

# CK Metric

*by* Iwan Binanto

---

**Submission date:** 18-Jan-2018 04:13PM (UTC+0700)

**Submission ID:** 862006451

**File name:** PaperSoftwareMetric\_IwanBinanto\_IEEE.doc (569.5K)

**Word count:** 2553

**Character count:** 13377

# Measuring the Quality of Various Version an Object-Oriented Software Utilizing CK Metrics

<sup>1,2</sup>Iwan Binanto

<sup>1</sup>Informatics Department,  
Faculty of Science and Engineering  
Sanata Dharma University  
Yogyakarta, Indonesia 55002  
iwan@usd.ac.id

<sup>2</sup>Harco Leslie Hendric Spits Warnars<sup>A</sup>, <sup>2</sup>Ford Lumban  
Gaol<sup>B</sup>, <sup>2</sup>Edi Abdurachman<sup>C</sup>, <sup>2</sup>Benfano Soewito<sup>D</sup>

<sup>2</sup>Computer Science Department,  
BINUS Graduate Program – Doctor Computer Science  
Bina Nusantara University  
Jakarta, Indonesia 11480  
spits.hendric@binus.ac.id<sup>A</sup>, fgaol@binus.edu<sup>B</sup>,  
edia@binus.edu<sup>C</sup>, bsoewito@binus.edu<sup>D</sup>

**Abstract**—There are many object-oriented software quality measurement techniques. One of them is CK metric. This paper utilizes CK metrics to measure qualities per version of an open source software namely Statcato. Measurements are done in each class, each version of the software. The measurement is used to analyze the quality of the software. This paper concludes that the quality of software has improved during its lifecycle, although its features increase.

**Keywords**— *object-oriented software metric; chidamber-kemerer metric; software quality; Software measurement; Software Size*

## I. INTRODUCTION

In software engineering, there are at least two concepts of developing software, which are traditional and object-oriented. Software development using Object Oriented concept has become popular. It facilitated the developers to build and maintain their software products easily since it is reusable, secure, and efficient as it decreases the size of system and number of logical constructs [1].

The more complex the software is developed, the harder it becomes to measure the quality. Software metrics are required to measure the quality of the developed software and also to estimate the cost and effort of software projects [2]. Radjenović, et. al. concluded that object-oriented metrics 49% were used, compared to traditional source code metrics in 27% or process metrics in 24% [3]. There are many existing object-oriented software metrics [4]. One of measurement metrics was developed by Chidamber and Kemerer [5], which is often called as CK metrics suite [6]. This is called as metric because it is an indicator of system performance [7].

CK metrics suite were most widely referred and used by researchers [1], [3]. It was used in half of the 106 studies that Radjenović, et. al. did [3]. CK metrics has been recognized by numerous studies in which said that these metrics were effective at measuring the concepts they represent [6], [7], [8], [9] and as the backbone for object-oriented design metric [10].

Today, open source software becomes popular and important in many applications, such as scientific and business. Many companies use this software in their own

work. Open source software was developed with different management style than the industrial, the quality and reliability of the code have to investigated [11].

One of open source software is Statcato, an open source and Java-based application which is naturally an Object-Oriented Software. It requires Java Runtime Environment (JRE) 1.6 or higher. Once any machine has Java installed, whether the computer runs Windows, Mac, Linux, or UNIX, it can be run [12]. The aim of this study is to evaluate Statcato to investigate the quality of this software during its lifecycle. The calculated metric values for various releases are used as a basis to evaluate the software quality of Statcato. We use Spinellis's tool namely CKJM to calculate CK metrics [13]. It is a great tool for this purpose.

## II. RELATED WORKS

Gupta, et. al. in [7] reviewed some of metrics for measuring Object Oriented software and gave conclusion that software metrics always was always needed to the performance of the system. Malhotra & Khanna in [14], studied the relationship between Object Oriented metrics and the vulnerability of open source software classes based on Java. Ferreira, et. al. in [4] provided a threshold definition for a set of object-oriented software metrics. They argued that the use of metrics for software quality measurement was less effective because many studies did not specifically investigate the threshold for object-oriented software metrics. This threshold can be applied to other metric software. Benlarbi, et. al. in [15] provided a threshold on the CK metric to make it easier to interpret the measurements. They had experiment on object-oriented in C++. Malhotra & Bansal in [9] built predictive models to identify software parts which have a high probability of generating errors. They calculated the threshold values of CK metrics by using a statistical model derived from logistic regression. Shaik, et. al. in [1] did assessment in Object Oriented design and found that there were many metrics to measure software quality, but CK metrics was the most widely used. Subramanyam & Khisnani in [16] provided empirical evidence that supported the role of Object-Oriented complexity metrics in determining software defects, particularly CK metrics. Yadav, et. al. [17] did analyze CK

