International Journal of Computing and Corporate Research

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

AN EMPIRICAL MATHEMATICAL MODEL AND ALGORITHMIC APPROACH FOR REUSABILITY FACTORS IN SOFTWARE REENGINEERING

Aanchal Prajapat

M. Tech. Scholar

Ambala College of Engineering and Applied Research

Ambala, Haryana, India

Abstract

'Software' refers to a program and all of the associated information and materials needed to support its installation, operation, repair and enhancement. 'Software engineering' refers to the disciplined applications of engineering, scientific and mathematical principles and methods to the economical production of quality software. The field of software engineering was born in 1968 in response to the chronic failures of large software projects to meet schedule and budget constraints. Software engineering process constitutes of – (1) requirement, (2) analysis, (3) design, (4) implementation, and (5) test work flows. One of the key issues in the management of the development process is the capability to measure and record the relevant attributes of the software products and of the process itself in a structured and coherent framework. The availability of the software metric helps manager to control the various activities of the development life cycle and contributes to the overall objective of software quality. Software metric is a measure of some property of a piece of software or its

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

specifications. Software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving software quality. This paper highlights various factors affecting the reusability and metrics of the source code and propose the mathematical model for the reusability.

INTRODUCTION

Process metrics are known as management metrics and used to measure the properties of the process which is used to obtain the software. Process metrics include the cost metrics, efforts metrics, and advancement metrics and reuse metrics. Process metrics help in predicting the size of final system & determining whether a project on running according to the schedule. Process metrics include:

- Cost metrics, measuring the cost of a project, or of some project activities (for example original development, maintenance, documentation).
- Effort metrics (a subcategory of cost metrics), estimating the human part of the cost and typically measured in person-days or person-months.
- Advancement metrics, estimating the degree of completion of a product under construction.
- Process non-reliability metrics, assessing the number of defects uncovered so far.
- Reuse metrics, assessing how much of a development benefited from earlier developments.

Product metrics are also known as quality metrics and is used to measure the properties of the software. Product metrics includes product non reliability metrics, functionality metrics, performance metrics, usability metrics, cost metrics, size metrics, complexity metrics and style metrics. Products metrics help in improving the quality of different system component & comparisons between existing systems.

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

Product metrics include two categories: external product metrics cover properties visible to the users of a product; internal product metrics cover properties visible only to the development team.

External product metrics include:

- Product non-reliability metrics, assessing the number of remaining defects.
- Functionality metrics, assessing how much useful functionality the product provides.
- Performance metrics, assessing a product's use of available resources: computation speed, space occupancy.
- Usability metrics, assessing a product's ease of learning and ease of use.
- Cost metrics, assessing the cost of purchasing and using a product.

Internal product metrics include:

- Size metrics, providing measures of how big a product is internally.
- Complexity metrics (closely related to size), assessing how complex a product is.
- Style metrics, assessing adherence to writing guidelines for product components (programs and documents).

REUSE PROBLEMS

a) Increased maintenance costs:

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

If the source code of a reused software system or component is not available then maintenance costs may be increased as the reused elements of the system may become increasingly incompatible with system changes.

b) Lack of tool support:

CASE toolsets may not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse into account.

c) Not-invented-here syndrome:

Some software engineers sometimes prefer to re-write components as they believe that they can improve on the reusable component. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.

d) Creating and maintaining a component library:

Populating a reusable component library and ensuring the software developers can use this library can be expensive. Our current techniques for classifying, cataloguing and retrieving software components are immature.

e) Finding, understanding and adapting reusable components:

Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they will make routinely include a component search as part of their normal development process.

