ISSN:0975-9646

A Report on the Analysis of Metrics and Measures on Software Quality Factors – A Literature Study

[1]Vanitha N, [2]ThirumalaiSelvi R

[1]*Department of Computer Science, Women's Christian College, Chennai, India*
[2] *Department of Computer Science, Govt Arts College for Men, Chennai, India*

Abstract - **Software quality is the degree to which a component, system or process meets specified requirements and meets customer or user needs or expectations. Software quality is best described as a combination of several factors. The aim of this paper was to investigate the measures available to determine different quality factors. The identification of factors and as well as the metrics and measures was done on the basis of the literature survey by studying and analysis various research papers. The results benefit software developers, researchers and academicians to easily identify the metrics used to measure the quality characteristics of the software. Furthermore, the work aimed at providing some suggestions, using the potential deficiencies detected as a basis.**

Keywords - **beta-factor , Clone detection, Structuredness.**

I. INTRODUCTION

In recent times, the growth of software has increased manifolds. Software products are developed for corporate world as well as for individuals. With the increase in the availability of software the focus has shifted on software quality evaluation and enhancement. Today's user is aware of the expectations from the software and during the selection of software product the user validates the quality of the software product, in terms of quality factors. Improvement of quality after the completion of software is unadvisable as it increases the cost and is almost remaking the product. To overcome this issue the evaluation of software product quality is proposed at developer's perspective during the formulation of software product [1]. Quality measurement is usually expressed in terms of metrics. Software metric is a measurable property which is an indicator of one or more of the quality criteria that we are seeking to measure [12]. Many of the studies in the past have focused on factors and sub factors that affect the software quality and much of the previous studies discuss about the metrics and measurements used to measure the level of particular quality factor in their papers. This paper assesses the measures and metrics of various quality factors used to determine the quality of the software systems and are discussed. This Literature Review aims to identify and analyze the metrics and measures for certain quality factors. The objectives of this review are to provide a general overview of the software metrics and measures and to guide researchers and readers to follow which metrics can be used to measure the different quality factors.

The remainder of this paper is organized as follows. *Section II* provides systematic literature review methodology *Section III provides* a background to the field and presents some relevant surveys. Results of the study are presented in *Section IV .Section V* concludes the paper.

II. METHODOLOGY

In this paper, systematic approach to reviewing the literature on the analysis of the metrics and measures of quality factors follow the approach identified by Kitchenham and Charters[15].

A. Research Questions

The aim of this literature review is to analyze the metrics and measures for certain quality factors. This analysis is based on the research questions in TABLE 1.

TABLE I
RESEARCH QUESTIONS ADDRESSED

Research Questions	
RQ1	Which quality factor can be easily approachable for measuring?
RQ2	Which measure should be used for certain metric to determine different quality factors?

B. Inclusion Criteria

A study of Journal paper or Conference proceedings published in English to be included in this review. From the publication of similar studies, only the most comprehensive or recent to be included.

C. Identification of Papers

Included papers were published between the year 2007 and 2014. Key word search, using the search engines were Google Schloar, Scopus, IEEExplore and ScienceDirect. These search engines covered the majority of software engineering publications and the search string used for this is given in references. Totally 52 papers were identified , 27 papers were rejected as not relevant to this research and included 25 papers finally.

III. BACKGROUND AND RELATED WORK

Metric is a unit used for describing or measuring an attribute. During testing and operational stages external metrics applied and during requirement, design and coding internal metrics applied, basically for non-executable software, to measure quality of intermediate deliverables.

Quality in use metrics identifies the metrics used to measure the effects of the combined quality characteristics for the user. More specifically, these metrics care about the quality in satisfaction of customers. The metrics for effectiveness, performance, productivity and safety in real environment fall in this category [10]. In the end, only external factors matter, but the key to achieving these external factors is in the internal ones: for the users to enjoy the visible qualities, the designers and implementers must have applied internal techniques that will ensure the hidden qualities [7].

