International Journal of Scientific Research and Management (IJSRM)
||Volume||07||Issue||02||Pages||EC-2019-223-237||2019||
Website: www.ijsrm.in ISSN (e): 2321-3418
Index Copernicus value (2015): 57.47, (2016):93.67, DOI: 10.18535/ijsrm/v7i2.ec03

Code Quality Metric of Searching Algorithms for Multi-Paradigm Languages

Sotonwa K. A.[1], Balogun M. O.[2], Olusi Titilayo[3]
[1]Department of Computer Science and Information Technology, Bells University of Technology
[2]Department of Electrical and Computer Engineering, Kwara State University
P.M.B 1530, .Malete, Ilorin, Kwara State, Nigeria
[3]Department of Computer Science, I. I. C. T., Kwara State Polytechnic
Ilorin, Kwara State, Nigeria

Abstract:
This study investigated the comprehensibility of a software code from a developer‟s point of view and proposed new metric accordingly. The factors that affect the complexity of procedural, object-oriented, and multi-paradigm codes were analyzed for this purpose. Addition to the investigated factors, various metrics and several aspects were combined in the proposed metric. The proposed metric were empirically validated in different paradigms.

Keywords: MCM, FPA, CQM, C++**,** Java, Python.

1. Introduction

A multi-paradigm programming language is a programming language that supports more than one programming paradigm. The idea of a multi paradigm language is to provide a framework in which programmers can work in a variety of styles, freely intermixing constructs from different paradigms. Hence, multi-paradigm languages gained popularity by merging elements of various programming paradigms into a cohesive language which utilizes programming and conceptual aspects from different paradigms [1]. According to Coplien [2], multi-paradigm design becomes an audit for that intuition and provides techniques and vocabulary to regularize the design [3]. Some of the most popular multi-paradigm languages are C++, Python, JavaScript. One of the advantages of Python is its readability; JavaScript has easy integration and lower maintenance cost [4] while C++ has a strong performance.

Function Point Analysis (FPA) is a method of functional size measurement. It assesses the functionality delivered to its users, based on the user's external view of the functional requirements. Function point analysis measures software by quantifying the functionality the software provides to the user based primarily on logical design. With this in mind, the objectives of function point analysis are to measure functionality that the user requests and receives and measure software development and maintenance independently of technology used for implementation[5], [6].

According to ISO9126, software quality could be broken down into six major characteristics, which are functionality, reliability, usability, efficiency, portability and maintainability and these quality characteristics are split to a number of quality sub-characteristics. The above mentioned product quality characteristics can be divided into two different sets: external and internal. Customers care about external quality characteristics such as: functionality, reliability, usability, efficiency, flexibility, friendliness, simplicity, etc. simply because these are the characteristics which can easily be seen by the use of the product.

On the other hand, developers care about internal quality characteristics such as: maintainability, portability, reusability, testability, etc. because these characteristics relate to their development efforts [7], [8] started with a volume of 55 quality characteristics which have an important influence

on quality, and called them "factors". For reasons of simplicity, McCall [9] then reduced the number of characteristics to the following eleven. A second set of quality factors was defined by [10] added to McCall's original list to reflect recent thinking [11] listed sixteen factors. A full list of both models is set out in the Table 1 Because of the format used to describe them; quality factors are often referred to as the ility factors.

Code quality is closely related with testing and measurement [12]. As the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules. Testing techniques tend to find defects, bottlenecks and weaknesses of a software system. Measurement aims to find the complexity in order to understand the effectiveness of the software‟s code. Requirement to improve the software quality is the prime objective, which promotes research projects on software metrics technology. It is always hard to control the quality if the code is complex [13], [14].

Table 1: Software Quality Model

McCall et al. (1976-77)	Boëhm (1978)
Efficiency	Usability
Integrity	Clarity
Reliability	Efficiency
Usability	Reliability
Accuracy	Modifiability
Maintainability	Reusability
Testability	Modularity
Flexibility	Documentation
Interface facility	Resilience
Transferability	Maintainability
	Portability
	Interoperability
	Understandability
	Integrity
	Validity
	Flexibility
	Generality
	Economy
	Portability
	Interoperability

Complex codes are hard to review, test, maintain and manage. As a consequence, those handicaps increase the maintenance cost and the cost of the product. Due to these reasons, it is strongly recommended that the complexity of the code should be controlled from the beginning of the software development process.

2. Literature Review

There are lots of metrics that have been developed until now. Each metric has its own advantages and disadvantages. Metrics may tend to measure quality, size, complexity, requirements, effort, productivity, cost and schedule, scrap and rework, and support [15].

