Enhancement to an OO metric-MCB measure\_2018

First author :- Second author:-

D . I. De Silva Dr. Dilshan

Email:dilshan.i@sliit.lk

Third author :- Fourth author :-

Name: M.A.P.M Perera Name: A.A.Akeel

IT number : IT20127800 IT number:IT20129958

Email:IT20127800@my.sliit.lk Email:IT20129958@my.sliit.lk

Fifth author:- Sixth author:-

Name: Jayawardhana S.A Name: S.W.L.P de Silva

IT number: IT20222772 IT number: IT20221850

Email:IT20222772@my.sliit.lk Email:IT20221850@my.sliit.lk

Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka.

***Abstract*--** Given the focal role that product improvement plays in the conveyance and use of data innovation, chiefs are progressively zeroing in on process improvement in the product advancement region. This request has prodded the arrangement of various new and additionally further developed ways to deal with programming improvement, with maybe the most noticeable being object-orientation (OO). For that this research paper illustrate the some important object-orientation factors and how those factors improving. Those factors are polymorphism, encapsulation, abstraction and method chaining. Throw out of this research paper illustrate the importance of those selected factors and how to demonstrate complexity calculation of those factors separately.

**Key words :** Object-oriented paradigm, complexity metrics, WCC measure

# **Introduction**

Many researchers propose several metrics to measure the complexity of object-oriented programs since 1980. These measurements are popular among software developers because of cost projection, reduce cost, manpower allocation, and program and programmer evaluation [1].

Knowing the level of complexity of the code is easier to predict the maintenance of a program when needed. And also, managing software complexity reduces the risk of defects in production. As well as it reduces maintenance costs because defects are identified earlier and prepared for it.

Some object-oriented metrics are Chidamber and Kemerers’ suite of six metrics [2], Chen and Luis’ suite of eight metrics [2], MOOD metrics [2], and Li’s suite of six metrics [2]. Class complexity measure [2], weighted class complexity measure [2], cognitive code complexity measure [2], CB measure [2], and MCB measure [2] are some of the OO metrics based on the cognitive aspect.

A variety of cognitive code-level (CCL) object-oriented (OO) complexity measures has been proposed in the literature [2], [3], [4], [5]. In addition, a survey on existing CCL OO complexity measures can be found in [6].

The next section discussed about…

**Polymorphism**

The complexity that arises due to the usage of polymorphism is considered by assigning a constant value to the total size (S) value. Zero for the method in the base class, value one for the overridden method, which is at the first derived class, and two for the overridden method at the next derived class. Similarly, the total size (S) value allocated for the overridden method increased by one for each derived class. Figure 1 illustrates a sample program that demonstrates the effect of polymorphism on the complexity of a program. Table 1 illustrates how the complexity calculates for a program using MCB was given in Figure 4. Table 2 illustrates how the complexity calculates according to the polymorphism given in Figure 1.

Graphical user interface, text, application

Description automatically generated

*Figure 1: Sample Program to demonstrate the effect of polymorphism*

*Table 1: Complexity Calculation of Program according to the MCB*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | class Animal{ |  |  |  |  |  |  |  |
| 2 | public void eat(){ | void, eat() | 2 | 0 | 0 | 1 | 1 | 2 |
| 3 | System.out.println("Eating...."); | System, . , out, . , println(), “Eating” | 6 | 0 | 0 | 1 | 1 | 6 |
| 4 | } |  |  |  |  |  |  |  |
| 5 | } |  |  |  |  |  |  |  |
| 6 | class Dog extends Animal{ |  |  |  |  |  |  |  |
| 7 | public void eat(){ | void, eat() | 2 | 0 | 0 | 2 | 2 | 4 |
| 8 | System.out.println("Dog is eating...."); | System, . , out, . , println(), “Dog is eating....” | 6 | 0 | 0 | 2 | 2 | 12 |
| 9 | } |  |  |  |  |  |  |  |
| 10 | } |  |  |  |  |  |  |  |
| 11 | class Cat extends Animal{ |  |  |  |  |  |  |  |
| 12 | public void eat(){ | void, eat() | 2 | 0 | 0 | 2 | 2 | 4 |
| 13 | System.out.println("Cat is eating...."); | System, . , out, . , println(), “Cat is eating....” | 6 | 0 | 0 | 2 | 2 | 12 |
| 14 | } |  |  |  |  |  |  |  |
| 15 | } |  |  |  |  |  |  |  |
| 16 | class AnimalTest{ |  |  |  |  |  |  |  |
| 17 | public static void main(String args[]){ | void, main() | 2 | 0 | 0 | 1 | 1 | 2 |
| 18 | Dog dog = new Animal(); | Dog, dog, =, new, Animal | 5 | 0 | 0 | 1 | 1 | 5 |
| 19 | dog.eat(); | Dog, ., eat() | 3 | 0 | 0 | 1 | 1 | 3 |
| 20 | Cat cat = new Animal(); | Cat, cat, =, new, Animal() | 5 | 0 | 0 | 1 | 1 | 5 |
| 21 | cat.eat(); | Cat, ., eat() | 3 | 0 | 0 | 1 | 1 | 3 |
| 22 | } |  |  |  |  |  |  |  |
| 23 | } |  |  |  |  |  |  |  |
|  |  | WCC value |  |  |  |  |  | 58 |

