IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 7 Issue 1, January 2020
ISSN (Online) 2348 – 7968 | Impact Factor (2019) – 6.248
www.ijiset.com

Software Complexity Metrics of Functional Languages using Binary Search Algorithm

Maria Adigun[1], Kehinde Sotonwa[2], Johnson Adeyiga[3] and Mojeed Abass[4]

Department of Computer Science and Information Technology, Bells University of Technology
Ota, Ogun State, Nigeria

Abstract

Software metrics are used by the various software organizations for evaluating and assuring code quality, operation and maintenance which measure various types of complexity like size metrics, control flow metrics and data flow metrics: LOC, McCabe and Halstead measures. Software complexities must be continuously calculated, followed and controlled to reduce complexity, maintainability and testing effort. The revival of data science due to the presence of large amount of data has resulted in the need for a good programming language. Hence the need to apply complexity metrics to existing functional languages so as to guide programmers in choosing tools for building data science applications. For application, binary search algorithm is considered; written in three different functional languages: Python, R and Scala. The results are compared and Scala is realized to be the most complex for all the metrics while Python and R are averagely at par.

Keywords: *Functional Languages, Complexity, Software Metrics, Binary Search Algorithm, Data Science.*

1. Introduction

Functional programming is a way of thinking about software construction by creating pure functions. It avoids concepts of shared state, mutable data observed in object oriented programming, it emphasizes on expressions and declarations rather than execution of statements. The languages are specially designed to handle symbolic computation and list processing applications. It is a programming language that is based on mathematical functions which uses conditional expressions and recursion to do perform the calculation [20]. Some of the popular functional programming languages include: Lisp, Python, Erlang, Haskell, Clojure, R, Scala, F#, SQL, Erlang etc., for the scope of this study Python, R and Scala are considered. [1][2][3][4][5][6]

The term software complexity is often applied to the interaction between a program and a programmer working on some programming tasks [7]. Software complexity is defined as "the degree to which a system or component has a design or implementation that is difficult to understand and verify [8] i.e. complexity of a code is directly dependent on the understandability. All the factors that make a program difficult to understand are responsible for its complexity [9]. Results based on real life projects have shown that there is a correlation between the complexity of a system and the number of faults [10][11].

Software metrics determine the degree of maintainability of software products, which is one of the important factors that affect the quality of any kind of software. Also, software metrics provide useful feedback to the designers to impact the decisions that are made during design, coding, architecture, or specification phases. Without such feedback, many decisions are made in an ad hoc manner [8] Software metrics have been found to be useful in reducing software maintenance costs by assigning a numeric value to reflect the ease or difficulty with which a program module may be understood. The most common software metrics used are LOC, McCabe and Halstead methods. [8]

Search algorithm is a combination of two words search and algorithm, according to oxford dictionary 7th edition; search can be defined as to look into, over or through something carefully in order to find somebody or something and algorithm is a logical sequence of steps for solving a problem, often flowchart that can be translated into a computer program. Search algorithm is an algorithm for finding an item with specified properties among a collection of items [12]. Among the common types of search algorithms is binary search algorithm which is an algorithm for locating the position of an element in a sorted list. It inspects the middle element of the sorted list: if equal to the sought value, then the position has been found; otherwise, the upper half or lower half is chosen for further searching. A binary search halves the number of items to check the iteration, so locating the item (or determining its

absence) takes logarithmic time. A binary search is a dichotomy divide and conquer search algorithm because it keeps on dividing the array into two halves, the binary search is very much faster than linear search, but it is not worth list [12] [13].

2. Related Works

Sotonwa *et al.* [14] compared software complexity of Line of Code, cyclomatic complexity metric and Halstead complexity metrics of linear and binary search algorithms using VB, C#, C++ and Java to measure the sample programs using length in lines of the program, LOC with comments, LOC without comments, McCabe method and the program difficulty using Halstead method. It was discovered that the four object-oriented programming languages is good to code linear and binary search algorithms. However, procedural language can also be applied on software metrics.