Typically the metrics can be classified as follows [15]:

i. Technical metrics: are used to measure the structure of the code, external characteristics, manuals and documentation.
ii. Defect metrics: measure defects of software. For instance, the number of defects found in a specific time.
iii. End-user satisfaction: is based on the value received from using the system.
iv. Warranty metrics: focus on revenues and expenditures related with correcting the software defects.
v. Reputation metrics: are related with user satisfaction.

Although, metrics can be classified variously, this paper categorises them according to paradigms.

3. Methodology

The study investigates the factors that affect the complexity of multi-paradigm codes and then developed a Code Quality Metric (CQM) for programming languages. For validation of the metric, the metric is applied to some searching algorithms codes written in C++, Java and Python.

3.1 The Metric

To measure CQM a new metric was developed for the procedural parts of multi-paradigm codes. The research was extended by considering also the object oriented parts of the codes. This means that the extended metric combines procedural and OO factors. The cognitive aspect is included also, as in procedural complexity measure. The factors which are responsible for the complexity of multi paradigm language are.

Factors of complexity of procedural part:

i. Variables and constants
ii. Basic Control Structures

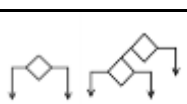
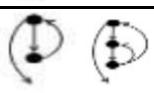
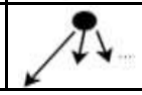
Factors of Complexity of object oriented Part:

i. Attributes and constants,
ii. Basic Control Structures; and
iii. Classes.

To cover a wider aspect of programming features Basic control structures are simplified as follows. The new version of BCS can be seen in Table 1. According to [16] and also [17], it is important to note that while calling a module, „2" is added to the total CWU of the called module. Hence, coupling factor is included in the structure. In the first part of the research, number of variables, operators, and constants are multiplied with structures. However, in the second stage of the research it is not necessary to multiply variables, operators or constants with structures. Otherwise the metric would force the programmers to be confused in too many details.

Due to later addition of numerous factors that affect the complexity, it is decided that following similar approach would totally make up an extremely complex metric. Thus, even though it is still possible to use PCCM, a simpler version of procedural complexity is also proposed. The metric is developed in a way that it can measure object oriented and procedural parts separately. Then, 0 should be assigned to the parts that are not related to OO paradigm.

Table 2: Basic control structure for Multi-paradigm Complexity Measure

Category	CWU	Flow Graph
Sequence	1	
Condition	2	
Nested sub-condition	1	
Loop	3	
Nested Sub-loop	2	
Module call	2	
Recursion	3	
Exception	1	

According to further investigations of the complexity factors, the study developed a metric for multi-paradigm codes as follows.

$$MCM: = CIclass + CDclass + PCCM \quad (1)$$

Where, CIclass = Complexity of Inherited Classes, CDclass = Complexity of Distinct Class PCCM = Procedural Complexity

Although, PCCM measures the procedural complexity, it is assumed that using PCCM is difficult. MCM includes various complexity factors. PCCM, being a part of MCM, would make the metric too complex and too difficult to apply. Therefore, it is recommended that Cprocedural is used in place of PCCM in the metric. However, it is possible to use MCM with PCCM for more detailed measurement. From *eqn (1):*

$$MCM = CIclass + CDclass + Cprocedural \quad (2)$$

where Cprocedural = Procedural Complexity

All these factors are defined as follows: Cclass can be defined as complexity of a class. Cclass takes a major role in the calculation of both CDclass and CIclass. For example, for calculating CIclass, CDclass first, it is needed to calculate Cclass. Cclass is defined as:

$$Cclass = W(attributes) + W(variables) + W(structures) + W(objects) - W(cohesion) \quad (3)$$

Where, Cclass = Complexity of Class

The reason of subtraction of cohesion is that it reduces the complexity and thus it is desirable from the point of view of software developers [5]

Where, weight of attributes or variables is defined as:

$$W(\text{variables or attributes}) = 4*AND + MND \quad 4)$$

Where AND = number of arbitrarily named distinct variable/ attributes and MND = Number of meaningfully named distinct variables/attributes

Weight of structure W (structures) is defined as:

$$W(structure) = W(BCS) \quad (5)$$

Where, BCS are basic control structure.

Weight of objects, Weight (objects) is defined as:

W (objects) = 2 *(6)*

Creating an object is counted as 2, because while creating an object constructor is automatically called. Thus, coupling occurs. Therefore, it is the same as calling a function or creating an object. Here it is meant to be the objects created inside a class. Moreover, a method that calls another method is another cause of coupling, but that fact is added to MCM value inside Weight (structures). Weight of cohesion is defined as:

W (cohesion) = MA/AM *(7)*

Where, MA = Number of methods where attributes are used and AM = Number of attributes used inside methods

While counting the number of attributes there is no any importance of AND or MND

CIclass can be defined as;

There are two cases for calculating the complexity of the Inheritance classes depending on the architecture:

i If the classes are in the same level then their weights are added.

ii If they are children of a class then their weights are multiplied due to inheritance property.

