

A New Metric for Code Readability

Rajendar Namani¹, Kumar J²

¹Department of CSE, DRK College of Engineering and Technology, Ranga Reddy, Andhra Pradesh, India

²Department of CSE, DRK Institute of Science and Technology, Ranga Reddy, Andhra Pradesh, India

Abstract: Code readability is very important in software development process. It has relationship with software quality. In this paper we define a new readability metric that can be used to measure the readability of given source code. From senior software engineers we collected some rules and then deduced them into a metric which can be used to measure the readability of code. Moreover we implemented some of the existing readability metrics such as Automated Readability Index (ARI), Gunning's Fox Index (FOG) and SMOG programmatically. The results of these metrics are compared with the each other. We have developed a prototype application that takes source code as input and applies metric. The metric generates readability statistics. The input code is given to 50 software engineers and asked them to provide the percentage of readability of code. The percentage given by them is compared with the proposed application generated statistics. The results revealed that both are closely matching and the proposed metric can be used in the real world applications.

Index Terms: Code readability, readability metrics, software development life cycle, and software quality.

I. Introduction

Readability of software has something to do with software development and also its quality. As a matter of fact, readability is the judgment of humans with respect to the ease of reading of given source code. Readability can promote software maintainability and the overall software quality can be ensured. As it is said in [1] maintenance of a software project takes 70% of its while life cycle cost. Aggarwal et al. [2] claim that to the maintainability of any project code and documentation readability is critical. The most time consuming activities belong maintenance phase of SDLC [3], [4], and [5]. Readability of source code is given more importance in such a way that Marcotty and Elshoff when to adding a new phase in SDLC known as core readability improvement phase [6]. According to Knight and Myers one phase of software inspection should be verifying source code for readability [7] which is meant for ensuring the reusability, portability and maintainability of source code. Haneef in [8] went to the extent of adding a documentation group to his development team. He also said that well established guidelines for code readability can help reviewers a lot. As described in [1] and [9], programmers have some sort of intuition with respect to the concept and features of program and thus readability is essential and comments in the programs promote code readability. According to Dijkstra, the readability of a program depends on many factors such as simplicity of control sequences, comments, top down approach and so on [10].

In this paper we define a new metric based on a set of rules collected from software engineers. The rules are kept part of formula of the new metric. This does mean that the new metric is developed using individual rules computations and then substituted into the main formula. After implementing the new metric, we evaluated the metric by inviting 50 software engineers asking them to provide their readability percentage (from human perspective). Then the results are averaged and compared with the results of the prototype application for the same source files. The comparison results are encouraging and the new metric can be used in real world software development communities.

II. Existing Readability Metrics

There are many existing readability metrics such as Automated Readability Index [11], SMOG [12] and Gunning Fog [13]. These three code readability metrics work for text files only. However, the proposed metric is described in the next section. The following sub sections focus on ARI, SMOG and Gunning Fog metrics.

II.1 The Automated Readability Index (ARI)

In this metric word difficulty and sentence difficulty ratios are used. The word difficulty refers to number of letters per word and sentence difficulty refers to the number of words present in a sentence. The first step is to establish the factors used and then relate them to other indices. The sentence structure related to factor is same to that of presently used indices. In the process the verification of relationship between the factors is self – evident. There are two factors associated with most readability factors. The first factor is related to sentence

structure which is made up of number of words. The second factor is related to word structure which is made up of letters. Word list has advantages, and it is slow and relatively inaccurate when it is applied to reading material of adults. The syllable count is not reliable. The equation to compute readability with ARI is

$$4.71 \frac{\text{characters}}{\text{Words}} + 0.5 \frac{\text{words}}{\text{sentences}} - 21.43 \quad (1)$$

SMOG

In 1969 G Harry McLaughlin created the SMOG readability metric. It estimates the the number of years of education one needs to comprehend a piece of written matter. This is an improvement over other readability formulae. SMOG stands for Simple Measure of Gobbledygook. Some believe that it stands for Robert Gunning’s FOG. SMOG formula is as shown below.

$$\text{SMOG grade} = 3 + \text{Square Root of Polysyllable Count} \quad (2)$$

The Gunning’s Fog Index

This readability metric is known as FOG Index developed by Robert Gunning. According to him the readability formula is

$$\text{Grade Level} = 0.4 (\text{ASL} + \text{PHW}) \quad (3)$$

where

(ASL = Average Sentence Length (i.e., number of words divided by the number of sentences)

PHW = Percentage of Hard Words).

III. Proposed Readability Metric

Readability metric helps in understanding the percentage of readability of given source code. This paper aims at creating new code readability metric. The methodology used to define the new metric is described here. First of all some rules that can be used to measure readability are obtained from senior software engineers. It does mean that we asked some senior software engineers to provide criteria for readability metric and collected their responses. After making an initial list, the list is validated and finally seven rules are considered to be part of new metric. The seven rules and their representative notations are presented in table 1. This methodology is inspired by [14].