metrics and their impact on object-oriented software. Suresh, et. al. in [6] argued that CK metrics were used to compute the system reliability, also gave an idea for future study about applying neural network to predict system reliability using CK metrics. Shawky & Abd-El-Hafiz in [8] utilized CK metric as one of their metrics to study some metrics that related to the quality of the generated code affected by development methodology, especially in agile approach. Suri and Singhal in [10] experimentally analyzed the influence of CK metric in object-oriented classes. Singh in [18] used CK metrics as one of metrics to measure the quality per version of open-source software namely JFreeChart. Some study utilized CKJM tool to calculate the CK metrics of the classes are in [9], [10], [13], [19], [20], [21].

### III. CHIDAMBER & KEMERER METRICS

Chidamber and Kemerer (CK) metrics used to measure object oriented software. It consist of 6 metrics which are calculated for each class in Object-Oriented software, which are: WMC, DIT, NOC, CBO, RFC and LCOM.

#### A. WMC (Weighted Method per Class)

The more methods in class, the more specific the class becomes. It also affects the derived class because it will inherit all of those methods. Keep WMC value low. It is because when WMC value is high, the class is harder to reuse and maintain [6], [7], [11], [18], [20]. WMC value must be less than 25 although it is still acceptable if it is more than 25 (but less than 40) [22].

#### B. DIT (Depth of Inheritance Tree)

The deeper the class hierarchy, the greater the number of methods inherited so that the class becomes more complex. However, it also becomes increasingly potential for inherited reuse methods. It affects the quality of the software directly as it increases reusability and the complexity at the same time [6], [7], [11], [18], [20]. The value 0 of DIT shows "root". If the DIT value is 2 or 3, it indicates higher usage. The DIT value of less than 2 can be said to be less good, in terms of advantage using object-oriented design and inheritance. On the other hand, a DIT value that is greater than 5 will provide high complexity, but it is expected from object-oriented design [22]. Suri & Singhal, however, in [10] said that all classes must inherit from existing class, so the DIT value, at least, is 1.

#### C. NOC (Number Of Children)

Increasing children classes means increasing reuse classes, which means the greater of abstraction of the parent class. Therefore, more testing is required [6], [7], [11], [18], [20]. There is no "good" or "bad" value for the NOC. The value becomes important if the class has a high value for other metrics. The complexity of the class will be passed to all child classes and the entire system [22].

#### D. CBO (Coupling Between Object Classes)

An increase of CBO indicates decreasing of class reusability, so this values should be kept as low as possible

[6], [7], [11], [18], [20]. CBO values should be less than 5. High values of CBO indicate that classes are difficult to understand, reuse or maintain. The higher value of CBO indicates that classes are more sensitive to change in other sections of design. Therefore, this makes maintenance more difficult. Low CBO values make the class easier to understand, less prone to spawning errors, encourage encapsulation and increase modularity [22].

#### E. RFC (Response For a Class)

RFC measures the number of methods that are invoked to respond messages received by an object. It shows how many classes communicate with other classes. The greater the number of classes giving the method, the greater the complexity of the class [6], [7], [11], [18], [20]. RFC value should be less than 50. If the RFC is high, the complexity increases and class becomes elusive [22].

#### F. LCOM (Lack of Cohesion in Method)

Cohesion is the extent to which methods are interconnected with one another. High LCOM values imply that classes are better divided into two or more separate classes, so it will keep LCOM value low [6], [7], [11], [18], [20].

### IV. ANALYSIS METRICS

CKJM tools are used to collect data from all classes in Statcato software and we calculate the mean of it as show in Table 1. By calculating the mean of data, we intend to show the movement of software development from version to version.