*Table 2: Complexity Calculation of the Program with polymorphism factor*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | class Animal{ |  |  |  |  |  |  |  |
| 2 | public void eat(){ | void, eat() | 2 | 0 | 0 | 1 | 1 | 2 |
| 3 | System.out.println("Eating...."); | System, . , out, . , println(), “Eating” | 6 | 0 | 0 | 1 | 1 | 6 |
| 4 | } |  |  |  |  |  |  |  |
| 5 | } |  |  |  |  |  |  |  |
| 6 | class Dog extends Animal{ |  |  |  |  |  |  |  |
| 7 | public void eat(){ | void, eat() | 3 | 0 | 0 | 2 | 2 | 6 |
| 8 | System.out.println("Dog is eating...."); | System, . , out, . , println(), “Dog is eating....” | 7 | 0 | 0 | 2 | 2 | 14 |
| 9 | } |  |  |  |  |  |  |  |
| 10 | } |  |  |  |  |  |  |  |
| 11 | class Cat extends Animal{ |  |  |  |  |  |  |  |
| 12 | public void eat(){ | void, eat() | 3 | 0 | 0 | 2 | 2 | 6 |
| 13 | System.out.println("Cat is eating...."); | System, . , out, . , println(), “Cat is eating....” | 7 | 0 | 0 | 2 | 2 | 14 |
| 14 | } |  |  |  |  |  |  |  |
| 15 | } |  |  |  |  |  |  |  |
| 16 | class AnimalTest{ |  |  |  |  |  |  |  |
| 17 | public static void main(String args[]){ | void, main() | 2 | 0 | 0 | 1 | 1 | 2 |
| 18 | Dog dog = new Animal(); | Dog, dog, =, new, Animal | 5 | 0 | 0 | 1 | 1 | 5 |
| 19 | dog.eat(); | Dog, ., eat() | 3 | 0 | 0 | 1 | 1 | 3 |
| 20 | Cat cat = new Animal(); | Cat, cat, =, new, Animal() | 5 | 0 | 0 | 1 | 1 | 5 |
| 21 | cat.eat(); | Cat, ., eat() | 3 | 0 | 0 | 1 | 1 | 3 |
| 22 | } |  |  |  |  |  |  |  |
| 23 | } |  |  |  |  |  |  |  |
|  |  | WCC value |  |  |  |  |  | 66 |

**3. Encapsulation and Its importance**

Encapsulation is defined as the ability to provide users with a well-defined interface to a set of functions in a way which hides their internal workings. The method of hiding details of a class is called abstraction[2].

Encapsulation is worried about the bundling of information and conduct to address a single element. To appropriately get to the nature of embodiment human comprehension is required what's more, the plan likewise should be completely appreciated. There are anyway two perspectives of the encapsulation that can be evaluated naturally. The principal concerns the degree to which a solitary class addresses a solitary element called Class Unity and second connects with the perceivability of a class information called information perceivability[23].

Encapsulation is basic to building huge complex programming, which can be kept up with furthermore, expanded. Many examinations have shown that the best expense in programming isn't the underlying advancement, yet the very long time spent in keeping up with the product. Very much epitomized parts are far more straightforward to keep up with. Once programming is set up, another extraordinary cost is expanding its usefulness. Epitome serves to limit this gamble. In a very much planned program, each article ought to have a solitary area of obligation. That article presents an interface, which characterizes the administrations the item gives.

Some advantages of using encapsulation include, the main advantage is security of data[23].

* Encapsulation protects an object from unwanted access by clients.
* Makes the application easier to understand.
* Encapsulation allows access to a level without revealing the complex details below that level.
* It reduces human errors.
* Simplifies the maintenance of the application

The strategy for concealing subtleties of a class is called abstraction. Classes can contain private, protected and public members. Although all the items in a class are private by default, software programmers can change the entrance levels when required. Those method are[23],

* **Public**: All objects can access the data.
* **Protected**: A variable or method that is protected can be accessed by code within the same class, by any classes that are in the same package and by all sub-classes in the same or other packages.
* **Private**: When the private access modifier is applied to an attribute or method, it can only be accessed by code within the same class.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

*Figure 2: Sample Program to demonstrate the effect of encapsulate*

Figure 2 sample program use to demonstrate complexity calculation of the program with encapsulation factor. For that table one demonstrate the complexity calculation according to the MCB. To demonstrate the encapsulate factors, private and protected attributes and method to assigning additional value one for the total size(S) and assigning additional value two to the total size(S) for public methods. According to the that description table two illustrate how the complexity calculate according to the encapsulate factors.

