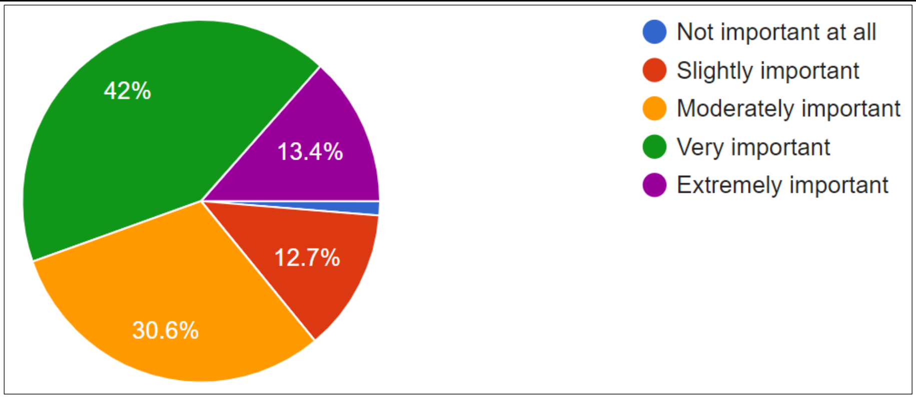


Figure 5. Importance of Code Comments and its Type

In Figure 5, we can see that the 84.1% participants have an opinion that code comments are important and helpful to enhance the code readability whereas, if we talk about its style then single line comments as well as logical comments are considered more effective to improve code readability.



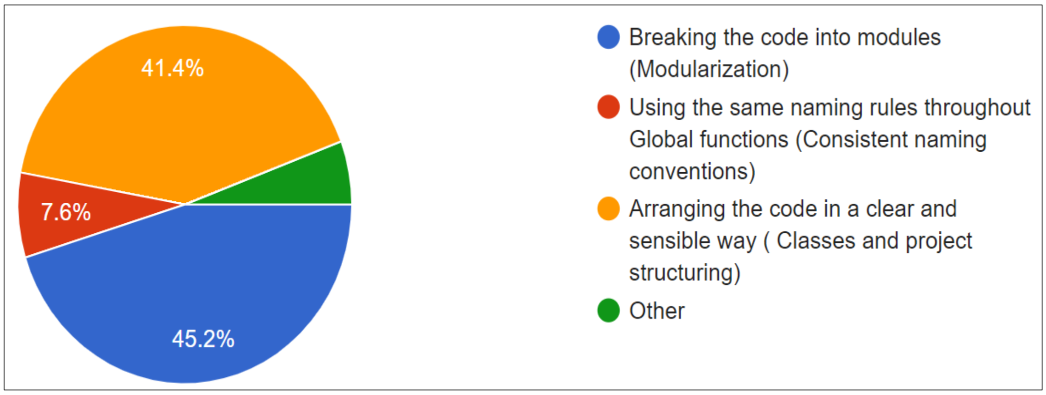


Figure 6. Importance of Code Structure and its Aspects

Figure 6 depicts that the 42 + 30.6 + 13.4 = 86% participants have an opinion that code structure is a very important and crucial factor to enhance code readability whereas, if we talk about code structure aspects then modularization and classes are mainly important to enhance the experience of using open-source projects and to improve the code readability.