TABLE I. MEAN VALUES OF METRICS FOR VARIOUS RELEASES OF STATCATO

Version	Metric	WMC	DIT	NOC	CBO	RFC	LCOM
0.91		3.884	0.974	0.018	2.681	15.03	70.37
0.92		4.309	0.932	0.932	2.832	18.35	59.34
0.93		4.341	0.945	0.029	2.835	18.75	65.38
0.94		4.228	0.933	0.022	2.825	18.8	52.09
0.95		4.379	0.952	0.032	2.833	18.21	70.97
0.96		4.388	0.95	0.03	2.83	18.2	70.61
0.97		4.395	0.948	0.03	2.838	18.29	70.65
0.98		4.37	0.945	0.028	2.822	18.1	71.74
0.99		4.381	0.944	0.028	2.825	18.16	71.89
0.910		4.38	0.943	0.03	2.821	18.11	72.54
0.911		4.357	0.941	0.03	2.83	18.13	74.2
0.912		4.368	0.931	0.026	2.817	18.58	67.2
1.0		4.368	0.931	0.026	2.817	18.58	67.21
1.01		4.368	0.931	0.026	2.817	18.59	67.21
1.02		4.39	0.935	0.026	2.851	18.72	67.59

#### A. WMC (Weighted Method per Class)

Fig. 1 shows that WMC is starting stable on version 0.95 upwards. In a negative view, the software does not have significant development. Positively, the software is close to stable for further development. In [23], it does seem that not many new features are developed.

### B. DIT (Depth of Inheritance Tree)

In the data that has been collected in Table 1, mean values of DIT are close to 1, in fact if mode and median are calculated and rounded then value of both of them is 1. Therefore, it can be concluded that DIT in this case is 1 and satisfies [10]. Fig. 2 shows the changes internal software for DIT. It appears that the trend is decreasing which indicates the class is less complex. Less complex class means easier to maintenance but less reusable.

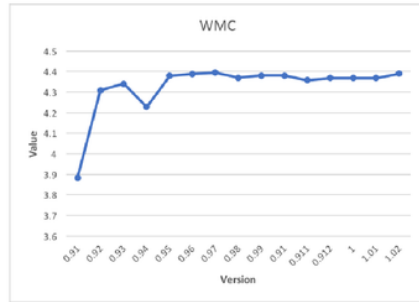


Fig. 1. Graphic of WMC

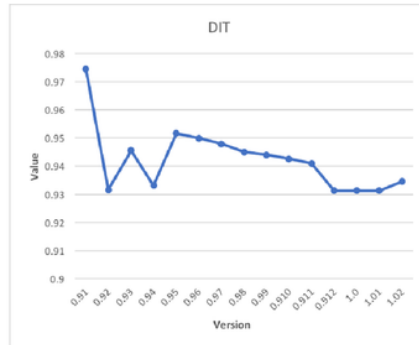


Fig. 2. Graphic of DIT

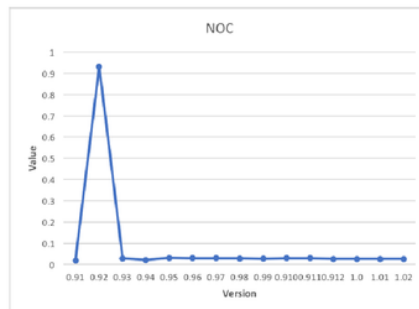


Fig. 3. Graphic of NOC

### C. NOC (Number Of Children)

Fig. 3 shows that NOC is starting stable on version 0.93 upwards. Referring to WMC metric above, in negative view, the software does not have significant development, or in

positive view, the software is close to stable for further development.

### D. CBO (Coupling Between Object Classes)

Fig. 4 shows that CBO is starting stable on version 0.95 upwards. The highest value of CBO is less than 4. It satisfies [22], so classes easy to understand and easy to maintain.