Table 1 – Rules involved in new readability metric

RULE	NOTATION
Lines of Code	LOC
Line Length	LL
Presence of comment lines in the program	NOCL
No. of Blank Lines	NOBL
Breaking the line after semicolon	NLAS
Blank space after directive statements	BSAD
No. of Methods	NOM

As can be seen in table 1, the notations for all seven rules are presented. These notations are used to deduce an equation for the new metric. The new code readability metric is represented by

$$\text{CR} = \text{LOC} + \text{LL} + \text{NOCL} + \text{NOBL} + \text{NLAS} + \text{BSAD} + \text{NOM} \quad (4)$$

where CR stands for Code Readability. Other notations are provided in table 1. As part of the methodology to define new metric and implement it, a prototype web application is built using Visual Studio which takes a source file as input and extracts values for LOC, LL, NOCL, NOBL, NLAS, BSAD and NOM. Afterwards those values are substituted in the equation 4 which is used to compute the readability of given source code. The result will be in percentage of readability. The more in this percentage, the more the source code readability is.

IV. Implementation And Results

The environment used to implement the proposed metric includes Visual Studio 2010, and SQL Server 2008. User interface is designed using WebForms while functionality is done using C# programming language which is part of Microsoft.NET framework. The Visual Studio is the IDE (Integrated Development Environment) used for rapid application development. The application takes C# source files as input and computes readability using the new readability metric proposed in this paper. The results of proposed readability metric for given C# source files are shown in fig. 1.



Fig. 1 – Results of new readability metric

As can be seen in fig. 1, the results of readability metric are shown. Different C# source code files got different results. Later the application supports applying the existing metrics with any given text file. When text file is given as input the readability metrics such as SMOG, Gunning Fog and Automated Readability Index. When a text file is given as input, the results are shown for all three metrics as shown in fig. 2.

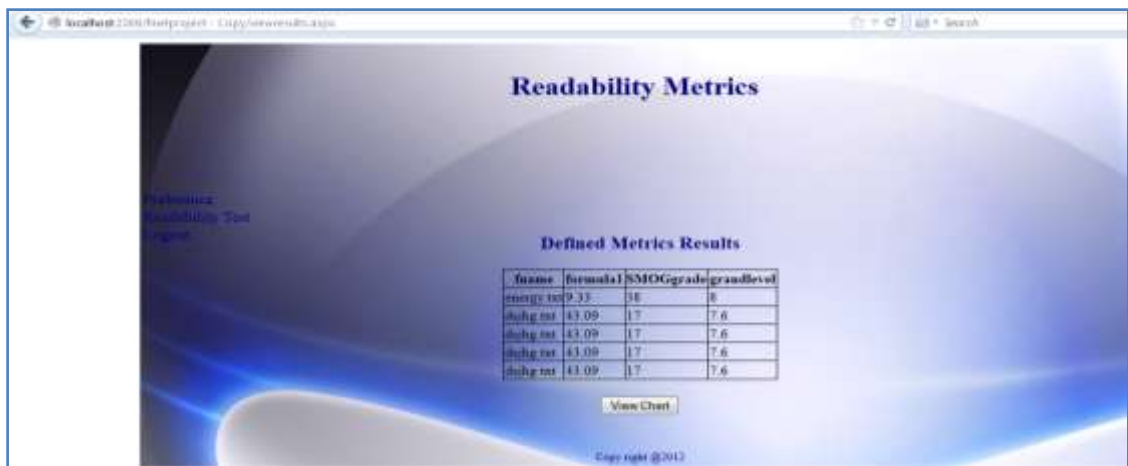


Fig. 2 – Results of Text Readability Metrics

As seen in fig. 2, the existing readability metrics are applied on various text input files. The results are shown in tabular format. The View Chart button initiates an action that presents the same results in the form of the graphs as shown in fig. 3.



Fig. 3 – Graphs showing results of text readability metrics

The three readability metrics suitable for text documents are applied to a set of input files. The results are shown in fig. 3. The results reveal that the three metrics use different parameters for measuring and the same is reflected in the results.

V. Evaluation Of Results

As discussed earlier, the results of proposed readability metric and also readability assessed by 50 human experts are presented in table 1. For each input file readability percentage is sought from 50 software engineers. Their percentage readability for each source file is averaged and tabulated in the last column of table 1. Column 2 shows the list of input files used in the experiments. Column 3 shows the results readability metric proposed and implemented in this paper.

Table 1 – Results of Readability Metric

sno	input file	code readability (programmatically)	code readability (human)
1	a.cs	40%	42%
2	b.js	53%	50%
3	c.cs	60%	65%
4	d.cs	56%	57%
5	aa.cs	68%	70%
6	bb.cs	65%	62%
7	cc.cs	35%	30%
8	Abc.cs	80%	78%
9	Xyz.js	68%	70%
10	A123.c	59%	58%

As can be seen in table 1, the results of new readability metric implemented in this paper and results given by human participants are presented. The same details are graphically visualized in fig.