Cafer, Akman and Sanjay [8] carried out a new approach research and concluded that none of the existing metrics can cover all the features of Python as well as cover all the aspects of complexity. Consequently, a new approach, which is unification of all the attributes, was presented sing Python, C++ and Java. Empirical validation was done through comparative study and applicability on a real project using three different object oriented languages to prove its real applicability and usefulness. Further research in other data analytic languages will be beneficial to the software community.

Balogun and Sotonwa [15] proposed a multi-paradigm complexity metric (MCM) for measuring software complexity of C++ and Python which combine the features of procedural and object oriented paradigms. The developed metric was applied on software complexity metrics (eLOC, cyclomatic complexity metric and Halstead measures). It was discovered that he developed metric have significant comparison with the existing complexity measures and can be used to rank numerous program and difficulty of various modules. However of all the types of LOC only eLOC was used; Future work may be geared towards evaluating data analytical language on software measures.

Abhinav and Goldie [16] carried out an in-depth research on Python, python libraries, its data structure, outlined both its advantages and disadvantages. The metric compared python to C, C++, Java, Ruby, C#, and Javascript in terms of lines of code. It would be more appropriate to compare languages on par with python and to more popularized languages used for data analytics. Line of code is just one of the code based metrics that could be used. Using Halstead, McCabe and measuring the degree of relationship would make it a more extensive research with greater basis for judgment.

Sotonwa *et al.* [17] matched rigorous object oriented application (VB, C#, C++, Java and Python) languages of linear and binary search algorithms using software complexity metrics (LOC, McCabe method and Halstead method) which allowed for consistency in the object oriented languages. Statistical evaluation was performed on the metrics using Person correlation coefficient which showed a high degree of correlation existence among (VB, C#, C++, Java and Python) and analysis of variance (ANOVA) showed that for VB, C#, C++, Java and Python is good to code linear and binary search algorithms. However, other object oriented programming languages can be used to justify this result.

3. Materials and Methods

The metrics were applied on binary search algorithms codes written Python R and Scala. Thee (3) different binary search algorithms codes were considered.

3.1 Evaluation of Software Complexity of Binary Search Algorithms

To find the complexity of variations of different implementation languages. The following approaches were applied:

A. Lines of code (LOC): counts every line of the program including comments, standalone brace, blank lines and parenthesis.

B. McCabe method: using cyclomatic complexity method

$$MC = V(G) = e - n + 2p \quad (1)$$

where e is the edges, n is the nodes and p is the connected component.

C. Halstead methods compute the following parameters:

μ_1= the number of unique operators
μ_2= the number of unique operands
N_1 = the total occurrences of operators
N_2= the total occurrences of operands

i. The length N of P:

$$N = N_1 + N_2 \quad (2)$$

ii. The vocabulary μ of P:

$$\mu = \mu_1 + \mu_2 \quad (3)$$

iii. Program Difficulty: using Halstead method

$$Dof\ P = (\mu_1 \div 2) * (N_2 \div \mu_2) \quad (4)$$

iv. Volume: using Halstead method

$$V = N *_{log_2} (\mu)_1 \quad (5)$$

v. Effort: E to generate program is calculated using Halstead method

$$E = D * V \quad (6)$$

where D is the Difficulty and V is the Volume

vi. Error: using Halstead method

$$B = V/_{X*} \quad (7)$$

where B, is the number of delivered bugs, V is the volume of the program and Halstead sets X* for a fixed value of 3000. Therefore B is defined as:

vii. $$B = E^{(2/3)}/3000 \quad (8)$$

where E is the effort to generate the program

viii. Time: using Halstead method

$$T = E/_{18} \quad (9)$$

where E is the Effort to generate program

3.2 Analysis of the Metrics

The sample of the analysis of the metrics is shown in Figure 1 for the binary code written in R language and the flow chart representation in Figure 2.

```
1 V, S, I <- numeric()
2 h=length(V)
3 Bin_search_recursive <- function(V,S,l,h) {
4   if(h < l) {
5     return(FALSE)
6   } else {
7     m <- floor((l + h) / 2)
8   if(V[m] > S ){
9     Bin_search_recursive(V, S,l,m-1)
10  }else if( V[m] < S ){
11    Bin_search_recursive(V, S, m+1, h)
12  }else
13    return(m)
14 }
15 }
```

