Journal of Science and Technology
ISSN: 2456-5660 Volume 5, Issue 2, March-April 2020, PP 54-59
www.jst.org.in

Impact of Dynamic Polymorphism on Quality of a System

Manju[1], Pradeep Kumar Bhatia[2]
[1](Research Scholar, Guru Jambheshwar University of Science and Technology, Hisar,India)
[2](Professor, Guru Jambheshwar University of Science and Technology, Hisar,India)

Abstract: *Polymorphism has a great impact on the quality of a software .It is mainly of two types, compile time or static polymorphism based on method overloading and runtime or dynamic based on method overriding. If a class is highly polymorphic i.e. reusability of that class is high. If the class is less polymorphic then reusability of that class is low. There are only few metrics available in literature to find the polymorphism factor of a class. In this paper, we introduce a new metric to find polymorphism of a class at compile time and runtime and compare with the existing metric. The purposed metric is applied on 4 design patterns having different behaviour. Dynamic polymorphism is calculated with the help of purposed tool named DynaPoly implemented in AspectJ using aspect oriented programming in java. Eclipse platform is used to perform coding of tool using AspectJ After analysing the results, it is concluded that purposed metric plays a vital role to find reusability, hence quality of a system and gives better results than existing metrics.*

Keywords*: Polymorphism, quality, metric, reusability.*

Received on: 02-01-2020 Revised on: 25-02-2020 Published on: 09-04-2020

I. Introduction

In the OO paradigm, polymorphism mainly comes after the term inheritance. It allows us code sharing and reusing of code in a systematic way. Polymorphism means having the ability to takeseveral forms There are mainly two types of polymorphism exist in Object Oriented languages like C++,java, C# etc. First is compile time or static polymorphism.It can be achieved by function overloading or constructor overloading. Overloading means using the same name with different signature. Second is,runtime or dynamic polymorphism.It can be achieved by virtual functions in C++ or dynamic binding[4] of function in java,C# etc. Overrriding means a new definition is given by derived class to base class function. In this paper,we are concerned with static polymorphism that can be measured at compile time with the help of design patterns. With the help of design patterns, we can clearly measure number of overloading or overridden methods in a class.

In MOOD[2] metrics suite, one metric named POF is defined to find polymorphism factor of a system at compile time.POF(Polymorphism factor) is used to find polymorphism factor of a system based on number of overridden methods in class and number of descendants of a class. The definition[1] of metric is as follows:

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{I=1}^{TC}[M_n(C_i) \times DC(C_i)]}$$

Where Mo(Ci) is overriding Methods inclass Ci ,Mn(Ci) is new Methods inclass Ci , DC(Ci) is number of Descendants of Class Ci (derivedclasses) , TC is total number of Classes.Polymorphism arises from inheritance. Dynamic binding is a common message calltoone of severalclasses (in the samehierarchy)issupposed to reduce complexity andtoallow refinement of the class hierarchywithout side effects. On the other hand, if the value of polymorphic factor POF increases than complexity of program also increases and maintainability of the program decreases. Value of POF gives us a measure of to what extent system used polymorphism in its classes [6] i.e. to what extent code is reused as it is.

II. Proposed Metrics

There are many flaws in existing POF metric given by Abreu et al.[1] as explained below[3] in Figure. 1. It explain value of POF metric for subsystem S that is greater than 1 that is biggest flaw in POF metric.as value of overriding methods for class P ,Q and R is 1,2 and 2 respectively. Class P adds 2 new method and number of descendants for class P is 2,in the same way class Q adds 2 new methods and same for class R. Number of descendants for class Q and R is 0 for both.Therefore,

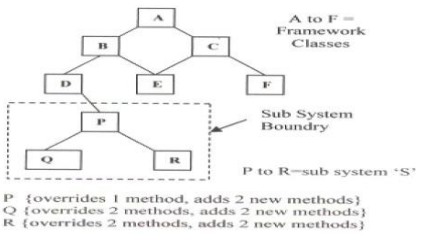

Figure. 1. Design of Subsystem S

POF for S=(1+2+2)/(2*2+2*0+2*0)=5/4>1

The purposed metric named Static Polymorphism has taken a count of both, method overloading and method overriding at compile time.It is a combination of two metric named CTP(Compile Time Polymorphism) and SPF(Static Polymorphism Factor). Definition of these metrics is explained below:

Figure. 1. Design of Subsystem S

CTP(Compile Time Polymorphism): It is ratio of total number of overloading methods in class to the total number of methods in class.

$$\mathrm{CTP(c)} = \frac{M_{OL(c)}}{M} \qquad (1)$$

Where M_{OL}(c) is the overloading methods in class and M is the total number of methods in class.

