IJCSMS International Journal of Computer Science and Management Studies, Special Issue of Vol. 12, June 2012
ISSN (Online): 2231-5268
www.ijcsms.com

Measurement of Reusability Ratio of Software Module by Using Pattern Based Metrics

Anil Kumar

Computer Science & Engineering, Vaish College of Engineering, Rohtak, India
anilbest2005@gmail.com

Abstract

There is no doubt that reusability of software modules play very important role in software development that not only reduces the resources, but also reduces effort, time and cost that are required during software development. There are numbers of technique available, which are used to determine reusability ratio, these technique help to determine reusability ratio, but not help to reduce the complexity. In this, proposed metrics are discusses that not only to determine this but side by side reduces the complexity.

Keywords: *OO Metrics, Proposed Metrics.*

1.1 Introduction

In this paper, here we will discuss a proposed metric that mot only measure reusability ratio but also helps us how to reduce the reusability ratio not only during the software development but also from the starting phase of software modules. By using proposed metrics we can determine reusability ratio of software modules while its actual requirement are takes place from software requirement specification to design' of software modules.

1.2 Problem Description

There is no doubt that OO metrics, such as MIF, AIF and PF are very important metric that will measure the reusability of software modules during software development. The value of PF indicates the reusability. However, if its values are continuously increases than after specific point, then it is very difficult to understand and test the software modules. Therefore, more effort, time and cost are required not only for understanding but also for further enhancement and testing.

1.3 OO Metrics

The OO metrics [1, 2], MIF, AIF and PF very important metrics that are used to determines reusability ratios of software modules. The MIF and AIF metrics measures the functionality of inheritance. The large value of MIF and AIF indicate that reusability increased, but if some modifications is required it is very difficult to understand/test the software modules. Therefore, more effort and time are required to understand and test for software modules during software development.

1.3.1 Method Inheritance Factor (MIF) Metric

MIF is the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods (locally defined plus inherited) for all classes.

$$MIF = \frac{\sum_{i=1}^{TC} Mi(Ci)}{\sum_{i=1}^{TC} Ma(Ci)}$$

$$Ma(Ci) = Md(Ci) + Mi(Ci)$$

Where Ci is number of classes
Ma (Ci) is the number of method defined in Ci
Md (Ci) is the number of method declared in Ci
Mi (Ci) is the number of method inherited in Ci
TC is the number class

1.3.2 Attribute Inheritance Factor (AIF) Metric

AIF is the ratio of the sum of inherited attributes in all classes of the system on to the total number of available attributes (locally defined plus inherited) for all classes.

$$AIF = \frac{\sum_{i=1}^{TC} Ai(Ci)}{\sum_{i=1}^{TC} Aa(Ci)}$$

$$Aa(Ci) = Ad(Ci) + Ai(Ci)$$

Where Ci is number of classes

Aa (Ci) is the number of attribute defined in Ci

Ad (Ci) is the number of attribute declared in Ci

Ai (Ci) is the number of attribute inherited in Ci

TC is the number class

1.3.3 Polymorphism Factor (PF) Metric

Another very important OO metric is PF, which measure the reusability of methods in classes. The value of PF indicates reusability, if its value very high then it is very difficult to understand and test the software modules. Therefore more effort and time are required for test and understand.

PF is the ratio of the actual number of possible different polymorphic situation for class Ci to the maximum number of possible distinct polymorphic situations for class Ci

$$PF = \frac{\sum_{i=1}^{TC} Mo(Ci)}{\sum_{i=1}^{TC} \lfloor Mn(Ci) * DC(Ci) \rfloor}$$

$$Mo(Ci) = Mn(Ci) + Mo(Ci)$$

Where Ci is number of classes

Mn (Ci) is the number of new method defined in Ci

Mo (Ci) is the number of override method in Ci

DC is the number of docents class in Ci

TC is the number class

1.4 Proposed Metrics

The pattern metrics [3], DIP, NOP and CBP play very efficient role to determine reusability ratio not only during the coding phase of software modules, but also during the design phase of software modules. Also play important role to reduce the reusability ratio that is not effecting the time and cost, but also help to easy understand and test the software modules.

DIP metric measures number of packages inherited in base package and number of classes that exist in the package. More methods is likely to inherit, and then it is difficult in predicting the behavior. Therefore modules should be designed in such a way during reengineering, that they reduce the number of methods of package likely to inherit.

The NOP metric measures the complexity by determining the number of immediate sub package of a package and number of immediate sub-class of a class or derived class. The complexity increases as the result of the metric increases.

The CBP metric measures the number of base package and number of base classes of packages that inherit in derived package class. It will measure inheritance complexity during reengineering of software module. If its value increases, then complexity also increases

1.4.1. Depth of Inheritance Package (DIP) Metric:

DIP metric measure the weight of package inherited in base package that is again related to size complexity of modules.

$$DIP = \frac{1 + \sum_{i=1}^{D} P(D)}{DIT}$$

Where:

For d is total number of

package inherited

P (D) number of base package

1.4.1.1. Depth of Inheritance Tree (DIT) Metric:

DIT metric measure the weight of class in the inheritance hierarchy, the greater the number of methods it is likely to inherit, making it more complex to predict its behaviors.

$$DIT = 1 + \sum_{i=1}^{I} (P(I))$$

Where:

For I is total number of inherited method

P (I) is number of inherited hierarchy

1.4.2. Number of Package (NOP) Metric:

NOP metric measure the weight of immediate sub-package of a package that is again directly proportional to the size complexity of modules.

$$NOP = \frac{1 + \sum_{i=1}^{n} P(N)}{NOC}$$

Where:

For n is total number sub-package

P (N) number package

1.4.2.1. Number of Children (NOC) Metric:

NOC metric measure the weight of immediate sub-class of a class or derived class that is again directly proportional to size complexity of modules.

$$NOC = 1 + \sum_{i=1}^{ch} P(CH)$$

Where:

For ch total number of subclass

P (CH) is number of immediate sub class

1.4.3. Count of Base Package (CBP) Metric:

CBP metric measure the weight of base package that is again directly proportional to size complexity of modules.

$$CBP = \frac{1 + \sum_{i=1}^{b} P(B)}{CBC}$$

Where:

b for total number package

P (B) for number of base package

1.4.3.1 Count of Base Class (CBC) Metric:

CBC metric measures the weight of base class, which is used to measure inheritance complexity.

$$CBC = 1 + \sum_{i=1}^{C} P(B(C))$$

Where:

For c is total number base class

For P (B (C)) is number base class defined within the package

Conclusion

The proposed metrics help the software developer to design the software modules in such a way that will reduces reusability ratio, that help to easier to understanding and testing. Due to this it will save cost, time and resources that are required during development of software modules.

References

[1] Rambaugh., A Theory of Object-Oriented Design: The building-blocks of OOD and notations for representing them (with focus on design patterns.)
[2] Abreu, F. B. e., "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.
[3] Ashok Kumar and Anil Kumar," Complexity Measurement During Reengineering by Using Pattern Based Metrics", International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 2, No. 6, December 2011, ISSN: 2079-2557