International Journal of Software Engineering and Its Applications
Vol.8, No.2 (2014), pp.173-188
http://dx.doi.org/10.14257/ijseia.2014.8.2.17

A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment

K.P. Srinivasan[1] and Dr. T.Devi[2]

[1]*Associate Professor of Computer Science*
C.B.M. College, Kovaipudur, Coimbatore – 641 042, India
[2]*Head, Department of Computer Applications*
School of Computer Science and Engineering
Bharathiar University, Coimbatore – 641 046, India
[1]*kpsriznivasanmail@gmail.com, tdevi5@gmail.com*

Abstract

In software engineering, almost for the past three and a half decades, software measurements and metrics have been the subject of a variety of criticisms and many software metrics are proposed and given with inadequate methods for implementation and verification of results. All the engineering systems except software engineering have used the measure and measurement systems in day to day activities for the production of their product. In order to measure process efficiency and product effectiveness in software engineering, this research paper introduces the procedure based metrics system for software measurement and proposes complete and comprehensive object-oriented design metrics for measuring the quality of the design. The proposed set has been formulated drawing upon the more significant properties and attributes. The proposed six metrics set will be more comprehensive and complete than that of any one of the previous sets. The set of design quality attributes measured by the suite of metrics are functionality, understandability, effectiveness, flexibility, extendibility and reusability. The set of design properties measured using the metrics are encapsulation, inheritance, coupling, cohesion and complexity. Each proposed metric has a range of 0 to 1 and desired values for measuring the design and hence, this is result-oriented metrics. The computation formulae for design metrics have been proposed. The proposed metrics set and methodology is complete since it is a result based metrics set. When compared with previously defined metrics set, this metrics introduces improvements in methodology of execution and gives the solution for the usage of metrics in software engineering.

Keywords: *Software Quality Assessment, Software Measurement Procedure, Software Quality Attributes, Result Based Software Metrics, Object-Oriented Metrics, Software Metrics*

1. Introduction

The software metrics is an important research topic in software engineering [7, 10, 11, 15]. The software measurement and metrics gives better results for software development but, this research faces difficulties and criticisms due to hardship in proving the work and methodology [18, 22]. It is time to propose new software metric for software development process and product. Presently, in software development, object-oriented software development is mostly used and to assess the Object-Oriented Design (OOD) during design phase, a new set of object-oriented metrics for quality attributes of object-oriented design has been proposed here. The object-oriented design metrics and software measurement have been

ISSN: 1738-9984 IJSEIA

discussed in Section 2. In Section 3, a comprehensive set of object-oriented metrics for design quality attributes has been proposed. The proposed metrics suite will allow calculating the metrics values in an easy manner when applied in object-oriented design. These proposed metrics suites are formulated drawing upon the most significant characteristics of object-orientation and it will be more comprehensive and complete than that of any of the software metrics sets already proposed by researchers. In Section 4, the procedure based metric system is discussed in detailed and introduced for the execution of proposed set of object-oriented design metrics suite, quality attributes, design properties, range and desired values for object-oriented design for easy evaluation. In Section 5, the analysis of empirical study and results of proposed metrics in three different projects are discussed. In Section 6, the proposed metrics are compared with metrics set proposed by eminent researchers in object-oriented software metrics and Section 7 concludes with the merits of comprehensive metrics set.

2. Object-Oriented Metrics and Software Measurement

The metrics proposed by eminent researchers are called C-K metrics [5, 15, 19], MOOD metrics [1, 9, 15, 21], L-K metrics [13, 15], QMOOD metrics [3] for object-oriented design, Halstead metrics [8] and McCabe's metrics [14] for traditional software. The difficulties faced by metrics researchers are after software crisis period and mostly aim on proving their soundness of software metrics in software engineering and further, researchers are trying to move towards applied software metrics in software industry applications and to prove the usefulness of software metrics in software industry. A vast literature review has been conducted for this research [1-24] and it is identified that software metrics research faced more difficulties towards proving usefulness in industry, theoretical validity, empirical validity, defining precise metrics, understanding, methodology of execution, execution time is more to find the metrics values, metrics are executed only by experts, and accuracy on results.

The software measurement is a mechanism for characterizing, evaluating, and predicting various software processes and products. The only way to improve any software process is to measure specific attributes of the process and develop a set of metrics based on quality attributes and then use the metrics to provide indicators that will lead to strategy for improvement. Software measurement plays an important role in understanding and controlling software development processes and products in software engineering. A software organization establishes a software measurement program for many reasons. Those include collecting good management information for guiding software development and developing innovative and advanced techniques. In general, many of the failures are caused by the inherent complexity of the software development process, for which there often is no sound explanation. The problems can be ameliorated however, by applying software metrics. This requires both the development of improved software metrics and improved utilization of such metrics. Unfortunately, the current state of software metrics poses many questions and difficulties in the states of their "usefulness, application methodology, easy understanding and result oriented" [22]. This research paper identifies the concepts to improve the software metrics for object-oriented design in software engineering.

