Volume 2, No. 3, May-June 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
CASE STUDY AND REPORT
Available Online at www.ijarcs.info

Analysis of CK metrics and Li and Henry's metrics for UML Class diagrams

Narendra Pal Singh Rathore *
Computer Science
SSSIST, Sehore
Sehore, India
Enggnarendra1029@gmail.com

Prof. Ravindra Gupta
Computer Science
SSSIST, Sehore
Sehore, India
Ravindra_p84@rediffmail.com

Abstract: For the last couple decade's software quality changed in many levels. The demand for increased software quality has resulted in quality being more of differentiator between products than it ever has been before. For this reason, software developers need objective and valid measures for use in the evaluation and improvement of product quality from the initial stages of development. Class diagrams are a key artefact in the development of object-oriented (OO) software because they lay the foundation for all later design and implementation work. It follows that emphasizing class diagram quality may significantly contribute to higher quality OO software system. The primary aim of this work, therefore, is to present a survey, as complete as possible, of the existing relevant works regarding class diagram metrics. In this survey paper we present study on CK metrics and Li and Henry's metrics for class diagrams.

Keywords: class diagram, object oriented, UML, metrics, software quality

I. INTRODAUCTION

In the OO paradigm one of the key artifacts is the class diagram. The class diagram constitutes the backbone of the OO development and provides a solid foundation for the design and the implementation of software. Therefore, class diagram quality has great influence over the system that is ultimately implemented. Quality in software products is characterized by the presence of different external attributes such as functionality, reliability, usability, efficiency, maintainability and portability. [21]. But these attributes can only be measured late in the OO software development life cycle. Therefore, it is necessary to find early indicators of such qualities based, for example, on the structural properties of class diagrams. [1] This is the context where software measurement is fundamental, because measures can allow us to evaluate class diagram quality characteristics in an objective way, thus avoiding a bias in the evaluation process.

Measuring class diagram quality allows OO software designers:

• To identify weak design spots when it costs less to improve them, rather than repair consequent errors at later implementation phases.

• To choose between design alternatives in an objective way.

•To predict external quality characteristics such as, maintainability, reusability, etc, and improve resource allocation based on these predictions.

Although in the OO software measurement arena the need for measures that can be applied in the early phases of the development process is emerging, up until a few years ago the work done in this sense was scarce because most software measurement researchers focused on the measurement of code and advanced design.[2] [22][23]

The aim of this work is to present a survey of the existing literature of OO measures that can be applied to measure internal quality attributes of class diagrams, considering the following proposals:

Chidamber and Kemerer. [3][4][5]

As the Unified Modeling Language (UML) has emerged as a modeling standard, and in general has been widely accepted by most software development organizations, we will focus this work on UML class diagrams. [7] A precise demarcation of analysis, design, and implementation activities is not easy, due to widespread adoption of iterative and fountain life cycles, which tend, sometimes deliberately, to blur their distinctions. [6] For our current purposes, we shall consider the UML class diagram, at its initial stages of development, to be composed of the following UML constructs:

• Packages.
• Classes.
• Each class has attributes and operations.
• Attributes have their name.
•Operations only have their signature, i.e. their name and definition of parameters.
•Relationships: Association, Aggregation3, Generalization and Dependencies

For defining a Metrics for Software some issues that must be taken into account. [8][9][10][11][23]

•Metrics must be defined pursuing clear goals (using for example the GQM method. [12][14]
•Metrics must be theoretically validated, by addressing the question —is the measure measuring the attribute it is purporting to measure?
• Metrics must be empirically validated, by addressing the question —is the measure useful in the sense that it is related to other external quality attributes in the ways expected.