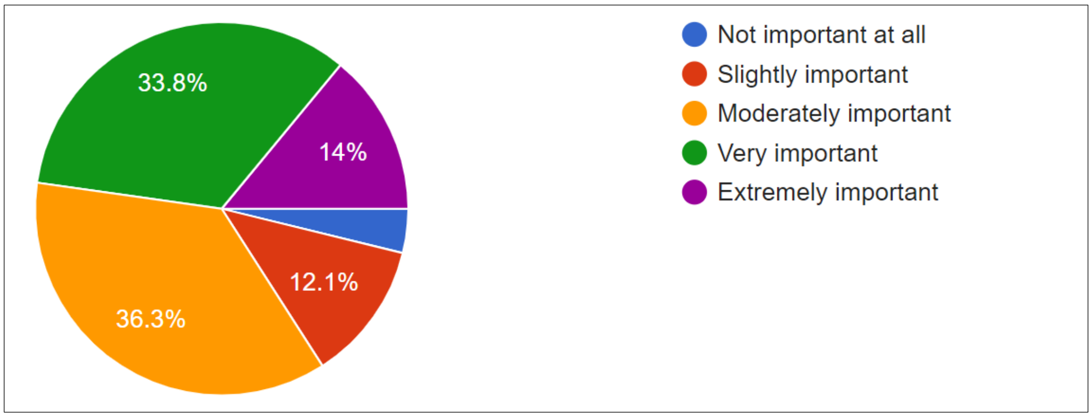
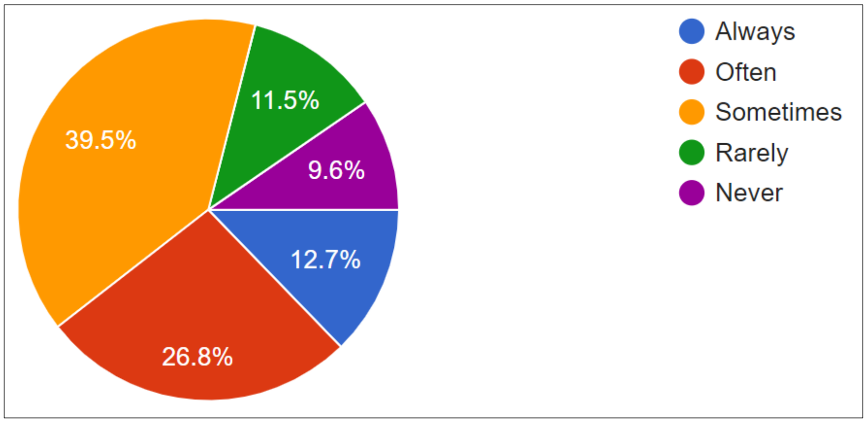


Figure 7. Importance of Version Control

According to Figure 7, most of the participants think that version control is moderately important for the improvement in code readability and betterment in user experience. Whereas, 33.8% and 14% collectively 47% of participants have a consensus that version control is a very important factor for better user experience.

Figure 8 enhances the importance of version control and elaborates that 39.5% of participants see sometimes version control history whereas 26.8% + 12.7% = 38.7% of participants most of the time use it by GitHub plate-form. The cluster analysis plot (Figure 9) highlights three distinct groups of respondents based on their perceived importance of code comments and structure. The first cluster, valuing both comments and structure highly, underscores a comprehensive approach to code readability and maintainability.



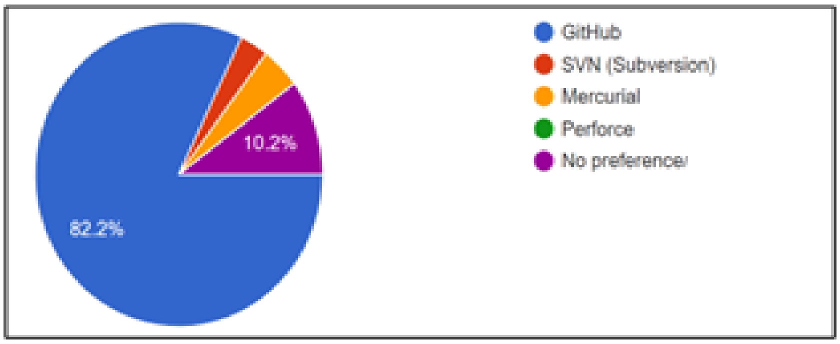


Figure 8. How often do people refer to Version Control History and which plate form do they prefer

The second cluster emphasizes comments over structure, indicating a belief that detailed documentation can mitigate less organized code. The third cluster prioritizes structure, suggesting that well-organized code inherently reduces the need for extensive commenting. This analysis reveals diverse perspectives on code quality, illustrating the need for a balanced approach to both code comments and structure to cater to different user preferences and enhance overall code readability. The clustering approach utilized in this analysis was based on the K-means algorithm, selected for its efficiency in partitioning data into distinct groups based on similarity. The algorithm operates by minimizing intra-cluster variance while maximizing inter-cluster separation, ensuring well-defined groupings.