SPF (Static Polymorphism Factor): It is ratio of total number of overriding methods in class to the total number of methods in class.

$$\mathrm{SPF(c)} = \frac{M_{OR(c)}}{M} \qquad (2)$$

Where M_{OR}(c) is the overridden methods in class and M is the total number of methods in class.

SP(Static Polymorphism): It is defined as sum of CTP and SPF taken from equation (1) and (2).

$$\mathrm{SP(c) = CTP + SPF}$$

Where CTP is Compile Time Polymorphism and SPF is Static Polymorphism Factor.If the value of purposed metric is 1 then class is highly polymorphic and code is reused in a significant manner. If the value of SP is 0 then code is not reused at all. Therefore purposed metric plays a vital role to find reusability of class hence quality of a software system.

DPF (Dynamic Polymorphism Factor): It is ratio of total number of times overloading methods executed at runtime to the total number of times methods of a class executed at runtime of a class.

$$\mathrm{DPF(c)} = \frac{\sum_{i=1}^{m} OL_i}{\sum_{i=1}^{m} n_i} \qquad (3)$$

Where m is the total number of methods in class, OL_i is the number of times overloading method i executed at runtime, n_i is the number of times method i executed at runtime.

RTP (Run Time Polymorphism): It is ratio of total number of times overridden methods executed at runtime to the total number of times methods of a class executed at runtime of a class.

$$\mathrm{RTP(c)} = \frac{\sum_{i=1}^{m} OR_i}{\sum_{i=1}^{m} n_i} \qquad (4)$$

Where m is the total number of methods in class, OR_i is the number of times overridden method i executed at runtime, n_i is the number of times method i executed at runtime.

DP(Dynamic Polymorphism): It is defined as sum of DPF and RTP taken from equation (3) and (4).

$$\mathrm{DP(c) = DPF + RTP}$$

Where DPF is Dynamic Polymorphism Factor and RTP is Run Time Polymorphism. If the value of purposed metric is 1 then class is highly polymorphic and code is reused in a significant manner at runtime also. If the value of DP is 0 then code is not reused at all at runtime. Therefore purposed metric plays a vital role to find reusability of class at runtime hence quality of a software system.

III. Case Study

Consider a design pattern of system named C1[5]coded in java to calculate purposed metric. In Figure. 2. Class A has 5 methods and all the methods are overloading methods. Class B contains 1 method that is not overridden method. If you look at the method in class B,in first look it seems like overridden method but it is not because base class does not contain any definition for m method that is exactly same as in derived class. Class c contains 2 methods and both the methods are overridden methods. Class D is an individual class with 1 method.

```
Class A {
void m (int i);
void m (float f);
void m (int i , int f);
void p ();
```

```
void p (int x);
}
Class B  extends A
{ void m (); }
Class C  extends B
{ void m ();
void p (); }
Class D {
void m (); }
```

Dynamic Polymorphism is calculated with the help of purposed tool named DynaPolyimplemented in AspectJ[8] using aspect oriented programming in java. Eclipse platform is used to perform coding of tool using AspectJThere are some advantages of Aspect oriented approach over the other approaches to trace the events at runtime.[9]

1. The process of building a tracing or profiling frame work using aspect oriented approach is relatively simpler than any other approach.
2. This approach does not generate a large amount of data.
3. This approach produces results which are valid among all programme execution environment.
4. This approach is relatively in expensive and much more practical in nature.