3. Comprehensive Object-Oriented Design Metrics Suite

At present, object-oriented design metrics (OOD Metrics) play a role in object-oriented software development particularly in design phase because design determines the structure of the software and finishing product. In general, in software development

cycle, once the design has been completed, it is a difficult task to change the design and modifying the design from used design is time consuming and expensive. Hence, high quality of design based on metrics evaluated design gives accurate final product and reduces software cost in software development. It does not only cause the cost of its own creation, but it also influences the cost of the following phases on coding, testing, and maintenance phases [2, 4]. During the software development cycle, early well software design is good for trying to make changes and extensions in the maintenance phase. In order to create a quality of object-oriented design, a quantity of quality assessment attribute is required in design phase. In past research, a strong correlation between design metrics and maintainability of systems has been identified. In order to improve software design in design phase, design measurement based on software metrics is important and vital in software development. At present, improving quality of software product, performance and productivity has become a primary goal of almost every software industry and organization. The process of developing software, maintaining and modification of developed systems has in many cases have been poorly implemented, resulting in time and cost overruns [3, 10] Based on the above view, in order to measure the design, this research paper proposes a comprehensive and complete set of metrics for object-oriented design based on research study, experience and discussions with the software experts [6, 18-23]. The proposed set of object-oriented design metrics is shown in Figure 1. The proposed OOD metrics are explained below with important viewpoints.

3.1. Metric 1: Methods-Per-Class Factor (MPCF)

The *Method-Per-Class Factor (MPCF*) is defined as the ratio of the *Number of Public Methods (NPM)* to the sum of the *Number of Public Methods (NPM)* and *Number of Non Public Methods (NNPM)* in the class. *Method-Per-Class Factor* excludes inherited methods. Since influence of the inherited methods is taken into account later in the MIF metric (Metric 3), they are not included in the count for MPCF. A large value of MPCF shows that ① the class may have too much functionality, ② the reusability of the class will increase, and ③ implementation of methods is good.

$$MPCF = \frac{NPM}{NPM + NNPM}$$

3.2. Metric 2: Attributes-Per-Class Factor (APCF)

The Attribute-Per-Class Factor (APCF) is defined as the ratio of the Number of Private (Protected) Attributes (NPA) to the sum of the Number of Private Attributes (NPA) and Number of Non Private Attributes (NNPA) in the class. Attribute-Per-Class Factor excludes inherited attributes. Since influence of the inherited attributes is taken into account later in the AIF metric (Metric 4), they are not included in the count for APCF. A large value of APCF indicates that ① the object has more properties in it, ② the high potential impact on children, and ③ the time and effort of the construction of a class.

$$APCF = \frac{NPA}{NPA + NNPA}$$

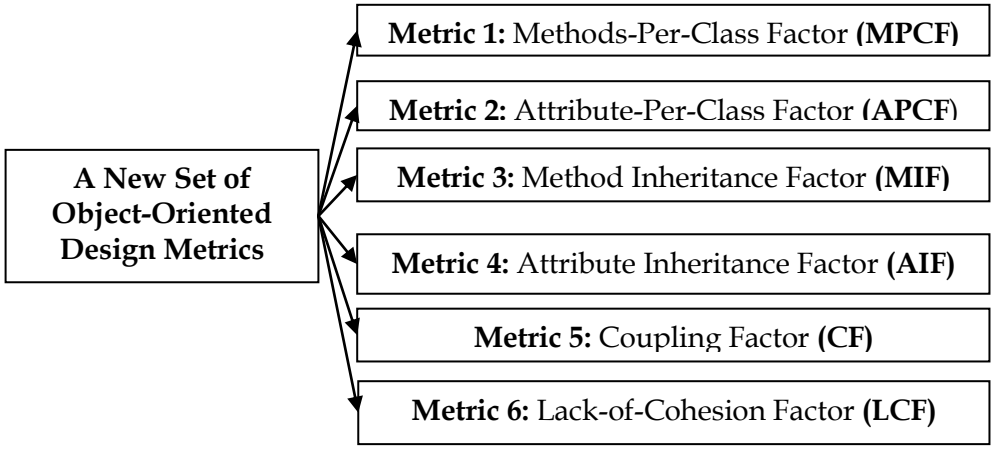

Figure 1. Comprehensive Set of Object-Oriented Design Software Metrics Suite

3.3. Metric 3: Method Inheritance Factor (MIF)

The *Method Inheritance Factor (MIF)* is defined as the ratio of the *Number of Inherited Methods (NIM)* to the sum of the *Number of Inherited Methods (NIM)* and the *Number of Defined Methods (NDM)* in the class. A large value of MIF indicates that ① more difficult to predict the behavior of the class, ② violated the abstraction implied by the super class, and ③ generally formed a designing difficulty.

$$MIF = \frac{NIM}{NIM + NDM}$$

3.4. Metric 4: Attribute Inheritance Factor (AIF)

The *Attribute Inheritance Factor (AIF)* is defined as the ratio of the *Number of Inherited Attributes (NIA)* to the sum of *Number of Inherited Attributes (NIA)* and the *Number of Defined Attributes (NDA)* in the class. A large value of AIF indicates that the ① subclass have design problem, ② the subclass abstraction is not of high quality, and ③ the testing will be difficult.

