IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.8, Issue.5, May 2020 E-ISSN: 2347-2693

Performance Evaluation of Platforms Using Software Metrics

K.S. Aparna[1], Vaibhav Kumar K.[2*], Sushmitha G.[3], Vanam Saikiran[4]

[1,2,3,4]Department of Computer Science and engineering, RYMEC, VTU, Ballari, India

*Corresponding Author: vybhavkumar03@gmail.com , Tel.: +91-8861834245

DOI: https://doi.org/10.26438/ijcse/v8i5.178181 | Available online at: www.ijcseonline.org

Received: 20/May/2020, Accepted: 25/May/2020, Published: 31/May/2020

Abstract—In this paper, the quality of software and its performance is evaluated using software metrics. The software product estimation is purely based on software metrics and is being applied on java and python platforms. Number of lines of code (NLOC), Cyclomatic complexity number (CCN), Token count and parameter count have been utilized to analyze the complexity of a program. The application is developed to test the codes of two different platforms and their complexities are compared using doubly linked list program as an example. The results are recorded and compared based on these software metrics to uncover the vary in results for similar projects.
The paper is divided into five sections. In section one we are discussing about software metrics and it's importance in deciding the performance of different programming platforms. In section two we are discussing about different software metrics that are used to evaluate the performance of the programs. In section 3 we are discussing the previous works and related works. In section four we are taking an example and trying to evaluate the performance of the programs that are written in java and python languages and discussing about the results. In section five we are discussing about the future work and the conclusion of our developed tool.

Keywords— metrics, Cyclomatic complexity number (CCN), Number of lines of code (NLOC), Analyzer (developed tool), parameter-count,token-count

I. INTRODUCTION

Software metrics are used to determine the various software properties for example, cost, effort, reliability, reusability, feasibility and maintenance. Programming building is a genuinely intelligent and vital plan process as a result of today's dynamic condition which is very erratic and on a basic level, not completely specifiable ahead of time. Successful programming quality assessment requires determinants that depict what quality is and how it tends to be followed back to the improvement procedure or the result itself. The programming industry is step by step advancing towards a time of high development; where casual ways to deal with the quality investigation cannot work anymore. Because of the progressive development [1], clients additionally perceive its worth and they are not ready to settle on the subjective viewpoints. Despite this, the inward nature of an item may go unchecked or be purposely undermined on occasion. Programming measurements are crude markers to code quality which gives us the way to take star dynamic activities at the most punctual stage conceivable, at whatever point venture is getting off course. Java is most popular and widely used programming language and it is a platform independent language. It is an object-oriented programming. Python works on different platforms and these two programming languages has been taken as input to our model. Further, Software measurements are instruments applied to a bit of programming or its plan details to accomplish reproducible quantitative estimations, which might be additionally applied in cost estimation, venture planning, investigating, quality confirmation.

II. DIFFERENT SOFTWARE METRICS AND ITS IMPACT

Software metrics are most valuable entities in the whole software life cycle. They provide the measurement for software development. By using metrics, the software quality can be improved. The analyzer is a software application giving code measurements, for example, cyclomatic complexity (CC) and the number of lines per program; it assists with recognizing the functions of the parameters and when applied to our code it surely assisted with finding trouble instances of confusing code, anyway measures, for example, Code complexity[13] have never been unambiguously related with imperfect or breakable code and it appears to feature code deserving of audit instead of expressly deficient code. The Code complexity stays a decent proportion of the number of experiments important to completely test a bit of code.

NLOC (Number of lines): This one of the most important length parameters which represent the total number of line of codes along with the comments. Parameter of the LOC measure claim [7] that LOC is an "artifact" of all software development projects that can be counted. NLOC is typically used to estimate or how much effort is required to develop a program as well as to estimate and evaluate other aspects of cost and quality.

Cyclomatic complexity number: It is a software metric used to indicate the complexity in the code and it is used for white box testing. Initially created by Thomas McCabe, this generally utilized measure checks straight autonomous ways through a progression of control chart [8]. This can be found by checking language watchwords and administrators which influence on source code multifaceted nature. Cyclomatic intricacy has an establishment in diagram hypothesis and gives us a very valuable sensible measurement. Cyclomatic complexity can be calculated in two ways

$V(G) = E - N + 2$ (1), $V(G) = P + 1$ (2)

Where,
E is Number of Edges
N is Number of node

Where,
P is predicted conditions

Functions Count: All the capacities, strategies, or subroutines are considered under this physical well as coherent measurement. When contrasted with LOC, it is increasingly important a size-metric because somewhat, it is free of the programming language selected. It is anything but difficult to figure and serves best as an interface metric. The benefits of using this measurement to support management decision-making in order to ensure the quality of the program.

