International Journal of Computer Applications (0975 – 8887)
Volume 26– No.7, July 2011

Unified Design Quality Metric Tool for Object-Oriented Approach including other Principles

Poornima U S
Assistant Professor
Acharya Institute of
Management and Sciences,
Peenya, Bangalore-INDIA

ABSTRACT

Object Oriented methodology is an emerging trend in software development for scientific and business applications. Design of the solution domain has an impact on the overall quality of the software. Merging of all individual design quality metric tools as a package along with other design principles like abstraction and stability could serve the developer better as plug-ins for IDEs.

General Terms

Object Oriented Design, Metrics and Tools.

Keywords

Programming_in_the_large, solution domain, metrics, tools.

1. INTRODUCTION

Programming_in_the_large is the current development scenario where the problem domain is data-centric rather than services. The solution domain is populated with data and classified as classes based on commonality. The class attributes represent the overall behavior of the software. Different classes in the solution domain does not offer rich set of services, whereas, allowing them to either share or importing the services makes the solution domain complete. The quality of the software depends on the design of a class. Much effort goes in repairing the bad design if identified in later stages of development process. Object Oriented Design metrics are helpful in identifying faulty design at early stage of software development. Many tools are available individually to measure a Java program before and after implementation. This paper presents the concept of merging design metric tools as a package along with other design principles like abstractness and stability.

2. DIRECT AND INDIRECT METRICS

Software metric measures the quality of either the process or product under development ([1] [2] [3]). Process metrics checks the software development planning and scheduling so that process should not exceed the calendar. Product metrics are either direct or indirect measures of developed software. The direct measure checks LOC, Execution time, Memory usage, cost and effort on development, and number of defects. The indirect measures are on the software environment includes security, reliability, scalability, portability and maintainability.

3. OBJECT-ORIENTED DESIGN QUALITY METRICS

Success of the software is mainly dependent on its design. A significant number of Object Oriented design quality metrics are defined among which CK metrics (Chidamber and Kemerer) [4] are popular in the literature.

3.1 Weighted Methods per Class (WMC)

It measures the complexity of a class- operational attribute, methods, in terms of effort and time for development and maintenance. Complexity of a class is a cumulative sum of complexity of all its methods. The objective is to keep it low to uphold design quality.

$$WMC(C) = \sum_{i=1}^{n} ci(Mi)$$

Where C is a class and M is a class method.

3.2 Depth of Inheritance Tree (DIT)

It measures the *vertical* growth of a class. Inheritance supports reusability; however the complexity is directly proportional to the distance between leaf and parent class. Deeper tree structure is prone to higher complexity as it is difficult to access end class behavior.

3.3 Number Of Children (NOC)

It is a metric to measure the *horizontal* growth of a class. The immediate subclasses in a hierarchy show the greater reusability. System functional quality is highly dependable on abstractness of the parent class. Much effort is required in testing if tree grows in both directions.

3.4 Coupling Between Object classes (CBO)

It measures the interdependency between the classes. An object of a class can use the service or object of another class. The objective is to reduce the much interdependency (cross coupling) to increase the clarity of the solution.

3.5 Response For a Class (RFC)

It measures response set of a class. When an object of a class sends a message, the methods executed inside and outside of a class are counted. The amount of effort in debugging, testing and maintenance is depending on response count.

$|RS| = \{ M \} \cup_{\text{all } i} \{ R_i \}$

where {Ri} = set of methods called by method *i* and {M}=set of all methods in the class.

3.6 Lack of Cohesion in Methods (LCOM)

It measures the quality of a class in a solution domain. Cohesion refers the degree of interconnectivity between attributes of a class. A class is cohesive if it can not be further divided in to subclasses. It measures the method behavior and its relevance where it is defined. Pair of methods using data object proves the cohesiveness where as the methods not participating in data access makes it less cohesive. Consider C is a class and M1, M2...Mn are its methods using set of class instances. I1={a,b,c,d}, I2={a,b,c} and I3={x,y,z} are set of instances used by the methods M1,M2 and M3 respectively. If intersection of object set is non-empty then the method using them is cohesive and their relevance in the class is proved. i.e. $I1 \cap I2 = \{a, b, c\}$ means M1 and M2 are cohesive. But intersection of I1, I3 and I2, I3 is empty set. High count in LCOM shows less cohesiveness and class need to be divided to subclasses.

Apart from CK and MOOD, other metrics [5] based on Object Oriented principles are also assess the design quality.

1 *Stability* measures a class or class category in terms of changes they incorporate if it is done within. Highly interdependent class or category is risky to manage as changes have ripple effect on other classes. System is more stable when its dependency on outside classes is less than the others using it. The aim is to minimize the dependency to make system more scalable and maintainable.

2 *Abstractness* is a metric for measuring flexibility of a software. It provides a basement on which variety of services are implemented, all belongs to the same category. Software architecture can be made flexible by including more abstract classes.

