Volume 01, Issue 01 (Jan-Feb 2021),, PP:18-20
www.ijsreat.com

Dynamic Polymorphism Influence on Superiority of a System

K.Hari Krishna Prasad[1], S.N.MurtyKodukulla[2]
[1]*Aditya Institute of Technology and Management, Tekkali, India.*
[2]*Sri GCSR CollegeCollege, India.*

Abstract: *Polymorphism colossally influences the idea of an item .It is essentially of two sorts, accumulate time or static polymorphism considering system over-troubling and runtime or dynamic considering procedure supplanting. If a class is significantly polymorphic for instance reusability of that class is high. If the class is less polymorphic, reusability of that class is low. There are simply several estimations open recorded as a hard copy to track down the polymorphism component of a class. In this paper, we familiarize one more estimation with track down polymorphism of a class at request time and runtime and contrast and the ongoing estimation. The purposed estimation is applied on 4 arrangement plans having different approach to acting. Dynamic still up in the air with the help of purposed gadget named DynaPoly completed in AspectJ using point of view arranged programming in java. Eclipse stage is used to perform coding of equipment using AspectJ After breaking down the results, it is derived that purposed estimation expects an essential part to track down reusability, consequently nature of a system and gives further developed results than existingmetrics.*

Keywords*:* Polymorphism, quality, metric, reusability

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.

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 forboth.Therefore,

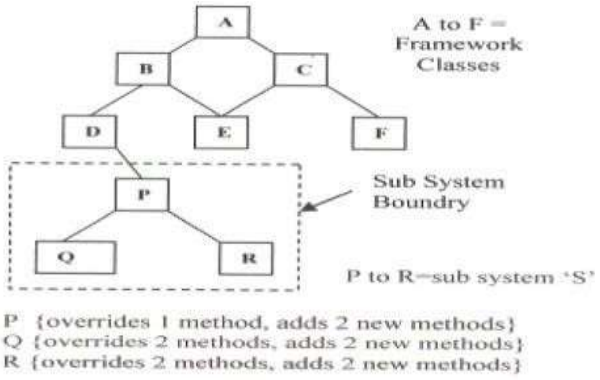


Figure. 1. Plan of SubsystemS

The purposed estimation 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.

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

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.

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

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).

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

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.

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

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.

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

Where m is unquestionably the quantity of procedures in class, ORi is the times supplanted technique executed at, ni is the times system I executed at runtime.

DP(Dynamic Polymorphism): It is portrayed as measure of DPF and RTP taken from condition (3) and (4).

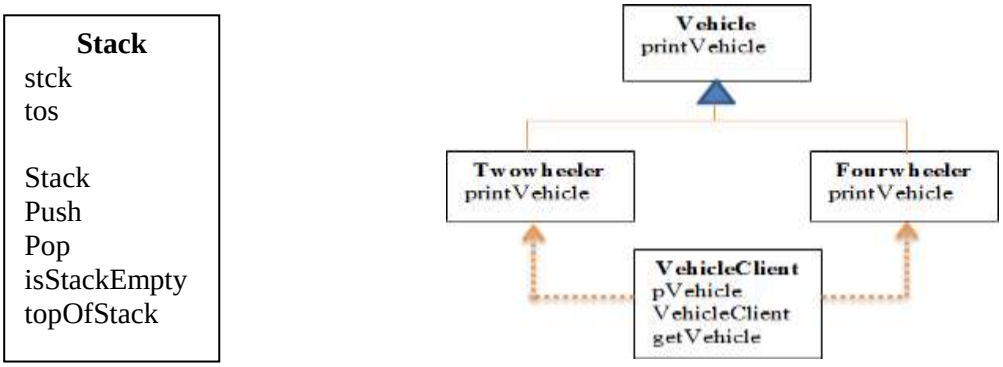
$$\mathrm{DP(c)} = \mathrm{DPF} + \mathrm{RTP}$$

Where DPF is Dynamic Polymorphism Factor and RTP is Run Time Polymorphism. If the value of purposed estimation is 1, class is extraordinarily polymorphic and code is reused in a colossal manner at runtime as well. If the value of DP is 0, code isn't reused using any and all means at runtime. Consequently purposed estimation expects a fundamental part to find reusability of class at runtime in this way nature of an item structure.

II.Experimental Study

A preliminary report is coordinated on 4 plan named P1,P2, P3 and P4 showed in Figure.4. All the 4 Design pattern[11] have different approach to acting. As we presumably know POF is a system based estimation yet our

purposed estimation is a class based estimation. So to find the polymorphism of a system we can add the potential gains of SPand DP of classes contained insystem.



(a) (b)

Figure. 3. (a) plan P1 (b) Design Pattern P2

First arrangement plan P1, contains all of the methods in 1 class so worth of POF is 0 for instance there is no polymorphism in P1 at accumulate and runtime both thus worth of SP and DP is 0 for P1.P2 contains 4 classes as one class as base class and 2 decided classes. Fourth class is individual class joined with other 2 classes. Decided worth of POF for P2 is also 0 as there is single level heritage exist and no family members of construed classes. P3 contains 5 classes with one base class and 3 construed class.Fifth class is individual class joined with other 3 classes.P4 contains 4 classes ,1 base class and 2 decided class. Below average class is furthermore connected by fourth class as shown in Figure.4. Worth of POF for P3 is in like manner 0 since single level of heritage exist there. POF worth of setup plan P4 is 0.5 or half. Decided potential gains of purposed estimation are shown in Table 1.DM is the times systems for a class executed atruntime.

VI.Conclusion

In this paper, a point by point study is done on 4 arrangement configuration having different approach to acting as one relies upon no heritage, two arrangement configuration considering single inheritance and continue to go one relies upon staggered inheritance.Results shows that P3 setup configuration has generally critical degree of polymorphism at assemble time and runtime both considering the way that reusability of code is most raised in this plan among all. Thusly, we can say that purposed estimations accepts an irreplaceable part to find reusability ofsoftware system.

References

[1] TobiasMayer,TracyHall, "MeasuringOOsystem:AcriticalAnalysisofMoodMetrics," SoftwarequalityJournal1999(Springer)
[2] 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.
[3] 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.
[4] S Benlarbi, WL Melo, "Polymorphism measures for early risk prediction," Proceedings of International Conference on Software Engineering,1999.
[5] https://moodmetrics.blogspot.com/2019/11/mood-metrics-suite.html
[6] *AspectJ<http://www.eclipse.org/aspectj>*