A Comprehensive Assessment of Object-Oriented Software Systems Using Metrics Approach

Sanjay Kumar Dubey
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University, Sec-125, NOIDA, India

Prof. (Dr.) Ajay Rana
Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University, Sec-125, NOIDA, India

***Abstract*—Demand for efficient software is increasing day by day and object-oriented design technique became able to fulfill this demand because it is the most powerful mechanism to develop efficient software systems. It can not only help in reducing the cost but also helps in the development of high quality software systems. Software developers need appropriate metrics to develop efficient software system. Object-oriented metrics can play important role in this aspect due to their importance in the development of successful software applications. This paper assesses the object-oriented software system using metrics approach to precisely define the qualitative characteristics of the software system.**

Keywords-System, Metrics, Model, Software, Object-oriented

I. INTRODUCTION

Object-oriented design and development is very popular approach in today's scenario of software development environment. This approach improves software productivity, reusability and flexibility of software systems. Object-oriented systems are gaining popularity as efficient software systems day by day because object-oriented techniques reduce the size of system and number of logical constructs. Object-oriented software usually contains large number of attributes, which can provide more comprehensive descriptions of software's internal nature and structure. These software systems made up of interacting objects that remains in their own local state and operate on their own information. The various concepts like complexity, usability, reusability, testability, understandability etc. are used to enhance the quality of software system, which are also very much related with object-oriented features and can be used to increase the efficiency of object-oriented systems.

Software metrics have become essential in some disciplines of software engineering, because they are used to measure software quality and to estimate the cost and effort of software projects [29]. Generally the metrics are used to indicate the software quality in early stage of software development life cycle (SDLC) to monitor the cost impact of modification and improvement in software system but most of metrics, available for object-oriented software analysis normally be used in later phase of SDLC [10]. Since object-oriented metrics require through understanding of object-oriented concepts and there is no single metric that shows all the features of object-oriented software system, so this paper studies various object-oriented metrics available in various literatures and presents comprehensive scenario of them. It also addresses the following questions: (i) what are the concepts behind object-oriented design methodology (ii) what are various metrics found in the literature for object-oriented software system?

II. OBJECT-ORIENTED DESIGN METHODOLOGY

The design methods provide a set of techniques for analyzing, decomposing, and modularizing software system architectures. There is wide applicability of object-oriented design in today's scenario of software development environment because it promotes better design and view a software system as a set of interacting objects. Object-oriented design must exhibit four features: inheritance, data abstraction, dynamic binding, and information hiding [13]. The components of various object-oriented software are given in Table1.

TABLE I. THE COMPONENTS OF OBJECT-ORIENTED SOFTWARE [8]

Objects	Build	Classes
Objects	Have (are composed of)	Attributes
Objects	Inherit	Attributes
Objects	Have (are composed of)	Methods
Objects	Inherit	Methods
Objects	Send	Messages
Objects	Receive	Messages
Messages	Are	Data
Messages	Are	Relations

It is necessary to establish some basic standards and guiding principles that application developer should follow to achieve expected benefits and advantages of object-oriented technology. This technology may be use in measurement of the metrics of object-oriented software. There are several design methodologies that suggested the guiding principle for many ways to develop object-oriented system.

The Booch method [5] describes the analysis and design phases of an object-oriented system implementation. This method offers a path from requirements to implementation by using object-oriented analysis and design and emphasizes the distinction between logical view and physical view of a system. Jacobson's Object Oriented Software Engineering (OOSE)