*Table 3: Complexity Calculation of Program according to the MCB*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | **public** **class** Rectangle { |  |  |  |  |  |  |  |
| 2 | **protected** String name; | String | 1 | 0 | 0 | 1 | 1 | 1 |
| 3 | **private** **int** width; | Int | 1 | 0 | 0 | 1 | 1 | 1 |
| 4 | **private** **int** height; | Int | 1 | 0 | 0 | 1 | 1 | 1 |
| 5 | **public** Rectangle(String n, **int** w, **int** h) { | rectangle | 1 | 0 | 0 | 1 | 1 | 1 |
| 6 | name = n; | Name , = , n | 3 | 0 | 0 | 1 | 1 | 3 |
| 7 | width = w; | Width, =, w | 3 | 0 | 0 | 1 | 1 | 3 |
| 8 | height = h; | Height, =, h | 3 | 0 | 0 | 1 | 1 | 3 |
| 9 | } |  |  |  |  |  |  |  |
| 10 | **public** **void** print() { | Void, print() | 2 | 0 | 0 | 1 | 1 | 2 |
| 11 | System.***out***.println("\nThe name of shape is: " +name); | System, . , out , . , println() , \n , “The name of shape is:” , + , name | 9 | 0 | 0 | 1 | 1 | 9 |
| 12 | } |  |  |  |  |  |  |  |
| 13 | **private** **void** show() { | Void , show() | 2 | 0 | 0 | 1 | 1 | 2 |
| 14 | System.***out***.println("\nArea = width \* height "); | System , . ,out , . , println , \n , “Area = width \* height” | 7 | 0 | 0 | 1 | 1 | 7 |
| 15 | } |  |  |  |  |  |  |  |
| 16 | **public** **int** area() { | Int , area() | 2 | 0 | 0 | 1 | 1 | 2 |
| 17 | **return** width\*height; | width , \* , height | 3 | 0 | 0 | 1 | 1 | 3 |
| 18 | } |  |  |  |  |  |  |  |
| 19 | } |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |
| 21 | **public** **class** main { |  |  |  |  |  |  |  |
| 22 | **public** **static** **void** main(String[] args) { | Void , main() | 2 | 0 | 0 | 1 | 1 | 2 |
| 23 | Rectangle R = **new** Rectangle("Rectangle", 4 , 6); | Rectangle , R , = , new , Rectangle() | 5 | 0 | 0 | 1 | 1 | 5 |
|  | System.***out***.println("\nThe area of rectangle is: " + R.area()); | System, . , out, . , println() , "\nThe area of rectangle is: " , + , R , . , area() | 10 | 0 | 0 | 1 | 1 | 10 |
|  | } |  |  |  |  |  |  |  |
|  |  | WCC Value |  |  |  |  |  | 55 |

*Table 4: Complexity Calculation of the Program with Encapsulation factor*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | **public** **class** Rectangle { |  |  |  |  |  |  |  |
| 2 | **protected** String name; | String | 2 | 0 | 0 | 1 | 1 | 2 |
| 3 | **private** **int** width; | Int | 2 | 0 | 0 | 1 | 1 | 2 |
| 4 | **private** **int** height; | Int | 2 | 0 | 0 | 1 | 1 | 2 |
| 5 | **public** Rectangle(String n, **int** w, **int** h) { | rectangle | 3 | 0 | 0 | 1 | 1 | 3 |
| 6 | name = n; | Name , = , n | 3 | 0 | 0 | 1 | 1 | 3 |
| 7 | width = w; | Width, =, w | 3 | 0 | 0 | 1 | 1 | 3 |
| 8 | height = h; | Height, =, h | 3 | 0 | 0 | 1 | 1 | 3 |
| 9 | } |  |  |  |  |  |  |  |
| 10 | **public** **void** print() { | Void, print() | 4 | 0 | 0 | 1 | 1 | 4 |
| 11 | System.***out***.println("\nThe name of shape is: " +name); | System, . , out , println() , \n , “The name of shape is:” , + , name | 8 | 0 | 0 | 1 | 1 | 8 |
| 12 | } |  |  |  |  |  |  |  |
| 13 | **private** **void** show() { | Void , show() | 3 | 0 | 0 | 1 | 1 | 3 |
| 14 | System.***out***.println("\nArea = width \* height "); | System , . ,out , . , println , \n , “Area = width \* height” | 7 | 0 | 0 | 1 | 1 | 7 |
| 15 | } |  |  |  |  |  |  |  |
| 16 | **public** **int** area() { | Int , area() | 4 | 0 | 0 | 1 | 1 | 4 |
| 17 | **return** width\*height; | width , \* , height | 3 | 0 | 0 | 1 | 1 | 3 |
| 18 | } |  |  |  |  |  |  |  |
| 19 | } |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |
| 21 | **public** **class** main { |  |  |  |  |  |  |  |
| 22 | **public** **static** **void** main(String[] args) { | Void , main() | 4 | 0 | 0 | 1 | 1 | 4 |
| 23 | Rectangle R = **new** Rectangle("Rectangle", 4 , 6); | Rectangle , R , = , new , Rectangle() | 5 | 0 | 0 | 1 | 1 | 5 |
|  | System.***out***.println("\nThe area of rectangle is: " + R.area()); | System, . , out, . , println() , \n , “The area of rectangle is: " , + , R , . , area() | 11 | 0 | 0 | 1 | 1 | 11 |
|  | } |  |  |  |  |  |  |  |
|  |  | WCC Value |  |  |  |  |  | 67 |