From Figure. 2. Calculated values of purposed metrics are as following:

$M_o(A) = 5, M_{or}(A)=0, SP=5/5=1, DP=2/5=0.4$

$M_o(B) = 0, M_{or}(B)=0, SP=0/1=0, DP=0/1=0$

$M_o(C) = 0, M_{or}(C)=2, SP=2/2=1, DP=1/2=0.5$

$M_o(D) = 0, M_{or}(D)=0, SP=0/1=0, DP=0/0=0$

SP metric value is 1 for class A and C at compile time and 0.4 and 0.5 at runtime respectively. Working of DynaPoly is shown in Figure. 3.This shows that class A and C are highly polymorphic at compile time and less polymorphic at runtime. But in contrast class B and D has 0 values for both compile time and runtime.Therefore, class B and D does not contain any degree of polymorphism i.e. not considered as reusable classes.

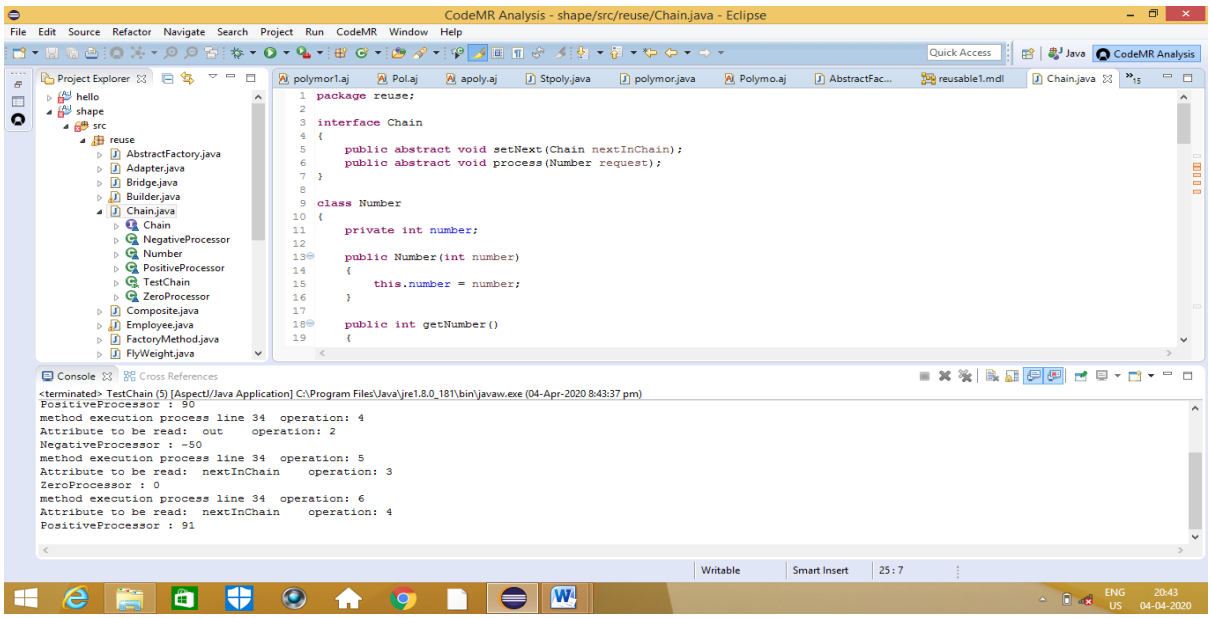

Figure. 2. Working of Purposed Tool DynaPoly

IV. Experimental Study

An experimental study is conducted on 4 Design pattern named P1,P2, P3 and P4 shown in Figure.4. All the 4 Design pattern[11] have different behaviour. As we know POF is a system based metric but our purposed metric is a class based metric. So to find the polymorphism of a system we can add the values of SPand DP of classes contained in system.