method [9] proposed pyramid model for the process of developing object-oriented design, in which tools provide support for the activities in three categories: architecture, method and process. Object Modeling Technique (OMT) approach described by Rumbaugh *et al.* [14], allows system designers to conceptualize the overall system architecture. OMT leads to three different models: object model, dynamic model and functional model of the system. Delatte *et al.* [1] developed Hierarchical Object Oriented Design (HOOD) method. The main process in HOOD, called the Basic Design Step, is based on the identification of objects by means of object-oriented design techniques. The purpose of HOOD is to develop the design as a set of objects, which together provide functionality to the program. Coad-Yourdon [30, 31] proposed object-oriented analysis and design method, which is a step by step method for developing object-oriented models. These steps are: finding class & object, identifying structures, defining subjects, defining attributes, and defining services. Reenskaug *et al.* [39] developed an analysis and design method which emphasizes the role played by objects in the system. This role is dependent on the requirements of the system rather than the properties of the object, thus a single object may perform different roles at different stages of the system. Wirfs-Brock [36] developed the object-oriented approach called Responsibility-Driven Design. They suggested that for each class, different responsibilities are defined and to fulfill the responsibilities of the classes, they need to demonstrate collaboration with other classes. The object agency [40] developed a set of validation measures for various object-oriented design approaches. These measures include concepts, notations, processes and pragmatics. Several other different measures for object oriented designs have been validated by [6, 33, 41]. For software system, Design-Level Cohesion is proposed by [12]. To describe the quality of software system, more structures related to the design properties of object-oriented system is given by [17, 18, 19, 20].

III. OBJECT-ORIENTED METRICS

The concept of object oriented programming, which is based on object-oriented metrics, is closely links the design and implementation phases of software system. Various object-oriented metrics have been proposed in literature [26]. Metrics proposed by Abreau [3, 4], J. Bansiya *et al.* [10], Briand *et al.* [17], Chidamber and Kemerer [37], Lorenz *et al.*[27], W. Li *et al.* [42, 43] are some of the metrics suit that are mostly referenced in various literatures.

Chidamber and Kemerer (CK) [37] are the mostly referenced researchers. They defined six metrics viz. Weighted Methods per Class (WMC), Response sets for Class (RFC), Lack of Cohesion in Methods (LCOM), Coupling Between Object Classes (CBO), Depth of Inheritance Tree of a class (DIT) and Number of Children of a class (NOC). CK metrics were defined to measure design complexity in relation to their impact on quality attributes such as usability, maintainability, functionality, reliability etc. Several studies have been conducted to validate CK metrics. For example Basili *et al.* [41] investigated the CK metrics and validated that five metrics of them appear to be useful to predict class fault proneness. Theoretical validation of CK metrics is given by [6, 15] and several experimental studies have been carried out to validate CK metrics for e.g. [2, 7, 11, 16, 18, 19, 22, 23, 24, 28, 33, 38, 40, 41, 43]. Table 2 shows the summary of CK metrics.

TABLE II. THE METRICS SUITE OF CHIDAMBER AND KEMERER [37]

CK Metric	Definition
WMC	Number of methods of a certain class without inherited methods
RFC	Number of methods that can be performed by a certain class regarding a received message
LCOM	Number of disjunctive method pairs of a certain class
CBO	Number of couplings between a certain class and all other classes
DIT	Maximal depth of a certain class in an inheritance structure
NOC	Number of direct subclasses of a certain class

CK metrics are aimed at assessing the design of object-oriented system rather than implementation. This make them more suited to object-oriented paradigm as object-oriented design put great emphasis on the design phase of software system. The relation between important object-oriented software quality concepts, CK metrics and object-oriented (OO) features is given in Table 3 [21].

TABLE III. RELATIONSHIP AMONG CK METRICS, OBJECT-ORIENTED SOFTWARE QUALITY CONCEPTS AND OBJECT-ORIENTED FEATURES

CK Metric	Concept	OO Feature
WMC	Complexity, Usability, Reusability	Class/Method
RFC	Design, Usability, Testability	Class/Method
LCOM	Design, Reusability	Class/Method
CBO	Design, Reusability	Coupling
DIT	Reusability, Understandability, Testability	Inheritance
NOC	Design	Inheritance

Lorenz *et al.* [27] defined metrics to measure the static characteristics of software design. These metrics divided in the categories of class size, class inheritance and class internal. Size-oriented metrics for the object-oriented classes focus on counts of attributes and operations. Inheritance-oriented metrics focus on the manner in which operations are reused in hierarchy class. Internal class-oriented metrics look at cohesion and code-oriented issues.

MOOD metric set model, proposed by Abreu [3] is another basic structural method of the object-oriented paradigm. They were defined to measure the use of object-oriented design methods such as inheritance (MIF (Method Inheritance Factor), AIF (Attribute Inheritance Factor)) metrics, information hiding (MHF (Method Hiding Factor), AHF (Attribute Hiding Factor)) metrics, and polymorphism (POF (Polymorphism Factor), COF (Coupling Factor)) metrics. Abreu firmly suggested that metrics definitions and dimensions should be justified as they play important role in designing the object-oriented metrics.