**Abstraction**

Abstraction(Wa) is the most common way of concealing implementation details and show just fundamental information to the client. Abstraction can accomplished with by abstract classes or interfaces[11]. Abstract class can't used to make objects.

Abstract technique can utilized in an abstract class, doesn't have a body [11].The body give by sub class(inherited class).if we use abstraction it lessens the complexity. avoids from duplication of code .As well increment the reusability. This is what the abstraction factor considered in the new measurement means for the complexity of a program

1.Using Abstract classes

Weighted allocations

|  |  |
| --- | --- |
| Statements | Weight for each line |
| Abstract class weight | 0 |
| Sub class extends abstract class | 1 |
| Normal class doesn’t extend abstract class | 2 |

2.Using interfaces

Weighted allocations

|  |  |
| --- | --- |
| Statements | Weight for each line |
| The weightage of the interface | 0 |
| If class implements three interfaces or more than three interfaces | 1 |
| If class implements two interfaces or more than two interfaces | 2 |
| If class doesn’t have any interfaces | 3 |

* The equation to constructed to find total weights

∑ = summation for n number of classes and interfaces

**Object oriented complexity metric** = **( Wa +Wcs + Wv + Wi +Wio)**

**N**

**Wa-Weight due to Abstraction**

**Wcs-Weight due to control structures**

**Wv-Weight due to variables**

**Wi-Weight due to inheritance**

**Wio-Wight due to input outputs.**

**N-number of lines in code**

In Abstraction factor, we get it as Wa esteem. on the off chance that Abstract technique can utilize an abstract class. So it diminishes the complexity. we get weight as 0 on the off chance that it is an abstract class. we get 1 if it is a sub class broadening abstract class. We get weight as 2 assuming it is an ordinary class .Like that we appoint values for our new measurement utilizing abstraction.

In control structure factor, we get it as Wcs. There are more control structures some of them with allocate values are if-else = 1 , For Loop = 2, while Loop=3 .

In any case, better comprehension with settled levels then we can relegate esteems effectively and can diminish piece of complexity.

The Number of input output we get as Wio, In here count all System.out.println() statements. Here allocate 1 for all information sources and results articulations. Like that allocate 1 to all input/output proclamations for newly made measurement utilizing this variable.

In Measure by its variable element, we get it as Wv and it again isolated in to four sections, there are global variables(Wvs), local variables(Wvs),, primitive data types variables(Wpdt),, composite data type variables(cdts), we allocate values as, local variables and primitive data type variables as 1, global variables and composite data type variables as 2.

Utilizing these variable kinds we can comprehended program all the more obviously. Like that we appoint values utilizing Measure by its variables.

The Inheritance factor we get as Wi, It is instrument for communicating comparability among classes. At the point when the class acquire from another, that implies it can utilize its strategies and traits. Then we can lessen the source code and increment unwavering quality of code. In here we appoint values like statement inside the base class/root class= 0 , first determined class =1 and nth inferred class= n . Like that we can relegate values to inheritance to newly metric.

Like vise we get above factors final value and we get summation of that because we get complexity using these all factors so that, we add Wa , Wi, Wio, Wcs and Wv together.

**(Wa + Wi +Wio +Wcs+ Wv)**

Finally we get summation of four factors. So that it increase the complexity of our code, So we divide it by number of lines in our code .Then it reduce complexity of our code.