A. Dynamic Metrics and Measures

Dynamic Metrics are used to measure specific runtime properties of programs, components, subsystems and systems. According to Tahir *et al*[2] , Sandhu *et al*[4] and Choi *et al* [16], the following are some of the metric type found to predict the qualities related to dynamic systems using the measures given in Table 3. A dynamic analyser tool has been developed using aspect – oriented programming (Aspectj) to perform dynamic analysis of java applications for the purpose of collecting run-time data needed for the dynamic cohesion metrics and dynamic coupling tracer application has been developed in Aspectj for the purpose of computation of the coupling [3].

TABLE II
DYNAMIC METRICS AND QUALITY FACTORS COVERAGE

Metrics	Quality Factors	Measures
Cohesion	Reusability	Measures each instance of variable by the number of time it is accessed
		Message passing load
Coupling	Understandability	Message passing load
	Reliability	Real-time Object Oriented Modelling (ROOM)Charts
	To predict faults	Base-Aspect coupling and Crosscutting Degree of an Aspect
Complexity	Understandability	Decision points in code
Polymorphism	Reusability	Polymorphic behaviour index = P / Total dispatches Where , Total dispatches = (P + NP) P = Unique polymorphic dispatches executed NP = Unique non-polymorphic dispatches executed
	Efficiency	Average Changing Rate of Virtual methods(ACRV)

B. Object –Oriented Design Metrics and Measures

According to Srinivasan *et al* [5] the following are the best measures to assess quality of Object –Oriented design in design phase.

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.

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.

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.

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.

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).

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. 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.

7) Package Cohesion: From the prior literature [13], it has been found that there is a strong interconnection between coupling and cohesion in the way of measuring its level.

Information – Flow – based coupling (ICP) and Conceptual Coupling Between Classes (CCBC) are the two coupling metrics to measure package Cohesion.ICP measure capture highest cohesion levels, which allows us to get information of how much information flowing between classes. CCBC measure captures lower cohesion levels, which allows us to identify semantically related classes in a module. Thus the aggregated measure of the above is used to determine classes that should belong together in a package.

TABLE III
OBJECT ORIENTED DESIGN METRICS AND QUALITY FACTORS COVERAGE

Object – Oriented Design Metrics	Quality Factors	Measures
Encapsulation	Understandability	APCF
Abstraction	Effectiveness, Extendibility	MIF,AIF
Inheritance	Effectiveness, Extendibility	MIF,AIF
Coupling	Understandability	CF
Cohesion	Understandability	LCF
Package Cohesion	Understandability	ICP, CCBC
Complexity	Functionality, Reusability	MPCF

C. Sub-factor Quality Metrics

1) Measuring Structuredness: Alan Gillies [12] proposed that well-structured code will be easier to maintain or adapt and it may be calculated in terms of the average length of code modules within the program.

Structuredness α modularity = $\frac{lines\ of\ code}{number\ of\ modules}$

According to Alan Gillies [12] McCall's approach is more quantitative, using scores derived from equations such as

McCall's structuredness metric = $\frac{n01}{ntot}$

n01 = no. of modules containing one or zero exit points only

ntot = total number of modules

Scores are normalized to a range between 0 and 1.

2) Measuring Readability: In order to assess how documentation may assist in the usability of a piece of software [12]. According to Alan Gillies [12] readability can be calculated using Flesch –Kincaid Readability Index and Fog Index measurement methods:

Grade level = 0.39 + b-c

Where a= the number of words in the sentence

b= the mean number of syllables per 100 words

c= 15.59

The score averages 7 to 8

The other method is the Fog Index (Gunning , 1968):

Fog Index = 0.4a + b

Where a- the number of words in a sentence

b = the percentage of words with more than two syllables

3) Measuring Reusability: Gaffney and Durek [1989] proposed model for software reuse and shows the cost of reusing software components as follows [7]:

Relative cost for software development = [(Relative cost of all new code) * (Proportion of new code)] + [(Relative cost of reused software) *(Proportion of reused software)] .

4) Measuring Reliability: Reliability is measured as the probability that a system will not fail to perform its intended functions over a specified time interval [7].

Mean Time between Failures is the most commonly used measure. By measuring the complexity of the software when the change is designed, software reliability can be estimated indirectly. Software reliability can be measured from the following reliability metrics defined by Holmberga *et al* [14] are:

The probability that the software is imperfect (not fault-free): P(SW imperfect)

The probability distribution for number of residual faults: P(N =n), N = number of faults

The probability (or failure rate) of the critical digital system failure (due to a software fault): P(SW failure)

The conditional probability of a common cause failure (called also beta-factor).