4. EXISTING TOOLS

Development of a software begins from data collection and analysis through design to implementation. Quality of work is measured by using tools based on defined metrics ([6] [7] [8]).

4.1 Design quality metric tools

4.1.1 JDepend

This tool checks the dependency among the packages in the solution framework. It generates the count of number of package classes used by the current package (Ce) and being used (Ca) by other packages. The design parameters like number of Classes and Interfaces, Abstractness, Instability and Dependency cycle are also shown. It also reports the cyclic dependency between the packages. It is a plug-in with Eclipse IDE. Limitation of this tool is poor report format. It displays the afferent and efferent couplings of each analyzed Java package in a plain format which need to be more expressive using graphical tools. Moreover it fails in analyzing the metrics in full extent for which it is designed.

4.1.2 Classycle

This tool is helpful in identifying static cyclic dependency between classes or packages. Unlike JDepend, it works at class level and analyses the compiled Java code not the source file. Directed graph of classes or package dependency is generated and is further analyzed to find the cyclic dependency. The output in the XML can be better visualized with other graphical tools available.

4.1.3 Chidamber & Kemmerer Java Metrics

It is an open source tool to access CK Object Oriented Design quality metrics. It analyses compiled Java file and generates output in text format.

4.1.4 *CCCC* tool analyses and generates the report on general code metrics like LOC along with design metrics by Chidamber & Kemerer.

4.1.5 ES2 collects metrics like size, coupling and inheritance. It inspects ".java" files and collects the information from interface specifications. It fails in analyzing other quality metrics.

4.1.6 SDMetric tool is a design metric tool for measuring all types of UML diagrams. It collects the information from system level to sub system level, packages and in detail towards classes and objects. It reports the information in user-friendly tables and charts.

4.2 Code analyzers

4.2.1 PMD is an open source tools helpful for identifying bugs, unused variables and codes. It also checks code redundancy to improve overall quality of the software.

4.2.2 QJ-Pro is Java source code review tool targets the developer for not using language standards. It checks the code for no error when the code modified. It ensures the indirect measure of software quality like reliability, portability and maintainability. It is used during software development and testing phase of Software Development Life Cycle.

4.2.3 LOC counts number of lines in a source code including blank lines and comment lines. It is a basic code analyzer to assess size and complexity of software.

4.2.4 SLOC counts only physical lines of code in a module in a verity of languages. It works on both Unix and windows environment can be easily installed and used. It will automatically estimate effort, time and cost of the project development. The basic cost estimation model, COCOMO, is used based on LOC.

4.2.5 Resource Standard Metrics (RSM) is a source code analyzer checks both quantity and quality of a software. It reports design information of package, classes, methods and size of each module. The output is in text, HTML and CSV format.

5. COMPARETIVE STUDY

It is known from the study that very few commercial and open source tools are available for quality evaluation of object oriented software design. Each tool is performing well with defined metrics individually but failed to cover all proposed in the literature. Output format of the report generated by few tools are not having friendly features which will have an impact on assessing the result.

Table 1. Tools v/s metrics evaluation

Tools	Metrics					Formatted O/P
	WMC	DIT	NOC	CBO	LCOM	
JDepend		√	√			√
Classycle		√				√
ckjm	√	√	√	√	√	
CCCC		√	√	√		
RSM	√	√	√			√
ES2	√	√	√	√		√

6. LITERATURE SURVEY

Since metrics are quantitative measure of a software design and implementation, a number of individual tools have been developed and made accessible as open source tools. A few commercial tools are also available for measuring process and product metrics.

A paper by P. Edith Linda et al. (2011) focused on different tools available and proposed a web page so that all the tools are accessible at the same place.

Dr. Rakesh Kumar and Gurvider Kaur (2011) have done a comparative study on the complexity of Object Oriented Design metrics proposed by Shyam R. Chidamber , Chris F. Kemerer and Li.

Dr. M.P Thapaliyal and Garima Verma (2010) have done an empirical analysis on few metric data and their relationship with software defects.

Rudiger Lincke et al. (2008) have done analysis and comparison of the output of different tools on different projects. It is shown that the output of product metrics is tool dependent.

Linda Westfall, The Westfall Team (2005) defines 12 steps to useful software metrics suitable for organization. It focuses on refining the metrics for organization so that better product can be developed.

Sandeep Purao and Vijay Vaishnavi (2003) present a rigorous survey on product metrics. It focuses on understanding and classification of ongoing research in Object Oriented metrics.