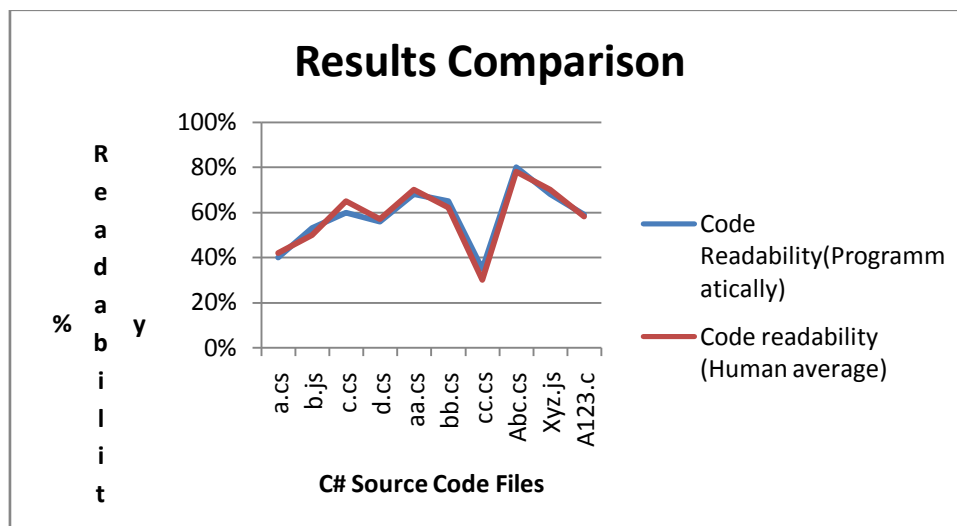


Fig. 4 – Comparison of Results of Readability Metric

As shown in the graph in fig. 4, it is evident that the proposed readability metric gives result which is very closer to the results given by human participants. This reveals that the new metric implemented is useful in finding readability of source code. Software engineers as part of their development life cycle can use the proposed metric to compute readability. Thus readability can be improved and in turn this improves the quality of software product.

VI. Conclusion



This paper defines a new readability metric for measuring readability of source code. The methodology used to achieve this is that from senior software engineers a set of rules are obtained. These rules are then deduced to formula that represents a new metric. A prototype application is built to demonstrate the effectiveness of the new code readability metric. In the prototype application we have given support to apply some of the existing metrics such as SMOG, Gunning Fog, and Automated Readability Index (ARI). The result

of new metric which is applied to programmatically to C# source code is evaluated through the results obtained from humans. The source code files are given to 50 software engineers and asked them to provide readability percentage. The values given by them are averaged and compared with the system generated results. The comparison results revealed that

References

- [1] B. Boehm and V.R. Basili, "Software Defect Reduction Top 10 List," Computer, vol. 34, no. 1, pp. 135-137, Jan. 2001.
- [2] K. Aggarwal, Y. Singh, and J.K. Chhabra, "An Integrated Measure of Software Maintainability," Proc. Reliability and Maintainability Symp., pp. 235-241, Sept. 2002.
- [3] L.E. Deimel, Jr., "The Uses of Program Reading," ACM SIGCSE Bull., vol. 17, no. 2, pp. 5-14, 1985.
- [4] D.R. Raymond, "Reading Source Code," Proc. Conf. Center for Advanced Studies on Collaborative Research, pp. 3-16, 1991.
- [5] S. Rugaber, "The Use of Domain Knowledge in Program Understanding," Ann. Software Eng., vol. 9, nos. 1-4, pp. 143-192, 2000.
- [6] J.L. Elshoff and M. Marcotty, "Improving Computer Program Readability to Aid Modification," Comm. ACM, vol. 25, no. 8, pp. 512-521, 1982.
- [7] J.C. Knight and E.A. Myers, "Phased Inspections and Their Implementation," ACM SIGSOFT Software Eng. Notes, vol. 16, no. 3, pp. 29-35, 1991.
- [8] N.J. Haneef, "Software Documentation and Readability: A Proposed Process Improvement," ACM SIGSOFT Software Eng. Notes, vol. 23, no. 3, pp. 75-77, 1998.
- [9] P.A. Relf, "Tool Assisted Identifier Naming for Improved Software Readability: An Empirical Study," Proc. Int'l Symp. Empirical Software Eng., Nov. 2005.
- [10] E.W. Dijkstra, A Discipline of Programming. Prentice Hall PTR, 1976.
- [11] The Automated Readability Index (ARI): <http://www.readabilityformulas.com/automatedreadability-index.php>
- [12] The smog readability formula: <http://www.readabilityformulas.com/smog-readability-formula.php>
- [13] The Gunning's Fog Index (or FOG) Readability Formula: <http://www.readabilityformulas.com/gunning-fog-readability-formula.php>
- [14] Raymond P.L. Buse and Westley R. Weimer, "Learning a Metric for Code Readability", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. 4, JULY/AUGUST 2010.

AUTHORS

	<p>Rajendar Namani is a student of DRK College of Engineering and Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech degree in Computer Science and Engineering and M.Tech Degree in Computer Science and Engineering. His main research interest includes Software Engineering.</p>
	<p>Kumar J is a student of DRK Institute of Science and Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech degree in Computer Science and Engineering and M.Tech Degree in Computer Science and Engineering. His main research interest includes Software Engineering.</p>