Code Readability Management of High-level Programming Languages: A Comparative Study

Muhammad Usman Tariq[1]
Abu Dhabi School of Management
Abu Dhabi, UAE

Muhammad Bilal Bashir[2], Muhammad Babar*[3], Adnan Sohail[4]
Computing & Technology Department
IQRA University, Islamabad, Pakistan

***Abstract*—Quality can never be an accident and therefore, software engineers are paying immense attention to produce quality software product. Source code readability is one of those important factors that play a vital role in producing quality software. The code readability is an internal quality attribute that directly affects the future maintenance of the software and re-usability of same code in similar other projects. Literature shows that readability does not just rely on programmer's ability to write tidy code but it also depends on programming language's syntax. Syntax is the most visible part of any programming language that directly influence the readability of its code. If readability is a major factor for a given project, the programmers should know about the language that they shall choose to achieve the required level of quality. For this we compare the readability of three most popular high-level programming languages; Java, C#, and C++. We propose a comprehensive framework for readability comparison among these languages. The comparison has been performed on the basis of certain readability parameters that are referenced in the literature. We have also implemented an analysis tool and performed extensive experiments that produced interesting results. Furthermore, to judge the effectiveness of these results, we have performed statistical analysis using SPSS (Statistical Package for Social Sciences) tool. We have chosen the Spearman's correlation ad Mann Whitney's T-test for the same. The results show that among all three languages, Java has the most readable code. Programmers should use Java in the projects that have code readability as a significant quality requirement.**

***Keywords*—Source code; high-level programming languages; Java; C++; C#; code readability; code readability index**

I. Introduction

Software engineering is different in nature as compared to other engineering domains. Products may remain in use even if there are some imperfections in them. But a software product may go through several revisions even after development is completed until software becomes faults free. Otherwise customer may not accept and use it. Customers these days are very smart and want to know what is going inside the software and what does affect the future maintenance and cost.

Software go through several updates after the first version due to some reasons; a feature was not implemented that was required, a feature was incorrectly implemented, or a new feature is now required. This is known as maintenance and research shows that around 70% of the product cost is spent on the maintenance [2] as shown in Fig. 1. Software engineers need to ensure that the software they produce is easy to maintain. There are many factors that affect software maintainability and source code readability is one of them. Readability is how quickly a reader can read and understand the written text. Elements that make the text difficult to read and understand include; long lines, insufficient contrast, and long paragraph with no segmentation.

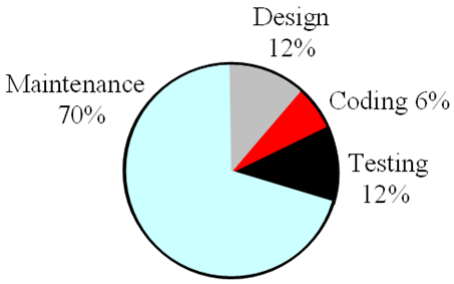

Fig. 1. Cost Distribution among Software Process Activities [12]

In a software product, readability means the ability to read documentation and source code [10]. The documentation serves as the means of communication among the stakeholders. But the research shows that agile teams focus on working software as compared to the documentation while communicating with the clients [10]. Collection of computer instructions that are written in high-level programming language is called source code. Source code is the significant part of software readability in terms of re-usability, cost, maintenance, and robustness. Software industry is facing problems to minimize the software development cost, which is affected by many factors. Researchers are trying to identify those factors and ways to eliminate or at least reduce their impact to reduce the overall cost. According to Collar et al. [11] improved readability saves developer's time while reading the code that eventually helps in bringing down the overall development cost. Readability is important not only during development time to improve software quality [1] but also during maintenance because reading the code is the first stage of maintenance [3]. Research also shows that the maintainability of a software is measured by the readability and understand-ability of code. [12].

If for a given project, project manager foresees that a large number of programmers will be required, programmers are geographically distributed, programmers will be changing over the period of time, new programmers will be hired, or customers will change the requirements then code readability becomes a major concern. Generally code readability is calculated using proportion between number of lines and the

comments that are written for the programmer. The project manager should select a programming language, which is not only suitable for project's functional requirements but should also offer required level of readability. This selection is vital because the correct selection will positively affect the quality of the software.

In this research we have conducted a comparative study on readability of high-level programming languages. We have chosen Java, C++, and C# for this purpose. According to the TIOBE programming community index [15], Java, C++ and C# are among the top five high-level programming languages. These languages are maximally used, so we have computed the readability value of these languages. For this first we have devised a comprehensive framework and used it for the analysis. The analysis is three-fold, we have not only used general text readability indexes, code readability indexes, but also have included the expert opinions. The end results clearly shows that Java has been the best as far as readability is concerned among all.