$$AIF = \frac{NIA}{NIA + NDA}$$

3.5. Metric 5: Coupling Factor (CF)

NAC is the *Number of Actual Couplings* with other classes and NPC is the *Number of Possible Couplings* of this class with other classes of the system. Clearly, the number of possible couplings of a class with other classes of the system is one less than the number of classes. Coupling Factor for a class is defined as Number of other classes to which coupled / (Number of classes – 1). Since, inheritance is already considered in MIF (Metric 3) and AIF (Metric 4) metrics, inheritance is excluded in determining the couplings. A large value of CF indicates that ① the testing will be difficult, ② maintenance will be difficult and will lead to higher defects, and ③ it is not easier to reuse the class in another application.

$$CF = \frac{NAC}{NPC}$$

3.6. Metric 6: Lack-of-Cohesion Factor (LCF)

NDMP is the *Number of Dissimilar Method Pairs* in the class and NPMP is the *Number of Possible Method Pairs* in the class. If two methods access one or more common attributes, then these two methods are similar. And if two methods have no commonly accessed attribute, then these two methods are dissimilar. When there are many similar method pairs in a class, then there is a good cohesion in the class. Lack-of-Cohesion is defined as if m is the number of methods in the class, then the number of possible method pair is m (m-1)/2. A large value of LCF indicates that ① a class is not desirable, ② classes should probably be split into more sub classes, and ③ increases complexity of the development process.

$$LCF = \frac{NDMP}{NPMP}$$

4. New Procedure for Comprehensive Object-Oriented Metrics

A procedure based metrics system for object-oriented design metrics has been proposed with simple steps and clear methodology. The design experts of a particular domain can design a formal object-oriented design for the betterment of quality of the software and it is very difficult to measure the object-oriented design quality. The software metrics proposed by previous researchers were without any procedure for execution of the metrics. Hence, it creates problem in understanding and execution. If executed, it creates ambiguity on execution of software metrics. In order to avoid the main criticism, this research paper proposes a procedural based approach [18, 20, 22] for object-oriented design metrics. This procedure adopts and proposes the quality attributes, design properties and desired values of the object-oriented design. The procedure based metrics system is used to find the effectiveness of the object-oriented design based on design properties such as quality attributes relationships of the design. Figure 2 gives the procedure for the execution of object-oriented design metrics. This procedure yields metric values of each class and finds the design properties. Each of the execution steps is explained in a detailed manner.

Step 1: Select the object-oriented design to measure the quality and effectiveness.
Step 2: Select the quality attributes of the object-oriented design to measure the particular domain.
Step 3: Identify the design properties of an object-oriented design for identifying and related to the quality attributes selected in step 2.
Step 4: Form the related object-oriented design metrics and desired values to quantify the design properties of step 3 and find design metrics-quality attribute relationship.
Step 5: Calculate the metric value of each metric and tabulate their values for each class of the entire system for easy operations and find effectiveness of the design.
Step 6: Find the design attribute effectiveness using the metric values obtained from step5 and further using computation formula. Analyze and check the design attributes using the desired values and design properties weights.
Step 7: Closely examine the design attributes from computation formula for quality attributes of class and metrics values, modify the individual classes if necessary.

Figure 2. Procedure Based Metrics System for Design Metrics Suite

Execution of Step 1: Select the formal object-oriented design for measuring the quality effectiveness. In this step, selected object-oriented designs are measured using the quality attributes of final product that is object-oriented design.

Execution of Step 2: Select the quality attributes of the object-oriented design to measure the particular domain. The set of design quality attributes measured by the proposed suite of metrics of the Section 3 are: Functionality, Understandability, Effectiveness, Flexibility, Extendibility and Reusability.

Figure 3 shows the design quality attributes related to object-oriented design and these attributes are measurable easily using proposed metrics set of section 3. The definitions of the object-oriented design quality attributes illustrated in Figure 3 are given below:

Functionality: The operations carried and assigned to the classes of a design.

Understandability: This attribute enable to find the degree of understanding of the design work.

Effectiveness: This refers to a designer's ability to achieve the desired requirements.

Flexibility: This attribute refers to characteristics of the incorporation of changes in the design.

Extendibility: This attribute refers to the incorporation of new requirements in the existing design.

Reusability: This refers to the characteristics of reuse of the already available design.