If there are m levels of depth in the object oriented code and level j has n classes then the Cognitive Code Complexity (CCC) of the system [18] is given as:

$$C_{class} = \prod_{j=1}^{m} \left\lceil \sum_{k=1}^{n} CC_{jk} \right\rceil \quad (8)$$

CDclass can be defined as:

CDclass = Cclass(x) + Cclass(y) + ----- *(9)*

Note: All classes, which are neither inherited nor derived from another, are parts of CDclass even if they have caused coupling together with other classes.

Cprocedural can be defined as;

Cprocedural = W (variables) + W (structures) + W (objects) – W (cohesion) *(10)*

Weight of variable W (variable) is defined as:

From *eqn (4) W (variables) = 4* AND + MND* The variables are defined globally.

Weight of structure W (structures) is defined as:

W (structures) = W (BCS) + object.method *11)*

Where, BCS are basic control structure, and those structures are used globally. „object.method‟ is calling a reachable method of a class using an object.„object.method‟ is counted as 2, because it is calling a function written by the programmer. If the program consists of only procedural code, then the weight of the „object.method‟ will be 0.

Weight of objects W (objects) is defined as W (objects) = 2 from *eqn. (6).* Creating an object is counted as 2, as it is described above. Here it is meant to be the objects created globally or inside any function which is not a part of any class. If the program consists of only procedural code, then the weight of the „objects‟ will be 0.

W (cohesion) =NF = NV *(12)*

Where, NF is number of functions, and NV means number of variables. Coupling is added inside W (structures) as mentioned in the beginning of the description of the metric.

After completing the calculation of MCM, the program‟s Function Point Analysis (FPA) is also needed to be calculated before CQM because to obtain code quality, FPA is divided by MCM. FPA calculations are used to determine the cost for a single unit based on calculation of UFP which is broken into two main types of functions: data function and transaction function. Data Functions:

i. Internal Logic File (ILF): An EIF has the inherent meaning that it is externally maintained, an interface has to be developed to get the data from the file.

ii. External Interface File (EIF): n ILF has the inherent meaning that it is internally maintained, it has some logical structure and it is stored in a file. i.e. EIF counted for an application and there must be an ILF in another application

Transaction Functions:

i. External Input (EI): is a transaction function in which Data goes "into" the application from outside the boundary to inside.
ii. External Output (EO): External Output (EO) is a transaction function in which data comes "out" of the system.
iii. External Inquiry (EQ): External Inquiry (EQ) is a transaction function with both input and output components that result in data retrieval.

However, FPA is chosen for its ease of use and popularity. To measure FPA, there are 14 questions to be answered for system characteristics and each of these questions are rated on the following Degree of Influence (DI): 0 = No Influence, 1 = Incidental, 2 = Moderate, 3 = Average, 4 = Significant and 5 = Essential respectively.

Above questions should be numbered from 1 (low) to 5 (high). After answering above questions, the following table should be filled for the weighting factor:

Table 3: Calculation of Value Adjustment Factor

S/No	General System Characteristics (GSC)	DI
1	Does the system require backup and recovery?	1
2	Are the specialized data communications required to transfer information to or from the application	0
3	Are there distributed processing functions?	0
4	Is performance critical?	0
5	Will the system run in an existing, heavily utilized operational environment?	2
6	Does the system require on-line data entry?	0
7	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	0
8	Are the ILFs updated on-line?	0
9	Are the inputs, outputs, files, or inquiries complex?	2
10	Is the internal processing complex?	3
11	Is the code designed to be reusable?	4
12	Are conversion and installation included in the design?	0
13	Is the system designed for multiple installations in different organizations?	0
14	Is the application designed to facilitate change and for ease of use by the user?	2
Total Degree of Influence (TDI)		14

Table 4: Complexity of components of Weighting Factor

Function Type	Weighting Factor				
	Estimated Count	Low	Average	High	FP Count
EIs	1	3	4	6	4
EOs	1	4	5	7	5
EQs	0	3	4	6	0
ILFs	0	7	10	15	0
ELFs	0	5	7	10	0
UFP					9

$$FPA = UFP * VAF \quad (13)$$

Where *FPA* is the Function Point Analysis, *VAF* is Value Adjustment Factor and *UFP* is Unadjusted Function Point

$$VAF = TDI *(0.65 + 0.01) \quad (14)$$

Where *TDI* is Total Degree of Influence Then;

$$CQM= (FPA / MCM) * 10,000 \quad (15)$$

Where CQM is the Code Quality Metric.