Probability of imperfection can be estimated by the following ways described in the paper Holmberga *et al* [14]:

- Empirical evidence of residual faults in the similar software systems.
- Derive an estimate of the residual number of faults by modeling the V&V development process.
- Measuring the Software complexity.

5) Measuring Portability: According to Mallikarjuna *et al* [7], analysis of porting costs involves analyzing the match between the interfaces of the software unit and those of the target. A figure for degree of portability can then be computed for a specific software unit is measured as follows.

DP = (cost to port / cost to redevelop)

If this value is greater than one, then porting is more cost effective than redevelopment. Moreover, porting costs will be inversely proportional to the DP; a value of one represents "perfect portability".

TABLE IV
STRUCTURAL AND SUB-FACTOR QUALITY METRICS AND QUALITY FACTORS COVERAGE

Metrics	**Quality Factors**	**Measures**
Structuredness	Usability	McCall's Approach
Readability	Usability	Flesch –Kincaid Readability Index
Cost for Software Development	Reusability	Gaffney & Durek 's cost model
Complexity	Reliability	Mean Time between failures
Portability	Reliability	DP = (cost to port / cost to redevelop**)**
Structural Metrics	Understandability	Correlation Analysis
	Modifiability	Regression Analysis

D. Structural Metrics for Process Models

Some of the structural metrics for Process Models explained in the prior literature Garcia *et al* [8] are as follows:

- Number of nodes : number of activities and routing elements in a model.
- Diameter : The length of the longest path from a start node to an end node.
- Density : Ratio of the total number of arcs to the maximum number of arc.
- The Coefficient of Connectivity : Ratio of the total number of arcs in a process model to its total number of nodes.
- The Average Gateway Degree : Expresses the average of the number of both incoming and outgoing arcs of the gateway nodes in the process model.

- The Maximum Gateway Degree : Captures the maximum sum of incoming and outgoing arcs of these gateway nodes.
- Separability :Ratio of the number of cut-vertices on the one hand to the total number of nodes in the process model on the other.
- Sequentiality: Degree to which the model is constructed out of pure sequences of tasks.
- Depth :Maximum nesting of structured blocks in a process model.
- Gateway Mismatch :The sum of gateway pairs that do not match with each other, e.g. when an AND-split is followed by an OR-join.
- Gateway Heterogeneity : Different types of gateways are used in a model.
- Cyclicity :The number of nodes in a cycle to the sum of all nodes.
- Concurrency : The maximum number of paths in a process model that may be concurrently activate due to AND-splits and OR-splits.

1)Measuring Modifiability and understandability: From Garcia *et al* [8], it has been found that understanding time is strongly correlated with number of nodes, diameter, density, average gateway degree, depth, gateway mismatch, and gateway heterogeneity .

The correlation analysis indicates that there is a significant relationship between the time and efficiency of understandability and structural metrics.

For efficiency for modifiability, it has been found that significant correlations with average (.745, .005) and maximum gateway degree (.763, .004), depth (-.751, .005), gateway mismatch (-.812, .001) and gateway hetero-geneity (.853, .000) is needed [8].

Gracia *et al*[8] proposed that the correlation analysis suggests it is necessary to investigate the quantitative impact of structural metrics on the respective time, accuracy and efficiency dependent variables of both understandability and modifiability. This goal was achieved through the statistical estimation of a linear regression analysis with the experimental data and the best indicators for modifiability are gateway mismatch, density and sequentiality ratio using regression analysis.

E. Technical Documentation Quality Metrics

From Wingkvist *et al* [6] it has been found, that comparing the text on paragraph level and XML structures, Clone detection determines the similarity between documents and size of two documents that is unique which are the indications of one of the quality of maintainability in technical documentation.

