Journal of Computer Science 1 (3): 437-444, 2005
ISSN 1549-3636
8 2005 Science Publications

Quality Metric Development Framework (^{q}MDF)

[1] K. Mustafa and [2]R. A. Khan
[1]Department of Information Technology, Al Husain Bin Talal University, Jordan
[2]Department of Computer Science, Jamia Millia Islamia, N. Delhi, India

Abstract: Several object-oriented metrics have been developed and used in conjunction with the quality models to predict the overall quality of software. However, it may not be enough to propose metrics. The fundamental question may be of their validity, utility and reliability. It may be much significant to be sure that these metrics are really useful and for that their construct validity must be assured. Thereby, good quality metrics must be developed using a foolproof and sound framework / model. A critical review of literature on the attempts in this regard reveals that there is no standard framework or model available for such an important activity. This study presents a framework for the quality metric development called Metric Development Framework (^{q}MDF), which is prescriptive in nature. ^{q}MDF is a general framework but it has been established specially with ideas of object-oriented metrics. ^{q}MDF has been implemented to develop a good quality design metric, as a validation of proposed framework. Finally, it is defended that adaptation of ^{q}MDF by metric developers would yield good quality metrics, while ensuring their construct validity, utility, reliability and reduced developmental effort.

Keywords: Object Oriented Metrics, Quality Models, OO Paradigm, OO Structure

INTRODUCTION

Several research works in the object oriented design metrics arena were produced in recent years[1-9, 14,17,31-35]. However, widespread adaptation of object oriented metrics in numerous application domains should only take place if metrics may be shown to be theoretically valid, in the sense that they accurately measure the attributes of software for which they were designed to measure and have also been validated empirically[10]. But there is a general agreement among the experts that metrics have not been even validated theoretically and what to talk of experimental validation. Most of the metrics are accepted by practitioners on ‘heavy usages and popularity’ and by academic experts on empirical validation. In such a scenario many of the available metrics may not be used properly and have been discarded. Reasons may be that these could not find a place among practitioners or not found to be valid by experts on empirical findings. Now the question is ‘Why such metrics which could not become acceptable, were developed?’- leading to all the efforts on development going vain. Therefore, there appears a need for such development to be sound enough and backed by valid procedures to avoid such embarrassing situations. That is, a sound and standard framework or model for metrics development may be quite useful. And that it could lead to the development of good quality metrics.

Development of object oriented metrics: It is evident from the review of literature that few of the researchers have proposed the criteria for developing the desired metrics. We could not explore a standard framework or model for designing object oriented metrics. Therefore, it appeared worthwhile for looking at major developments and discover the direct or indirect use of criteria, methods, guideline etc. Some of the major attempts have been critically reviewed (SATC’s Approach[11], Jagdish Bansiya and Carl Devis’s Attempt[2], Kitchenham’s Approach[12], Abreu’s Approach[14] and Victor’s Approach[15]) and the feeling gets confirmed that there are no known comprehensive and complete models or frameworks or concerned approach that may be used to design quality object oriented metrics. In order to further assure the conclusion, a question regarding the availability of standard framework for designing the desired metrics was posed to various researchers and practitioners. Their responses were analyzed and conclusion was drawn that no such framework is available, and no major attempts on its development has been reported. Despite the apparent diversity of object oriented design framework, an opportunity to work out a desired framework for designing the object oriented metrics is knocking. The need to have a framework that may be used for development of quality metrics motivates the proposal of an appropriate framework. However, a set of common and desired features for