Within the framework that, many metrics that are applied to traditional functional development are also applicable to object-oriented development, Rosenberg et al. [21] developed nine metrics for object-oriented system, from which three were traditional metrics viz. Cyclomatic Complexity (CC), Lines of Code (LOC), Comment Percentage (CP) and rest six metrics were same as CK metrics. They validated the six CK metrics at SATC and gave the relation between important object oriented software quality concepts, quality metrics and object oriented features as shown in Table 2 [21].

TABLE IV. OBJECT ORIENTED SOFTWARE QUALITY CONCEPTS, QUALITY METRICS AND OBJECT ORIENTED FEATURES

Metric	OO Feature	Concept
CC	Method	Complexity
LOC	Method	Complexity
CP	Method	Usability, Reusability
WMC	Class/Method	Complexity, Usability, Reusability
RFC	Class/Method	Design, Usability, Testability
LCOM	Class/Method	Design, Reusability
CBO	Coupling	Design, Reusability
DIT	Inheritance	Reusability, understandability, Testability
NOC	Inheritance	Design

W. Li *et al.* [43] proposed a new metric suite which include Number of Ancestor Classes (NAC), Number of Local Methods (NLM), Class Method Complexity (CMC), Number of Descendent Classes (NDC), Coupling Through Abstract data type (CTA), and Coupling Through Message passing (CTM). These metrics measure different internal attributes such as coupling, complexity and size.

J. Bansiya *et al.* [10] defined Quality Model for Object Oriented Design (QMOOD) metrics. The metrics in QMOOD were given as Average Number of Ancestors (ANA), Cohesion Among Methods of class (CAM), Class Interface Size (CIS), Data Access Metric (DAM), Direct Class Coupling (DCC), Measure Of Aggregation (MOA), Measure of Functional Abstraction (MFA), Number Of Polymorphic methods (NOP), Design Size of Class (DSC), Number Of class Hierarchies (NOH), Number of Methods (NOM). Like MOOD metrics, the QMOOD metrics are defined to be computable early in the design process. The summary of above reviewed metrics is represented in Table 4.

TABLE V. OBJECT-ORIENTED METRICS FROM VARIOUS SOURCES

Source	Metrics
Chidamber *et al.* [37]	WMC, RFC, LCOM, CBO, DIT, NOC
Lorenz et al [27]	Class size, Class inheritance, Class internal
Abreu [4]	MIF, AIF, MHF, AHF, POF, COF
Rosenberg *et al.* [21]	CC, LOC, CP, WMC, RFC, LCOM, CBO, DIT, NOC
Li W. *et al.*[43]	NAC, NLM, CMC, CMC, NDC, CTA, CTM
Bansiya *et al.* [10]	ANA, CAM, CIS, DAM, DCC, MOA, MFA, NOP, DSC, NOH, NOM

M. El. Wakil *et al.* compared four metric suits. Their findings showed that how various suits stack up against their standards. Their comparison is showing in following Table 6 [25].

TABLE VI. COMPARISON OF FOUR METRICS BY M. EL WAKIL ET AL. [25]

Metric Models Should	Metric Suite			
	CK [37]	M. Lorenz and J. Kidd [27]	F.B. Abreu [4]	J. Bansiya & C.G. Davis [10]
depend on high level design features only, such as abstract class diagrams which allows assessment in the early stages of design.	No	Yes	Yes	Yes
state model objectives explicitly (state how they assess quality).	No	No	Error Density, Fault Density & Normalized Rework	Reusability, Flexibility, Understandability, Extendibility & Effectiveness
be precisely defined. Ambiguity in metrics definitions allows many interpretations for the same metric.	Yes (except WMC)	Yes	Yes	Yes
provide a clearly stated, formal expression describing how metrics coincide with the assessed characteristic.	No	No	Yes	Yes
provide an interpretation of the results. Till the values produced by a model are given interpretations that could be used in making decisions… no extra understanding is gained.	No	No	Yes	Yes
be validated empirically	Validated	No	Validated	Validated