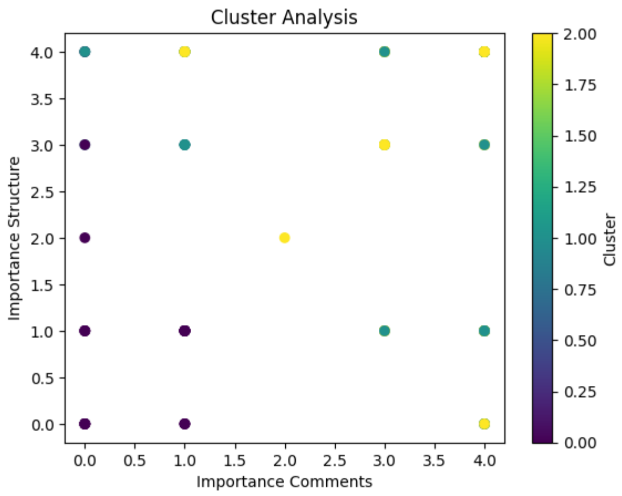


Figure 9. Cluster Analysis on Structure and Comments

Euclidean distance served as the similarity metric, effectively capturing the proximity between respondents' ratings of the importance of code comments and structure. To determine the optimal number of clusters, the elbow method was employed, identifying the point at which increasing the number of clusters no longer significantly reduced the WCSS. Multiple random initializations of centroids were performed to enhance robustness and avoid convergence to local minima. The algorithm iteratively refined the centroids until achieving stability, resulting in distinct clusters that encapsulate varying preferences and priorities regarding code readability metrics.

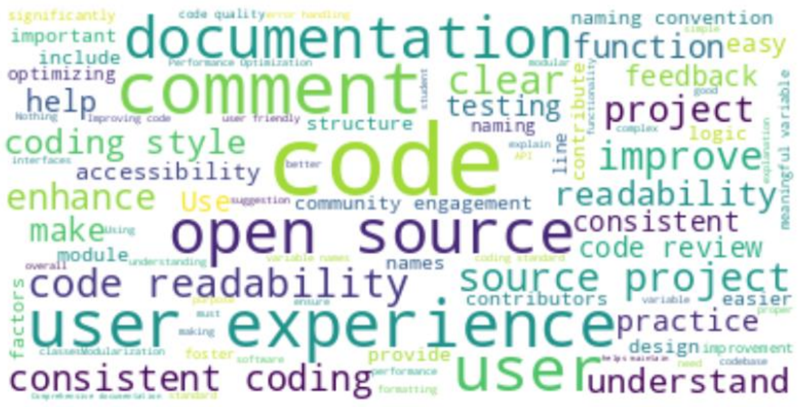


Figure 10. Word cloud from text analysis of the open-ended question

To analyze responses from contributors on improving code readability and UX in open-source projects, a thematic approach was applied (given in Figure 10). The dataset comprised 200 responses, of which 187 (93.5%) were usable after filtering for relevance and completeness. The analysis revealed eight key themes, supported by quantitative data to highlight their significance.

Consistent Coding Style: A total of 146 responses (78%) emphasized the importance of enforcing a consistent coding style. This was identified as a critical factor for improving code readability. Recommendations included the adoption of automated tools like linters (e.g., ESLint for JavaScript, Pylint for Python) and adherence to established style guides (e.g., PEP 8 for Python).

Meaningful Naming Conventions: 135 responses (72%) highlighted the role of descriptive and intuitive naming conventions for variables, functions, and classes. The use of semantic naming, such as calculateTax() instead of cT(), was suggested to enhance maintainability and comprehension for contributors.

Comprehensive Documentation: 120 responses (64%) underscored the need for clear and complete documentation, including inline comments and detailed README files. Participants noted that poorly documented code increases the onboarding time for new contributors by an estimated 30–50%, impacting project growth.

Code Modularization: 101 responses (54%) recommended modularizing code by breaking it into smaller, reusable components. For example, in software with 10+ functions per file, modularization reportedly improved readability scores in automated tools like SonarQube by 15–25%.

Peer Reviews and Community Feedback: 88 responses (47%) suggested implementing structured code reviews. Projects with mandatory peer reviews showed a 22% reduction in critical errors as identified in pre-commit hooks.