Rest of the paper is organized as follows. Section II presents brief description on literature review of existing text readability assessment techniques. Section III covers all the proposed techniques for code readability analysis. In Section IV, we present our novel framework to perform comparative analysis among programming languages. Section V presents experiment details and results. We analyze results using statistical techniques in Section VI. Finally we conclude the discussion in Section VII and future directions in Section VIII.

II. Literature Review

In this section, we present literature review of readability metrics to assess the natural languages. Readability tests not only determine readability but also predict the reading ease. Most of the tests are language neutral but some of them are used for certain languages. We have used four natural language metrics for code readability assessment on the basis of their popularity and they are described in this section along with some others.

A. Coleman–Liau Index

Colman–Liau is a readability index similar to automated readability index (ARI) [16] but different from other indexes used to estimate the readability of text. This index is developed by Pahal et al. [3]. This index considers letters per word rather than text as a whole. It was used to calculate readability mechanically from samples of hard copy text. It does not require characters from words and it only calculates the length in characters. The formula of Coleman-Liau index is given below:

$$CLI = 0.0588L - 0.296S - 15.8$$

In the above mentioned equation "L" is average number of letters, whereas, "S" is average number of sentences.

B. SMOG

SMOG stands for "Simple Measure of Gobbledygook". McLaughlin [14] created this index in 1969 in article, SMOG Grading. It estimates the time (years) to read the text required by any person. As compared to other readability metrics, SMOG is better and provide more accurate results. SMOG metric is calculated with the following formula:

$$SMOG = 3 + Squarerootof PolysyllableCount$$

C. Flesch-Kincaid Readability Index

The Flesch-Kincaid [17] index is improved version of Flesch Reading Ease Readability Formula [3]. It checks the reading ease of the give text. If the value is high, it means the text readability is high. But if the value is low then it means text is difficult to read. The grade level is calculated with the following formula:

$$FKRI = 206.835 - 1.015\left(\frac{Totalwords}{TotalSentences}\right) - 84.6\left(\frac{TotalSyllables}{TotalWords}\right)$$

Shorter sentences and words give best results. The score between 60 and 69 is considered average readability while score between 0 and 29 is considered confusing for the reader. The complete list of values and their interpretations is provided in Table I.

TABLE I. Value Ranges and Description [17]

Score	Grade Level
90-100	Very Easy
80-89	Easy
70-79	Fairly Easy
60-69	Standard
50-59	Fairly Difficult
30-49	Difficult
0-29	Very Confusing

D. The Gunning's Fog Index

Gunning [18] propose this index and it is also known as FOG index in short. It can be calculated by using the following formula:

$$FOG = 0.4(ASL + PHW)$$

The average sentence length is added to the percentage of hard word (PHW). And average sentence length (ASL) is calculated by ratio of words count to the total number of sentences. Ideal score for FOG readability is 7 or 8 and if score goes higher than 12, it is considered as hard to read text.

E. The Automated Readability Index (ARI)

Senter [19] design automated readability index (ARI) test to access the understandability of text. Word difficulty and sentences are used in ARI. ARI calculate the readability value and output will be compared with grade level. Here is the formula of ARI:

$$ARI = 4.71\left(\frac{Characters}{Words}\right) + 0.5\left(\frac{Words}{Sentences}\right) - 21.43$$

Characters are the number of letters and numbers. Words are the number of words and spaces and sentences are the number of sentences.

III. Code Readability Indexes

The most important parameter of maintainable software is readability, because changes in the system are made through source code [3]. Less readable source code is harder to maintain than a code that is readable. Most of the time managers reject the code due to lack of code readability. In this section we present some code readability index that we find in the literature.

A. Deepa and Dua (2015)

Deepa and Dua [4] explain that readability depends upon simple sequences and unnecessary loops complicate the program. In this paper code readability is calculated on the basis of software developer judgment. Authors use two copies of the same program for their study. First copy of the program is less readable as proper indentation was not applied whereas the second copy was well formatted using a beautifier tool. Authors also propose a new metric for readability assessment. They perform experiments using novel readability metric and find out that the program written and formatted properly with the help of beautifier has more readability as compared to the other one. The metric that authors use have some parameters including; lines of code, line length, and number of comment lines, number of blank lines, number of lines after semicolon, number of spaces after directive statement and number of method.