Figure 1: Binary Search in R

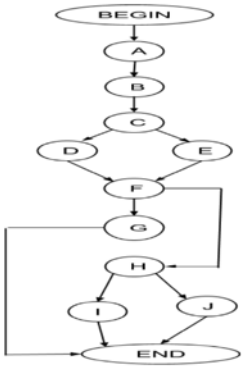

Figure 2: Flow Graph for Binary Search in R

Table 1: Comparison of Different Implementation of Binary Search Algorithm for McCabe. Error and Difficulty

Complexity Type	Python	R	Scala
McCabe	4	4	5
Error	0.1325	0.2296	0.34
Difficulty	2.91	4.65	8.68

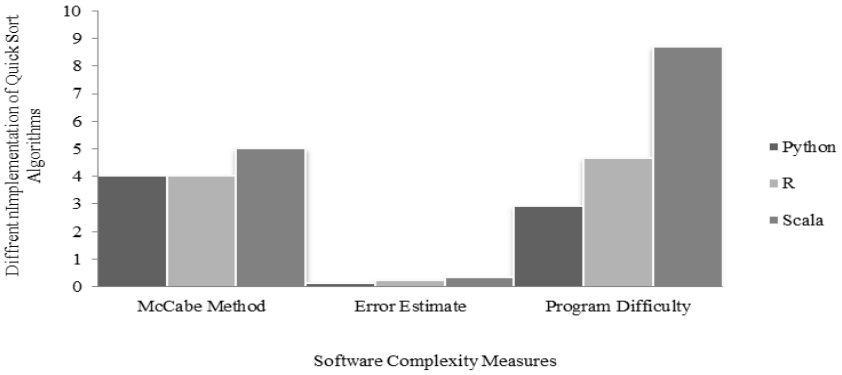

Figure 3: Relative Graph of Binary Search Algorithm for McCabe, Error and Difficulty

Figure 3 showed the relative graph of binary search in Python, R and Scala for McCabe, error and difficulty. For binary search of the three functional languages the McCabe method reveals that Python and R are at par in terms of their control flow complexity while Scala has the highest complexity. Scala is depicted to also generate more errors and more difficult as compared to Python and R.

Table 2: Comparison of Different Implementation of Binary Search Algorithm for LOC, Time and Volume

Complexity Type	Python	R	Scala
LOC	12	15	37
Time	525	1549	1046
Volume	285	414	599

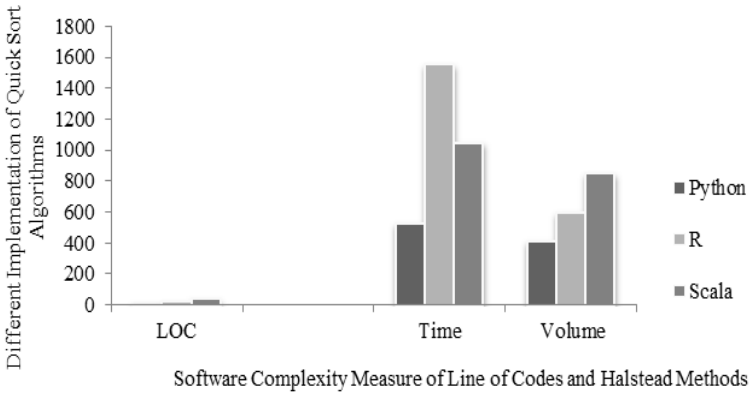

Figure 4: Relative Graph of Binary Search Algorithm LOC, Time and Volume

For Figure 4; in terms of LOC, time and volume, Python is lesser complexity. Deduced from the flow graph that when implementing binary search algorithm, Python is complex than R; Scala is the most complex language of the three functional languages.