TABLE V
TECHNICAL DOCUMENTATION QUALITY METRICS AND QUALITY FACTORS COVERAGE

Metrics	Quality Factors	Tool used to calculate metrics
Clone detection	Maintainability	VizzAnalyzer
Test success and Coverage	Usability	

Test coverage measurement statically analyzes the whole structure of a technical documentation, dynamically logs the documents and hyperlinks followed during testing, and correlates the static and dynamic information [6],which are the indications of one of the quality of usability in technical documentation. DocFactory is one of the technical documentation producers and VizzAnalyzer is an analysis tool to assess the technical quality of documents which supports metrics, such as, clone detection and coverage analysis. To visualize the metrics results tools such as Microsoft Excel and the yEd graph viewer, can be used.

IV. Results Of The Study

A. Answering Research Questions

RQ1: Which quality factor can be easily approachable for measuring?

Analyzing metrics and measures for different quality factors across the 25 studies in detail, suggests that understandability could be easily approachable for measuring. Fig. 1 showed the availability of many measures to estimate different quality factors. Our results provide some evidence to suggest that many metrics and measures are available to determine the quality for Object- oriented systems.

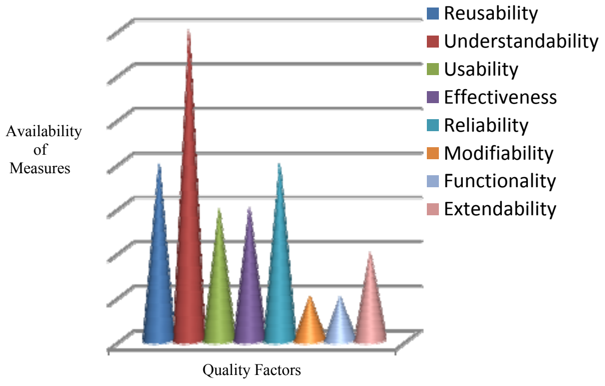

Fig. 1. Level of Availability of measures to estimate Quality factors

RQ2: Which measure should be used for certain metric to determine different quality factors?

Many different metrics and measures have been used in the 25 finally included studies. These mainly fall into dynamic metrics, source code metrics and metrics relating to documentation. In addition, dynamic metrics across the 12 studies we analyzed in detail suggests measures relatively well. However looking at the findings from individual studies, several authors report that quality is not only determined by process metrics, in the form of product and also even by documentation metrics. This study also explains well in our detailed comparison of quality factors and the metrics to determine the quality.

V. Conclusion

Metrics is an important topic in software engineering. Metrics have the potential to improve the quality of systems. As a result of this many metrics and measurement studies in software engineering have been published. In this paper analysis of 52 studies shows that large range of metrics were used but it is difficult for researchers to identify and analyze the metrics and measures of quality factors for similar software systems. The set of metrics presents the essential measures to determine the quality of software. It can be used by future quality prediction researchers, by journals and conference reviewers and software engineers. Of the 52 studies were viewed, only 25 satisfied our criteria and determine what impacts on quality factors. The results suggest that many metrics and measures are available to identify the understandability of the system. It has been also found that many metrics and measures are available for Object- oriented Systems. From this study, it has been also studied that there is interdependence between quality factors as measuring metrics determines more than one quality factors. Overall we conclude that many good measures are available in prior to determine quality in software systems have been reported in software engineering.

In future, progress or extension of this study can be done by focusing on other quality attributes or factors software metrics and measures and for other than object oriented systems.

REFERENCES

[1] Aman Kumar Sharma, Dr. Arvind Kalia, Dr. Hardeep , "An Analysis of Optimum Software Quality Factors", *IOSR Journal of Engineering, 2(4)* ,2012 , 663-669.

[2] Tahir, MacDonell, S.G., "A Systematic mapping study on dynamic software quality metrics", *Proc. 28th IEEE International Conference on Software Maintenance* ,Riva del Garda, Italy, 2012, 326-335.

[3] Object – oriented Static and Dynamic Software Metrics for Design and Complexity, V.Gupta ,2010.

[4] P.S. Sandhu and G. Singh, "Dynamic Metrics for polymorphism in Object Oriented Systems", *World Academy of Science, Engineering and Technology*, vol. 39, 2008.

[5] K.P. Srinivasan, Dr. T.Devi, "A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment", *International Journal of Software Engineering and Its Applications 8(2)*, 2014, 173-188.