B. Tashtoush (2013)

Tashtoush [5] develops an approach called "impact of programming features on code readability" (IPFCR). In this approach author studies the impact of various features and their effect on code readability. For evaluation he uses feature code readability tool (CRT). Author conducts the survey on a random number of expert programmers to access the level of impact. 25 readability features are proposed for survey; meaningful name, comments, spacing, indents, short scope, line length distribution, identifier name length, arithmetic formula, identifier frequency, if-else, nested if, switch, for loop, do while loop and nested loop [5]. Programmers evaluated features into positive and negative factors based on their understandability. The results are evaluated using SPSS statistical tool. ANOVA test is used to remove the biased from data. The top three features that come from survey were meaningful names, consistency and comments. And the lowest impact features were nested loops, arithmetic formula and recursive function. Some of them have neutral impact on readability.

C. Sivaprakasam and Sangeetha (2012)

Sivaprakasam and Sangeetha [7] have conducted a study that shows that readability has a global effect on software budget. In this paper authors define the relationship between software quality and source code readability. Mostly software metrics are used to measure the complexity of software. Authors have developed an automated readability tool, which is 80% more effective than human judgment. Authors have performed extensive experiments to evaluate the readability of code and for this they selected code snippets from the developed projects. The size of snippets is important because too small snippets may reflect incorrect or misleading scores. The scores authors have used range from 1 to 5 where 5 means more readable and 1 means least readable. Authors have ensured that all the snippets have some features including line length, number of character, identifier length indentation, loops and many other features. For a large number of experiments this technique is useful for conducting readability index.

D. Relf (2004)

Relf [8] examines in this paper that identifier naming standards that improve the code readability are acceptable by software professionals. Author claims that naming standards affect source code readability and that greatly impact code maintainability. To examine the impact of naming standards author collects 21 naming standards from research. These include multiple underscore characters, outside underscore character, numeric digits, naming convention anomaly, identifier encoding, short identifier name, long identifier name, number of words, class qualification, abstract words, constant qualification, numeric identifier name and some others. Author analyzes some codes written in ADA and Java programming languages and rates these programs on the basis of naming standards used from 1 to 5 (1 is strong acceptance and 5 is strong rejection). This study also states that expert programmers accept the naming standards more than the beginners.

E. DeYoung, Kampen, Topolski (1992)

An automated readability measure will be useful for developers during coding as it will continuously assessing their code and assisting them to improve. DeYoung et al. [9] examine the machine computable and human-judged program features. They identify that length of identifiers and are very useful in predicting code readability. Using analyzer generated quality of comments, logicality of control flow and meaningfulness of identifier names are studied to find out whether these predictors are worthy for readability estimation [9]. The proposed predictors increase the proportion of readability of judgments from 41% to 72%. Authors also claim that when logicality of control flow is added as a predictor, it produces better results as compared to human judgment but somehow these predictors are expensive to obtain.

F. Buse and Westley (2008)

Buse and Westley [2] perform a detailed empirical study to calculate readability of code. For this they have chosen 100 snippets and around 120 annotators that grade these snippets. The biggest issue in this research is that authors have used 19 parameters including line length, identifiers, identifier length,

indentation, keywords, numbers, comments, periods, commas, spaces, parenthesis, arithmetic operators, comparison operators, assignment, branches, loops, blank lines, occurrences of any character and occurrences of any single identifier, which are difficult to calculate. From these parameters authors have constructed automated readability measurement and proved that it will be 80% more effective than human judgment. Furthermore, he discusses that how readability has potential for improving programming language design with respect to software quality. Authors also suggest to decrease the parameters for readability analysis and sets this as future work for their research.

G. Relf (2005)

Relf [6] describes a practical study to show whether coder increases the readability of his programs if he gets support from source code editor that provides vibrant responses on his identifier naming practices. Software coder should adopt a standard for software interface to gain benefits. This paper is useful for both student and professional software coder for maintaining the code and significant for the improvement of code readability. Author uses only one parameter for code readability that is identifier naming practices.

H. Daryl, Hindle, and Devanbu (2011)

Daryl et al. [13] propose to use entropy for predictive modeling approach. Authors study that whether size of the code impact the readability of the code or not. They have used six parameters including mathematical equations, average number of comments, and maximum indention, maximum word, maximum line length and maximum occurrence character in the code snippets. Author also used Halstead's metrics to find the size of code on the mean readability. For mean readability total number of operators and operands are combined and formulate the Halstead's metrics. For measuring the Entropy total number of tokens and unique token is counted. Also Entropy model improves the performance in term of prediction and readability but byte entropy does not improve the prediction.