development of object oriented metrics can be drawn from the reported attempts on development of metrics.
Applying object oriented metric to new paradigms such as object oriented domain has been a dominant feature of academic research up to the present day[16,17]. The researchers community has developed a considerable number of object oriented metrics. The basic premise behind the development of object oriented metrics is that they can serve as early predictors of classes that contain faults or that are costly to maintain. CK suit is the most referred and most commercially used metrics collection tool available. Dr. Linda Rosenberg[11] at SATC validated the six CK metrics and found that the main features the metrics are incorporating the coverage of all the object oriented concepts. External complexity and internal object structure was found to be the much desired feature for designing the object oriented metrics[11]. Victor Laing[15] with Dr. Linda Rosenberg working for NASA at SATC suggest Orthogonality as one of the important characteristics of the desired metrics to be used for object oriented technology to find the minimal set of metrics. Abreu[18] proposed MOOD set of metrics, which allow the use of the attributes of the object oriented paradigm to be evaluated and reviewed. Abreu strongly suggests that metrics definition and dimension should be justified as it plays an important role in designing the object oriented metrics[14]. System size independence and language independence was also found to be the major contributor for designing the desired metrics[14]. Kitchenham's work in the direction of designing the object oriented metrics reveal that the dimensional consistency and use of correct unit and scale type be the essential feature of the design metrics[12]. It was suggested that the developed metrics must preserve all intuitive notions about the attributes and the way in which the metrics distinguish between entities[13].

Quality metrics: The researchers generally conclude that a good design metrics should possess all the object oriented characteristics. As the basic question regarding the design of constructs, effective use of constructs to decrease the architectural complexity, psychological complexity, application specific design and enhancement tests through structure may be raised at the time of designing the quality metrics. There are *researchers' views* that 'design metrics are liable to cover all the quality factors' and that it may enable us to tackle many pertinent questions. To minimize the resource, usage and improve upon cost effectiveness the minimal set of metrics appears to be the generic feature of design metrics. The inherent use of a relevance and value of good design metrics for the same system would not vary with time and people but with major paradigm shift in the field of software development. It is also evident that the metrics are supposed to preserve all the intuitive notions about the attributes and the way in which the metrics distinguish between the entities.
Therefore it appears that a set of essential and desirable features may be identified and hence assured by metric developers for the best cause of software engineering. The following abstractions may be listed as the essential features for designing the quality metrics:

Compliance: The ability to cover all aspects of quality factors and the design characteristics[1].

Orthogonality: The ability to represent different aspects of the system under measurement[15].

Formality: The ability to get the same value for the same systems for different people at different times through precise, objective and unambiguous specification[14].

Minimality: The ability to be used with the minimum number of metrics[15].

Implementability/Usability: The implementation technology independent ability.

Accuracy: A quantitative measure of the magnitude of error, preferably expressed as a function of relative error[25].

Validity: Validity refers to the degree to which a study accurately reflects or assesses the specific concept that the researcher is attempting to measure.

Reliability: The probability of failure free software operation for a specified period of time in a specified environment.

Interpretability: The ease with which the user may understand and properly use and analyze the metrics results.

It also appears relevant that the desired metrics should focus on internal object structure that reflects the complexity of each individual entity and on external complexity that measures the interactions among entities. In spite of these essential features of design metrics, some desired features has also been investigated.
There is no doubt that to ensure better processes a 'framework, method or roadmap' is generally used and have been found to be handy and quite fruitful. Further, it becomes evident through our explorations that a little

work has been reported on the subject. There is an ample opportunity and necessity as well, to work out a framework that may be prescriptive in nature and be easily usable to end up in the 'development of good quality metrics'.

The framework ^{q}MDF: Taking into account the need and significance of a roadmap or framework for developing metrics with 'essential and desirable features', an integrated and prescriptive framework ^{q}MDF is hereby proposed. ^{q}MDF has been attempted to be highly implementable and prescriptive in nature. It has been structured into a hierarchical description including premises, generic guidelines and metric development process to be followed in order as follows.

Premises: The following premises has been considered when the proposed framework is being used to design the quality metrics:

* Five quality indicators comprising efficiency, complexity, understandability, reusability and testability/maintainability cover all the factors that affects the software quality.
* An integrated approach to measurement of software quality is feasible and would prove to be optimal.
* A common set of features for the desired metrics may be used to form the basis for its development.
* The recourse optimization in SDLC depends on the early use of metrics and uncovering of errors as far as possible.
* The approach to measurement should be more applicable to identifying low quality software than the high quality code.