[6] Anna Wingkvist, Morgan Ericssony, Rudiger Lincke, Welf Lowe, "A Metrics-Based Approach to Technical Documentation Quality", Proc. *Seventh International Conference on Quality of Information and Communications Technology* , 2010, 476 – 481.

[7] C.Mallikarjuna, K. Sudheer Babu, P. Chitti Babu, "A Report on the Analysis of Software Maintenance and Impact on Quality Factors", *International Journal of Engineering Sciences Research-IJESR , Vol 05, Article 01335*, 2014.

[8] Laura Sanchez-Gonzalez, Felix Garcia, Jan Mendling, Francisco Ruiz, Mario Piattini, "Prediction of Business Process Model Quality based on Structural Met rics" , *Conceptual Modeling –ER 2010*, (CanadaSpringer Berlin Heidelberg ,2010) , 458-463.

[9] Vigdis By Kampenes, *Quality of Design, Analysis and Reporting of Software Engineering Experiments: A Systematic Review,* doctoral diss, University of Oslo,2007.

[10] O. Tolga Pusatli, Sanjay Misra , "A Discussion On Assuring Software Quality In Small And Medium Software Enterprises: An Empirical Investigation", *Portal of scientific journals of croatia,18(3)*,2011,447-452.

[11] Bakota, T., Péter Hegedus, P., Kortvelyesi,P.,Ferenc,R.,Tibor Gyimothy, "A Probabilistic Software Quality Model, Software Maintenance"(ICSM), *Proc. 27th IEEE International Conference*, 2011, 243-252.

[12] Alan Gillies , *Software Quality: Theory and Management* , 3rd edition, Lulu.

[13] G Bavota, A De Lucia, A Marcus, R Oliveto, "Using structural and semantic measures to improve software modularization", *Emprical Software Engineering, 18(5)*,2013, 901-932.

[14] Jan-Erik Holmberga, Peter Bishopb, Sofia Guerrab, Nguyen Thuyc , "Safety case framework to provide justifiable reliability numbers for software Systems", *Proc. 11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference*, 2012, pp. 10-TH2-2.

[15] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering "(version 2.3),Keele University, UK, Tech. Rep. *EBSE Technical Report EBSE-2007-01*, 2007.

[16] K.H.T. Choi and E. Tempero, "Dynamic measurement of polymorphism", *Australian Conference on Computer Science*, Ballarat, Victoria, 2007,211-220.

[17] Karine Mordal, Nicolas Anquetil, Jannik Laval, "Alexander Serebrenik,Bogdan Vasilescu, Stephane Ducasse" , Software Quality Metrics Aggregation In Industry, *Journal Of Software: Evolution And Process,25(10)*, 2013, 1117-1135.

[18] Rohitt Sharma, Paramjit Singh, Sumit Sharma, "An Approach Oriented Towards Enhancing a Metric Performance", *International Journal On Computer Science And Engineering (IJCSE)*, 4(5), 2012, 743-748.

[19] Jitender Kumar Chhabra and Varun Gupta, "A Survey of Dynamic Software Metrics", *Journal Of Computer Science and Technology, 25(5)*, 2010, 1016-1029.

[20] P. Cavano and J. A. McCall, "A framework for the measurement of software quality", *Proc. of the software quality assurance workshop on Functional and performance issues*, 1978, 133–139.

[21] Jehad Al Dallal, Lionel C. Briand, "A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(2), Article No. 8, 2012.

[22] Lawrence Putnam, Ware Myers, *Five Core Metrics: The Intelligence Behind Successful Software Management*, New York, Addition-Wesley ,2013.

[23] Sonia Montagud, Silvia Abrahao, Emilio Insfran, "A systematic review of quality attributes and measures for software product line", *Software Quality Journal, 20(3-4),* 2012, 425-486.

[24] Gregor Grambow, Roy Oberhauser, Manfred Reichert, "Contextual Injection of Quality Measures into Software Engineering Processes", *International Journal On Advances in Software, 4(1,2)*,2011,76-99.

[25] Farroq, S.U., Quadri, S,M.K., Ahmad, N., "Metrics, models and measurements in software reliability", *Proc. IEEE 10th International Symposium*, Herlany, 2012, 441-449.