IV. Framework for Comparative Analysis

In this section we present the framework we have proposed for performing the comparative analysis among the selected three programming languages (Java, C#, and C++).

The main objective of our work is to compare the readability of three of the top five most popular programming languages. In proposed framework we compare the human judgment with readability index: ARI (Automated Readability Index), SMOG, FOG, and FKG. The framework is presented in Fig. 2.

To perform comparison first we have to find the programing parameters that can affect the readability of source code. For this we select the constructs from the research work of Buse and Westley [2]. Second step is to compute the effect of these constructs on the readability of Java, C# and C++ languages. To calculate the effect, we have selected code snippets of Java, C# and C++ languages. After snippets selection, online survey is conducted, in which expert opinion is obtained and results are obtained for every selected programming construct. Selected snippets are measured with different readability indexes.

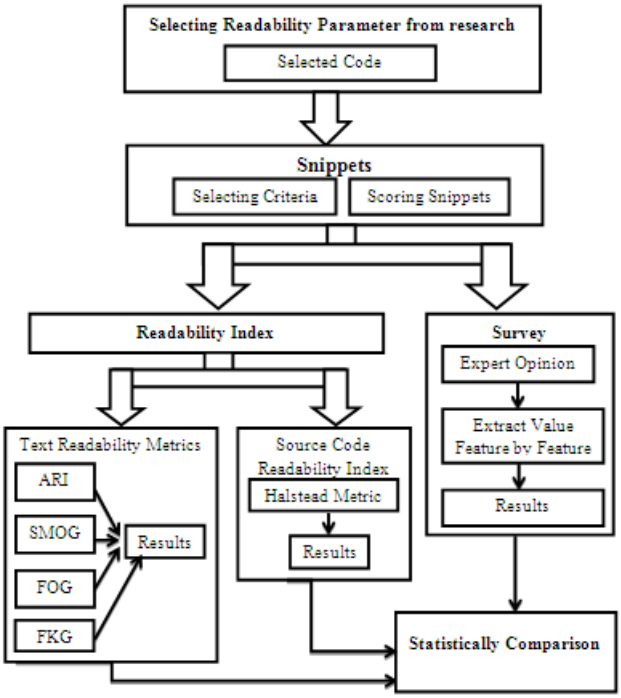

Fig. 2. The Proposed Framework

Text readability indexes include ARI (Automated Readability index), SMOG Fog, and Flesch Kincaid Grade level, while the code readability includes Halstead's complexity. The effect of readability by each construct is then calculated with these readability indexes.

A. Selection of Readability Parameters

Readability of code is normally linked with comments and naming standards and also called the important factor that impact readability but there are some other aspects the affect the readability. Number of parameters are used in coding that make the code possible and easy to build. There are number of parameters that we find in the literature [2] out of those we have chosen 14 to conduct this comparative study. Table II presents this list of selected parameters.

TABLE II. Parameters Used for Code Readability Comparison

Sr. No.	Parameter	Notation
1	Parenthesis	PAR
2	Indent	IND
3	Spaces	SPA
4	Class Distribution	CD
5	Arithmetic Equations	AE
6	For Loop	FL
7	Nested Loop	NL
8	Do-While Loop	DWL
9	IF-Else	IE
10	Switch	SWI
11	Blanks Lines	BL
12	Line Length (characters)	LL
13	Arrays	ARR
14	Comparison Operators	CO

B. Selection of Code Snippets

A small section of source code or text is called code snippet. Normally they are defined in effective unit of large programs model. In the readability model, first we select the code snippets of Java, C++ and C#. As we know snippets are the small portion of source code, thus we select a small human readable codes that are neither too short nor too long. Each snippet contains a parameter to check their readability impact of that we have discuss earlier. Snippet does not include comments, header functions, and blank lines because they are not meaningful. Secondly code snippets should be logically clear to respondent so, he/she can easily read them. Finally, these snippets are given to the annotators (explain functionality of codes). The ratings for the code snippets are assigned from 1 to 5 where 4 and 5 mean that code is more readable and rank 1 and 2 mean that code is less readable and rank 3 is for average. To perform the online survey, we have used Google Forms and Excel sheets. Respondent can choose one rank (1 to 5) against each language.

V. Comparative Analysis

In this section we perform detailed comparative analysis using the proposed framework presented in the previous section. First we present the details and results of the Survey that we have conducted with the help of programmers of different skill and level.