The division of FPA by MCM usually gives an extremely small number which may be difficult to make decisions on. Therefore, it may be suitable to multiply by 10, 000 in order to make the calculation easier to understand

3.2 Demonstration of the metric and its analysis

For demonstration of CQM, linear and binary search algorithms codes were considered, the codes were written in three different programming languages, which include C++, JAVA and Python. Since the metric consist of both procedural and object oriented parts, so in the tables the parts that are in dark colour represent the object oriented part of the code, from which the complexity of the

distinct class is calculated, while the light part represent the procedural part. The calculations for CQM are given in Tables A to F. The analysis of the metric for each code from the application of CQM is given at the end of each Table.

Table 5: C++ Linear Search Demonstration

	Att	Str	Var	obj	MA	AM	Cohesio	Compl e
#include "ClassOfSearchAlgos .h"					2	1	2	2
#include <iostream>								
#include <fstream>								
using namespace std;								
char a[8000][7]={{'\0'},{'\ 0'}};	1		4					5
int main() {		1						1
string wanted;			1					1
cout << "Search :";		3						3
cin >> wanted;		3	1					4
ClassOfSearchAlgos r1;				4				4
r1. load_data_from_file("C:\\Users\\Tolu \\workspace\\tolu\\src\ \Codes.txt");								
cout<<r1.linear_searc h(8000,(char*) wanted.c_str())<<"\n" ;	1	6	2					9
}								
char * ClassOfSearchAlgos: :linear_search(int array_size, char * wanted)		1						1
{								
string msg;			1					1
char stringvalue[25];			2					2
int number;			1					1
int high = 0;			2					2
bool fl = false;			5					5
high =array_size;			2					2
msg = "Using Linear Search, The Element Found at position ";		1	1					2
fl = false;			5					5
for (int i = 0; i < high; i++)		3	1 4					1 7
{								
ifatoi(wanted)== atoi(a[i]))	1	6	5					1 2
{								
number = i +1;			6					6
itoa(number, stringvalue, 10);		2	3					5
msg =msg + stringvalue + "," + " ";		2	3					5
fl = true;			5					5
}								
}								
if (fl == false)		2	5					7
{								
msg = "Element Not Found";		1	1					2
}								
return (char*) msg.c_str();		4	1					5
}								
voidClassOfSearchAl gos::load_data_from _file(char * filename)		1						1
{								
ifstream my_data(filename);			2	2				4
int i=0;			4					4
while(!my_data.eof())		5	1					6
{								
my_data >> a[i];	1	2	4					7
cout<< a[i]<<"\n";	1	2	5					8
i++;			4					4
}								
my_data.close();		2	1					3
}								

MCM = CIclass +CDclass + Cprocedural From the table there is no inherited class so, CIclass = 0

CDclass = 5+9+7+17+12+6+15+9+14+6+7+12+3 = 122

Cprocedural = 7+3+4+4+9 =27 MCM = 0 +122 +27 =149

i. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/149) *10,000 = 5,581.2080

Table 6: C++ Binary Search Demonstration

	Att	Str	Var	Obj	MA	AM	Cohesion	Complexi
#include "ClassOfSearchAlgos.h"								
#include <iostream>					**4**	**1**	**4**	**4**
#include <fstream>								
using namespace std;								
char a[8000][7]={{'\0'},{'\0'}};	**1**		**4**					**5**
								1
int main() {		**1**						**1**
string wanted;			1					1
cout << "Search :";		3						3
cin >> wanted;		3	1					4
ClassOfSearchAlgos r1, r2;				4				4
oad_data_from_file("C:\\Users\\Tolu\\workspace\\tolu\\src\\Codes.txt");								
cout<<r1.binary_search(8000,(**char***) wanted.c_str())<<"\n";	1	6	2					9
}								
char * ClassOfSearchAlgos::binary_search(int array_size, char * wanted)								
{								
char stringvalue[25];			2					2
string msg;			1					1
int number;			1					1
bool fl = false;			5					5
int low = 0;			2					2
int high = 0;			2					2
int mid = 0;			2					2
low = 0;			2					2
high =array_size-1;			3					3
while (low <= high)		3	2					5
{								
mid = (int)((low + high) / 2);		2	4					6
if (atoi(a[mid])==atoi(wanted))	1	6	2					9
{								
number = mid +1;			3					3
itoa(number, stringvalue, 10);		2	3					5
msg =msg + stringvalue + "," + " ";		2	3					5
fl = true;			5					5
}								
if (atoi(a[mid]) > atoi(wanted))	1	6	2					9
{								
high = mid - 1;			3					3
}								
Else								
{								
low = mid + 1;			3					3
}								
}								
if (fl == false)		2	5					7
{								
msg = "Element Not Found";		1	1					2
}								
return (char*) msg.c_str();		4	1					5
}								
void ClassOfSearchAlgos::**load_data_from_file**(**char** * filename)		**1**						**1**
{								
ifstream my_data(filename);			2	2				4
int i=0;			4					4
while(!my_data.eof())		5	1					6
{								
my_data >> a[i];	1	2	4					7
cout<< a[i]<<"\n";	1	2	5					8
i++;			4					4