Token count: A computer program is a collection of tokens, which may be functions of the parameters and when applied to our code it surely assisted with finding trouble instances of confusing code.

III. RELATED WORK

Estimation is basic in any building space, and there is no exclusion of programming designing. A few specialists in the past have applied programming measurements as key contributions to direct quality indicators. Henrike Barkmann [3] distinguishes the connection between's few measurements from notable article situated measurements suites, for example, CK measurements, McCabe Cyclomatic Complexity, and different size measurements, other than introducing potential edges. Yasunari Takai et al. [4] propose new programming measurements dependent on coding guidelines infringement to catch dormant blames in a turn of events. Dad Juda distinguishes a straight development pattern in programming size for manned space and airplane, which can sensibly anticipate programming size in comparative future projects, utilizing SLOC based information. Zhou Yuming and XU Baowen [5] research the connections of size and unpredictability measurements with the viability of opensource programming. S. Pradeep et aluses CK measurements, SLOC, COM measurements, and so on to examine the connection between programming measurements and imperfections. Domenico Cotroneo shows the connection between programming maturing and a few static highlights of the product. Cesar Couto, Christofer Silver find confirmations towards causality between programming measurements (as indicators) and the event of bugs. Yuming Zhou rethinks the capacity of unpredictability measurements to anticipate shortcoming inclination. Daniela Glasberg approves OO plan measurements on a business Java application. AK Pandey has utilized LOC, MVG, and Halstead measurements to arrange the product module as flaw inclined or not. Zhou et al infers that LOC and WMC (weighted technique McCabe multifaceted nature) are without a doubt preferable deficiency inclination indicator over other lesser-known intricacy measurements SDMC, AMC. In his enormous observational investigation of five Microsoft programming frameworks, Nagappan [6] found that disappointment inclined programming substances are measurably associated with code unpredictability measures.

IV. EXAMPLE AND RESULTS

For example, let's take a program that has been written in both the languages java and python. Here we took a Doubly Linked-list program and we got the following result Fig:1. By analysing that result the programmer will choose the language in which he has to be implement his project or any application.

4.1 The java program is as follows:

```
public class DoublyLinkedList {
class Node{
int data;
Node previous;
Node next;
public Node(int data) {
this.data = data;
}
}
Node head, tail = null;
public void addNode(int data) {
Node newNode = new Node(data);
if(head == null) {
head = tail = newNode;
head.previous = null;
tail.next = null;
}
else {
tail.next = newNode;
newNode.previous = tail;
tail = newNode
tail.next = null; }
}
public void display() {
Node current = head;
if(head == null) {
System.out.println("List is empty");
return;
}
System.out.println("Nodes of doubly linked list: ");
while(current != null) {
System.out.print(current.data + " ");
current = current.next;
}
35.  }
36.  public static void main(String[] args) {
37.  DoublyLinkedList dList =
     new DoublyLinkedList();
38.  dList.addNode(1);
39.  dList.addNode(2);
40.   40. dList.addNode(3);
41.  dList.addNode(4);
42.  dList.addNode(5);
43.  dList.display();
44.  }
45.  }
```

4.2 The python program is as follows:

```
1.          class Node:
2.          def __init__(self,data):
3.          self.data = data;
4.          self.previous = None;
5.          self.next = None;
6.           class DoublyLinkedList:
7.          def __init__(self):
8.          self.head = None;
9.          self.tail = None;
10.            def addNode(self, data):
11.        newNode =
Node(datadat
12.        if(self.head == None):
13.            self.head = self.tail = newNode;
14.            self.head.previous = None;
15.                self.tail.next = None;
16.            else:
17.            self.tail.next = newNode;
18.            newNode.previous = self.tail;
19.            self.tail = newNode;
20.                self.tail.next = None;
21.     def display(self):
22.     current = self.head;
23.       if(self.head == None):
24.            print("List is empty");
25.            return;
26.            print("Nodes of doubly linked list: ");
27.            while(current != None):
28.            print(current.data),;
29.            current = current.next;
30.            dList = DoublyLinkedList();
31.            dList.addNode(1);
32.            dList.addNode(2);
33.            dList.addNode(3);
34.  dList.addNode(4);
35.  dList.addNode(5);
36.  dList.display();
```

The analyzer can compute the cyclomatic multifaceted nature for programming written in different dialects. It lines up with unpredictability, although it tallies switch proclamations [11] by the number of cases and takes a gander at consistent articulations for fanning.