A. Survey

As mention earlier a set of snippets are selected for human judgment for estimating readability. In Table II, we have presented 14 language constructs that we have chosen to compare the readability of selected programming languages. For every construct we have prepared 6 to 7 pieces of codes for all three languages. Then they are presented to 100 programmers including IT professionals, Programmers, and Computer Science Students. According to their judgment they have ranked snippets. Participant have to rank each snippet from 1 to 5 where 1 is less readable and 5 is more readable.

Each snippet contains a parameter that affects the code readability. And against each parameter participant rank the code readability. Each participant was given the same questionnaires using Google Forms. To improve the visibility results of the survey are presented in bar-chart form in Fig. 3.

We can notice that as per the experts, code snippets written in Java are more readable for almost every selected programming construct. The results also show that C# performs better for two language constructs including DO-While and For Loop is more readable.

B. Code Readability Index

We have computed code readability index for all the selected code snippets against all the selected language constructs using Halstead's metric. Halstead's metric proposed by Maurice Howard Halstead is used to measure the complexity of a program. It depends upon the actual implementation of program which is computed from some operators and operands. It can also computes words size, errors and testing time for C++, C# and Java codes. The

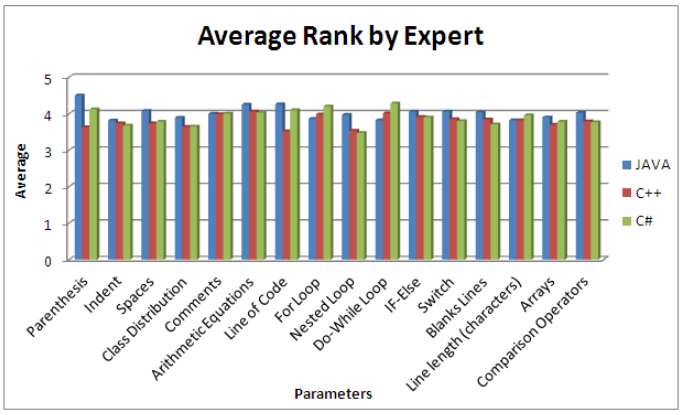

Fig. 3. Results of the Code Readability Comparison

parameters used by Halstead's metric are mentioned below:

n1: Number of unique operators
n2: Number of unique operands
N1: Total number of operators
N2: Total number of operands

The following list presents the various parameters and their expressions that are offered by Halstead's metric to compute different aspects of programs written in programming languages:

$$Vocabulary : n = n1 + n2$$

$$Size : N = N1 + N2$$

$$Volume : V = length * log2Vocabulary$$

$$Difficulty : D = \left(\frac{n1}{2}\right) * \left(\frac{N1}{n2}\right)$$

$$Efforts : E = Difficulty * Volume$$

$$ProgramLevel : L = V^*/V$$

$$TestingTime : T = \frac{Efforts}{S}, where S = 18 seconds$$

In order to apply the above mentioned metrics on the code snippets, we have developed a source code readability tool (SCRT). SCRT calculates the vocabulary of code, size, volume efforts, errors, testing time and difficulty of the code for all the programs. After calculating these different metrics we have presented the results in upcoming tables including Table III, Table IV, and Table V for Java, C#, and C++, respectively.

After obtaining the results of Halstead's matrices, we have plotted one of the aspects, which is "difficulty" with the help of a line chart to compare the results of all three languages. The results in Fig. 4 clearly show that C++ programs are more difficult to read and understand as compared to the programs written in Java or C#. Mostly Java seems to be less difficult among all the languages in nearly all the language constructs except for comparison operator and arithmetic expressions.

TABLE III. HALSTEAD'S METRIC RESULTS FOR JAVA LANGUAGE

Params	Vocab	Size	Volume	Difficulty	PRO Level	Quality
PAR	09	21	066.56	03.66	0.46	1.61
IND	19	78	331.33	08.05	0.254	1.037
SPA	8	16	048.00	01.50	0.57	1.115
CD	08	13	039.00	0.125	0.80	NA
CO	09	51	161.66	11.11	0.18	1.12
AE	14	85	323.62	11.25	0.16	1.064
FL	11	27	093.40	03.18	0.42	0.81
NL	13	41	151.71	04.03	0.330	0.60
DWL	13	25	092.51	02.42	0.54	1.76
IE	15	31	012.11	01.83	0.51	0.83
SWI	17	48	196.19	01.41	0.38	1.68
LL	22	43	191.75	03.86	0.51	NA
ARR	14	30	114.22	02.35	0.493	2.07
SCO	13	36	133.21	03.23	0.45	2.1