**(Wa + Wi +Wio +Wcs+ Wv)**

**N**

Sample Program to Calculate complexity measure

**public** **abstract** **class** Employee {

**private** String name;

**private** String email;

**private** String Employeeid;

**public** **abstract** **double** Salary();

**public** **abstract** **double** getEmployeedetails();

**public** Student(String name,String email,String Employeeid) {

System.***out***.println("Collect Salary");

**this**.name=name;

**this**.email=email;

**this**.Employeeid= Employeeid;

}

**public** String getName() {

**return** name;

}

**public** **void** Salary(**int** Salary1,**int** Salary2) {

System.***out***.println("There are two Salary");

}

**public** **void** Salary(**int** Salary1) {

System.***out***.println("There are one Salary");

}

}

**public** **class** Salary **extends** Employee {

**private** **double** Salary;

**public** Salary (String name, String email, String Employeeid,**double** Salary) {

**super**(name, email, Employeeid);

**this**. Salary = Salary;

}

@Override

**public** **double** Salary () {

System.***out***.println("Salary for :"+getName());

**return** 0;

}

**public** **double** getEmployeedetails()

{

System.***out***.println("name :"+getName());

**return** 0;

}

**public** **void** Salary (**int** Salary1,**int** Salary2) {

**int** sum= Salary1+ Salary2;

**if**(sum>100000) {

System.***out***.println("total Salary more than 100000 : "+sum);

}**else** {

System.***out***.println("total Salary less than 100000 : "+sum);}

}

}

Calculation of complexity of the program

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Line no | Program statements | Wa | Wc | Wcs | Wi | Wio | Wv | | |
| wvs | wpdt | wcdt |
| 1 | **public** **abstract** **class** Employee { | 0 |  | 0 | 0 | 0 |  |  |  |
| 2 | **private** String name; |  |  | 0 |  |  | 2 | 1 |  |
| 3 | **private** String email; |  |  | 0 |  |  | 2 | 1 |  |
| 4 | **private** String Employeeid; |  |  | 0 |  |  | 2 | 1 |  |
| 5 | **public** **abstract** **double** Salary (); |  |  | 0 |  |  |  |  |  |
| 6 | **public** **abstract** **double** getEmployeedetails(); |  |  | 0 |  |  |  |  |  |
| 7 | **public** Employee(String name,String email,String Employeeid) { |  |  | 0 |  |  |  |  |  |
| 8 | System.***out***.println("Collect Salary "); |  |  |  | 1 |  |  |  |  |
| 9 | **this**.name=name; |  | 1 |  |  |  |  |  |  |
| 10 | **this**.email=email; |  | 1 |  |  |  |  |  |  |
| 11 | **this**. Employeeid = Employeeid; |  | 1 |  |  |  |  |  |  |
| 12 | } |  |  |  |  |  |  |  |  |
| 13 | **public** String getName() { |  |  |  |  |  |  |  |  |
| 14 | **return** name; |  |  |  |  |  |  |  |  |
| 15 | } |  |  |  |  |  |  |  |  |
| 16 | **public** **void** Salary (**int** Salary1,**int** Salary2) { |  |  |  |  |  |  |  |  |
| 17 | System.***out***.println("There are two Salary "); |  |  |  | 1 |  |  |  |  |
| 18 | } |  |  |  |  |  |  |  |  |
| 19 | **public** **void** Salary (**int** Salary1) { |  |  |  |  |  |  |  |  |
| 20 | System.***out***.println("There are one Salary "); |  |  |  | 1 |  |  |  |  |
| 21 | } |  |  |  |  |  |  |  |  |
| 22 | } |  |  |  |  |  |  |  |  |
| 23 | **public** **class** Salary **extends** Employee { | 1 |  |  |  |  |  |  |  |
| 24 | **private** **double** Salary; |  |  |  |  |  |  |  |  |
| 25 | **public** Salary(String name, String email, String Employeeid,**double** Salary) { |  |  |  |  |  |  |  |  |
| 26 | **super**(name, email, Employeeid); |  |  |  |  |  |  |  |  |
| 27 | **this**. Salary = Salary; |  | 1 |  |  |  |  |  |  |
| 28 | } |  |  |  |  |  |  |  |  |
| 29 | **public** **double** Salary () { |  |  |  |  |  |  |  |  |
| 30 | System.***out***.println("Salary for :"+getName()); |  |  |  | 1 |  |  |  |  |
| 31 | **return** 0; |  |  |  |  |  |  |  |  |
| 32 | } |  |  |  |  |  |  |  |  |
| 33 | **public** **double** getEmployeedetails() |  |  |  |  |  |  |  |  |
| 34 | { |  |  |  |  |  |  |  |  |
| 35 | System.***out***.println("name :"+getName()+" Email: "+getEmail()+" studentid : "+getEmployeeid()); |  |  |  | 1 |  |  |  |  |
| 36 | **return** 0; |  |  |  |  |  |  |  |  |
| 37 | } |  |  |  |  |  |  |  |  |
| 38 | **public** **void** Salary (**int** Salary1,**int** Salary2) { |  |  |  |  |  |  |  |  |
| 39 | **int** sum= Salary1+ Salary2; |  |  |  |  |  | 1 | 1 |  |
| 40 | **if**(sum>100000) { |  |  | 1 |  |  |  |  |  |
| 41 | System.***out***.println("total Salary more than 100000 : "+sum); |  |  | 1 |  |  |  |  |  |
|  | }**else** { |  |  |  |  |  |  |  |  |
| 42 | System.***out***.println("total Salary less than 100000 : "+sum);} |  |  | 1 | 1 |  |  |  |  |
| 43 | } |  |  |  |  |  |  |  |  |
| 44 | } |  |  |  |  |  |  |  |  |
|  | Total | 1 | 4 | 3 | 6 | 0 | 7 | 4 | 0 |
|  | Total weight | 25 |