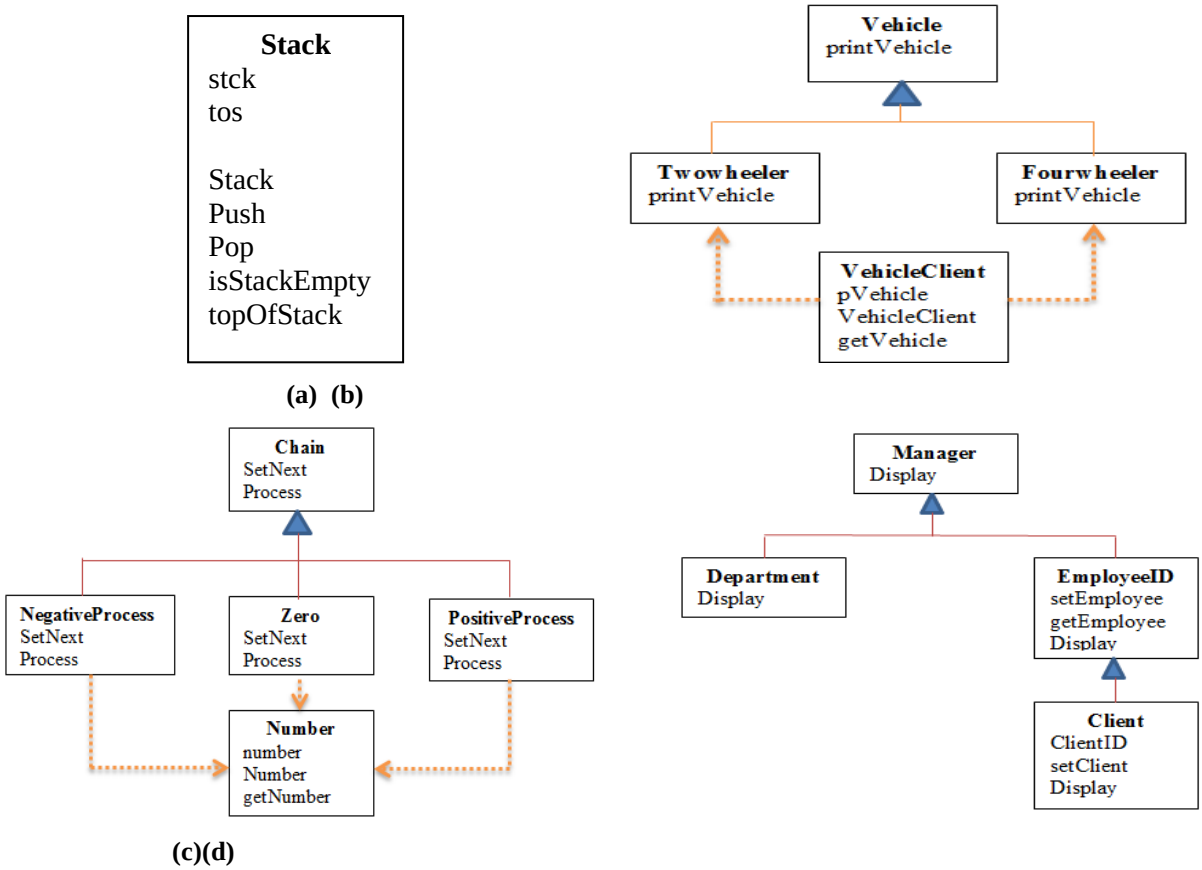

Figure. 3. (a) Design pattern P1 (b) Design Pattern P2 (c) Design Pattern P3 (d) Design Pattern P4

First design pattern P1, contains all the methods in 1 class so value of POF is 0 i.e. there is no polymorphism in P1 at compile and runtime both therefore value of SP and DP is 0 for P1.P2 contains 4 classes as one class as base class and 2 derived classes. Fourth class is individual class coupled with other 2 classes. Calculated value of POF for P2 is also 0 as there is single level inheritance exist and no descendants of derived classes. P3 contains 5 classes with one base class and 3 derived class.Fifth class is individual class coupled with other 3 classes.P4 contains 4 classes ,1 base class and 2 derived class. Third class is further extended by fourth class as shown in Figure.4. Value of POF for P3 is also 0 because single level of inheritance exist there. POF value of design pattern P4 is 0.5 or 50%. Calculated values of purposed metric are shown in Table 1.D_M is the number of times methods of a class executed at runtime.

Table 1. Purposed Metrics for various Design Patterns

Design Pattern	Class Name	M_{OL}©	M_{OR}©	M	CTP	SPF	SP	D_M	DPF	DRTP	DP
P1	Stack	0	0	5	0	0	0	16	0	0	0
P2	Vehicle	0	0	1	0	0	0	0	0	0	0
	TwoWheeler	0	1	1	0	1	1	1	0	1	1
	FourWheeler	0	1	1	0	1	1	0	0	0	0
	VehicleClient	0	0	2	0	0	0	2	0	0	0