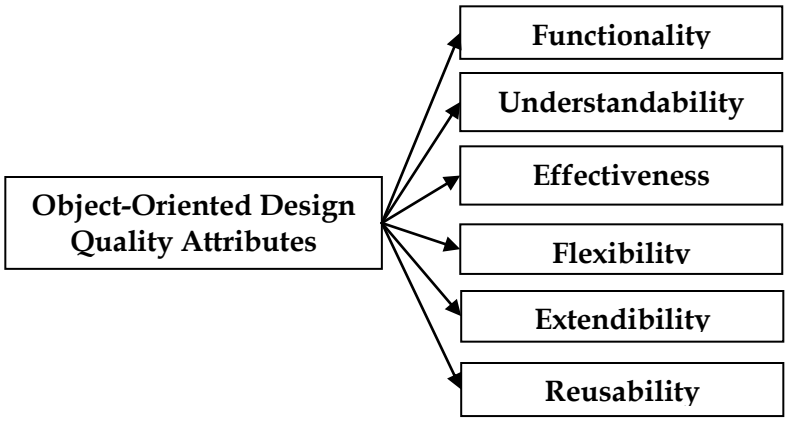

Figure 3. Object-Oriented Design Quality Attributes

Execution of Step 3: Identify the design properties of an object-oriented design and relate to the quality attributes of the design selected in step 2. This step forms the design properties for measuring quality of an object-oriented design. The set of design properties measured using the proposed metrics are: Encapsulation, Inheritance, Coupling, Cohesion and Complexity. Figure 4 shows the design properties of object-oriented design and these properties are measurable easily using proposed metrics set. The definitions of the object-oriented design properties given in Figure 4 are:

Encapsulation: This refers to the enclosing of data and method within a single construct.

Inheritance: This refers to relationship related to the level of nesting of classes.

Coupling: This refers to interdependency of the object on another object.

Cohesion: This refers to assess the relatedness of methods and attributes of the class.

Complexity: This refers to degree of difficulty of designing the design.

Execution of Step 4: Form the object-oriented design metrics and their desired range values to quantify the properties for the step 3 and find design metrics-quality attribute relationship. This step uses the proposed object-oriented design metrics given in section 3. Table 1 gives the relationships to design properties selected in step 3 and selected object-oriented design metrics of step 4. The Table 2 shows the proposed range of values and desired values of object-oriented design metrics and Table 3 gives the design metrics – quality attributes relationships. In order to identify the design properties that would yield a particular quality attribute, a detailed study and extensive analysis have been carried out from research work of S.R.Chidamber and C.F. Kemerer (1994), B. Kitchenham, S.L. Pfleeger and N. Fenton (1995), V.R. Basili, L.C. Briand and W.L. Melo (1996), J. Bansiya and C.G. Davis (2002), N.E. Fenton and Pfleeger (2004), Kan, S.H(2006), K.P. Srinivasan and T. Devi (2009, 2011, 2014) and R. Vir and P.S. Mann (2013) [3- 5, 7, 12, 18, 20, 22-24]. Table 3 gives the identification of each of the design properties on the quality attributes and a multiplication symbol mark **X** indicates that the design property has weight on the quality attribute.

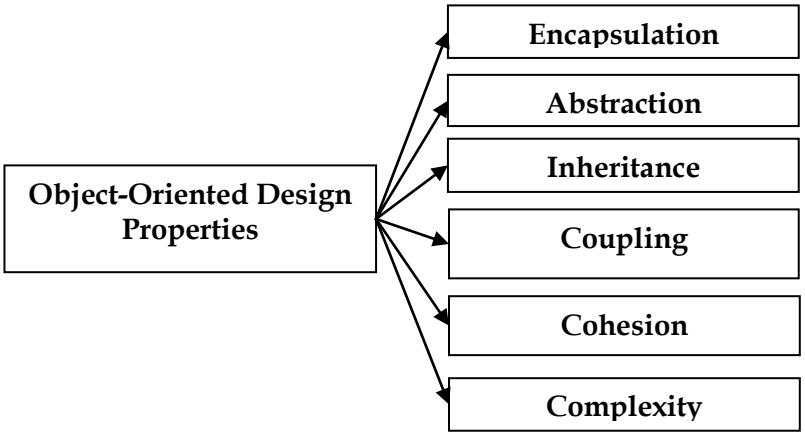

Figure 4. Object-Oriented Design Properties

Execution of Step 5: Calculate the object-oriented design metric values of each metric and tabulate the values for each class of the entire system for easy usage and find quality effectiveness of the design. Apply the metrics on the object-oriented design for quality measurement and get the values from the design and tabulate the values received from the measurement. Based on the metric values, the next step 6 calculates the design attributes quality effectiveness of the design.

Execution of Step 6: Find the quality of the design using the metric values and check the desired values and range of values and based on metric values, find the quality of the design metrics – quality relationships. A good system design will result in a value closer to desired value for each of the metric values of class and if a class has a closer desired value then the design needs not to be revised and improved. This step evaluates the quality of design using the desired values which are defined by the design experts for a particular domain environment and application. The desired values given in Table 2 are acceptably good.