The objective of our work is two-fold:

1. Provide practitioners with information on the available metrics for UML class diagrams, if they are empirically validated (from the point of view of the practitioners, one of the most important aspects of interest, i.e., if the metrics are really fruitful in practice).
2. Provide researchers with an overview of the current state of metrics for UML class diagrams, focusing on the strengths and weaknesses of CK Metrics and Li and Henry's metrics. Thus,

© 2010, IJARCS All Rights Reserved

researchers can have a broad insight into the work already done in the field of metrics for UML class diagrams.
This work is organized as follows: The existing proposals of OO metrics that can be applied to UML class diagrams are presented in Section 2. Section 3 presents an overall analysis of CK Metrics and Li and Henry's metrics proposals. Finally, Section 4 presents some concluding remarks and highlights in the field of metrics for UML class diagrams.

II. OBJECT ORIENTED PROGRAMMING AND METRICS

Object oriented programming and design are very important in today's environment. It provides generalized solutions for many problems in addition to many benefits like reusability, decomposition of problems into small easily understandable objects and also helps to perform modifications in future and to do functional extensions in already built systems. [15] Object oriented programming is more recent and more important in quality software programming than that of the old style procedural programming. In the last two decades object oriented software engineering receives much attention because object oriented technology is wide spread. [16][17] Object oriented technology and development requires different approach to design, implementation and to measure metrics compared to standard set of metrics. A large number of metrics have been developed and proposed by researchers and numerous tools are available to help to assess the design, quality, maintenance and to collect metrics from software programs. [18][19][20] Many object oriented metrics in literature lack in theoretical proof and some have not been validated. The metrics that evaluate object oriented programming are: classes, methods, inheritance, coupling and cohesion. Very few metrics are for object oriented interfaces. The goal of this paper is first, study on role of metrics for software quality and second study on CK Metrics and Li and Henry's metrics.

III. PROPOSALS OF METRICS FOR UML CLASS DIAGRAMS

We will now present those metrics proposals selected for consideration and that may best demonstrate the present-day context of metrics for UML class diagrams.

CK metrics [Chidamber91; Chidamber94]

Metrics. Chidamber and Kemerer proposed a first version of these metrics and later the definition of some of them was improved. [3][4] Only three of the six CK metrics are available for a UML class diagram (see Table 1).

TABLE 1
CK METRICS [4]

Metric name	Definition
WMC	The Weighted Methods per Class is defined as follows: $WMC=\sum_{i=1} c_i;$ Where $c_1, ..., c_n$ be the complexity of the methods of a class with methods M1, ...,Mn. If all method complexities are considered to be unity, the WMC = n, the number of methods7.
DIT	The Depth of Inheritance of a class is the DIT metric for a class. In cases involving multiple inheritances, the DIT will be the maximum length from the node to the root of the tree.
NOC	The Number of Children is the number of immediate subclasses subordinated to a class in the class hierarchy.

•**Goal.** CK metrics were defined to measure design complexity in relation to their impact on external quality attributes such as maintainability, reusability, etc.
•**Theoretical validation.** Chidamber and Kemerer corroborated that DIT and NOC both accomplish Weyuker's axioms for complexity measures. [4][13]. Briand classified the DIT metric as a length measure, and the NOC metric as a size measure.[25] Poels and Dedene have demonstrated by means of the DISTANCE framework that they can be characterized at ratio the scale level. [26]
•**Empirical validation.** Several empirical studies have been carried out to validate these metrics, among others we refer to the following:
• Li and Henry showed that CK metrics appeared to be adequate in predicting the frequency of changes across classes during the maintenance phase. [5]
•Chidamber and Kemerer have applied these metrics to two real projects obtaining the following observations: [4]
•Designers may tend to keep the inheritance hierarchies shallow, forsaking reusability through inheritance for simplicity of understanding.
•These metrics were useful for detecting possible design flaws or violations of design philosophy, and for allocating testing resources.
•Basili have put the DIT metric under empirical validation, concluding that the larger the DIT value, the greater the probability of fault detection. Also, they observed that the larger the NOC, the lower the probability of fault detection.
• Daly found that the time it took to perform maintenance tasks was significantly lower in systems with three levels of inheritance depth as compared to systems with no use of inheritance. [24]
•Chidamber has carried out studies on three commercial systems, in order to examine the relationships between CK metrics and productivity, rework effort and design effort. None of the three systems studied showed significant use of inheritance, so DIT and NOC tended to have minimal values. Chidamber suggested that low values of DIT and NOC indicate that the reuse opportunities (via inheritance) were perhaps compromised in favor of comprehensibility of the overall architecture of the applications.