Generic guidelines: The guidelines before following the process to develop the metrics may be listed as follows:

* Assure compliance/ adherence to collect a common set of essential and desirable features for the proposed metric.
* Identify and persist with all the attributes of good quality software.
* Identify and persist with all the quality factors affecting specifically the quality of object oriented software being measured/ predicted.
* Correlate the identified attributes with quality factors and accordingly design the metric.
* Assure to control somehow all the extraneous and intervening factors that may affect metric based prediction.

Metric development process: The development process of the metrics is comprised of seven phases together with prescriptive steps for each and has been depicted pictorially in ^{q}MDF, Fig. 1. Such a framework has been proposed on the basis of integral and basic components for designing good quality metrics. The first phase starts with the conceptualization. Planning for the desired metrics is treated as an important task and has been putforth as a second phase, followed by the phases termed as designing, validation, testing, review and revision and packaging. An attempt has been made to symbolically represent the spirit of designing a metric and make the framework prescriptive in nature followed by a brief description of each o f the phases comprising the depicted steps in the special reference to development of metrics (Fig. 1).

Conceptualization: One of the foremost task of any comprehensive problem-solving activity is conceptualization. That is the initial brainstorming activity envisaged and undertaken to understand the problem, jot down ideas for solution and to realize problem-related facts. Which in turn may be precisely stated and represented in meaningful formats, under the aegis of specifications. Importance of this phase lies in the fact it serves as the basis for evolving initial set of specifications to subsequent phases of development.

Planning: It is mandatory to have a plan, if one wants to succeed in a problem solving situation and hence also for development of metrics. A precisely defined plan provides guidance to the developer as it works as a roadmap. There is no doubt, that a metric will have little value if it is designed outside a well-developed structural framework.

Designing: Software metrics are an integral part of the state-of-the-practice in software engineering. Well designed metrics with documented objectives may help the organization to obtain the information it needs to continue to improve its products, process and services while maintaining a focus on what is important to that organization. Thus, designing is the most important and critical step towards the development of desired quality design metrics.

Validation: Theoretical validation of software metrics provides the supporting evidence as to whether a measure really captures the internal attributes that it purports to measure. The main goal of theoretical validation is to assess whether a metric actually measures what it purports to measure[19].

In the context of an empirical study, the theoretical validation of metrics establishes their construct validity, i.e. it 'proves' that they are valid measures for the constructs that are used as variables in the study. Unfortunately, as Van den Berg and Van den Broek[20] remark, even though several attempts have been made at proposing methods and principles to carry out the theoretical validation of metrics (mainly in the context of software engineering), there is not yet a standard, accepted way of theoretically validating a software metric. However, the most general approach that may be adapted is analytical.

Testing: Common wisdom, intuition, speculation and proof of concepts may not be reliable sources of credible knowledge[22], hence it is necessary to place the metrics under testing. Testing is one of the best empirical research strategies, performed through quantitative analysis of experimental data on implementation[23]. Testing is crucial for the success of any software measurement project[10,12].

Review and revision: This phase is informal and has been placed as the sixth phase with free-to-enter at any of the earlier phases. Basic idea of such a prescription is to have adequate enough exposure and then turn back for better review, in the light of all the previous phases. However, informal reviews and revisions may be carried out at any of the stages in the metric development process.

Packaging: This phase is the last and conclusive phase, of the metric development process. During this phase the developed metric is prepared with the needed accessories to become a ready-to-use product, like any other usable product.