Problem Statement

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

Reusability is the likelihood that a segment of source code can be used again to add new functionalities with slight or no modification. More the segment of source code is independent more it will be reusable. . In my Masters dissertation I designed coupling and cohesion metric which will help in calculating the independency of a source code

Objectives of the Proposed Work

- To devise an effective and unique software reusability metric based on cohesion and coupling in procedural as well as object oriented systems
- To propose and implement the adaptable and consistent software metric so that it can be used by the industry as well as academia
- To propose the effective methods and metrics for the measurement of cohesion and coupling in any source code
- To implement the parameter and associated aspects based reusability metric and develop the parser so that it can automatically calculate the cohesion, coupling and reusability.

Methodology Used

We have formulated the empirical mathematical models for the measurement of reusability that is solely based on the cohesion and coupling in the modules and the components.

The proposed mathematical model makes use of the following parameters and is calculated for multiple source codes :

- *Cumulative Weight CW*
- *Total Modules Accessing and Modifying the common Object*
- *Total Shared Objects or Global Variables*

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

- *Total Interfaces in OOP Based Source Code*
- *Total Friend Functions in Procedural as well as OOP Based Source Code*
- *Total Modules and Variables*

MUA *= # of pairs of methods that directly or indirectly use common attributes, or directly connected methods.*

MPV *= all sets or combinations possible # of methods.*

F_P *= Fuzzy Parameter*

NIC *are the pairs of methods that are indirectly connected.*

CGV *is the set of common global variables used by the set of methods.* {[count(global variables i.e g2,g3 used)]

CV *is the Cumulative Declarations of the Variables in the entire procedure / method*

Steps of Implementation

a) METRIC formulation: we had designed metrics for measuring coupling and cohesion in a software program. By using these metrics we can calculate the value of total coupling and cohesion among module of software program, which help in calculating the reusability of a software module or program. Higher the value of coupling more the module or component is dependent and hence lesser the module is reusable. While, higher the value of cohesion more the component or module is independent and hence more it is reusable. Coupling metric is calculated using following formula

International Journal of Computing and Corporate Research

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

Shared Object Coupling Metric (SOCM) =

$$CW(((MOB+TSO+GV+I+TFF)/TMV)*100)$$

Cohesion is calculated using:=

$$(MUA / MPV) + (\sum CGV / \sum VT) \pm (F_P (MUA+NIC) / MPV)$$

b) Generation of individual mathematical equation: parser will convert the statement based equation to notion based equation so that further calculations can be done easily

c) Development of java based parser: based on suffix and prefix parser will analysis the component

d) Generation of dependent and independent module: depending upon the keywords and parameter dependency of each line is analysis and the dependent and independent module are formed

e) Apply coupling and cohesion metrics over it

f) Fetch the result

1. SOCM (Shared Object Coupling Metric) :

$$CW(((MOB+TSO+GV+I+TFF)/TMV)*100)$$

*Cumulative Weight CW (((Total Modules Accessing and Modifying the common Object + Total Shared Objects or Global Variables + Total Interfaces + Total Friend Functions) / Total Modules and Variables)*100)*

MOB	Uses or Modifies	In consideration
Method1()	G2,g3	Yes
Method2()	G3	Yes
Method3()	-	No

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

Total count of MOB = 2

TSO /GV = 3 (i.e. g1,g2,g3)

I =1 (interface1)

TFF =1 samplefn()

TMV = 9 (a,b,g1,g2,g3,method1(),method2(),method3(),samplefn())

Calculation part

=((2+3+1+1)/9)*100

=(7/9)*100

=0.7778*100

=77.78%

++

2. STRONG CLASS COHESION

Metric Formulation : (MUA / MPV) + (∑CGV / ∑CV) ± (F_P (MUA+NIC) / MPV)

MUA *= # of pairs of methods that directly or indirectly use common attributes, or directly connected methods.*

MPV *= all sets or combinations possible # of methods.*

F_P *= Fuzzy Parameter*

NIC *are the pairs of methods that are indirectly connected.*

CGV *is the set of common global variables used by the set of methods.*
{[count(global variables i.e g2,g3 used)]

CV *is the Cumulative Declarations of the Variables in the entire procedure / method*
count ([local variable declared or used i.e a])}