Li and Henry's metrics [5]

- **Metrics.** Table 2 shows the metrics proposed by Li and Henry, which are defined at class level

TABLE 2
LI AND HENRY'S METRICS [5]

Metric name	Definition
DAC	The number of attributes in a class that have another class as their type.
DAC'	The number of different classes that are used as types of attributes in a class.

© 2010, IJARCS All Rights Reserved

NOM	The number of local methods.
SIZE2	Number of Attributes + Number of local methods

.

• **Goal.** These metrics measure different internal attributes such as coupling, complexity and size.

• **Theoretical validation**. Briand have found that DAC and DAC' do not fulfill all the properties for coupling measures proposed by Briand. [27][28] This means that neither DAC nor DAC' metrics can be classified according to Briand et al.'s framework; this defines the set of properties that length, size, coupling, complexity and cohesion metrics must fulfill.

• **Empirical Validation.** Li and Henry have applied these metrics (and others) to two real systems developed using Classic-ADA.[5] They found that the maintenance effort (measured by the number of lines changed per class in its maintenance history) could be predicted from the values of these metrics (and others like DIT, NOC, etc.).

• **Tool.** A metric analyzer was constructed to collect metrics from Classic-Ada designs and source code.

IV. GENERAL COMMENTS

After the individual analysis of both proposals, we can conclude that:

• The work on measures for UML class diagrams at a high-level design stage is scarce and is not yet consolidated.

• Although the metrics seem to be defined pursuing a clear goal, which is the complete list of desirable properties of "good" class diagrams, this is not totally clear.

• Even though CK metrics are shown overall to be empirically the most thoroughly investigated, results in some cases, especially those relating to the DIT metric, prove to be contradictory. In summary, evidence regarding the impact of inheritance depth on fault-proneness proves to be rather equivocal. This is usually an indication that that there is another effect that is confounded with inheritance dept. Further research is necessary to identify this confounding effect and disentangle it from inheritance depth in order to assess the effect of inheritance depth by itself.

• CASE tools should be integrated with metrics tools which support metrics like those presented above and allow users to define their own metrics. Thus, CASE tools really can guide and help designers to make decisions along the software development life cycle.

V. CONCLUSIONS

The main contribution of this work is a survey of most of the existing relevant works related to metrics for class diagrams at initial stages of development, providing practitioners with an overall view on what has been done in the field and which are the available metrics that can help them in making decisions in the early phases of OO development. This work will also help researchers to get a more comprehensive view of the direction that work in OO measurement is taking.

VI. REFERENCES

[1] Briand L., Arisholm S., Counsell F., Houdek F. and Thévenod-Fosse P.: "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions", Empirical *Software Engineering*, vol. 4, no. 4, pp. 387-404, 2000.

[2] Etzkorn L., Bansiya J. and Davis C.: "Design and Code Complexity Metrics for OO Classes·, vol. 12, no. 1, pp. 335-40, 1999.

[3] Chidamber S. and Kemerer C.: "Towards a Metrics Suite for Object Oriented Design".,Published in SIGPLAN Notices, vol. 26, no. 11, pp. 197-211, 1991.

[4] Chidamber S. and Kemerer C.: "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.

[5] Li W. and Henry S.: "Object-Oriented Metrics that Predict Maintainability", vol. 23, no. 2, pp. 111-122, 1993.

[6] De Champeaux D. The Technical Writer Hadbook,*Object Oriented Development Process and Metric*, Prentice Hall, 1997.

[7] Electronic Publication: Object Management Group. UML Specification Version 1.5, OMG Document formal/03-03-01.

[8] Briand L., Morasca S. and Basili V.: "An operational process for goal-driven definition of measures", *IEEE Transactions on Software Engineering*, vol. 28 no. 12, pp. 1106-1125, 2002.

