INTERNATIONAL EDUCATIONAL APPLIED RESEARCH JOURNAL (IEARJ)
Volume 03, Issue 07, July 2019
E-ISSN: 2456-6713

ANALYSIS OF SOFTWARE METRICS AND SOFTWARE TOOLS

Sanjeev Kumar

Abstract: ***Accurate measurement is a prerequisite for all engineering disciplines, and software engineering is not an exclusion. For decades seek engineers and researchers to express features of software with numbers in order to make easy software quality assessment. A large body of software quality metrics have been developed, and various tools exist to collect metrics from program representations. This large variety of tools allows a user to select the tool best suited, e.g., depending on its managing, tool support, cost. Conversely, this assumes Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.***

Keywords: Software metrics, Software measurement tools, Software projects, Design, Software quality.

INTRODUCTION:

A software metric is a mathematical definition mapping the entities of a software system to numeric metrics values. Furthermore, we understand a software metrics tool as a program which implements a set of software metrics definitions. It allows to assess a software system according to the metrics by extracting the required entities from the software and providing the corresponding metrics values. For finding a set of suitable software metrics tools, we conducted a free search on the internet. Our first criteria was the tools calculate any form of software metrics. In order to compare as many tools as possible, we chose to analyze Java programs. Furthermore, about half of the tools are rather simple \code counting tools. Investigating the legal status of the tools, we found that some of them are limited to analyze just a few files at a time, or we can simple not get hands on these programs.

SOFTWARE METRICS:

- **CBO** (Coupling between Object classes) is the number of classes to which a class is coupled.
- **DIT** (Depth of Inheritance Tree) is the maximum inheritance path from the class to the root class.
- **LCOM-CK** (Lack of Cohesion of Methods) (as originally proposed by Chidamber & Kemerer) describes the lack of cohesion among the methods of a class.
- **LCOM-HS** (Lack of Cohesion of Methods) (as proposed by Henderson-Sellers) describes the lack of cohesion among the methods of a class.
- **LOC** (Lines of Code) counts the lines of code of a class.
- **NOC** (Number of Children) is the number of immediate subclasses subordinated to a class in the class hierarchy.
- **NOM** (Number of Methods) is the methods in a class.
- **RFC** (Response for a Class) is the set of methods that can potentially be executed in response to a message received by an object of the class.
- **WMC** (Weighted Methods per Class) (using Cyclomatic Complexity as method weight) is the sum of weights for the methods of a class.

SOFTWARE METRICS TOOLS:

Our final selection left us with the following 10 software metrics tools:

Analyst4j is based on the Eclipse platform and available asa stand-alone Rich Client Application or as an Eclipse IDE plug-in. It features search, metrics, analyzing quality, and report generation for Java programs.

CCCC is an open source command-line tool. It analyzes C++ and Java files and generates reports on various metrics, including Lines of Code and metrics proposed by Chidamber & Kemerer and Henry & Kafura .

Chidamber & Kemerer Java Metrics is an open source command-line tool. It calculates the C&K object-oriented metrics by processing the byte-code of compiled Java files.

Dependency Finder is open source. It is a suite of tools for analyzing compiled Java code. analysis application that extracts dependency graphs and mines them for useful information. This application comes as

a command-line tool, a Swing-based application, a web application, and a set of Ant tasks.

Eclipse Metrics Plug-in 1.3.6 by Frank Sauer is an open source metrics calculation and dependency analyzer plugin for the Eclipse IDE. It measures various metrics and detects cycles in package and type dependencies.

Eclipse Metrics Plug-in 3.4 by Lance Walton is open source. It calculates various metrics during build cycles and warns, via the Problems View, of metrics 'range violations'.

OOMeter is an experimental software metrics tool developed by Alghamdi et al. It accepts Java/C# source code and UML models in XMI and calculates various metrics.

Semmle is an Eclipse plug-in. It provides an SQL like querying language for object-oriented code, which allows to search for bugs, measure code metrics, etc.

Understand for Java is a reverse engineering, code exploration and metrics tool for Java source code.

VizzAnalyzer is a quality analysis tool. It reads software code and other design specifications as well as documentation and performs a number of quality analyses.

Table 1: Evaluation of Software Metrics and Tools

TOOLS USED	Metrics								
	CBO	DIT	LCOM-CK	LCOM-HS	NOC	NOM	RFC	TCC	WMC
Analyst4j	X	X	X		X	X	X		X
CCCC	X	X			X	X			
Chidamber & kemmerers Java Metrics	X	X	X		X	X		X	
Dependency finder		X			X	X			
Eclipse Metrics Plugin 1.3.6		X		X	X	X			X
Ecipse Metrics 3.4			X	X					X
OO Meter	X	X	X		X			X	
Semmle		X	X	X	X	X	X		
Understand of Java	X	X	X		X	X			
Vizz Analyzer	X	X	X		X	X	X	X	X

There are four criteria for maintainability with the software metric tools and software metrics:

(i) Analyzability
(ii) Changeability
(iii) Stability
(iv) Testability.