4. Conclusions

Software complexity metrics are used to quantify a variety of software properties. Complexity measures can be used to predict critical information about testability, reliability, and maintainability of the software systems from automatic analysis of the source code. It plays a vital role to reduce the effort required in maintaining software, the effectiveness of testing and software quality. The more complex the software solution, the more errors it generates. It was found that for the six complexity metrics applied, Scala has the highest complexity value in all compared to Python and R. The metrics, however, do not give the same values.

Acknowledgments

Acknowledgement goes God the father, the Son and the Holy Spirit for making this work possible. Sincere appreciation also goes to entire staff of the Department of Computer Science and Information Technology, Bells University of Technology, Ota, Nigeria for their encouragement and supports.

References

[1] P. Hudak “Conception, Evolution and Application of Functional Programming Languages”, ACM Computing Survey, Vol. 21, No. 3, pp. 359-411. Doi:10.1145/72551.
[2] Chambers, John M. Programming with Data: A Guide to the S Language. Springer Verlag. pp. 67–70, 1998.
[3] Effective Scala, Scala Wiki. Retrieved 2012-02-21.
[4] Hartel, Pieter; Henk Muller; Hugh Glaser (March 2004). *"The Functional C Experience"*, Journal of Functional Programming. Vol. 14, No. 2,pp. 129–135.*2004. doi:10.1017/S0956796803004817*.;
[5] David Mertz. *Functiona*l Programming in Python, Part 3, *IBM developer Works.* Archived from *the original* on 2007-10-16. Retrieved 2006-09-17.(Part 1, Part 2)
[6] The useR! 2006 conference schedule includes papers on the commercial use of R". R-project.org. 2006-06-08. Retrieved 2011-06-20.
[7] Basil V.R. “Quantitative Software Complexity Models: A Summary in tutotial on Models and Methods for Software
[8] Cefer Ferid, Ibrahim Akman and Sanjay Misra “Estimating Complexity of a Source Code”, M.Sc. thesis, School of Natural and Applied Science, Software Engineering Department, Atilim University, Nakara, Turkey 2010.
[9] IEEE Standard, IEEE Computer Science, pp. 278, 1998.
[10] Munson J. and Khoshgoftaar T. “Software Metrics for Reliability Assessment”, in Handbook of Software Reliability Engineering, Michael Lyu(edt.), McGraw-Hill, , pp.493-529, 1996.
[11] Ammar H.H., Nikzadeh T. and Gugan J. “A Methodology for Risk Assessment of functional Specification of Software systems using Colored Petri Nets”, Proc. Of the Fourth International software Metrics Symposium Metrics, Albuquerque, New Mexico, pp.108-117, 1997.
[12] Donald K. “Data Structure and Program Design in C++” Prenice-Hall, ISBN0-13768995-0:280, 1997.

[13] Netty van Gasteren and Wim Feiien "The Binary Search Revisited, 1995..
[14] Sotonwa K. A., Olabiyisi S.O, and Omidiora E. O. "Comparative Analysis of Software Complexity of Searching Algorithms Using Code Based Metrics". International Journal of Scientific & Engineering Research, Vol. 4, Issue 6, June-2013 pp. 2983-2993.
[15] Balogun M. O. and Sotonwa K. A. "A Comparative Analysis of Complexity of C++ and Python Programming Languages Using Multi-Paradigm Complexity Metric (MCM)". International Journal of Science and Research (IJSR) ISSN: 2319-7064Vol. 8 Issue 1, January 2019, pp. 1832-1837.
[16] Abhinav Nagpal and Goldie Gabrani: 2019: Python for Data Analytics, Scientific and Technical Applications, Amity International Conference on Artificial Intelligence (AICAI), (IEEE), INAPEC Accession no. 18635640 DOI: 10.1109/AICAI.2019.8701341, Dubai, United Arab Emirate.
[17] Sotonwa K., Balogun M., Isola E., Olabiyisi S., Omidiora E. and Oyeleye C. "Object Oriented Programming Languages for Search Algorithms in Software Complexity Metrics". International Research Journal of Computer Science (IRJCS), Vol. 6, Issues 04, April 2019, pp.90-101.