}								
my_data.close();		2	1					3
}								
void sort(int array_size)	1							1
{								
int temp;			1					1
for (int i = 0; i < array_size; i++)	1	3	15					19
{								
for (int j = i; j < array_size; j++)	1	5	19					25
{								
if (a[j] < a[i])	2	2	9					13
{								
temp = a[i];	1		6					7
a[i] = a[j];	2		9					11
a[j] = temp;	1		6					7
}								
}								
}								
}								
int mid(int left, int right)	1							1
{								
if (left <= right)		2	3					5
{								
if ((left + right) % 2 == 0)		2	7					9
return (left + right - 1)/2;			7					7
Else								
return (left + right) / 2;			5					5
}								
Else								
{								
return -1;			2					2
}								
}								
}								

i. *MCM = CIclass +CDclass + Cprocedural*
From the table there is no inherited class so,
CIclass = 0
CDclass
=2+2+5+10+8+6+9+3+15+9+3+3+7+2+5+9+6+7+8+4+3 =126
Cprocedural
=18+9+2+19+25+13+7+11+7+6+9+7+5+2
=140
MCM = 0+126 + 140 = 266

ii. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/266) *10,000 = 3,126.3157

Table 7: Java Linear Search Demonstration

	Att	Su	Var	Obj	MA	AM	Cohesion	Complexi
public class LinearSearch {					2	1		
private static int[] x = new int[1];	1		1					2
public static String Search(**int** key)								
{								
String msg = **null**;			2					2
boolean fl = **false**;			5					5
msg = "Element Not Found";		1	1					2
fl = **false**;			5					5
msg = "Using Linear Search, The Element Found At position ";		1	1					2
for (**int** i = 0; i < *x*.length; i++)	1	3	13					17
{						1		
if (key==*x*[i])	1	2	5					8
{								
msg = msg + (i + 1) + "," + " ";			7					7
fl = **true**;			5					5
}								
}								
if (fl == **false**)	2		5					7
{								
msg = "Using Linear Search, The Element Not Found";		1	1					2
}								
return msg;			1					1

}								
public static void load_data_from_file(String filename)		1						1
{								
int i=0;			6					6
{								
msg = "Using Linear Search, The Element Not Found";		1	1					2
}								
return msg;			1					1
}								
public static void load_data_from_file(String filename)		1						1
{								
int i=0;			6					6
File file=new File(filename);			3					3
Scanner my_data = null;		2						2
try								
{								
my_data = new Scanner(file);			3	2				5
while(my_data.hasNextLine()		3		2				5
{								
i=i+1;			11					11
nt[] tempReDim1 = new int[i];			6					6
if (x != null)	1	2	2			1		6
System.arraycopy(x, 0, tempReDim1, Math0,.min(x .length, tempReDi1.length));	2	4	4					10
x = tempReDim1;	1		2					3
x[i-1] =Integer.valueOf(my_data.nextLine());	1	4	7					12
System.out.println(x[i-1]);	1	2	6					9
}								
my_data.close();		2						2
}								
catch(FileNotFoundException e)				2				2
{								
System.out.println("The file: " + file + " does not exist.");		6	2					8
System.exit(0);		2	1					3
}								
}								
import java.io.BufferedReader;								
import java.io.InputStreamReader;								
public final class TestProgram {								
static int[] a = new int[1];	1		2					3
/**								
* @param args								
*/								
public static void main(String[] args) throws Exception		2		2				4
{								
int key=1;			2					2
String filename="C:\\Users\\Tolu\\workspace\\tolu\\src\\Codes.txt";		1	5					6
LinearSearch ls= new LinearSearch();		1	2	2				5
ls.load_data_from_file(filename);		1		2				3
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));		1		4				5
System.out.print("\nsearch: ");								
String wanted = in.readLine();		1	1					2
System.out.println(ls.Search(Integer.valueOf(wanted)));				4				4
}								
}								

i. MCM = CIclass +CDclass + Cprocedural From the table CIclass = 0
CDclass = 9+9+17+8+7+5+10+12+10+11+6+6+10+3+12 + 9+4+8+3 = 159
Cprocedural = 7+13+8+6 = 34
Therefore, MCM =0 +159 + 34 = 193

i. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/193) *10,000 = 4,308.8083