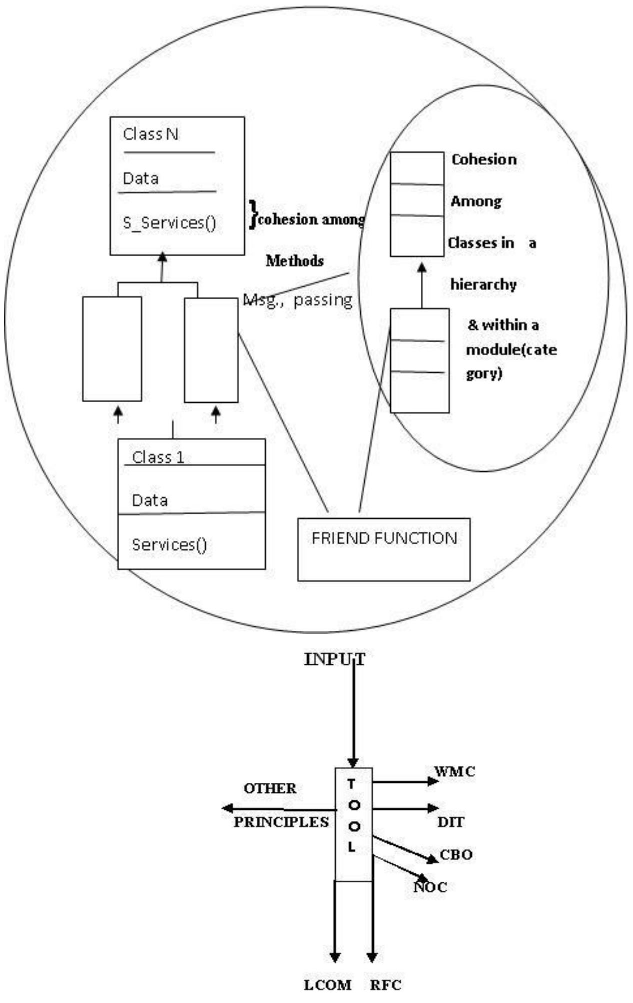
Stojanovic M and El Eman K (2001) developed a tool, ES2, for collecting the design quality metrics for C++ and JAVA source code. This analyzer is basically implemented on top of Source Navigator IDE for analyzing large amount of code with cross-references and links amongst classes.

Dr. Linda H. Rosenberg focuses on Traditional and Object Oriented Metrics adapted for Object Oriented environment to evaluate the principle object oriented structures and concepts.

7. CONCLUSION AND FUTURE SCOPE

Automation of complex business application with accuracy and customer friendly is becoming a big challenge for the developers. The objective of this paper is to create a common platform for all design quality metrics as plug-ins for IDE to make the application software more scalable and maintainable. The tools available as a commercial product or open source are limited in functionality or including both code and architecture analysis.

Fig.1 Solution Domain of an Object Oriented System



8. ACKNOWLEDGMENTS
I would like to thank Mr. Nandakumar V Purohit, a Technical Team Leader for his guidance on practical approach and Mr. Mallikarjuna Shastri P M (Phd.) for the suggestions on improving the quality of this paper.

9. REFERENCES
[1] Sandeep Purao and Vijay Vaishnavi "Software Metrics for Object-Oriented systems", ACM Computing Surveys, Vol 35, No2, JUNE 2003, pp 191-221.

[2] Seyyed Mohsen Jamali, "Object Oriented Metrics –A Survey Approach", January 2006.

[3] Dr. Rakesh Kumar and Gurvinder Kaur, "Comparing Complexity in Accordance with Object Oriented Metrics", *International Journal of Computer Applications (0975 – 8887) Volume 15– No.8, February 2011*.

[4] Shyam R. Chidamber and Chris F. Kemerer "A Metrics Suite for Object Oriented Design", IEEE TRANSACTION ON SOFTWARE ENGINEERING, VOL 20,No 6, JUNE 1994.

[5] Robert Martin, "OO Design Quality metrics", October 28, 1994

[6] Rüdiger Lincke, Jonas Lundberg and Welf Löwe, "Comparing Software Metric Tools", 2008 ACM 978-1-59593-904-3/08/07.

[7] P. Edith Linda, V. Manju Bashini, S. Gomathi, "Metrics for Component Based Measurement Tools", International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011.

[8] Stojanovic M and El Eman K, "ES2: A Tool for Collecting Object-Oriented Design Metrics from C++ and Java Source code", National Research Council of Canada, June 2001.

[9] Dr. M.P Thapaliyal and Garima Verma, "Software Defects and Object Oriented Metrics–An Empirical Analysis" *International Journal of Computer Applications (0975 – 8887) Volume 9– No.5, November 2010.*

[10] Linda Westfall, The Westfall Team, "12 Steps to Useful Software Metrics", The Westfall Team, 2005

[11] Dr. Linda H. Rosenberg, "Applying and Interpreting Object Oriented Metrics", Track 7 - Measures/Metrics.

AUTHOURS PROFILE
Ms. Poornima U. S has done B.E. and MTech. in Computer Science and Engineering, presently working as Assistant Professor in the Department of Master of Computer Applications at AIMS, Peenya Bangalore. She has got 13 years teaching experience and guiding post graduate students on live projects for department automation and NGOs.