qMDF tryout: The metric development process prescribed in qMDF has been followed to develop an integrated design metric. As an outcome of the 'in-order' successful implementation, an integrated class based metric, WCC (Weighted Class Complexity Metric) has been developed and validated using ten commercial software projects[36,21]. A glimpse of the activities undertaken in developing a metric using the framework may be had in the following descriptions.
There appeared to be a need for developing a single integrated object oriented metric, encompassing all the object oriented design constructs, which may be used in the early stage of development to give a good indication of software quality. The ability to cover all aspects of quality factors and the design characteristics, to represent different aspects of the system under measurement, to get the same value for the same systems for the different people at different time, to be used with the minimum number of metrics, to have an empirical validation and ability of failure free operation are identified as the essential features of the desired object oriented metric.
Early estimation of quality is feasible only with the early use of object oriented design metrics. Mapping the identified object oriented design characteristics with quality attributes makes the development of design quality metrics feasible. The use of qMDF also supports the feasibility of the development of such metrics. The generic feature of the developed design metric is its use to minimize the recourse, usage, improvement upon cost effectiveness and attain enhanced quality. A sound basis in the form of object oriented design attributes, concepts, metrics exists to pave the way for the development of a new metric. Moreover, it also become clear that the inputs required for measures may be available well in advance in the design phase.

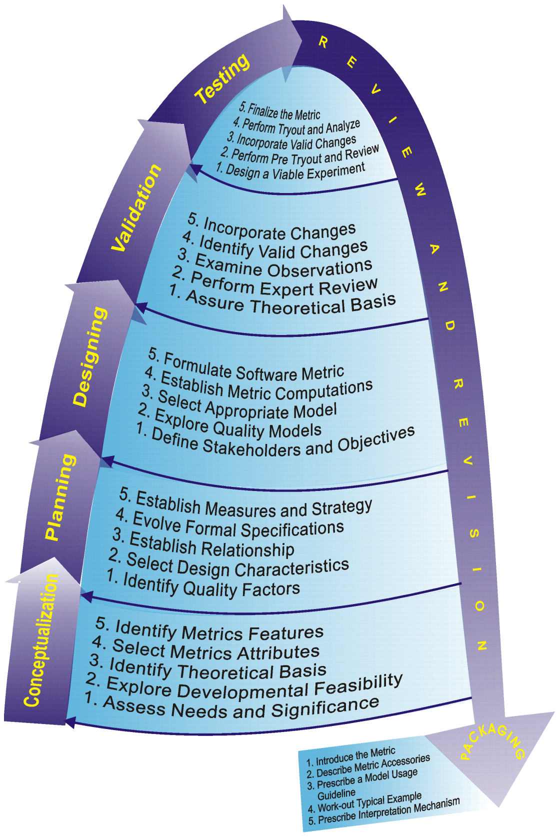
Testing
5. Finalize the Metric
4. Perform Tryout and Analyze
3. Incorporate Valid Changes
2. Perform Pre Tryout and Review
1. Design a Viable Experiment
Validation
5. Incorporate Changes
4. Identify Valid Changes
3. Examine Observations
2. Perform Expert Review
1. Assure Theoretical Basis
Designing
5. Formulate Software Metric
4. Establish Metric Computations
3. Select Appropriate Model
2. Explore Quality Models
1. Define Stakeholders and Objectives
Planning
5. Establish Measures and Strategy
4. Evolve Formal Specifications
3. Establish Relationship
2. Select Design Characteristics
1. Identify Quality Factors
Conceptualization
5. Identify Metrics Features
4. Select Metrics Attributes
3. Identify Theoretical Basis
2. Explore Developmental Feasibility
1. Assess Needs and Significance
REVIEW AND REVISION
PACKAGING
1. Introduce the Metric
2. Describe Metric Accessories
3. Prescribe a Model Usage Guideline
4. Work-out Typical Example
5. Prescribe Interpretation Mechanism

Fig.: 1 **Metric Development Framework**

As for as metric attributes are concerned, some of the important attributes related to the metric have been identified for the proposed metric. Such attributes include the following important inherent characteristics:

* Qualitative interpretation;
* Catering to all aspects of object oriented design;
* Covering all quality factors;

* Early usability in SDLC; and
* Optimizing the quality estimation processes.