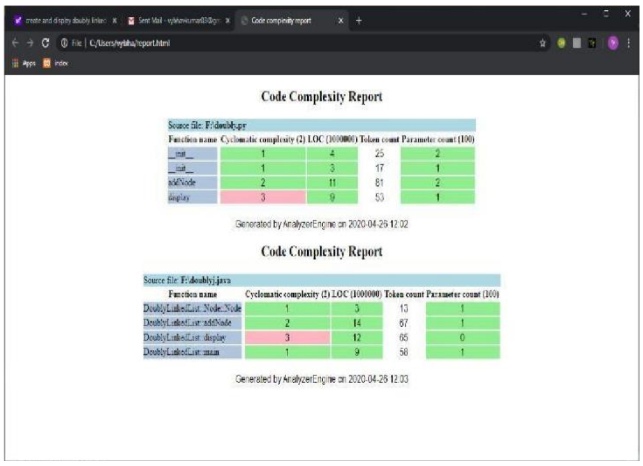
Result discussion of java program:

1. Mathematically, the cyclomatic complexity is calculated by using
   It can be calculated for each function for example,
   In 4.1line number 25. Public display()
   There are two condition nodes so
   V(G) = 2+1 = 3 (3)
   Here P =2
2. Lines of each class has been calculated by using the length.
3. Token count for each class is
   The total number of tokens in function display()
   N = 53 (4)
4. The number of parameters in function display() are calculated by parameters passed in a function.

Result discussion of python program:

1. The cyclomatic complexity for the function display()
   In 4.2 line number 21. Of the function display()
   There are two condition nodes so
   V(G) = 2+1 =3 (here p=2) (5)
2. Lines of each class has been calculated by using the length.
3. Token count for each class is
   The total number of tokens in function display()
   N = 65 (6)
4. The number of parameters in function display() are calculated by parameters passed in a function.

As explained in the above the results for other functions are calculated by using our software application. The main difference we observed is some functions having high complexity and high parameter and token counts. By analyzing all the results the python program is having the good performance.



Code Complexity Report

Source file: **F:\doubly.py**

Function name	Cyclomatic complexity (2)	LOC (1000000)	Token count	Parameter count (100)
__init__	1	4	25	2
__init__	1	3	17	1
addNode	2	11	81	2
display	3	9	53	1

Generated by AnalyzerEngine on 2020-04-26 12:02

Code Complexity Report

Source file: **F:\doublyj.java**

Function name	Cyclomatic complexity (2)	LOC (1000000)	Token count	Parameter count (100)
DoublyLinkedList::Node::Node	1	3	13	1
DoublyLinkedList::addNode	2	14	67	1
DoublyLinkedList::display	3	12	65	0
DoublyLinkedList::main	1	9	58	1

Generated by AnalyzerEngine on 2020-04-26 12:03

Figure 1. Doubly Linked-List program report

V. FUTURE WORK AND CONCLUSION

There are a lot more things that need to be done in the future with the software application of Analyzer to include a more number of parameters to be tested and it should cover more programming languages. This paper describes the progression of four software metrics on a set of two programming languages java and python programming platforms through software automated analysis tools. Cyclomatic complexity centres around the unpredictability of the program. Quality measures [16] can help to find out which lines of code need to be made more elegant and simpler, which lines need more extensive testing and in line with a lot of experiments going on in the same direction, we need to build things easier to use we need to build things easier to use and should have an economic value. This economical value plays a very important role in deciding the different platforms required for building the various applications within cost, time and budget constraints. These software metrics play very major role in deciding the performances of various emerging platforms.

REFERENCES

[1] H. Barkmann, R. Lincke, and W. Löwe. *"Evaluation of Software Quality Metrics In Open-Source Projects"*. In Proceedings of The 2009 IEEE International Workshop on Quantitative Evaluation of large-scale Systems and Technologies (QuEST09), Bradford, UK, 26-29th May 2009.

[2] Yasunar Takai, Takashi Kobayashi, kiyoshi Agus *"Software Metrics based on Coding Standards Violations"*, In Proc. the Joint Conference of the 21th International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement (IWSM/MENSURA2011) pp.273-278, Nara, Japan, 3-4 Nov. 2011.