**Object oriented complexity metric** = **(Wa +Wcs + Wv + Wi +Wio)**

**N**

= 25/44

= 0.5681

**Methods Chaining**

**Method chaining** is a common syntax to invoke multiple methods calls in OOPs. Each method in chaining returns an object. It violates the need of intermediate variables.In other words, the method chaining can be defined as if we have an object and we call methods on that object one after another is called method chaining. For example,

1. obj.method1().method2().method3();

In the above statement, we have an object (obj) and calling method1() then method2(), after that the method3(). So, calling or invoking methods one after another is known as method chaining.[16]

Figure 1 illustrates a sample program that demonstrates the effect of methods chaining on the complexity of a program. Table 5 illustrates how the complexity calculates for a program using MCB was given in Figure 3. Table 6 illustrates how the complexity calculates according to the methods chaining given in Figure 3.

Graphical user interface, text, application, email

Description automatically generated

Figure 3 Sample Program to demonstrate the effect of Methods Chaining

Table 5 Complexity Calculation of Program according to the MCB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | public class Student |  |  |  |  |  |  |  |
| 2 | { |  |  |  |  |  |  |  |
| 3 | private int id; | int, id | 2 | 0 | 0 | 1 | 1 | 2 |
| 4 | private String name; | String, name | 2 | 0 | 0 | 1 | 1 | 2 |
| 5 | private int age; | int, age | 2 | 0 | 0 | 1 | 1 | 2 |
| 6 | private int std; | int, std | 2 | 0 | 0 | 1 | 1 | 2 |
| 7 | public Student setId(int id) | setld | 1 | 0 | 0 | 1 | 1 | 1 |
| 8 | { |  |  |  |  |  |  |  |
| 9 | this.id = id; | this, ., id, =, id | 5 | 0 | 0 | 1 | 1 | 5 |
| 10 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 11 | } |  |  |  |  |  |  |  |
| 12 | public Student setName(String name) | setName | 1 | 0 | 0 | 1 | 1 | 1 |
| 13 | { |  |  |  |  |  |  |  |
| 14 | this.name = name; | this, ., name, =, name | 5 | 0 | 0 | 1 | 1 | 5 |
| 15 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 16 | } |  |  |  |  |  |  |  |
| 17 | public Student setAge(int age) | setAge | 1 | 0 | 0 | 1 | 1 | 1 |
| 18 | { |  |  |  |  |  |  |  |
| 19 | this.age = age; | this, ., age, =, age | 5 | 0 | 0 | 1 | 1 | 5 |
| 20 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 21 | } |  |  |  |  |  |  |  |
| 22 | public Student setStd(int std) | setStd | 1 | 0 | 0 | 1 | 1 | 1 |
| 23 | { |  |  |  |  |  |  |  |
| 24 | this.std = std; | this, ., std, =, std | 5 | 0 | 0 | 1 | 1 | 5 |
| 25 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 26 | } |  |  |  |  |  |  |  |
| 27 | public void detail() | Void, detail | 2 | 0 | 0 | 1 | 1 | 2 |
| 28 | { |  |  |  |  |  |  |  |
| 29 | System.out.println("Student Detail is:\n"); | System, ., out, ., println() | 5 | 0 | 0 | 1 | 1 | 5 |
| 30 | System.out.println("Id: "+id+ "\nName: "+name+"\nAge: "+age+ "\nStandard: "+std); | System, ., out, ., println() | 5 | 0 | 0 | 1 | 1 | 5 |
| 31 | } |  |  |  |  |  |  |  |
| 32 | public static void main(String args[]) | void, main() | 2 | 0 | 0 | 1 | 1 | 2 |
| 33 | { |  |  |  |  |  |  |  |
| 34 | Student student = new Student(); | Student, student, =, new, Student() | 5 | 0 | 0 | 1 | 1 | 5 |
| 35 | student.setId(1123).setName("Amal Siriwardena").setAge(15).setStd(8).detail(); | Student,.,setId(),.,setName,.,setAge ,.,setStd,.,detail() | 11 | 0 | 0 | 1 | 1 | 11 |
| 36 | } |  |  |  |  |  |  |  |
| 37 | } |  |  |  |  |  |  |  |
|  |  | WCC value |  |  |  |  |  | 66 |