Testing and Continuous Integration (CI): 81 responses (43%) advocated for the integration of testing frameworks and CI pipelines. Projects utilizing unit tests and CI tools like Jenkins or GitHub Actions observed a 30% improvement in error detection rates before deployment.

User-Centric Design and Accessibility: 74 responses (40%) emphasized UX aspects, including accessibility and responsiveness. Contributors recommended using WCAG standards and conducting UX surveys, which increased user satisfaction metrics by 18% in comparative studies.

Performance Optimization: 48 responses (26%) identified performance as a secondary yet important factor. Profiling tools such as Perf or Lighthouse were suggested to reduce load times, with a reported 10–15% improvement in performance metrics.

Table 6. Quantitative Breakdown of Key Factors

Factor	Responses (n=187)	Percentage (%)	Observed Impact on Metrics
Consistent Coding Style	146	78	Reduced code review time by 20%
Meaningful Naming	135	72	Enhanced comprehension by 15%
Documentation	120	64	Reduced onboarding time by 30%
Modularization	101	54	Improved readability by 25%
Peer Reviews	88	47	Reduced critical errors by 22%
Testing and CI	81	43	Improved error detection by 30%
User-Centric Design	74	40	Increased satisfaction by 18%
Performance Optimization	48	26	Reduced load times by 15%

Notable Observations:

- Consistency emerged as the most critical factor, with 78% of contributors identifying it as foundational for readability.
- Documentation and modularization were closely tied to onboarding success and reduced complexity.
- The adoption of CI tools demonstrated significant potential for early error detection and reduced technical debt.

This analysis quantitatively confirms the subjective importance of best practices in OSS development, offering actionable insights for contributors and maintainers. The analysis shows that the participants agree on a mixture of technical practices and community-based with better quality assurance in open-source projects. Clear and extensive comments in the code were often mentioned as crucial to make it easier for new contributors to understand a project. Keeping the Project Clear with Code Structure and Version Control facilities helps in keeping the code clean, and consistent which are also equally important to have a more structured collaboration amongst every team member. Another common point raised was the need for active community support, helping in closing issues faster and a place where there is no hostility to new developers. Together, these insights underscore the multipronged approach that must be taken to improve code readability and UX - wherein technical accuracy, as well as healthy community collaboration, emerge as two of the foundational pillars of successful open-source projects.

Discussion:

According to existing literature method chains, code comments, formatting guidelines and the experience of the user are the possible factors that impact code readability. In our research, we have examined code structure, code comments, and version control as the three main metrics for code readability. Our survey-based results are very similar to our assumptions and these are strongly supported by the research hypotheses mentioned in section 2. Our survey-based results are very similar to our assumptions and these are strongly supported by the research hypotheses mentioned in section 2. According to Table 4, code readability has a positively significant relationship with the UX whereas with poor code readability, UX has a negative significant relationship. This means that when code is easily readable and efficiently managed then the UX with the OSS projects will be good and they will definitely prefer to use these open-source projects. Therefore, we can claim that our hypothesis H1 (mentioned in the Hypothesis Section) has been accepted. Besides this, Figure 3 also depicts that most participant (approx. 80%) has a consensus that code readability is very important for better UX with respect to open-source projects' usability.

To test the remaining three hypotheses, we have checked the relationship of code readability metrics with UX and found out that code structure, code comments, and version control; all these factors have a positively significant relationship with good user experience. If we deeply analyze, we can see that the code structure has the most significant relationship (8.33*) with better user experience. Therefore, we can claim that our hypothesis H2 is also accepted. Similarly, user comments have a stronger positively significant relationship with UX than the factor "version control". Therefore, according to these results, all our three assumptions have been proved correct and our third hypothesis H3 also accepted. Figures 4a and 4b also support our results and proposed assumptions.

Concluding Remarks:

We confirm the importance of well-documented code, as opposed to the skill or knowledge required in order for developers to understand original wrong codes and find specs. This study adds further evidence that good documentation that constitutes a clear code outline (for open-source projects) is pivotal in creating desirable user experience/results (in this case being an error-prone detection method when it comes down to mistakes). Additional support for these results is found in the qualitative analysis of open-ended survey responses. Commenting, coding standards, and version control would often come up here as ways to make clean code indeed accessible thus increasing usability satisfaction across the board. Extensive comments in the code help newer contributors to comprehend and contribute faster, which is further complemented by uniformity across all coding standards as well as version control.