Software Assurance Technology Center, SATC, has proposed five of quality differentiators/attributes for the coding and design phase. These are, Efficiency, Complexity, Understandability, Reusability and Testability/ Maintainability. It is evident from the discussion that no universally agreed-upon definition for each of high-level quality attributes exists. It was observed that each of the design constructs affect certain quality attributes. This is depicted in Fig. 2. We considered the SATC's five quality factors to carry forward the development, as these five quality factors cater the overall quality of the software system.
Dr. Linda Rosenberg[11], at SATC (NASA), described three aspects of object-oriented paradigm: Encapsulation, Polymorphism and Inheritance. Polymorphism and Inheritance are two aspects unique to the object oriented approach, while encapsulation is not. Therefore, the three fundamental properties required for an object oriented approach was considered and in the process of development of an integrated metric. Numerous software metrics related to software quality assurance have been proposed in past and are still being proposed. General review of metrics suggested by various researchers/practitioners (MOOD[18]/ MOOSE[28]/ QMOOD[1] / EMOOSE[29] etc.) and evaluation of object oriented concepts by these suggested metrics has been presented in[21].

A critical examination of the existing design metrics revealed that all metrics have relevance with respect to a class, i.e. all metrics eventually conduct measures taking class as a basis. This is hardly surprising as 'class' is the fundamental concept of object oriented software[21,33,34]. It appeared that the Dromey's model[30] is more appropriate for the development of a good quality metric with the help of ^{q}MDF, as it maps the identified set of design characteristics and quality attributes.
The survey result depicts that all the metrics have relevance with respect to a class. This motivated effort towards developing a single class based metric, Weighted Class Complexity (WCC), which would give a cumulative measure of all the aspects of object oriented design and would thereby give an indication of 'quality' of a class in terms of complexity. This single metric when averaged would enable computing the average complexity of software and finally the quality. The simplest relationship that evolves out of it is as follows:

$$\text{Ave. Complexity of S/W} = \frac{\sum \text{WCC}}{\text{Number of classes (total)}}$$

The complexity in this context has more of a physiological meaning rather than complexity as a quality attribute. Thus WCC should take into account each/or most of the design constructs, i.e. WCC should be integrated with an encapsulation, inheritance, coupling and cohesion factors. Supporting evidence has been presented in authors' own study[21], as to whether a measure really captures the internal attributes that it purports to measure. A sound theoretical basis is given and advocated for the validity to support the claim.

A representative (randomly selected) sample of data was used to validate the proposed metric as per experimental design and statistical analysis of data gathered through the tryout has been interpreted. For this, a set of ten projects was used in the software industry. We labelled the applications as: System A, System B, System C, System D, System E, System F, System G, System H, System I and System J. All these systems were commercial software implemented in C++/Java and consisted of approximately 10-20 classes. The industry professionals themselves have used full-scale code analysis system for estimating the quality of these systems. Table 1 summarizes the quality ranking of these software systems given by industry professionals.
In order to investigate the correlations and relationships between the object oriented metric WCC and software quality, a correlation and a multiple linear regression analysis has been conducted for ten projects. Table 2 summarizes the results of the correlation analysis for the integrated metric set over the ten software systems. The column lists the correlation values for each pair of metrics in the integrated metric set and rows list the system. In the table Metric 1 ^ Metric 2 shows the correlation between Metric 1 and Metric 2.

The multiple linear regression model was fitted to the minimal set of the metric and shown in equation 1 for system A, B, C, D, E, F, G, H, I and J respectively and the results are given in Table 3.

$$\text{WCC} = a + b_{\text{RFC*Level}}\ (\text{RFC*Level}) + b_{\text{LCOM}}\ \text{LCOM} \quad (1)$$

The standardized beta weight (β_i's) and raw score beta weight (b_i's) has been calculated and shown in Table 3.

The computed standardized beta weights (β_i's) in Table 3 for all the systems show that the RFC*Level component has most significant contribution on WCC. It is also evident from the raw score beta weights (b_i's). LCOM component also has a considerable significant contribution on WCC which is depicted through both the beta weight (β_i's) and raw score beta weight (b_i's). Examining the F ratio in Table 3, it is clear that the regression shown in equation 1 is significant at .01 level of significance for the Systems A, D, E, F, H and J and at .05 for the Systems B, C, G and I.