In order to be able to use the software quality model with all tools, we can only include metrics which are calculated by all tools. We should also have as many metrics as possible: we should have at least one coupling, one cohesion, one size, and one inheritance metric included to address the biggest areas of quality-influencing properties. We further involve as many tools as possible.

Maximizing the number of tools and metrics involved, we came to include 4 tools and 5 metrics. The tools are: Analyst4j, C&K Java Metrics, Vizz Analyzer, and Understand for Java. The metrics involved are: CBO, a coupling metric, LCOM-CK, a cohesion metric, NOM, a (interface) size metric, and DIT and NOC, inheritance metrics. The composition of the quality model should not have a large influence on the results, as long as it is the same for each tool and project. The relations and weighting of metrics to criteria can be seen arbitrarily.

Thus, the metrics values are aggregated and abstracted by the factors and the applied weights to the maintainability criteria, which describe the percentage of classes being outliers in the system, thus having bad maintainability.

CONCLUSIONS AND FUTURE SCOPE:

In this thesis, new static and dynamic metrics have been proposed, evaluated and validated for measurement of cohesion, coupling and complexity for object-oriented software. Major contributions of the work reported in this thesis are:

- New metrics for the measurement of package cohesion and package coupling have been proposed and validated.
- Aspect-oriented approach of collection of run-time data has been found to be better than other approaches for this purpose.
- New dynamic cohesion and coupling metrics have been proposed and validated.
- A dynamic analyzer tool has been developed using aspect-oriented programming (Aspectj) to perform dynamic analysis of Java applications.

- New spatial complexity measures for the estimation of comprehensibility of Java programs have been defined.

- New object-oriented cognitive-spatial complexity metrics have been proposed and evaluated.

One goal of cognitive complexity metrics is to be able to predict where the risk of error is high. This can be due to two factors. When STM is overloaded, errors occur. Also, when a weakly supported concept is important, the risk of error is raised. In this way, understandability can be measured as either too much information in a short time or not enough information over a longer period. Future directions for my research include empirical testing of text comprehension exercises involving text of a technical nature. I also plan to look at how chunking is pattern matching and develop some metrics based on pattern matching.

REFERENCES:

1. V. Gupta, "Object-Oriented Static and Dynamic Metrics for Design and Complexity", Ph.D. dissertation, Department of Computer Engineering, National Institute of Technology, Kurukshetra, India, pp. 1 -30. (references) Available: http://www.nitkkr.ac.in/clientFiles/FILE_REPO/2012/MAY/12/1336804909539/varun_co-phd.pdf viewed on 30th June, 2016.
2. C. Serban, "Metrics in Software Assessment", Ph.D. dissertation, Faculty of Mathematics and Computer Science, Babes-Bolyai University,Cluj-Napoca, Romania, 2012, pp. 1 -36. (references)
3. H.B. Klasky, "A Study of Software Metrics", M.S. thesis, Graduate Program in Electrical and Computer Engineering, Graduate School-New Brunswick, Rutgers. The State University of New Jersey, May, 2003, pp. 1 -50. (references) Available: http://www.cac.rutgers.edu/TASSL/Thesis/Hilda_Thesis.pdf. (references)Viewed on 30th June 2016.
4. S. Moser, "Measurement and Estimation of Software Processes", PhD Thesis, University of Berne, Switzerland, pp. 1 -31, November, 1996. Available: http://scg.unibe.ch/archive/phd/moser-phd.pdf viewed on 30th June, 2016. (references)
5. L.H. Rosenberg and L.E. Hyatt, "Software quality metrics for object-oriented environments", Crosstalk Journal, vol. 10, no. 4, pp.1 -6, 1997.
6. E.M. Bouwers, "Metric-based Evaluation of Implemented Software Architectures", M.S. thesis, Department of Computer Science, University of Delft, Rijswijk, 2013, pp. 1 - 195. Available: http://www.st.ewi.tudelft.nl/~arie/phds/Bouwers.pdf , viewed on 30th June, 2016. (references)
7. T. Hall and N. Fenton, "Implementing effective software metrics programs", IEEE Software, pp. 55 – 64, April 1997.
8. C. Catal and B. Diri, "A systematic review of software fault prediction studies", Expert Systems with Applications, Elsevier, vol. 36, no. 4, pp. 7346 – 7354, May, 2009.
9. P.K. Singh, O.P. Sangwan, A.P. Singh, and A. Pratap, "An assessment of software testability using fuzzy logic technique for aspect oriented software", International Journal of Information Technlogy and Computer Science, vol. 3, pp. 18 - 26, February, 2015.
10. T. Azuma, "Automation technology for software development at NTT data", vol. 12, no. 12. pp. 1 – 8, December, 2014.