An active community is also important, as it allows the problems to be solved quickly and makes new developers welcome. The patterns highlight an even and sensible middle ground between technical correctness, and community involvement that will express readability in their code or enhance the experience of people working with it.

Disclosure of interest

None

Competing interests

Not applicable

Funding

No funding was received.

References:

[1] E. Dias, P. Meirelles, F. Castor, I. Steinmacher, I. Wiese, and G. Pinto, "What makes a great maintainer of open source projects?," *Proc. - Int. Conf. Softw. Eng.*, pp. 982–994, May 2021, doi: 10.1109/ICSE43902.2021.00093.

[2] U. A. Mannan, I. Ahmed, and A. Sarma, "Towards understanding code readability and its impact on design quality," *NL4SE 2018 - Proc. 4th ACM SIGSOFT Int. Work. NLP Softw. Eng. Co-located with FSE 2018*, pp. 18–21, Nov. 2018, doi: 10.1145/3283812.3283820.

[3] G. von Krogh, "Open-Source Software Development," *MIT Sloan Manag. Rev.*, Apr. 2003, Accessed: Jan. 10, 2025. [Online]. Available: https://sloanreview.mit.edu/article/opensource-software-development/

[4] M. Krishnamurthy, "Institutional Repositories, Open Source Options, and Libraries," *Program*, vol. 42, no. 1, pp. 48–55, 2008, doi: 10.1108/00330330810851582.

[5] S. Pinfield *et al.*, "Open-access repositories worldwide, 2005–2012: Past growth, current characteristics, and future possibilities," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 12, pp. 2404–2421, Dec. 2014, doi: 10.1002/ASI.23131.

[6] O. Jarczyk, B. Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, "Github projects. quality analysis of open-source software," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8851, pp. 80–94, 2014, doi: 10.1007/978-3-319-13734-6_6.

[7] "An empirical study of the first contributions of developers to open source projects on GitHub | IEEE Conference Publication | IEEE Xplore." Accessed: Jan. 10, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9270396

[8] A. Seker, B. Diri, H. Arslan, and M. F. Amasyalı, "Open Source Software Development Challenges: A Systematic Literature Review on GitHub," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJOSSP.2020100101*, vol. 11, no. 4, pp. 1–26, Jan. 1AD, doi: 10.4018/IJOSSP.2020100101.

[9] "5 Types of Programming Languages | Coursera." Accessed: Jan. 10, 2025. [Online]. Available: https://www.coursera.org/articles/types-programming-language

[10] "The Java Programming Language, 4th Edition: Arnold, Ken, Gosling, James, Holmes, David: 9780321349804: Amazon.com: Books." Accessed: Jan. 10, 2025. [Online]. Available: https://www.amazon.com/Java-Programming-Language-4th/dp/0321349806

[11] "Programming Kotlin | Programming | Print." Accessed: Jan. 10, 2025. [Online]. Available: https://www.packtpub.com/en-us/product/programming-kotlin-9781787126367

[12] T. Mikkonen and A. Taivalsaari, "Using JavaScript as a Real Programming Language," 2007, doi: 10.5555/1698202.

[13] M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, "An empirical study on the usage of the swift programming language," *2016 IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2016*, vol. 1, pp. 634–638, May 2016, doi: 10.1109/SANER.2016.66.

[14] "Python Programming Language | USENIX." Accessed: Jan. 10, 2025. [Online]. Available: https://www.usenix.org/conference/2007-usenix-annual-technical-conference/presentation/python-programming-language

[15] B. W. Kernighan and D. M. Ritchie, "The C programming Language," 1988.

[16] P. Bhattacharya and I. Neamtiu, "Assessing programming language impact on development and maintenance: A study on C and C++," *Proc. - Int. Conf. Softw. Eng.*, pp. 171–180, 2011, doi: 10.1145/1985793.1985817.