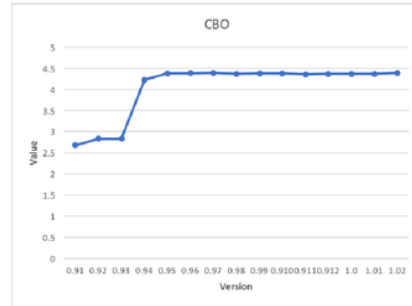


Fig. 4. Graphic of CBO

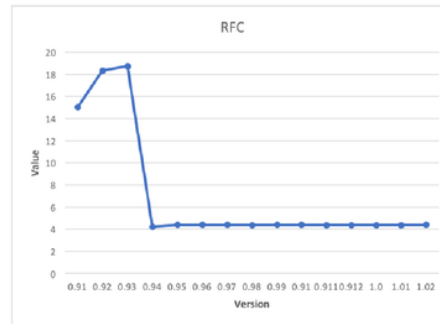


Fig. 5. Graphic of RFC

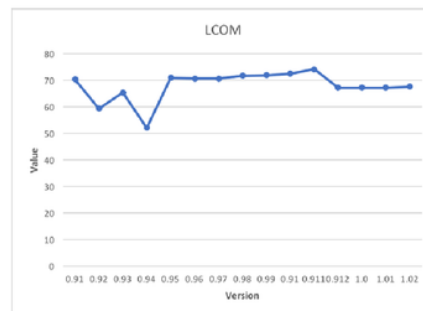


Fig. 6. Graphic of NOC

### E. RFC (Response For a Class)

Fig. 5 shows that CBO is starting stable on version 0.94 upwards. Referring to WMC and NOC metric above, in negative view, the software have not significant development,

but positively, the software is close to stable for further development.

#### F. LCOM (Lack of Cohesion in Method)

Fig. 6 shows that LCOM value is decreased since version 0.912. It means that developers care about the cohesion for a better development.

### V. CONCLUSION

We can say that the quality of the latest version of Statcato is better than the predecessor, although the feature is increasing. It can be summarized that in positive view, the value of WMC, NOC, and RFC metrics is stable, which is a good condition. While DIT and CBO metrics show ease to maintain and LCOM metric indicates developer concerns. Therefore, it can be concluded that Statcato has good quality and has a good development to continue improving during its lifecycle, although its feature is increasing. For future works, we recommend to calculate the threshold per metric to better clarify the assessment of the metrics.

### REFERENCES

- [1] A. Shaik, M. C. S. E. Tech, and D. C. S. E. Ph, "Object oriented software metrics and quality assessment: Current state of the art," *Int. J. Comput. Appl.*, vol. 37, no. 11, pp. 6–15, 2012.
- [2] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach 3rd Ed.* CRC Press Taylor & Francis Group, 2015.
- [3] D. Radjenović, M. Heričko, R. Torkar, and A. Živković, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, Aug. 2013.
- [4] K. A. M. Ferreira, M. A. S. Bigonha, R. S. Bigonha, L. F. O. Mendes, and H. C. Almeida, "Identifying thresholds for object-oriented software metrics," *J. Syst. Softw.*, vol. 85, no. 2, pp. 244–257, 2012.
- [5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, Jun. 1994.
- [6] Y. Suresh, J. Pati, and S. K. Rath, "Effectiveness of Software Metrics for Object-oriented System," *Procedia Technol.*, vol. 6, pp. 420–427, Jan. 2012.
- [7] S. Gupta and A. Lazrus, "Comparative Study and Review on Object Oriented Design Metrics," *Int. J. Sci. Res. Eng. Technol.*, vol. 6, no. 1, pp. 59–62, 2017.
- [8] D. M. Shawky and S. K. Abd-El-Hafiz, "Characterizing software development method using metrics," *J. Softw. Evol. Process*, vol. 28, no. 2, pp. 82–96, Feb. 2016.
- [9] R. Malhotra and A. J. Bansal, "Fault prediction considering threshold effects of object-oriented metrics," *Expert Syst.*, vol. 32, no. 2, pp. 203–219, 2015.
- [10] B. Suri and S. Singhal, "Investigating the OO characteristics of software using CKJM metrics," in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, 2015, pp. 1–6.
- [11] P. Singh, K. D. Chaudhary, and S. Verma, "An Investigation of the Relationships between Software Metrics and Defects," *Int. J. Comput. Appl.*, vol. 28, no. 8, pp. 13–17, Aug. 2011.
- [12] M. Yau, "Statcato Features." [Online]. Available: <http://www.statcato.org/statcato/index.php/about/features>. [Accessed: 23-Oct-2017].
- [13] D. Spinellis, "Tool Writing: A Forgotten Art?," *IEEE Softw.*, vol. 22, no. 4, pp. 9–11, Jul. 2005.
- [14] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and change proneness," *Int. J. Mach. Learn. Cybern.*, vol. 4, no. 4, pp. 273–286, 2013.
- [15] S. Benlarbi, K. El Emam, N. Goel, and S. Rai, "Thresholds for Object-Oriented Measures," in *ISSRE '00: proceedings of the 11th international symposium on software reliability engineering*, 2000, no. March 2000, p. 4.
- [16] R. Subramanyam and M. S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE Trans. Softw. Eng.*, vol. 29, no. 4, pp. 297–310, 2003.
- [17] P. Yadav, K. Hussain, and A. Gambhir, "Analysis of object Oriented Metrics," *International J. Sci. Res. Publ.*, vol. 3, no. 7, 2013.
- [18] G. Singh, "Metrics for measuring the quality of object-oriented software," *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 5, p. 1, 2013.
- [19] S. Kaur, K. Kaur, and N. Kaur, "An Empirical Investigation of Relationship between Software Metrics," in *2015 Second International Conference on Advances in Computing and Communication Engineering*, 2015, pp. 639–643.
- [20] S. Gupta and A. Lazrus, "OBJECT ORIENTED DESIGN METRICS FOR DESIGN AND COMPLEXITY ANALYSIS," *Int. J. Eng. Sci. Res. Technol.*, vol. 6, no. 5, pp. 803–809, 2017.
- [21] J. Al Dallal, "Accounting for data encapsulation in the measurement of object-oriented class cohesion," *J. Softw. Evol. Process*, vol. 27, no. 5, pp. 373–400, 2015.
- [22] L. H. Rosenberg and R. Stapko Albert Gallo, "Risk-based Object Oriented Testing," *24th SWE, 1999-Citeseer*, 1999. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.7509&rep=rep1&type=pdf>. [Accessed: 26-Oct-2017].
- [23] M. Yau, "Versions and Updates." [Online]. Available: <http://www.statcato.org/statcato/index.php/downloads/versions-and-updates>. [Accessed: 26-Oct-2017].