Table 6 Complexity Calculation of the Program with Methods Chaining factor

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Program Statements | Tokens | S | Wc | Wn | Wi | Wt | WC |
| 1 | public class Student |  |  |  |  |  |  |  |
| 2 | { |  |  |  |  |  |  |  |
| 3 | private int id; | int, id | 3 | 0 | 0 | 1 | 1 | 3 |
| 4 | private String name; | String, name | 3 | 0 | 0 | 1 | 1 | 3 |
| 5 | private int age; | int, age | 3 | 0 | 0 | 1 | 1 | 3 |
| 6 | private int std; | int, std | 3 | 0 | 0 | 1 | 1 | 3 |
| 7 | public Student setId(int id) | setld | 2 | 0 | 0 | 1 | 1 | 2 |
| 8 | { |  |  |  |  |  |  |  |
| 9 | this.id = id; | this, ., id, =, id | 5 | 0 | 0 | 1 | 1 | 5 |
| 10 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 11 | } |  |  |  |  |  |  |  |
| 12 | public Student setName(String name) | setName | 2 | 0 | 0 | 1 | 1 | 2 |
| 13 | { |  |  |  |  |  |  |  |
| 14 | this.name = name; | this, ., name, =, name | 5 | 0 | 0 | 1 | 1 | 5 |
| 15 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 16 | } |  |  |  |  |  |  |  |
| 17 | public Student setAge(int age) | setAge | 2 | 0 | 0 | 1 | 1 | 2 |
| 18 | { |  |  |  |  |  |  |  |
| 19 | this.age = age; | this, ., age, =, age | 5 | 0 | 0 | 1 | 1 | 5 |
| 20 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 21 | } |  |  |  |  |  |  |  |
| 22 | public Student setStd(int std) | setStd | 2 | 0 | 0 | 1 | 1 | 2 |
| 23 | { |  |  |  |  |  |  |  |
| 24 | this.std = std; | this, ., std, =, std | 5 | 0 | 0 | 1 | 1 | 5 |
| 25 | return this; | this | 1 | 0 | 0 | 1 | 1 | 1 |
| 26 | } |  |  |  |  |  |  |  |
| 27 | public void detail() | Void, detail | 3 | 0 | 0 | 1 | 1 | 3 |
| 28 | { |  |  |  |  |  |  |  |
| 29 | System.out.println("Student Detail is:\n"); | System, ., out, ., println() | 5 | 0 | 0 | 1 | 1 | 5 |
| 30 | System.out.println("Id: "+id+ "\nName: "+name+"\nAge: "+age+ "\nStandard: "+std); | System, ., out, ., println() | 5 | 0 | 0 | 1 | 1 | 5 |
| 31 | } |  |  |  |  |  |  |  |
| 32 | public static void main(String args[]) | void, main() | 3 | 0 | 0 | 1 | 1 | 3 |
| 33 | { |  |  |  |  |  |  |  |
| 34 | Student student = new Student(); | Student, student, =, new, Student() | 5 | 0 | 0 | 1 | 1 | 5 |
| 35 | student.setId(1123).setName("Amal Siriwardena").setAge(15).setStd(8).detail(); | Student,.,setId(),.,setName,.,setAge ,.,setStd,.,detail() | 11 | 0 | 0 | 1 | 1 | 11 |
| 36 | } |  |  |  |  |  |  |  |
| 37 | } |  |  |  |  |  |  |  |
|  |  | WCC value |  |  |  |  |  | 76 |

**Conclusion**

Complexity measurements can be utilized to anticipate the nature of a product framework. A bunch of object oriented factors are utilized to present the new object oriented complexity metric. This research incorporates the clarifications of what each component considered in the new measurement mean for the complexity of a program and the clarifications of how the new measurement catches the complexity presented by each element. Every one of the four factors were depicted thinking about significant elements.

Then the research incorporates the new object oriented complexity metric which was presented utilizing those arrangement of factors and the clarification of how the new measurement was produced thinking about those variables. Four sample programs are utilized to assess the complexity level utilizing the newly presented complexity metric. The newly proposed measurement is determined utilizing source codes; subsequently, this measurement can be a decent indicator of reusability, understandability, testing endeavors and future support endeavors. This recently acquainted measurement can be utilized with effectively foster

an complexity measuring tool.