Table 8: Java Binary Search Demonstration

	Att	Str	Var	Obj	MA	AM	Cohesion	Complexity
					3	1		
private static int[] x = new int[1];	1							1
public static String Search(int wanted, boolean sortArray)		1						1
{								
String msg = null;			3					3
int first = 0;			3					3
int last = 0;			3					3
int mid_pt = 0;			3					3
msg = "Using Binary Search, The Element Not Found";		1	2					3
first = 0;			3					3
last = x.length-1;	1		4					5
if (sortArray==true)		2	3					5
{								
sort();		2						5
}								
while (first <= last)		3	3					6
{								
mid_pt = mid(first, last); ;								
if (x[mid_pt]==wanted)	1	2	3					6
{								
msg = "Using Binary Search, The Element Found at position " + (mid_pt +1)+","+"";		3	7					9
}								
if (x[mid_pt]>wanted)	1	2	3					6
{								
last = mid_pt - 1;			5					5
}								
Else								
{								
first = mid_pt + 1;			5					5
}								
}								
return msg;		2						2
}								
public static void load_data_from_file(String filename)		1						1
{								
int i=0;			6					6
File file=new File(filename);			3					3
Scanner my_data = null;		2						2
try								
{								
my_data = new Scanner(file);			3	2				5
while(my_data.hasNextLine())		3		2				5
{								
i=i+1;			11					11
int[] tempReDim1 = new int[i];			6					6
if (x != null)	1	2	2			1		5
System.arraycopy(x, 0, tempReDim1, 0,	2	4	4					10

Math.min(x.length, tempReDim1.length));								
x = tempReDim1;	1		2					3
x[i-1] =Integer.valueOf(my_data.nextLine());	1	4	7					11
System.out.println(x[i-1]);	1	2	6					9
}								
my_data.close();		2						2
}								
catch(FileNotFound Exception e)				2				2
{								
System.out.println(" The file: " + file + " does not exist.");		6	2					8
System.exit(0);		2	1					3
}								
}								
private static void sort()	1							1
{								
int temp;			1					1
for (int i = 0; i < x.length; i++)	1	3	15					19
{								
for (int j = i; j < x.length; j++)	1	5	19					25
{								
if (x[j] < x[i])	2	2	9					13
{								
temp = x[i];	1		6					7
x[i] = x[j];	2		9					11
x[j] = temp;	1		6					7
}								
}								
}								
}								
private static int mid(int left, int right)	1							1
{								
if (left <= right)		2	3					5
{								
if ((left + right) % 2 == 0)		2	7					9
return (left + right - 1)/2;			7					7
Else								
return (left + right) / 2;			5					5
}								
Else								
{								
return -1;			2					2
}								
}								
}								
import java.io.BufferedReader;								
import java.io.InputStreamReader;								
public final class TestProgram {								
static int[] a = new int[1];	1		2					3
/**								
* @param args								
*/								
public static void main(String[] args) throws Exception		2		2				4
{								
int key=1;			2					2
String filename="C:\\Users\\Tolu\\workspace\\tolu\\src\\Codes.txt";		1	1					2
BinarySearch bs= new BinarySearch();		2	1					3
bs.load_data_from_file(filename);				2				2
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));		1						1
System.out.print("\n search: ");		1	1					2
String wanted =			2					2

in.readLine();								
System.out.println(bs.Search(Integer.valu eOf(wanted), false));			1	4				5
}								
}								

i. MCM = CIclass +CDclass + Cprocedural From the table CIclass = 0
CDclass =
2+18+15+6+11+9+11+7+7+5+10+11+6+15
+14+9+4+8+5+19+25+13+7+11
+7+6+9+7+5+2 = 284
Cprocedural = 7+4+5+3+7 =26
MCM =0 +284 +26= 310

ii. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/310) *10,000 = 2,682.5806

Table 9: Python Linear Searching Demonstration

	Att	Str	Var	Obj	MA	AM	Cohesio	Compl ex
import time					2	1		
a = ["0"]	1							1
class ClassOfSearchAlgos:								
def linear_search(self, item_list, wanted):								
fl = False			5					5
msg = "Using Linear Search, The Element Found At position "		1	1					2
for i in range(len(item_list)):		7	5					12
if int(wanted) == int(item_list[i]):		6	6					12
msg = msg + str(i) + "," + " "		4	6					10
fl = True			5					5
if fl == False : msg = "Using Linear Search, The Element Not Found"		3	6					9
return msg			1					1
def load_data_from_file(f		1						1
ilename):								
i = 0			5					5
with open(filename) as fp:			1	2				3
for line in iter(fp.readline, ''):		7	1					8
a.append("0")	1	2	1					4
i = i + 1			9					9
a[i] = line	1		5					6
i = i + 1			9					9
a[i] = line	1		5					6
print(a[i])	1	2	4					7
load_data_from_file("C:\EclipseWorkspaces\csse120\Tolu\codes.txt")		3						3
wanted = input("search: ")		3	1					4
x = ClassOfSearchAlgos()				2				2
print (x.linear_search(a, wanted))	1	4	1					6

i. MCM = CIclass +CDclass + Cprocedural From the table CIclass = 0
CDclass =
5+2+12+12+10+5+9+1+1+15+3+8+4+9+6+
7 =99
Cprocedural = 1+3+4+2+6 =
16 MCM =0 +99 +16 =115

ii. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/115) *10,000 = 7,231.3043