# CK Metric

## ORIGINALITY REPORT

13%

SIMILARITY INDEX

8%

INTERNET SOURCES

8%

PUBLICATIONS

5%

STUDENT PAPERS

## PRIMARY SOURCES

1	Thamilvaani Arvaree @ Alvar. "Algorithm analyzer to check the efficiency of codes", ICIMU 2011 Proceedings of the 5th international Conference on Information Technology & Multimedia, 11/2011 Publication	1%
2	<a href="http://citeseerx.ist.psu.edu">citeseerx.ist.psu.edu</a> Internet Source	1%
3	<a href="http://zenodo.org">zenodo.org</a> Internet Source	1%
4	Submitted to Binus University International Student Paper	1%
5	<a href="http://cse.spsu.edu">cse.spsu.edu</a> Internet Source	1%
6	Submitted to University of Auckland Student Paper	1%
7	<a href="http://www.statcato.org">www.statcato.org</a> Internet Source	1%

[people.ucalgary.ca](http://people.ucalgary.ca)

8

Internet Source

1%

9

Priscila P. Souza, Bruno L. Sousa, Kecia A. M. Ferreira, Mariza A. S. Bigonha. "Applying software metric thresholds for detection of bad smells", Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse - SBCARS '17, 2017

Publication

1%

10

Submitted to Universiti Teknologi Malaysia

Student Paper

1%

11

ijcsse.org

Internet Source

1%

12

International Journal of Quality & Reliability Management, Volume 22, Issue 2 (2006-09-19)

Publication

&lt;1%

13

www.ijest.info

Internet Source

&lt;1%

14

Singh, Gagandeep, and Hardeep Singh. "Effect of software evolution on metrics and applicability of Lehman's laws of software evolution", ACM SIGSOFT Software Engineering Notes, 2013.

Publication

&lt;1%

15

a4academics.com

Internet Source

&lt;1%

16

[www.iceberg-sqa.eu](http://www.iceberg-sqa.eu)

Internet Source

&lt;1 %

17

Submitted to International Islamic University  
Malaysia

Student Paper

&lt;1 %

18

Gaurav Kumar, Pradeep Kumar Bhatia.  
"Neuro-Fuzzy Model to Estimate & Optimize  
Quality and Performance of Component Based  
Software Engineering", ACM SIGSOFT  
Software Engineering Notes, 2015

Publication

&lt;1 %

19

[www.inse.org](http://www.inse.org)

Internet Source

&lt;1 %

20

[communities.computer.org](http://communities.computer.org)

Internet Source

&lt;1 %

21

[research.ijcaonline.org](http://research.ijcaonline.org)

Internet Source

&lt;1 %

22

Shatnawi, R.. "A Quantitative Investigation of  
the Acceptable Risk Levels of Object-Oriented  
Metrics in Open-Source Systems", IEEE  
Transactions on Software Engineering, 2010.

Publication

&lt;1 %



Exclude bibliography    On