TABLE IV. HALSTEAD'S METRIC RESULTS FOR C# LANGUAGE

Params	Vocab	Size	Volume	Difficulty	PRO Level	Quality
PAR	08	19	57.0	3.75	0.45	1.41
IND	13	66	244.22	8.46	0.21	0.34
SPA	14	20	76.14	0.53	0.78	1.12
CD	10	17	56.47	0.1	0.73	NA
CO	12	49	175.66	5.83	0.25	1.32
AE	15	73	285.20	7.8	0.20	1.04
FL	11	27	93.41	3.63	0.41	0.85
NL	13	40	148.01	3.76	0.33	0.62
DWL	13	24	88.81	2.15	0.56	1.83
IE	15	26	101.57	1.16	0.61	2.31
SWI	15	42	164.08	1.06	0.38	0.59
LL	20	42	181.52	5.4	0.46	0.97
ARR	14	29	110.41	2.14	0.51	1.89
SCO	11	29	100.32	3.00	0.32	0.52

C. Text Readability Index

Now we calculate the readability of the code with various different text readability indexes. There are many metrics available for the same and among those we have chosen some most popular metrics listed below. After that we have applied them on all the selected code snippets of all three programming languages. The results are presented in Table VI, Table VII, and Table VIII. Before presenting the results, below are the metrics that we have applied to calculate text readability indexes:

- ARI
- FOG
- FKG Level
- SMOG

TABLE V. HALSTEAD'S METRIC RESULTS FOR C++ LANGUAGE

Params	Vocab	Size	Volume	Difficulty	PRO Level	Quality
PAR	11	23	79.56	3.63	0.69	1.54
IND	20	51	220.41	2.97	0.39	0.35
SPA	14	32	121.83	3.21	0.47	0.47
CD	12	50	226.17	5.02	0.47	1.244
COM	09	51	161.66	12.26	0.228	0.34
AE	20	82	354.39	10.5	0.173	0.25
FL	14	36	137.06	4.57	0.40	1.01
NL	17	52	212.54	7.05	0.26	0.31
DWL	15	36	140.64	5.66	0.41	0.549
IE	19	40	169.91	3.36	0.48	0.55
SWI	19	50	212.39	3.13	0.39	0.42
LL	31	101	500.37	8.70	0.26	1.57
ARR	16	41	164	5.06	0.40	0.43
SCO	15	36	140.64	3.96	0.33	0.45

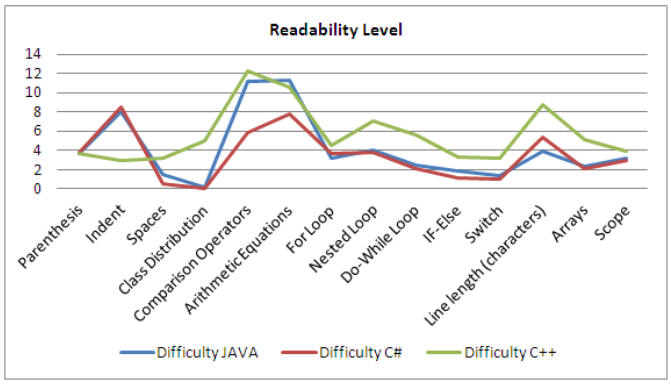

Fig. 4. Comparison of Readability Difficulty among Java, C#, and C++

TABLE VI. TEXT READABILITY INDEX FOR JAVA LANGUAGE

Parameters	ARI	FOG	FKG	SMOG	Average
PAR	11.30	08.41	02.28	08.09	07.52
IND	20.14	11.82	01.73	11.18	11.21
SPA	08.91	12.67	07.49	07.79	09.21
CD	18.90	12.62	11.24	09.24	13.00
AE	01.46	04.68	-00.40	09.00	03.68
FL	15.73	07.60	03.14	13.00	09.86
NL	01.57	05.16	01.50	07.89	04.03
DWL	01.63	06.11	02.61	08.56	04.72
IE	04.11	06.54	02.11	08.09	05.21
SWI	03.39	06.40	01.40	08.83	05.05
SCO	08.39	05.83	01.62	10.14	06.49
LL	06.80	06.15	00.99	08.65	05.64
ARR	18.54	21.85	09.29	11.06	15.18
COM	05.86	05.90	01.66	08.09	05.37