**Reference**

|  |
| --- |
| [1] JOSEPH K. KEARNEY, ROBERT L. SEDLMEYER, WILLIAM B. THOMPSON,, "SOFTWARE COMPLEXITY MEASUREMENT," 11 1986. |
| [2] D. I. De Silva, N. Kodagoda,S. R. Kodituwakku, A. J. Pinidiyaarachchi , "Enhancements to an OO Metric: CB Measure," *Journal of Software,* vol. 13, 2018. |
| [3] D. S. Kushwaha and A. K. Misra, "Cognitive information complexity measure of object-oriented software: a practitioner's approach," in *Fifth WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, Madrid, Spain, 2006. |
| [4] S. Misra and K. I. Akman, "Weighted class complexity: a measure of complexity for object oriented system," *Journal of Information Science and Engineering,* vol. 24, 2008. |
| [5] S. Misra, I Akman, and M Koyuncu, "An inheritance complexity metric for object-oriented code: a cognitive approach," *Indian Academy of Sciences,* vol. 36, no. 3, pp. 317-337, 2011. |
| [6] Dilshan I. De Silva, Saluka R. Kodituwakku, and Amalka J. Pinidiyaarachchi, "An Analytical Study of Cognitive Code-Level Object-Oriented Complexity Measures," *International Journal of Computer Applications,* vol. 183, no. 45, pp. 8-14, 2021. |

[7] Misra, S. and Akman, I., 2008. Weighted class complexity: a measure of complexity for object oriented system. *Journal of Information Science and Engineering*, *24*, pp.1689-1708.

[8] Misra, S. and Adewumi, A., 2018. Object-oriented cognitive complexity measures: An analysis. *Intelligent systems: Concepts, Methodologies, Tools, and Applications*, pp.1324-1347.

[9] De Silva, D.I., Kodagoda, N., Kodituwakku, S.R. and Pinidiyaarachchi, A.J., 2015, September. Limitations of an object-oriented metric: Weighted complexity measure. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 698-701). IEEE.

[10] Misra, S. and Kiliç, H., 2007. WEAK MEASUREMENT THEORY AND MODIFIED COGNITIVE COMPLEXITY MEASURE.

[11] Thamburaj, T.F. and Aloysius, A., 2015. Cognitive Weighted Polymorphism Factor: A Comprehension Augmented Complexity Metric. *International Journal of Computer and Information Engineering*, *9*(11), pp.2342-2347.

[12] Abreu, F.B. and Carapuça, R., 1994, October. Object-oriented software engineering: Measuring and controlling the development process. In *Proceedings of the 4th international conference on software quality* (Vol. 186).

[13] “Cyclomatic complexity - Wikipedia”. [Online]. Available: https://en.wikipedia.org/wiki/Cyclomatic\_complexity. [Accessed: 16-Apr.-2022].

[14] “Understanding Cyclomatic Complexity & its Importance in Code ...”. [Online]. Available: https://blog.thedigitalgroup.com/understanding-cyclomatic-complexity-and-its-importance-in-code-analysis-metrics. [Accessed: 16-Apr.-2022].

[15] “Cyclomatic Complexity - GeeksforGeeks”. [Online]. Available: https://www.geeksforgeeks.org/cyclomatic-complexity/. [Accessed: 16-Apr.-2022]

[16] “Method Chaining In Java with Examples - GeeksforGeeks”. [Online]. Available: https://www.geeksforgeeks.org/method-chaining-in-java-with-examples/. [Accessed: 16-Apr.-2022].

[17] ‘”What is Abstraction in OOPs? Java Abstract Class & Method”. [Online] Available:<https://www.guru99.com/java-data-abstraction.html>.[Accessed: 16-Apr-2022].

[18] “Method chaining in java?” [Online].Available:https://www.linkdin.com/pulse/method-chaining-java-mehdi-ali-soltani.[Accessed: 16-Apr-2022].

[19] “Applicability of three complexity metrics – IEEE Xplore”. [Online] Available :https://ieeexplore.ieee.org/document/6421409?arnumber=6421409. [Accessed: 16-Apr-2022]

[20] “(PDF) Linear mass scans in quadrupole ion traps using the inverse ...”. [Online]. Available: https://www.academia.edu/33463097/Linear\_mass\_scans\_in\_quadrupole\_ion\_traps\_using\_the\_inverse\_Mathieu\_q\_scan. [Accessed:16-Apr.-2022].

[21] “Jamie Cullum Croons in ’The Halcyon’ WWII Drama: Watch Behind ...”. [Online]. Available: https://www.billboard.com/music/music-news/jamie-cullum-the-halcyon-behind-the-scenes-video-watch-7973749/. [Accessed: 16-Apr.-2022].

[22] “COGNITIVE COMPLEXITY”. [Online]. Available: https://www.sonarsource.com/docs/CognitiveComplexity.pdf. [Accessed: 16-Apr.-2022].

[23] “Definition of Encapsulation in Computer Programming”. [Online]. Available: https://www.thoughtco.com/definition-of-encapsulation-958068. [Accessed: 16-Apr.-2022].