IV. CONCLUSION AND FUTURE WOTK

This paper assessed various metrics suit for object-oriented software system. Assessment shows that metrics provide guidelines to indicate the progress that a software system has made and the quality of design. Using these guidelines, we can develop more usable and maintainable software system to fulfill the demand of efficient system for software applications. By observing the growing popularity of object-oriented software, we are going to develop a model, which will predict the usability and maintainability of object-oriented software system in efficient manner. Since collecting and analyzing the data, design quality of software can predict easily, so after

developing the model it will be test for suitability to fit in object-oriented scenario, on the basis of analysis of the data.

REFERENCES

[1] B. Delatte, M. Heitz, and J. F. Muller, HOOD Reference Manual 3.1, Masson, Paris, 1993.

[2] B. Unger and L. Prechelt, The impact of inheritance depth on maintenance tasks – Detailed description and evaluation of two experimental replications, Technical Report, Karlsruhe University: Karlsruhe, Germany, 1998.

[3] F. B. Abreu and R. Carapua, "Candidate Metric for OOS within taxonomy framework, Journal of System & Softwrae, Vol. 26, No. 1, July 1994.

[4] F. B. Abreu, "The MOOD Metrics Set", In Proc. ECOOP'95, Workshop on Metrics, 1995.

[5] G. Booch, Object-oriented analysis and design, Benjamin-Cummings, U.S.A, pp.107-215, 1994.

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

[7] G. Poelsand and G. Dedene, "Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models", 5th European Conference on Software Maintenance and Reengineering (CSMR 2001), Lisbon, Portugal, 2001, pp. 20-29.

[8] H. Sneed, Encapsulating Legacy Software for Reuse in Client/Server Sstem, In proceedings of WCRE-96, IEEE press, 1996, Monterey.

[9] I. Jacobson, Object-Oriented Software Engineering, Addison-Wesley, 1992 .

[10] J. Bansiya and C.G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, Vol. 28, No. 1, 2002.

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

[12] J. M. Bieman, and B. K. Kang, "Measuring Design-Level Cohesion", IEEE Transactions on Software Engineering, Vol. 24, No. 2, pp. 111-124, 1998.

[13] J. Pinson Lewis and Richard S. Wiener, An Introduction to Object-oriented Programming and Smalltalk, Addison- Wesley pp 49-60, 1988.

[14] J. Rumbaugh, M. Blaha, W. Lorensen, F. Eddy, and W. Premerlani, Object-Oriented Modeling and Design, Prentice-Hall, 1991

[15] L. C. Briand, S. Morasca and V. Basili, "Property-Based Software Engineering Measurement", IEEE Transactions on Software Engineering, Vol. 22, No. 6, pp. 68-86, 1996.

[16] L. C. Briand, J. W. Daly, V. Porter, and J. Wust, A Comprehensive Empirical Validation of Product Measures for Object-Oriented Systems. Technical Report, ISERN-98-07, 1998.

[17] L. C. Briand, J. W. Daly and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Transactions on Software Engineering, Vol. 25, No. 1, pp. 91–121, 1999.

[18] L. C. Briand, J. W. Daly, V. Porter, and J. Wust, "Exploring the Relationships Between Design Measures and Software Quality in Object Oriented Systems", Journal of Systems and Software, Vol. 51, No. 3, pp. 245-273, 2000.

[19] L. C. Briand and J. Wust, "The Impact of Design Properties on Development Cost in Object-Oriented Systems", Proc. 7th Int'l Software Metrics Symposium (METRICS 01), IEEE CS Press, 2001.

[20] L. C. Briand, W. L. Melo and J. Wust, "Assessing the Applicability of Fault Proneness Models Across Object-Oriented Software Projects", IEEE transactions on Software Engineering, Vol. 28, No. 7, 2002.

[21] L. H. Rosenberg and L. Hyatt, "Software Quality Metrics for Object-Oriented Environments", Crosstalk Jounal, 1997.

[22] L. Prechelt, B. Unger, M. Philippsen and W. Tichy, "A controlled experiment on inheritance depth as a cost factor for code maintenance", The Journal of Systems and Software, Vol. 65, 2003, pp. 115-126.