Table 10: Python Binary Search Demonstration

	Att	Str	Var	Obj	MA	AM	Cohesion	Complex i
import time					2	1	2	
a = ["0"]	1							1
def load_data_from_file(f ilename):		1						1
i = 0			5					5
with open(filename) as fp:			1	2				3
for line in		7	1					8

iter(fp.readline, ''):								
a.append("0")	1	2	1					4
i = i + 1			9					9
a[i] = line	1		5					6
print(a[i])	1	2	4					7
class ClassOfSearchAlgos:								
def binary_search(self, item_list, wanted):								
msg = "Element Not Found"		1	1					2
first = 0			2					2
last = len(item_list)		2	2					4
while first <= last:		3	2					5
mid = int((first + last) / 2)		2	4					6
if int(item_list[mid]) == int(wanted):		6	3					9
msg = "Using Binary Search, The Element Found at position " + str(mid) + "," + " "		3	2					5
return msg			1					1
if int(item_list[mid]) > int(wanted) :		6	3					9
last = mid - 1			3					3
else:								
first = mid + 1			3					3
msg = "Using Binary Search, The Element Not Found "		1	1					2
return msg			1					1
load_data_from_file("C:\EclipseWorkspaces\csse120\Tolu\codes.txt")		3						3
wanted = input("search: ")		3	1					4
x = ClassOfSearchAlgos()				2				2
print (x.linear_search(a, wanted))	1	4	1					6
print (x.binary_search(a, wanted))	1	4	1					6

i. *MCM = CIclass +CDclass + Cprocedural*
From the table CIclass = 0
CDclass = 2+2+4+5+6+9+5+1+9+3+3+2+1= 52
Cprocedural =
1+1+5+3+8+4+9+6+7+3+4+2+6+6 = 65
MCM =0 + 52+ 65 = 117

ii. *FPA = UFP * VAF*
= 9 x 9.24 = 83.16

iii. *CQM = (FPA/MCM) * 10,000*
= (83.16/117) *10,000 = 7,107.6923

4. Results and Discussion

For empirical validation of CQM, six searching algorithms codes were analyzed. Each code contained both object oriented and procedural parts. Since they include different structures, therefore they contain most of the characteristics of a system required for the validation of the modified metric. Complexity values for each of the codes were summarized in Table 10.

Table 11: Complexity Values for C++, Java and Python in Searching Algorithms

	C++ LS	C++ BS	Java LS	Java BS	Python LS	Python BS
MCM	149	266	193	310	115	117
FPM	83.2	83.2	83.16	83.2	83.2	83.2
CQM	5581.2	3126.3	4308.8	2682.6	7231.3	7107.7

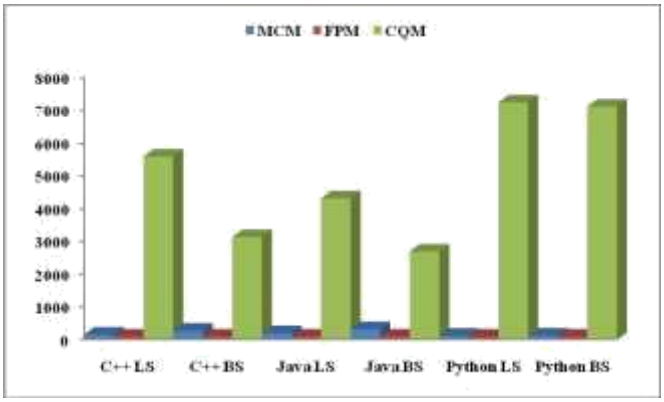


Figure 1: Relative Graph

4.2 Discussion

The variables, meaningful named and arbitrary named, basic control structures, for example sequence, branch, iteration and function call are directly related to complexity of a code. If a structure does not contain any variable, operator and constant, then its PCCM value becomes zero. In all the codes used for validation, no class was inherited, so the complexity for the inherited class is zero throughout.