Table 1. Design Metrics for Design Properties of the Design

Design Property	Design Metrics
Complexity	MPCF
Encapsulation	APCF
Inheritance, Abstraction	MIF
Inheritance, Abstraction	AIF
Coupling	CF
Cohesion	LCF

Table 2. Ranges and Desired Value for Design Metrics

Design Metric	Range of Metric	Desired Value
MPCF	0 to 1	1
APCF	0 to 1	1
MIF	0 to 1	0
AIF	0 to 1	0
CF	0 to 1	0
LCF	0 to 1	0

Table 3. Design Metrics – Quality Attributes Weighted Relationships

Design Metric	Functionality	Understandability	Effectiveness	Flexibility	Extendibility	Reusability
MPCF	X					X
APCF		X	X	X		
MIF			X		X	
AIF			X		X	
CF	X	X		X		X
LCF		X	X			X

The design metric to quality attribute relationship shows the relative significance of design properties that influence a quality attribute of the design. Table 2 shows the desired and range values of each metric. The Tables 4 and 5 show the computation formula for quality attributes and desired values of computation formula for quality attributes respectively. The computation formula for quality attribute and the corresponding desired value will help in identifying the quality of attributes.

Execution of Step 7: Closely examine the quality of the design of class based on desired value and computed formulas values and modify the individual classes if necessary. After checking the design with desired values of each class, it must be improved if any class is if unacceptable desired values. Then those classes are closely examined and improved if possible for better design.

Table 4. Computation Formula for Quality Attributes

Quality Attribute	Computation Formula
Functionality	(MPCF+(1-CF))/2
Understandability	(APCF+(1-CF)+(1-LCF))/3
Effectiveness	(APCF+(1-MIF)+(1-AIF)/3
Flexibility	(APCF+(1-CF))/2
Extendibility	((1-MIF)+(1-AIF))/2
Reusability	(MPCF+(1-CF)+(1-LCF))/3

Table 5. Quality Attributes, Range and Desired Values of Computation Formula

Quality Attributes	Range of Attribute	Desired Value
Functionality	0 to 1	1
Understandability	0 to 1	1
Effectiveness	0 to 1	1
Flexibility	0 to 1	1
Extendibility	0 to 1	1
Reusability	0 to 1	1

This section proposes the procedure for execution of proposed metrics in order to measure the effectiveness of design. This section also proposes the quality attributes, design properties, range and desired values of each metric and relationships of design metrics-quality attributes. Any object-oriented designer or metrics expert can check their designs for better development of software. This section gives the strong and unambiguous execution steps and methodology of execution on software measurement field for improvement of usage of software metrics in software industry and organization. Next section explains the empirical study, analysis and results of proposed metrics.

5. Empirical Study, Analysis and Results

The implementation of empirical studies is conducted for three different projects for object-oriented design metrics. Project 1 is software that assists Air Gourmet in deciding whether to continue supplying special meals to passengers who request them. The product will allow the client to enter a reservation, and generate the information needed to administer the special meal program. Project 2 is a software that assists a company named Martha Stockton Greengage Foundation in making decisions whether to give loan to a couple by keeping property on mortgage [16, 17]. Project 3 is an object-oriented system called Trader System. The projects are referred here as Project 1, Project 2 and Project 3. The descriptive statistics such as Minimum (Min), Maximum (Max), Mean, Median and Standard Deviation [6] are calculated for object-oriented design metrics. The computation formula for quality attributes of project 1, project 2, and project 3 are calculated. The following observations are made from applying the metrics on project 1. The Table 6 provides common descriptive statistics for the metric distributions of Project 1. Figure 5 shows the distributions of analyzed object-oriented design metrics for project 1. The MPCF and APCF values are medium in the project 1. This shows that attributes and methods of a class are at medium level and functionality, reusability are a minimum and properties of the classes on impact of class through inheritance is medium. The MIF and AIF values are medium in project 1. This shows that inheritance used in all the classes of project1 are medium level and abstraction of the classes is maintained and testing time is normal. The MIF value is average for project 1 is observed and that shows that there are very less methods in a super class. The MIF and AIF measures can provide a measure of information hiding incorporated by software designers. The value of AIF is medium suggesting medium use of inheritance. The CF measures the complexity of the software by counting the number of classes and coupling of the classes to other classes and CF value is less in project 1, hence classes are easy to understand and maintain. CF has minimum level indicating that classes are declared without other class's accesses. The LCF values are maximum level because the number of dissimilar pairs of methods having access to common attributes is more than the number of pairs of method having common attributes. It implies that classes are less cohesive.

Table 6. Descriptive Statistics of Project 1 Design Metrics

Metric	Min	Max	Mean	Median	Standard Deviation
MPCF	0.3	1.0	0.5	0.4	0.2
APCF	0	1.0	0.5	0.9	0.5
MIF	0	0.8	0.4	0.3	0.4
AIF	0	1.0	0.5	0.6	0.4
CF	0.1	0.5	0.2	0.2	0.2
LCF	0.7	1.0	0.9	0.9	0.1

The Table 7 provides the descriptive statistics of computation formula of Project 1. The Figure 6 shows the distributions of analyzed computation formula result of project 1 compared with project 2 and project 3. The understandability, effectiveness, extendibility and reusability values are average in the project 1 attributes. This shows that declared attributes and methods of the classes are of medium level. The functionality and flexibility is good and maximum compared to all other attributes of project 1. The computation formula of attributes and design metrics of project 1 are average values that reflect that project 1 is normal in object-oriented design. The Table 8 provides descriptive statistics for the metric distributions of Project 2 and Figure 5 shows the distributions of analyzed object-oriented design metrics for all projects.