[3] Paul A. Judas, Lorraine E. Prokop. *"A historical compilation of software metrics with applicability to NASA's Orion spacecraft flight software sizing"*. ISSE 7(3): 161-170 (2011).

[4] Yuming Z., Baowen X. *"Predicting the Maintainability of Open Source Software Using Design Metrics"*. Wuhan University Journal of Natural Sciences, Vol. 13 No.1, PP 14-20. 2008.

[5] S Pradeep, Chaudhary K D and V Shrish. *"An Investigation of the Relationships between Software Metrics and Defects"*. International Journal of Computer Applications 28(8):13-17, August 2011. Foundation of Computer Science, New York, USA.US

[6] M.Karanam, L.Gottemukkala "*Software Fault Detection Using Improved Relief Detection Method"*,Vol 4, Issue 5, pp.1-4, October 2016.

[7] A. Fitzsimmons and T. Love, *"A review and evaluation of software science,"* Computing Survey, vol. 10, no. 1, March 1978.

[8] S. D. Conte, H. E Dunsmore, and V. Y. Shen, *" Software engineering metrics and models"* Benjamin/ Cummings Publishing Company, Inc., 1986.

[9] B. A. Nejmeh, *"NPATH: A measure of execution path complexity and its applications,"* Comm. of the ACM, vol. 31, no. 2, pp. 188-210, February 1988.

[10] T.A. McCabe, *"A complexity measure,"* IEEE Transactions on Software Engineering, vol. 2, no. 4, pp. 308-320, December 1976.

[11] Woodward, Hennell, M.A., and Heldey, D.A.*: "A measure of control flow complexity in program text",* IEEE Transactions on Software Engineering, 1979, 5, (1). pp. 45 - 50.

[12] Hall, N.R., and Preiser, S.: *"Combined network complexity measures",* IBM Journal of Research & Development, 1984, 28, (I), pp. 15 – 27.

[13] Aanchal, Sonu Kumar "*Metrics for software components in object oriented environments: A survey",* Vol 1, Issue 2, march-april-2013.

[14] Wiener, R., and , R.*: "Software engineering with Modula-2 and Ada"* (Wiley, 1984).

[15] Dahl, O.J., Dijkstra, E.W., and Hoare, C.A.R.: *"Structured programming"* (Academic Press, 1972).

[16] Dijkstra, E.W.:„Goto statement considered harmful", Communications of ACM, 1968, 18, (8), pp. 453 – 457.

[17] O, Omidiora. E.O, Balogin M.O. *"A complexity metric for multi-paradigm programming languages",* vol.4, No.12, December 2014.

[18] Ermiyas Birihanu Belachew, Feidu Akmel Gobena, Shumet Tadesse Nigatu,*"Analysis of software quality using software metrics",* vol.8, No.4/5, October 2018.

[19] Gurudev Singh, Dilbag Singh, Vikram Singh *"A study of software metrics"* vol.11, January 2011.

[20] Neha saini, Sapna kharwar, Anushree Agarwal,*"A Study of significant software metrics"*,vol.3, No.12, July 2014.

Authors Profile

Mrs. k s Aparna working as assistant professor in the Department of Computer Science and Engineering at Rao Bahadhur Y Mahabaleswarappa Engineering college, Affiliated to VTU, Belagavi, approved by AICTE, New Delhi,and Govt. Of Karnataka, certified by NAAC with B++, Cantonment, Ballari - 583101

Mr. Vaibhav Kumar K pursuing BE in Department of Computer Science and Engineering at Rao Bahadhur Y Mahabaleswarappa Engineering college, Affiliated to VTU, Belagavi, approved by AICTE, New Delhi,and Govt. Of Karnataka, certified by NAAC with B++, Cantonment, Ballari - 583101

Miss. Sushmitha G pursuing BE in Department of Computer Science and Engineering at Rao Bahadhur Y Mahabaleswarappa Engineering college, Affiliated to VTU, Belagavi, approved by AICTE, New Delhi,and Govt. Of Karnataka, certified by NAAC with B++, Cantonment, Ballari – 583101

Mr. Vanam Saikiran pursuing BE in Department of Computer Science and Engineering at Rao Bahadhur Y Mahabaleswarappa Engineering college, Affiliated to VTU, Belagavi, approved by AICTE, New Delhi,and Govt. Of Karnataka, certified by NAAC with B++, Cantonment, Ballari – 583101.