P3	Chain	0	0	2	0	0	0	0	0	0	0
	Number	0	0	2	0	0	0	2	0	0	0
	NegativeProcessor	0	2	2	0	1	1	6	0	(1+4)/6	0.83
	ZeroProcessor	0	2	2	0	1	1	5	0	(1+3)/5	0.8
	PositiveProcessor	0	2	2	0	1	1	3	0	(0+2)/3	0.66
P4	Manager	0	0	1	0	0	0	0	0	0	0
	Department	0	1	3	0	1/3	0.33	2	0	0	0
	Employee	0	1	1	0	1	1	1	0	1	1
	Client	0	1	2	0	1/2	0.5	2	0	1/2	0.5

V. Analysis Results

From Table 1,We can see that there is a great difference between values of existing metric POF and purposed metrics for 4 design pattern .Values of these metrics for 4 design pattern is shown in Table 2.

Table 2. Comparison of Existing and Purposed Metrics

Sr.no.	Design Pattern	POF	SP	DP
1	P1	0	0	0
2	P2	0	2	1
3	P3	0	3	2.29
4	P4	50%	1.83	1.5

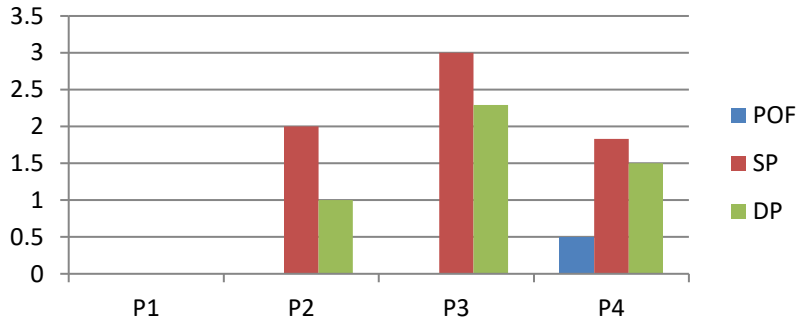

Figure.3.Comparison of POF and SP

As shown in Figure.3. It clearly shows that there is no polymorphism in P1,P2 and P3 according to existing metric or no code is reused but in actual there is code reused in P2 and P3. So existing metric is not able to find the actual degree of polymorphism.For P2,P3 and P4 value of compile time and runtime polymorphism is totally different. It shows purposed metrics have their own existence and behaviour of polymorphism is different at compile time and run time. Therefore, purposed metric plays a significant role to find reusability hence quality of a software system.

VI. Conclusion

In this paper, a detailed study is done on 4 design pattern having different behaviour as one is based on no inheritance, two design pattern based on single inheritance and last one is based on multilevel inheritance.Results shows that P3 design pattern has highest degree of polymorphism at compile time and runtime both because reusability of code is highest in this design pattern amongst all. Hence, we can say that purposed metrics plays a vital role to find reusability ofsoftware system.

References

[1] prof. jubair j. al-ja'afer, "khaireddin m. sabri, metrics for objectoriented design(mood) to assess java programs," king abdullah ii school of information 2004.
[2] Tobias Mayer,Tracy Hall, "Measuring OO system : A critical Analysis of Mood Metrics," Software quality Journal 1999(Springer)
[3] AmandeepKaur, Satwinder Singh, Dr.K.S.Kahlon, Dr.ParvinderS.Sandhu, "Empirical Analysis of CK &Mood Metric Suit," International Journal of Inovation , Management and Technology, Vol.1No.5,2010.

[4] Kelvin H.T. Choi, Ewan Tempero, "Dynamic Measurement of Polymorphism",proceedings of the Thirtieth Australasian Computer Science Conference ,Ballarat, Victoria, Australia,Vol.62,211-220, 2007.
[5] S Benlarbi, WL Melo, "Polymorphism measures for early risk prediction," Proceedings of International Conference on Software Engineering,1999.
[6] https://moodmetrics.blogspot.com/2019/11/mood-metrics-suite.html
[7] AspectJ<http://www.eclipse.org/aspectj>
[8] https://o7planning.org/en/10257/java-aspect-oriented-programming-tutorial-with-aspectj
[9] https://www.eclipse.org/aspectj/doc/next/progguide/printable.html
[10] https://www.geeksforgeeks.org/inheritance-in-java/
[11] https://www.geeksforgeeks.org/chain-responsibility-design-pattern/