Table 7. Descriptive Statistics of Project 1 Computation Formula

	Functionality	Understandability	Effectiveness	Flexibility	Extendibility	Reusability
MIN	0.4	0.1	0.1	0.2	0.1	0.3
MAX	1.0	0.7	0.9	0.9	1.0	0.9
MEAN	0.7	0.5	0.6	0.7	0.6	0.5
MEDIAN	0.6	0.6	0.7	0.9	0.5	0.5
STD.	0.2	0.3	0.3	0.3	0.4	0.2

Table 8. Descriptive Statistics of Project 2 Design Metrics

Metric	Min	Max	Mean	Median	Standard Deviation
MPCF	0.8	1.0	1.0	1.0	0.1
APCF	0	1.0	0.6	1.0	0.5
MIF	0	0.4	0.1	0.3	0.2
AIF	0	0.4	0.2	0.4	0.2
CF	0	0.8	0.3	0.3	0.3
LCF	0	1.0	0.6	0.7	0.3

The MPCF value is high and APCF, LCF values are medium for project 2. This shows that methods of a class are high and functionality and reusability are maximum. The MIF, AIF, and CF values are low in value for project 2. This shows that minimum level of inheritance is used in all the classes of the project 2. As the MIF value is low for project 2 there are very few methods in a super class. The MIF and AIF measures can provide the amount of information hiding incorporated by software designers. The low AIF value indicates a medium use of attribute inheritance. The CF measures the coupling of the classes to other classes and CF value is low in project 2, hence classes are easy to understand and maintain. The LCF values are of medium level because the number of dissimilar pairs of methods having access to common attributes is more than the number of pairs of methods having common attributes. It implies that classes are less cohesive. Table 9 provides the descriptive statistics of computation formula of Project 2 and Figure 6 shows the distributions of analyzed computation formula result of project 2 compared with project 1 and project 3. The understandability, effectiveness, and reusability values are medium in the project 2. The functionality and flexibility is medium and extendibility is maximum value in project 2. The

results of computation formula of attributes of project 2 design reflect that project 2 is normal in object-oriented design. Table 10 provides common descriptive statistics for the design metric distributions of Project 3 and Figure 5 shows the distributions of analyzed mean values of object-oriented design metrics for project 3. The APCF value is maximum for project 3. This shows that the properties of the classes and impact of class through inheritance are high.

Table 9. Descriptive Statistics of Project 2 Computation Formula

	Functionality	Understandability	Effectiveness	Flexibility	Extendibility	Reusability
MIN	0.6	0.1	0.7	0.5	0.6	0.4
MAX	1.0	0.8	1.0	1.0	1.0	1.0
MEAN	0.8	0.6	0.8	0.7	0.9	0.7
MEDIAN	0.9	0.7	0.7	0.5	1.0	0.7
STD.	0.1	0.3	0.1	0.3	0.2	0.2

Table 10. Descriptive Statistics of Project 3 Design Metrics

Metric	Min	Max	Mean	Median	Standard Deviation
MPCF	0.3	1.0	0.8	1.0	0.3
APCF	0.7	1.0	0.9	0.9	0.1
MIF	0.4	1.0	0.7	0.8	0.3
AIF	0	0.9	0.7	0.8	0.3
CF	0	0.8	0.2	0.1	0.3
LCF	0	1.0	0.3	0.7	0.4

The MPCF, MIF and AIF values are above average value for project 3. This shows that inheritance used in the classes of project 3 are of average and medium level. MIF value of Project 3 is above average and it shows that there are a good number of methods in a super class. The MIF and AIF measures can indicate the amount of information hiding. The value of AIF being medium suggests that there is medium use of attribute inheritance. For project 3, CF measures are low compared to all other metrics values. CF value is less in project 3, hence classes are easy to understand and reuse. Low LCF value indicates that classes are cohesive in Project 3. The Table 11 provides the descriptive statistics of computation formula of Project 3 and Figure 6 shows the distributions of analyzed mean values for project 3. The understandability, flexibility, and reusability values of Project 3 are high and these attributes indicate that the design is good. The functionality and effectiveness are medium for Project 3. The extendibility is low value in Project 3 compared to all other attributes of the project 3. The design attributes values of project 3 reflect that the project 3 is good in object-oriented design. The summary statistics of mean values of design metrics for Project 1 to Project 3 are shown in Figure 5. It shows all the corresponding metric values of the projects. In Figure 5, the metrics value of classes is on Y-axis and the defined metrics MPCF, APCF, MIF, AIF, CF, and LCF are given in X-axis.