Table 1: Quality Ranking of Systems

Projects (Systems)	Classes	Quality Ranking
A	11	Low
B	9	Low
C	12	Low
D	19	High
E	18	High
F	15	Low
G	10	Low
H	11	Low
I	9	Low
J	11	Low

Table 2: Correlation Analysis Summary

Systems	WCC^ LCOM	WCC^ (RFC* Level)	LCOM^ (RFC* Level)
A	.02	.88	-.06
B	.28	.98	.13
C	.24	.59	.01
D	.57	.46	.23
E	.45	.99	.29
F	.21	.25	.67
G	.82	.88	.66
H	.54	.59	.61
I	.68	.98	.53
J	.78	.88	.66

Table 3: Regression Analysis Summary

	β_{RFC} *Level	β_{LCOM}	b_{RFC} *Level	b_{LCOM}	a	F ratio
A	.88	.07	.70	.21	1.94	16.5
B	.95	.16	.77	.15	2.68	6.11
C	.58	.23	.58	.78	1.87	4.18
D	.35	.49	.17	.07	-1.17	.29
E	.93	.17	.75	.98	.35	9.11
F	.43	.07	.44	.45	1.50	4.21
G	.59	.42	.65	2.07	-.45	5.91
H	.41	.29	.41	.94	2.28	14.22
I	.85	.22	.85	.71	1.62	6.19
J	.73	.35	.60	1.06	1.15	15.11

Table 4: χ^2 Test Observations

	High	Low	Total
WCC	8_A	2_B	10
Industry Rating	2_C	8_D	10
Total	10	10	20
Value of χ^2 is 5.0			

Examining Table 2 shows that for all the systems, all of the metrics are highly correlated with each other, with WCC and (FRC*Level) being the most significantly correlated. In order to further assure, χ^2 test has been used for testing the null hypothesis stated as follows:

H_0: Quality estimates obtained through WCC are not significantly comparable/close to those obtained from industrial quality experts.

H_a: Quality estimates obtained through WCC are significantly comparable/close to those obtained from industrial quality experts.

WCC values of all the ten projects have been tested using the Chi-Square Test (χ^2). The Chi-Square test observations for all the ten systems are listed in Table 4 by using equation 2 applicable for small samples, as frequencies of cells are fewer than 10. The assumptions made for WCC values are low for less than or equal to four and high for greater than four and the degree of freedom may be calculated by using the formula df=(row-1)(column-1).

$$\chi^2 = \frac{N[|AD-BC|-N/2]^2}{(A+B)(C+D)(A+C)(B+D)} \quad (2)$$

In equation 2, A, B, C and D are being replaced by 8_A, 2_B, 2_C and 8_D respectively. The computed value of χ^2 is greater than the critical value of χ^2 for 1 degree of freedom at .05 level of significance, which is 3.84. The test indicates that there is a significant relationship between the WCC value and industry rating for quality of all the systems at the .05 level of significance. Hence, the null hypothesis is rejected and it leads to the inference that 'WCC gives quite comparable result regarding quality for all the systems to those obtained by using full-scale code analyzer, by the organization'. Further, the proposed metric may be used to discover the underlying errors in software design at the early stage of software development life cycle leading to reduce effort on quality assurance and avoidance of unnecessary overhead. It may also help to evaluate the quality of software and provide the cost estimates of a software project that facilitate the estimation and planning of new activities. The metric may be used to determine the effect of the object technology; especially re-use technology applied in the software development according to some quantitative evaluation such as productivity, quality, lead-time, maintainability, etc.

Observations: A close look at the components constructing the theoretical framework, studies and experimental tryout related to the design metrics led to the following observations:

* Strong theoretical basis for designing the metric is required.
* A low-level design metrics may be defined in terms of design characteristics.