It can be observed from the Table and the graph, that the complexity values realized showed that CQM are higher than those FPA and MCM. The higher the CQM the better the code is. Increment of MCM decreases CQM. This means when MCM is constant, higher level of functionality increases the code quality. Higher FP indicates more functionality and more merits. Similarly, when FP is constant, higher level of cognitive complexity decreases the code quality. Higher MCM indicates less comprehensibility, and more difficult code maintenance. The most desired combination is higher FP and lower MCM; the efficient way of programming the above specified class diagram is writing the code in Python language while the least efficient language is Java for this example.

It should be noted that these number do not claim that one language is better than the other one. Each language has its own merit. For one requirement, a specific language may have an advantage of efficiency, for another requirement, other language may be more suitable. The above program is very simple. For this reason, even though being much older C++ may have more advantages than Java. If a big program was under consideration, Java might have much higher CQM value than C++ due to being a more modern language.

Conclusion

In this study a complexity metric is proposed to include the factors that affect the cognitive complexity of a code. The proposed metric is formulated to manifest the factors of the procedural complexity and OO complexity characteristics. The metric was applied to codes written in procedural language and OO language for MCM since multi-paradigm encompasses both of them. The metric was extending to calculate FPA so that both cognitive and functional complexity aspects of a code could be calculated. By MCM, the code comprehensibility is measured, while the functionality is measured by FPA.

After combining MCM and FPA in one metric, CQM was proposed. From the empirical validation it is obvious that CQM has a platform-independent structure.. However, each language having its own programming merit, The CQM value does not contain an upper boundary, for there is no limit in the quality of a software system. Briefly, the proposed metric is useful especially in comparing software products which are developed to accomplish more or less the same tasks. Small size companies usually develop software products of the same category. Thus, most of their products are similar to each other. The proposed metric can be suggested to be used to improve the quality in such companies.

References

[1] Knutson, C.D., Budd, T.A., Vidos, H.: Multiparadigm Design of a Simple Relational Database,V.35(12), ACM, 2000.

[2] Coplien, J.O.: Multi-paradigm Design for C++, Addison Wesley, 1999.

[3] Miller, D.R.: „James O. Coplien Multi-Paradigm Design for C++“ Book review and commentary, January 2001, (last accessed 22.03.2010) Available at: http://www.inkdrop.net/docs/multiParadigm.pdf

[4] Python Programming Language – Official Website. (last accessed 18.02.2010). Available at: http://www.python.org/

[5] Roger S. P.: Software Engineering – A practitioner‟s approach, 6th Edition. McGraw-Hill (2005)

[6] Cohn, M.: Estimating with Use Case Points (last accessed. 23.02.2010) Available at:

http://www.methodsandtools.com/archive/archive.php?id=25

[7] ISO (2001): ISO/IEC 9126-1: Software Engineering -Product Quality - Part 1: Quality Model International Organization for Standardization, Geneva, Switzerland.

[8] Steve Klner (2013): Complexity Metrics and Difference Analysis for Better Application Management pg 1-33 Bill Hewlett, Hewlett-Packard.

[9] McCall, J.A., Richards, P.K. and Walters, G.F. (1977): Factors in Software Quality, I-III, Rome Air Development Centre, Italy

[10] Boëhm, B. (1978): Characteristics of software quality, 1 of TRW Series on Software Technology, North-Holland, Amsterdam, Holland.

[11] Yahya Tashtoush, Mohammed Al-Maolegi, Bassam Arkoki (2014): The Correlation among Software Complexity Metrics with Case Study: International Journal of advanced Computer Research (ISSN (print): 2249-7277 ISSN (online:7970) Volume 4, No. 2 Issue 15 June 2014.

[12] Alexander Serebrenik (2015): Software Metrics Complexity: Technishe University of Technology, Department of Mathematics an Computer Science (Tu/e) 2015: ISET/W&1.

[13] Francalanci, C. and Merlo, F. (2010): The Impact of Complexity on Software Design Quality and Costs: An Explanatory Empirical Analysis of Open Source Applications.

[14] Banker, R.D., Datar, S.M., and Zweig, D. (1989): Software Complexity and Maintainability CiteSeer Scientific Literature Digital Library and Search Engine.

[15] Software Technology Support Centre Software Estimation, Measurement, and Metrics (last accessed 16.03.2010) Available at: http://www.stsc.hill.af.mil/resources/tech_docs/gsam3/chap13.pdf

[16] Wang, Y., Shao, J.: A New Measure of Software Complexity Based on Cognitive Weights. Can. J. Elec. Computer Engineering, (2003) 69-74.

[17] Misra S., Akman, I: Unified Complexity Metric: A measure of Complexity, Proc. Of National Academy of Sciences Section A. (2010)

[18] Misra, S., Akman, I.: A Model for Measuring Cognitive Complexity of Software, Springer-Verlag Berlin Heidelberg 2008, pp. 879-886.