Table 11. Descriptive Statistics of Project 3 Computation Formula

	Functionality	Understandability	Effectiveness	Flexibility	Extendibility	Reusability
MIN	0.5	0.6	0.3	0.9	0.1	0.6
MAX	1.00	1.0	0.7	1.0	1.0	1.0
MEAN	0.8	0.9	0.5	0.9	0.4	0.9
MEDIAN	0.9	0.9	0.5	0.9	0.3	0.8
STD.	0.2	0.1	0.1	0.1	0.3	0.1

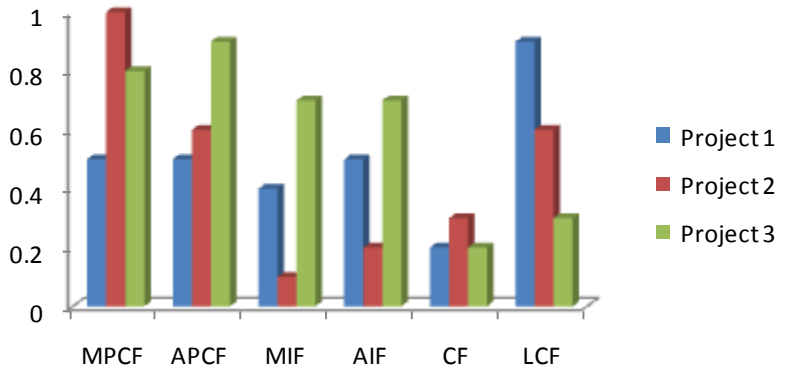

Figure 5. Analyzed Design Metrics of Project 1, Project 2 and Project 3

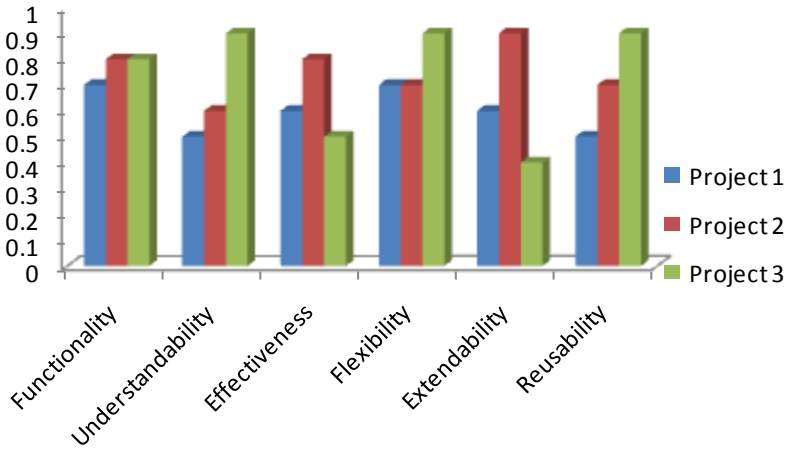

Figure 6. Metric Values of Project 1, 2 and 3 Attributes

6. Comparison and Achievements of Comprehensive Metrics Suite

The comparison of proposed metrics for object-oriented design with C-K metrics [5, 15, 19], MOOD metrics [1, 9, 21], L-K metrics [14, 15] and QMOOD metrics [3] are shown in Table 12. As, it can be seen from Table 12, that the proposed metrics are better than the existing metrics in characteristics such as level of understanding and quantifiability of results.

Table 12. Comparison of Comprehensive Object-Oriented Design Metrics

Description	C-K Metrics	MOOD Metrics	L-K Metrics	QMOOD Metrics	Comprehensive Metrics
Level of Understanding	Moderate	Moderate	Moderate	Moderate	Easy
Procedure for Calculation	No	No	No	No	Yes
Quantifiability of Results	Partial	Partial	Partial	Partial	High
Computation Formula	No	No	No	Yes	Yes
Range Values Proposed	No	Available	Partially Available	Partially Proposed	Completely Proposed
Desired Value Proposed	Not Mentioned	Available	Partially Available	Partially Proposed	Completely Proposed

The proposed procedure based object-oriented design metrics system has the following achievements and improvements over previous metrics available in literature: ①The complete metrics set having the range and desired values for measuring the design. ②The range value for each metric is between 0 and 1 hence result oriented.③The computation formula values between 0 and 1 hence result oriented. ④The quality attributes of design are given for quality assessment of the design. ⑤These metrics viewpoints are given for assessment. ⑥These metrics can be used for small, medium and large designs. ⑦The procedure followed is simple, clear, understandable, unambiguous and consistent. ⑧The procedural approach is given for execution of software metrics in easy manner. ⑨The metric set is more comprehensive and complete hence major properties such as abstraction, encapsulation, inheritance, complexity, coupling and cohesion are analyzed. ⑩The metrics values are easily obtainable hence student can check their design.

7. Conclusion

This research paper proposes a comprehensive set of six object-oriented design metrics for measuring design and it also proposes the design attributes, design properties, desired metric values, and range of metrics values for object-oriented design. These proposed metric set will be more comprehensive, complete and quickly measure the properties of object- oriented design. This paper has introduced a procedure based metrics system for execution of proposed comprehensive object-oriented metrics and the achievements of proposed design metrics in comparison with the existing metrics have been clearly brought out.