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

MUA -- directly connected methods

Main=> method1(), method1()=>method2(), method2() => method3()

Total =3

MPV – All possible connected pairs

Main() => method1()=>method2()=>method3()

Total = 4

CGV / CV

In method 1 = 2/1

{[count(global variables i.e g2,g3 used)] /count ([local variable declared or used i.e a])}

In method 2 = 1/1

In method 3 = 0/2

= 2+1+0

=3

NIC

Lists the indirectly connected pairs. Main() calls method1, method1() calls method2(), method2() calls method3(). So main is indirectly connected to method2() and method3(),

Method1() is indirectly connected to method3(). Total = 3.

MUA=3 , MPV = 4.

(F_P (MUA+NIC)/MPV) = Fp(1.5) (Fuzzy value should be between 0 to 1)

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

=(3/4) +(2+1+0)

=3.75

CONCLUSION AND FUTURE WORK

The notion of reusability is an old idea that has been around since human beings became involved in problem solving. Reuse, formal systematic reuse that is, is essential for the development and maturity of any engineering field. Software engineering, unlike other engineering fields, has not developed into a mature discipline yet. The practitioner in electrical or aerospace engineering, for example, routinely uses (i.e. reuses) formal models, formulas, and engineering handbooks previously developed, tested, verified, standardized, and accepted by their respective communities. Software engineers (i.e. craftsmen), in contrast, tend to reinvent at every opportunity by placing their creativity at the wrong level. Engineering is the art of mastering trade-offs and that is where software engineers should concentrate their creativity, not in creating more clever and intricate programs or structures. The problem in software engineering is not lack of reuse but lack of wide spread systematic reuse. Software reusability has considerable effect on software quality. Software quality increases as reuse of software components increases. But software quality cannot be improved unless it can be measured. In this research work we have derived a new approach to measure the software reusability of the attributes of software source code. We have also empirically studied and implemented the approach using C++ and Java Based Source Code. Software reuse has been practiced informally since programming was invented. Programmers have been reusing algorithms, sections of code from previous programs, and subroutines from functional collections. They also adapt and reverse-engineer existing systems. All this, however, is done informally and very much ad-hoc. Substantial quality and productivity pay-off from reuse are only achieved if conducted systematically and formally.

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

REFERENCES

[1] Budhija and Satinder Pal Ahuja," Review of Software Reusability" International Conference on Computer Science and Information Technology (ICCSIT'2011) Pattaya Dec. 2011

[2] Ivica Crnkovic," Component-based Software Engineering – New Challenges in Software Development" http://www.idt.mdh.se/personal/icc

[3] G.N.K.Suresh Babu And Dr.S.K.Srivatsa," ANALYSIS AND MEASURES OF SOFTWARE

[4] Sajjan G. Jarallah S. Alghamdi," MEASURING SOFTWARE COUPLING" The Arabian Journal for Science and Engineering, Volume 33, Number 1B April 2008 p119-129

[5] Gui Gui*, Paul. D. Scott," Measuring Software Component Reusability by Coupling and Cohesion Metrics" JOURNAL OF COMPUTERS, VOL. 4, NO. 9, SEPTEMBER 2009

[6] DR. P. K. SURI, NEERAJ GARG**,"** Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse" IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009 p237-248

[7] Liguo Yu, Kai Chen, Srini Ramaswamy, "Multiple-parameter coupling metrics for layered component-based software" Software Qual J(2009) 17:5–24DOI 10.1007/s11219-008-9052-9

[8] Neha Shiva, Lubna Abou Shala." Software Reuse: Research and Practice" International Conference on Information Technology (ITNG'07) 0-7695-2776-0/07 IEEE Computer Society 2007

[9] Manik Sharma, Gurdev Singh, Anish Arora And Parneet Kaur,"A Comparative Study of Static Object Oriented Metrics" International Journal of Advancements in Technology Vol. 3 No.1 (January 2012)