TABLE VII. TEXT READABILITY INDEX FOR C# LANGUAGE

Parameters	ARI	FOG	FKG	SMOG	Average
PAR	09.90	03.86	02.75	08.00	06.12
IND	17.38	12.91	02.10	11.30	10.92
SPA	11.88	12.00	07.47	08.00	09.83
CD	21.04	13.30	12.21	09.55	14.03
AE	10.23	06.35	03.31	09.85	07.44
FL	15.05	08.70	05.76	13.63	10.78
NL	04.91	07.02	03.98	08.29	06.05
DWL	05.42	08.65	05.56	08.91	07.13
IE	07.05	08.07	04.4	08.47	06.99
SWI	07.58	07.14	03.71	09.48	06.97
SCO	08.65	07.16	03.87	10.81	07.62
LL	05.97	08.04	03.56	09.08	06.66
ARR	19.05	20.23	09.56	11.24	15.02
COM	06.80	06.15	0.999	08.65	05.64

The obtained results after computing text readability indexes, are plotted with the help of bar-chart. Fig. 5 shows the results for all three programming languages against all programming constructs. The results again show that Java language codes are more readable as compare to C# and C++. But in some constructs such as comparison operators, arithmetic equations and scope C# is more readable as per text readability index.

VI. STATISTICAL ANALYSIS

In this section we present statistical analysis that we have performed on the results obtained after experiments. For this we have chosen *T-test* for the same. The *T-test* is used to compare the two sample means. Where one sample means

TABLE VIII. TEXT READABILITY INDEX FOR C++ LANGUAGE

Parameters	ARI	FOG	FKG	SMOG	Average
PAR	8.38	04.66	01.69	07.35	05.52
IND	09.3	11.91	08.29	09.48	09.74
SPA	03.08	06.54	01.51	07.79	04.73
CD	15.09	13.02	09.90	11.00	12.25
AE	02.94	06.22	02.05	08.09	04.82
FL	19.72	18.83	13.98	11.94	16.12
NL	02.69	08.65	03.99	07.69	05.75
DWL	05.38	10.80	06.43	08.47	07.77
IE	03.19	08.14	04.16	08.09	05.89
SWI	04.38	08.82	05.30	08.65	06.79
SCO	09.3	11.20	08.20	09.4	09.53
LL	-00.39	07.20	03.36	07.35	04.38
ARR	09.88	10.31	05.06	12.27	09.38
COM	04.59	08.66	03.6	08.00	06.22

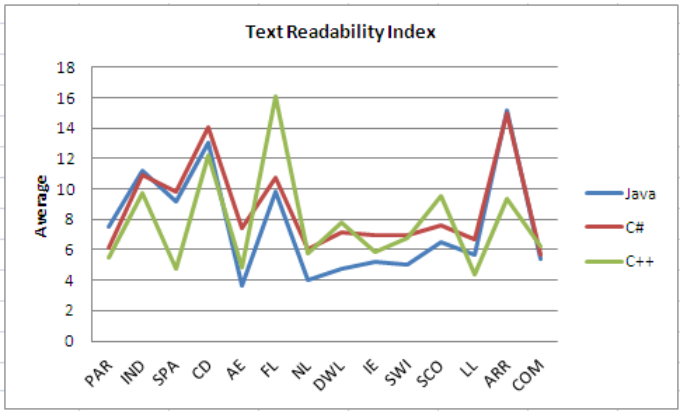

Fig. 5. Comparison of Text Readability Index among Java, C#, and C++

can be paired with other sample mean observation. In paired T-Test each entity is measured twice, result will be given in pairs. Table IX presents Halstead's arithmetic mean, standard deviation and standard mean error are given for all three languages (Java, C#, C++). Table X presents paired correlation between Java and Halstead index, C# and Halstead index and C++ and Halstead index of C++. Where correlation r ¿ 0.50 shows the strong relationship and r ¡ 0.50 shows the weak positive relationship.

TABLE IX. MEAN, STD. DEVIATION, AND STD. ERROR MEAN

	Mean	N	Std. Deviation	Std. Error
Java	4.0089	99	0.27024	0.02716
Halstead Java	3.3005	99	0.81316	0.08173
C#	3.8054	99	0.51242	0.05150
Halstead C#	3.7690	99	0.82096	0.08251
C++	3.8856	99	0.32749	0.03291
Halstead C++	3.8871	99	0.41251	.04146

TABLE X. CORRELATION BETWEEN MEAN, STD. DEVIATION, AND STD. ERROR MEAN

	N	Correlation	Sig.
Java & Metric Readability-Java	99	0.730	0.191
C# & Metric Readability-C#	99	0.529	0.001
C++ & Metric Readability-C++	99	0.610	0.553

The statistical results show that Java programming language has been found being more readable as compared to other programming languages.