[9] Morasca S.: Software Measurement, *Handbook of Software Engineering and Knowledge Engineering*, (S.K. Chang, ed.), Chapter 2: Software Measurement, World Scientific, pp. 239-276, 2001.

[10] Fenton N. and Neil M., Anthony Finkelstein: "Software Metrics: a Roadmap", *Future of Software Engineering*, Ed., ACM, pp. 359-370, 2000.

[11] Calero C., Piattini M., and Genero M.: "Empirical validation of referential integrity metrics", *Information and Software Technology*, vol. 43, pp. 949- 957, 2001.

[12] Van Solingen R. and Berghout E.: The Goal/Question/Metric Method: A Handbook, A practical guide for quality improvement of software development, McGraw- Hill, 1999

[13] Weyuker E.: "Evaluating Software Complexity Metrics", *IEEE Transactions on Software Engineering*, vol. 14, no. 9, pp. 1357-1365, 1998.

[14] Basili V. and Rombach H.: "The TAME project: towards improvement oriented software environments", *IEEE Transactions on Software Engineering*, vol. 14, no. 6, pp. 728-738, 1988.

[15] Rene Santaolaya Salgado, Olivia G. Fragosco Diaz, Manuel A. Valdes Marrero, Issac M. Vaseuqz Mendez and Shiela L. Delfin Lara, "Object Oriented Metric to Measure the Degree of Dependency Due to Unused Interfaces", ICCSA 2004, Springer LNCS 3046, P.No: 808-817,2004.

[16] Terry .C. and Dikel .D.,"Reuse Library Standards Aid Users in Setting up Organizational Reuse Programs",Embedded System Programming Product News,1996.

[17] Pradeep Kumar Bhatia, Rajbeer Mann, " An Approach to Measure Software Reusability of OO Design". March 29, 2008..

[18] Santonu Sarkar, Member, IEEE, Avinash C. Kak, and Girish Maskeri Rama," Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software, IEEE Transactions on Software Engineering, Vol. 34, No. 5, Sep-Oct 2008.

[19] Rudiger Lincke, Jonas Lundberg and Welf Lowe,"Comparing Software Metrics Tools", ISSTA'08, July 20-24, 2008, ACM 978-1- 59593-904-3/07.

[20] Nachiappan Nagappan, Thomas Ball and Andreas Zeller," Mining Metrics to Predict Component Failures",

© 2010, IJARCS All Rights Reserved

Verification and Measurement Group, Microsoft Research, 2005, Redmond, Washington.

[21] ISO/IEC 9126-1: "Information Technology- Software Product Quality – Part 1: Quality Model", 2001.

[22] Henderson-Sellers B.: Handbook on *Object-oriented Metrics - Measures of Complexity*, Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[23] Fenton N. and Pfleeger S.: Handbook on *Software Metrics: A Rigorous Approach*, 2nd. edition. London, Chapman & Hall, 1997.

[24] Daly J., Brooks A., Miller J., Roper M. and Wood M.: "An Empirical Study Evaluating Depth of Inheritance on Maintainability of Object-Oriented Software. Empirical Software Engineering", vol. 1, no. 2, pp. 109-132, 1996

[25] Briand L., Morasca S. and Basili V.: "Property-Based Software Engineering Measurement", *IEEE Transactions on Software Engineering*, vol. 22, no. 6, pp. 68-86, 1996

[26] Poels G. and Dedene G.: "DISTANCE: A Framework for Software Measure Construction", *Research Report DTEW9937*, Dept. Applied Economics, Katholieke Universiteit Leuven, Belgium, 46 p., 1999.

[27] Briand, L., Daly J. and Wüst J.: "A Unified Framework for Coupling Measurement in Object-Oriented Systems", *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, 1999.

[28] Basili V., Briand L. and Melo W.: "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions of Software Engineering*, vol. 22, no. 10, pp. 751-761, 1996.

© 2010, IJARCS All Rights Reserved