[10] Amandeep Kaur, Satwinder Singh, Dr. K. S. Kahlon and Dr. Parvinder S. Sandhu," Empirical Analysis of CK & MOOD Metric Suit" International Journal of Innovation, Management and Technology, Vol. 1, No. 5, December 2010

[11] Shyam R. Chidamber and Chris F. Kemerer, " A metaics suite for object oriented design" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 20, NO. 6, JUNE 1994

[12] William B. Frakes and Kyo Kang," Software Reuse Research: Status and Future" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 7, JULY 2005 p529-536

[13] REUSABILITY" International Journal of Reviews in computing p41-46

[14] file:///I:/Pen%20Drive/reusability/cbse/Componentbased%20Software%20Development%20%28CBSD%29.htm

[15] Green R M (1994) The Ethical Manager, Macmillan Publishing Gotterbam and Rogerson 1998, "The Ethics of Software Project Management", in Ethics and Information Technology, ed. G&an Collste, New Academic Publisher.

[16] William B. Frakes and Kyo Kang IEEE Transaction on software Engineering, Software Reuse Research: Status and Future, VOL. 31, NO. 7, JULY 2005

[17] http://ecomputernotes.com/software-engineering/discuss-in-detail-coupling-and-cohesion

[18] http://home.adelphi.edu/sbloch/class/adages/coupling_cohesion.html

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

[19] http://wiki3.cosc.canterbury.ac.nz/index.php/Coupling_and_cohesion

[20] http://www.ustudy.in/node/79

[21] Sarbjeet Singh, Sukhvinder Singh, Gurpreet Singh "Reusability of the Software" International Journal of Computer Applications (0975 – 8887) Volume 7– No.14, October 2010

[22] Gurdev Singh,, Dilbag Singh, Vikram Singh," A Study of Software Metrics" International Journal of Computational Engineering & Management, Vol. 11, January 2011

[23] P. Joshi and R.K. Joshi, "Microscopic Coupling Metrics for Refactoring", Proceedings of the Conference on Software Maintenance and Reengineering CSMR 2006, 22-24 March 2006, pp.145–152.

[24] S. R. Chidamber, D. P. Darcy, and C. F. Kemerer, "Managerial Use of Metrics for Object-Oriented Software: An exploratory analysis". IEEE Transactions on Software Engineering, 24(1998), pp. 629–639.

[25] Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue, "Prediction of Fault-Proneness at Early Phase in Object-Oriented Development", Second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, May 1999, pp. 253–258.

[26] Mie Mie Thet Thwin and Tong-Seng Quah, "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics", Journal of Systems and Software, 76(2)(2005), pp. 147–156.

[27] G. Myers, Composite/Structured Design. Van Nostrand Reinhold, 1978.

[28] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous & Practical Approach. 2nd edn. Reading, 1997.

ISSN (Online) : 2249-054X

Volume 3 Issue 4 July 2013

Manuscript ID : 2249054XV3I4072013-07

[29] Norman Fenton and Austin Melton, "Deriving Structurally Based Software Measures", J. System Software, (12) 1990 pp. 177–187.

[30] H. Dhama, "Quantitative Models of Cohesion and Coupling in Software", Journal of System and Software, 29(1)(1995) pp. 65–74.

[31] J. B. Lohse and S. H. Zweben, "Experimental Evaluation of Software Design Principles: An Investigation into the Effect of Module Coupling on System Modifiability", Journal of System and Software, 4(1)(1984), pp. 301–308.

[32] D. H. Hutches and V. R. Basili, "System Structure Analysis: Clustering with Data Bindings", IEEE Transactions on Software Engineering, 11(8)(1985), pp. 749–757.

[33] J. Offut, M.J. Harrold, and P. Kotle, "A Software Metric System for Module Coupling", Journal of System and Software, 20(3)(1993), pp. 295–308.