References

[1] F. B. Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality," Proceedings of the 3rd International Software Metrics Symposium, IEEE, Berlin, Germany, **(1996)**.
[2] R. K. Bandi, V. K. Vaishnavi and D. E. Turk, "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics," IEEE Transactions on Software Engineering, vol. 29, no. 1, **(2003)** January, pp. 77-87.
[3] J. Bansiya and C. G. Davis, "Hierarchical Model for Object-Oriented Design Quality Development", IEEE Transactions on Software Engineering, vol. 28, no. 1, **(2002**), pp. 4-17.
[4] 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, **(1996)** October, pp. 751-761.
[5] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design", IEEE Transactions on Software Engineering, vol. 20, no. 6, **(1994)** June, pp. 476-493.
[6] T. Devi and K. P. Srinivasan, "Statistical Techniques in Software Quality Measurement and Metrics", Proceeding of UGC Funded National Conference on Recent Advances in Statistics and Computer Applications, Bharathiar University, Coimbatore, **(2010)** December.
[7] N. E. Fenton and S. L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach", Thomson Asia, **(2004)**.
[8] M. H. Halstead, "Elements of Software Science", Elsevier, **(1977)**.
[9] R. Harrison, S. J. Counsel and R. V. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics", IEEE Transactions on Software Engg., vol. 24, no. 6, **(1998)**, pp. 491-496.
[10] C. Jones, "Applied Software Measurement: Global Analysis of Productivity and Quality", Tata McGraw-Hill Edition, India, **(2008)**.
[11] S. H. Kan, "Metrics and Models in Software Quality Engineering", Pearson Education, India, **(2006)**.
[12] B. Kitchenham, S. L. Pfleeger and N. Fenton, "Towards a Framework for Software Measurement Validation", IEEE Transactions on Software Engineering, vol. 21, no. 12, **(1995)**, pp. 929-943.
[13] M. Lorenz and J. Kidd, "Object-Oriented Software Metrics", Prentice Hall, **(1994)**.
[14] T. J. McCabe, "A Complexity Measure", IEEE Transactions on Software Engineering, vol. se-2, no. 4, **(1976)** December, pp. 308-320.
[15] R. S. Pressman, "Software Engineering A Practitioner's Approach", 5th Ed, Mc-Graw Hill, **(2001)**.
[16] S. R. Schach, "Software Engineering with Java", Tata McGraw-Hill Edition, India, **(1998)**.
[17] S. R. Schach, "Object-Oriented and Classical Software Engineering", Tata MGHill Ed, India, **(2002)**.
[18] K. P. Srinivasan and T. Devi, "Procedure for Selection of an Efficient Object-Orientated Design", Proceeding of UGC Funded National Conference on Recent trends in Software Engineering, Sullamussalam Science College, Mallapuram, Kerala, (Best Paper of the Conference), **(2009)**.
[19] K .P. Srinivasan, T. Devi and V. Thiagarasu, "Analysis of Chidamber - Kemerer Metrics for Object-Orientated Design", Proceeding of National Conference on Emerging trends in Computer Science, Avinasilingam University, Coimbatore, **(2009)**.
[20] K. P. Srinivasan and T. Devi, "Design and Development of a Procedure to Test the Effectiveness of the Object-Oriented Design", International Journal of Engineering Research and Industrial Applications, vol. 2, no. VI, **(2009)**.
[21] K. P. Srinivasan and T. Devi, "A Case Study Approach for Application of MOOD Metrics in Object-Oriented Design", Proceedings of the International Conference on Global Computing and Communication, Hindustan University, Chennai, **(2009)** December.
[22] K. P. Srinivasan and T. Devi, "Design and Development of a Procedure for new Object Oriented Design Metrics", International Journal of Computer Applications, vol. 24, no. 8, **(2011)**, pp. 30-35.
[23] K. P. Srinivasan and T. Devi, "A Novel Software Metrics and Software Coding Measurement in Software Engineering", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 1, **(2014)** January.
[24] R. Vir and P. S. Mann, "A Hybrid Approach for the Prediction of Fault Proneness in Object Oriented Design Using Fuzzy Logic", Journal Academic Industrial Research, vol. 1, no. 11, **(2013)** April, pp. 661-666.

Authors

K. P. Srinivasan received his Master of Computer Applications degree from Bharathiar University, Coimbatore, India in 1993 and M.Phil degree in Computer Science from the Bharathiar University, Coimbatore, India in 2001. Presently, he is working as an Associate Professor of Computer Science in C.B.M. College, Kovaipudur, Coimbatore under Bharathiar University, Coimbatore, India since 1997. He is doing his research work in Software Engineering. He has published five conference papers and five journal papers. His current research interests are in the areas of Software Engineering and Object-Oriented Systems.

Dr T. Devi received Master of Computer Applications Degree from P.S.G. College of Technology, Coimbatore, India in 1987 and the Ph.D. degree from the University of Warwick, United Kingdom in 1998. Presently, she is working as an Associate Professor and Head of the Department of Computer Applications, School of Computer Science Engineering, Bharathiar University, Coimbatore, India. Prior to joining Bharathiar University, she was an Associate Professor in Indian Institute of Foreign Trade, New Delhi, India. She has contributed more than 140 papers in various Journals and Conferences. Her current research interests are in the areas of Software Engineering and Concurrent Engineering.