VII. CONCLUSION

Code readability influences maintenance of a software at great deal. Due to its salient importance, we have conducted a comparative study to estimate readability of the codes produced by Java, C#, and C++ programming languages. We identify important language constructs that affect the code readability and then propose a novel framework to compare the codes using three different dimensions. First we have performed an expert survey involving programmers and experts to judge the readability of codes. Then we have applied code readability and text readability indexes to again calculate readability of the same programs. We have computed these indexes using a source code readability tool (SCRT). The experiment results show that Java language produces more readable code as compared to C# and C++. Only for a few language constructs like comparison operators and arithmetic operator. We have also statistically analyzed the results using SPSS tool to verify the effectiveness of experiments. This analysis also verifies that Java language code is more readable than C# and C++.

VIII. FUTURE WORK

In future we are planning to extend our analysis on other famous languages also including Python and VB.NET. Other than these, we are also looking to conduct an analysis on programming languages that are used specifically for mobile application development.

REFERENCES

[1] S. Fakhoury, D. Roy, A. Hassan and V. Arnaoudova, "Improving Source Code Readability: Theory and Practice," 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), pp. 2-12, Montreal, QC, Canada, 2019.

[2] Buse, R.PL., Westley R.W. "A metric for software readability." Proceedings of the 2008 international symposium on Software testing and analysis, pp. 121-130, Seattle, WA, USA, July 20-24, 2008.

[3] Pahal, Ankit, and Rajender S. Chillar. "Code Readability: A Review of Metrics for Software Quality."International Journal of Computer Trends and Technology (IJCTT) – Volume 46 Number 1- April 2017

[4] Deepa D., Dua A. K. "Evaluation of Quality of Source Code By Code Readability. International Journal of Advanced Research in Computer Science and Software Engineering, 2015.

[5] Tashtoush, Y. "Impact of programming features on code readability." International Journal of Software Engineering and its Applications. 7(6):441-458. November 2013.

[6] Relf, P.A., "Tool assisted identifier naming for improved software readability: an empirical study", In International Symposium on Empirical Software Engineering, Noosa Heads, Qld., Australia, November 17-18, 2005.

[7] Sivaprakasam., P, Sangeetha., V. "Improving software qualitythrough the development ofcode readability" International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 6, August 2012.

[8] Relf, P.A., "Achieving software quality through source code readability", Quality Contract Manufacturing LLC., 2004.

[9] DeYoung, G.E., Kampen, G.R. and Topolski, J.M. "Analyzer-generated and human-judged predictors of computer program readability." Proceedings of the 1982 conference on Human factors in computing systems. pp 223-228, Gaithersburg, Maryland, USA, March 15-17, 1982.

[10] Sivaprakasam, P., and V. Sangeetha. "An accurate model of software code readability." International Journal of Engineering Research and Technology.ESRSA Publications (2012).

[11] Collar Jr, Emilio, and Ricardo Valerdi. "Role of software readability on software development cost." 2006.

[12] Aggarwal, Krishan K., Yogesh Singh, and Jitender Kumar Chhabra. "An integrated measure of software maintainability." Reliability and maintainability symposium, 2002.Proceedings.Annual.IEEE, 2002.

[13] Daryl, P., Hindle, A., and Devanbu, P., "A simpler model of software readability", In Proceedings of the 8th working conference on mining software repositories, pp. 73-82, Waikiki, Honolulu, HI, USA, May 21-22, 2011.

[14] McLaughlin, G. Harry. "SMOG grading-a new readability formula." Journal of reading 12.8 (1969): 639-646.

[15] TIOBE. https://www.tiobe.com/tiobe-index/. (accessed on October 17, 2019).

[16] Automated Readability Index (ARI). https://en.wikipedia.org/wiki/Automated_readability_index. (accessed on October 18, 2019).

[17] Kincaid, J.P., Fishburne, R.P., Rogers, R.L., & Chissom, B.S., "Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formula) for Navy enlisted personnel." Research Branch Report 8–75. Chief of Naval Technical Training: Naval Air Station Memphis. 1975.

[18] Gunning, Robert., "The Technique of Clear Writing". McGraw-Hill. pp. 36–37. 1952.

[19] Senter, R.J.; Smith, E.A. (November 1967). "Automated Readability Index". Wright-Patterson Air Force Base: iii. AMRL-TR-6620. Retrieved March 18, 2012.