* Quality of software may be assessed as an aggregation of the framework's individual high-level quality attributes.
* A minimal set of metric is required to be developed to cover all the aspects of design characteristics and quality factors.
* If the metrics is non-size metrics, it should allow comparisons across different projects.
* Metrics should be defined in such a way that different people at different times or places get the same values for the same systems.
* Metrics development process avoids the development of a metrics with subjective rating like very low, low, average, high very high etc.
* The development process helps to evaluate the quality of software and provide the cost estimates of a software project, which facilitate the estimation and planning of new activities.
* The developed metrics will be able to indicate the faulty classes in early stage of development life cycle to decrease the rework.
* Viable experiments should be designed to validate the developed metric.
* Pre-tryout and tryout should be conducted on developed metric and the result gained from tryout be analyzed and interpreted.
* Informal review and revisions should be carried out throughout entire phases of metric development process.
* A metric usages guideline, a brief introduction and the metric computation mechanism should be provided to the user of that metric.

Apart from above concrete observations, it appeared conclusive that ^{q}MDF works well, at least in the cited experimental tryout, as assured by the quality of WCC. In the absence of any other framework it may be used by metric developers across the fraternity and in turn gets standardized and may be improved.
Further, experimental tryouts and statistical analyses at a large scale with typical representative samples may be needed to standardize the metric WCC. More developmental activities using the framework may be carried out by the researchers and practitioners. Review of already developed or underdevelopment metrics may be guided by the framework, and this framework may form the basis for the development of better-refined roadmaps/models.

REFERENCES

1. Bansiya J., 2002. A Hierarchical Model for object- oriented Design Quality Assessment, IEEE Transaction on software engineering, 28: 4-17.
2. Dumke, R. Reiner, 1995. A Measurements framework for Object- Oriented Software Development, submitted to Annals of software Engineering , Vol. 1.
3. Henderson Sellers, B., 1995. Identifying internal and external characteristics of classes likely to be useful as structural complexity metrics, Proceedings of 1994 intern. Conf. On Object Oriented Information Systems OOIS 94, London, Springer-Verlag, London, pp: 227-230.
4. Abreu, F. Brito and Carapuca, Rogerio, 1993. Candidate Metrics for Object- Oriented Software within a Taxonomy Framework, Proceedings of AQUIS'93 (Achieving Quality In Software), Venice, Italy, October 1993: selected for reprint in the Journals of Systems and Software, 23: 87- 96.
5. Letha Etzkom and Harry Delugach, 2000. Towards a Semantic Metrics Suite for Object-Oriented Design, 0-7695-0774-3/00 IEEE, pp: 71-80.
6. Tahvildari Ladan and Kontogiannis Kostas, 2003. A Metric-Based Approach to Enhance Design Quality Through Meta-Pattern Transformations, Proceedings of the Seventh European Conference On Software Maintenance And Reengineering (CSMR'03) 0-7695-1902-4/03 IEEE, pp: 183-192.
7. Tahvildari Ladan and Kontogiannis Kostas, 2003. A Metric-Based Approach to Enhance Design Quality Through Meta-Pattern Transformations, Proceedings of the Seventh European Conference On Software Maintenance And Reengineering (CSMR'03) 0-7695-1902-4/03 IEEE, pp: 183-192.
8. El Emam Khaled, 2001. A Primer On Object-Oriented Measurement, 1530-1435/01 IEEE, pp: 185-187.
9. Irwin Warwick and Churcher Neville, 2003. Object Oriented Metrics: Precision Tools and Configurable Visualizations, Proceedings of the Ninth International Software Metrics Symposium (METRICS'03) 1530-1435/03 IEEE, pp: 112-123.
10. Schneidewind N. F., 1992. Methodology for validating software metrics, IEEE Software Engineering, 18: 410-422.
11. Rosenberg Linda, Software Quality Metrics for Object Oriented System Environments, A report of SATC's research on OO metrics.
12. Kitchenham B.A, N. Fenton and S. L. Pfleeger, 1995. Towards a framework for software measurement validation, IEEE Transaction Software Engineering, 21: 929-944.
13. Rachel Harrison, 1998. An Evaluation of the MOOD Set of Object Oriented Software Metrics, IEEE Transaction on Software Engineering, vol. 24.