[23] M. Alshayeb, and M. Li, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes", IEEE Transactions on Software Engineering archive, Vol. 29, 2003, pp.1043 – 1049.

[24] M. Cartwright, An Empirical view of inheritance, Information and Software Technology, Vol. 40, No. 4, 1998, pp. 795-799.

[25] M. El Wakil, A. El Bastawissi, M. Boshra and A. Fahmy, Object-Oriented Design Quality Models – A Survey and Comparison. 2nd International Conference on Informatics and Systems, 2004.

[26] M. G. Bocco, M. Piattini and C. Calero, "A Survey of Metrics for UML Class Diagrams", Journal of Object Technology, Vol. 4, 2005, pp. 59-92.

[27] M. Lorenz and J. Kidd, Object-Oriented Software Metrics, Prentice Hall, 1994.

[28] M. Tang, M. Kao and M. Chen, An Empirical Study on Object-Oriented Metrics, 6th IEEE International Symposium on Software Metrics, 1998.

[29] N. E. Fenton and S. L. Peeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing Company, Boston, Massachusetts, USA, 1997.

[30] P. Coad and E. Yourdon, Object-Oriented Analysis, Yourdon Press, Prentice Hall, New Jersey, 1990.

[31] P. Coad and E. Yourdon, Object-Oriented Design, Yourdon Press, Prentice Hall, New Jersey, 1991.

[32] R. Harrison, S. Counsell and R. Nithi, "Experimental Assessment of the Effect of Inheritance on the Maintainability of Object-Oriented Systems", The Journal of Systems and Software, Vol. 52, 2000, pp. 173-179.

[33] R. Harrison, S. Counsell and V. Reuben, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics", IEEE Transactions on Software Engineering, Vol. 24, No. 6, pp. 491-496, 1998.

[34] R. Subramanya and M. S. Krishnan, "Empirical of CK Metrics for Object-Oriented Design Complexity: Implication for Software Defects", IEEE Transaction on Software Engineering, Vol. 29, 2003, pp. 297-310.

[35] R. W. Selby and V. R. Vasili, "Analyzing Error-Prone Systems Structure", IEEE Transactions on Software Engineering, Vol. 17, 1991, pp. 141-152.

[36] R. Wirfs-brock, B. Wilkerson, and L. Weiner, Designing Object-Oriented Software, Prentice-Hall, 1990.

[37] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476–493, 1994.

[38] 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, Vol. 24, No. 8, pp. 629-637, 1998.

[39] T. Reenskaug, E. Andersen A. Berre, A. Hurlen, A. Landmark, O. Lehne, E. Nordhagen, E. Ness-Ulseth, G. Oftedal, A. Skaar, and P. Stenslet , "OORASS: seamless support for the creation and maintenance of object oriented systems", Journal of Object Oriented Programming, Vol. 5, No. 6, 1992, pp. 7-41.

[40] The Object Agency, A comparison of Object–Oriented Development Methodologies, 1996. http://www.toa.com.

[41] V. R. Basili, L. C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators". IEEE Transactions on Software Engineering, Vol. 22, No. 10, pp. 751-761, 1996.

[42] W. Li, and S. Henry, "Object-Oriented Metrics that Predict Maintainability". Journal ofSystems and Software, Vol. 23, No. 2, pp. 111-122, 1993.

[43] W. Li, "Another Metric Suite for Object Oriented Programming", The Journal of Systems and Software, Vol. 44, No. 2, pp. 155-162, 1998.

AUTHORS PROFILE

Sanjay Kumar Dubey is Assistant Professor in the Department of Computer Science and Engineering in Amity University Uttar Pradesh, India. His research area includes Software Engineering and Usability Engineering. He is pursuing his Ph. D. in Computer Science and Engineering from Amity University, since July, 2008.

Prof. (Dr.) Ajay Rana is Professor and Director, ATPC, Amity University Uttar Pradesh, India. He is Ph. D. (2005) in Computer Science and Engineering from Uttar Pradesh Technical University, India. His research area includes Software Engineering and Software Testing. He has published more than 40 research papers in reputed National & International Journals. He is the author of several books in the area of computer science. He has received numbers of best papers/case studies medals and prizes for his work.