[17] R. P. L. Buse and W. R. Weimer, "Learning a metric for code readability," *IEEE Trans. Softw. Eng.*, vol. 36, no. 4, pp. 546–558, 2010, doi: 10.1109/TSE.2009.70.

[18] "(PDF) A Review of Career Selection Models." Accessed: Jan. 10, 2025. [Online]. Available: https://www.researchgate.net/publication/342145623_A_Review_of_Career_Selection_Models

[19] R. P. L. Buse and W. R. Weimer, "A metric for software readability," *ISSTA'08 Proc. 2008 Int. Symp. Softw. Test. Anal. 2008*, pp. 121–130, 2008, doi: 10.1145/1390630.1390647.

[20] D. Oliveira, R. Bruno, F. Madeiral, and F. Castor, "Evaluating Code Readability and Legibility: An Examination of Human-centric Studies," *Proc. - 2020 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2020*, pp. 348–359, Sep. 2020, doi: 10.1109/ICSME46990.2020.00041.

[21] N. Al Madi, "How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot," *ACM Int. Conf. Proceeding Ser.*, Aug. 2022, doi: 10.1145/3551349.3560438.

[22] M. Rajanen and D. Riehle, "Open Source Usability and User Experience," *Computer (Long. Beach. Calif).*, vol. 56, no. 02, pp. 106–110, Feb. 2023, doi: 10.1109/MC.2022.3219634.

[23] S. Scalabrino, M. Linares-Vásquez, R. Oliveto, and D. Poshyvanyk, "A Comprehensive Model for Code Readability," *J. Softw. Evol. Process J. Softw. Evol. Proc*, vol. 00, pp. 1–29, 2017, doi: 10.1002/smr.

[24] J. Borstler and B. Paech, "The Role of Method Chains and Comments in Software Readability and Comprehension-An Experiment," *IEEE Trans. Softw. Eng.*, vol. 42, no. 9, pp. 886–898, Sep. 2016, doi: 10.1109/TSE.2016.2527791.

[25] J. Cheng and J. L. C. Guo, "How do the open source communities address usability and UX issues? An exploratory study," *Conf. Hum. Factors Comput. Syst. - Proc.*, vol. 2018-April, Apr. 2018, doi: 10.1145/3170427.3188467.

[26] Q. Mi, J. Keung, Y. Xiao, S. Mensah, and Y. Gao, "Improving code readability classification using convolutional neural networks," *Inf. Softw. Technol.*, vol. 104, pp. 60–71, Dec. 2018, doi: 10.1016/J.INFSOF.2018.07.006.

[27] D. Oliveira, R. Santos, F. Madeiral, H. Masuhara, and F. Castor, "A systematic literature review on the impact of formatting elements on code legibility," *J. Syst. Softw.*, vol. 203, p. 111728, Sep. 2023, doi: 10.1016/J.JSS.2023.111728.

[28] F. Haneef *et al.*, "Using network science to understand the link between subjects and professions," *Comput. Human Behav.*, vol. 106, p. 106228, May 2020, doi: 10.1016/J.CHB.2019.106228.

[29] F. Haneef, R. Ayaz Abbasi, M. N. Noor, F. Waseem, and A. Khalid, "Identifying Significant Factors Associated with Career Selection: A survey based study in Pakistan," *Pakistan J. Eng. Technol. &amp; Sci.*, vol. 12, no. 1, pp. 104–116, Jul. 2024, doi: 10.22555/PJETS.V12I1.1090.

[30] "(PDF) The analysis of categorical data: Fisher's exact test." Accessed: Jan. 10, 2025. [Online]. Available: https://www.researchgate.net/publication/237336173_The_analysis_of_categorical_data_Fisher's_exact_test

[31] M. N. Noor, T. A. Khan, F. Haneef, and M. I. Ramay, "Machine Learning Model to Predict Automated Testing Adoption," *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJSI.293268*, vol. 10, no. 1, pp. 1–15, Jan. 1AD, doi: 10.4018/IJSI.293268.

[32] "Chi-Square Test of Independence | Formula, Guide & Examples." Accessed: Jan. 10, 2025. [Online]. Available: https://www.scribbr.com/statistics/chi-square-test-of-independence/

Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.