14. Marinescu Radu and Rat¸iu Daniel, 2004. Quantifying the Quality of Object-Oriented Design: the Factor-Strategy Model Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04) 1095-1350/04IEEE, pp: 192-201.
15. Laing, V, C. Coleman and Manager SATC, 2001. Principal Components of Orthogonal Object Oriented Metrics (323-08-14), White paper Analyzing the Results of NASA Object Oriented Data.
16. Fenton, N. And M. Neil, 1999. New Directions in Software Metrics, Fenton's Web Page,.
17. Van Gurp, J., 2000. Automating Software Architectures Assessment, Lillchammer, Norway.
18. Brito, Abreu. F. and Carpuca, Rogerio, 1994. Candidate Metrics for Object Oriented Software within a Taxonomy Framework., Proceeding of AQUIS'93, Venice, Italy, October 1993; selected for reprint in the Journal of Systems and Software, 23: 87- 96.
19. Fenton, N. E. and S. L. Pfleeger, 1997. Software Metrics: A Rigorous & Practical Approach, International Thomson Computer Press, London, United Kingdom .
20. Van Den Berg and Van Den Broek, 1996. Axiomatic Validation in the Software Metric Development Process, Chapter 10: Software Measurement, Edited by Austin Melton, Thomson Computer Press.
21. Khan, R. A., K. Mustafa and S. Yadav, 2004. Quality Assessment of Object Oriented Code in Design Phase, Proceedings, QAI 4th Annual International Software Testing Conference, Pune, India.
22. Basili, V., F. Shull and F. Lanubile, 1999. Building knowledge through families of experiments, IEEE Transactions on Software Engineering, 25: 435-437 .
23. Robson, C. , 1993. Real world research: A resource for social scientists and Practitioners-researchers, Blackwell.
24. Wohlin, C., P. Runeson, M. Höst, M. Ohlson, B. Regnell and A. Wesslén, 2000. Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers.
25. Fairly Richard, 2003. Software Engineering Concepts, Tata McGraw-Hill.
26. Khan, R. A. and K. Mustafa, 2004. A review of SATC research on OO Metrics, Proceedings, National Conference on Software Engineering Principles and Practices, Patiala, India.
27. Dagpinar Melis and H. Jahnke Jens, 2003. Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison, Proceedings of the 10th Working Conference on Reverse Engineering (WCRE'03) 1095-1350/03 IEEE, pp: 155-164.
28. Chidamber, S. R. and C.F. Kemerer, 1993. MOOSE: Metrics for Object Oriented Software Engineering, Workshop on Processes and Metrics for Object-Oriented Software Development (OOPSLA'93*),* Washington DC, EUA.
29. Henry, Li W., S. D. Kafura, R. Schulman, 1995. Measuring Object-Oriented Design, JOOP, pp: 48-55.
30. Dromey, R. G., 1995. A Model for Software Product Quality, IEEE Transaction on Software Engineering , 21: 146-162.
31. Bruntink Magiel and Deursen Arie van, 2004. Predicting Class Testability using Object-Oriented Metrics, Proceedings of the Fourth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'04) 0-7695-2144-4/04 IEEE, pp: 136-145.
32. Evanco William M., 2003. Comments on -The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics, IEEE Transactions On Software Engineering, 29: 630-650.
33. El Emam Khaled, Benlarbi SaõÈda, Goel Nishith and N. Rai Shesh, 2001. The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics, IEEE Transactions On Software Engineering, 27: 630-650.
34. Ferenc Rudolf, Siket Istv´an and Gyim´othy Tibor, 2004. Extracting Facts from Open Source Software, Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04) 1063-6773/04 IEEE, pp: 60-69.
35. Bluemke Ilona, 2001. Object oriented metrics useful in the prediction of class testing complexity, 1089-6503/01IEEE, pp: 130-136.
36. Khan, R. A. and K. Mustafa, 2004. High Level Design Quality Assessment of Object Oriented Codes, accepted for publication in the proceedings in 2nd International Workshop on Verification and Validation of Enterprise Information System VVEIS Porto, Portugal.