Software metrics to assess Software Quality

A Systematic Riterature Review

Josyula Jitendra

931023-9159

**E-mails**: jijo15@student.bth.se,

Aduri Raghu Ram

930817-4151

**E-mail**: raad15@student.bth.se

1. GROUP MEMBER PARTICIPATION

The group members participated in idea creation and in report writing with the following amount of involvement.

|  |  |  |
| --- | --- | --- |
| **Group Member** | **Idea Creation** | **Report Writing** |
| Josyula Jitendra | ***50%*** | ***50%*** |
| Aduri RaghuRam | ***50%*** | ***50%*** |

Table 1: Participation

*Abstract*— This literature review is proposed in the context of assessment of software quality in object oriented programming. So to assess software quality one of the way is using software metrics like CK, MOOD and QMOOD metrics. Here the main objective is to review different software metrics to attain software quality and address the problems associated with each metric. For this notion the data is collected from IEEE Xplore database using an appropriate search string. The articles related to CK, MOOD and QMOOD metrics in two different platforms C++ and JAVA are taken into consideration for this literature. Our results indicate that CK, QMOOD metrics have similar components and can detect the faulty classes more easily when compared to MOOD metrics. We finally conclude that metrics can minimize the flaws and complexity associated with a software, but they cannot eradicate them.

# Introduction

CONTEXT:

Today’s software is mainly based on object oriented design and development which involves a performance enhancement and risk management. According to Kitchenham every one have their own perspective on quality. There is neither a simple measure nor a universally accepted definition for software quality [1]. The Object Oriented (OO) approach in software development help in the improvement of software quality, project time and also in terms of project outcomes [2]. Assessment of software product quality depends on Efficiency, complexity, understandability, reusability, testability, maintainability characteristics [3] for OO programming languages. Using software metrics is one of the effective way in order to analyze software quality characteristics.

BACKGROUND:

For performing this review we have considered only three software metric suits like CK, MOOD and QMOOD metrics.

CK METRIC SUIT:

In 1991 Chidamber and Kemerer (CK) defined the CK metric suit and a revised version containing some definitions of metrics was published in 1994.This metric suit contains six metrics which are described in the table 2a [3].

MOOD METRIC SUIT:

Brito e Abreu proposed a metric suit - Metrics of Object Oriented Design (MOOD) which contains six metrics. In which only four metrics are considered for our literature review which are described in the table 2b [3].

QMOOD METRIC SUIT:

Bansiya and Davis proposed a metric suit- Quality Model for Object Oriented Design (QMOOD). This metric establish a connection between object oriented design attributes and design properties which are described in the table 2c [3].

OBJECTIVES:

The major Objective of this review is to understand different software metrics and their approaches to assess the software quality. Some of the problems encountered in this area, were considered as the knowledge gap for our literature review.

METHODS:

With the help of research questions and Kitchenham’s review guidelines [ ] we performed data extraction from IEEE Xplore database. Quality assessment and data extraction are discussed in detail in further sections.

RESULTS:

From our study on the research journal articles we have presented the overview of each article in a table (section VI).

|  |  |
| --- | --- |
| **Metric Name Value** | **Definition** |
| Weighted Method Per Class (WMC) | Sum of complexities of local methods of a class. For simple WMC, when all complexities are unity, same as number of class methods |
| Depth of Inheritance Tree (DIT) | Max number of edges between a given class and a root class in an inheritance graph (0 for a class which has no base classes). |
| Num. Children (NOC) | A Count of the of direct children of a given class |
| Coupling Between Objects (CBO) | Counts other class whose attributes or Methods are used by the given class plus those use the attributes or methods of the given class |
| Response For a Class | A count of all of local methods of a class plus all of methods on other classes directly called by any of the methods on the class |
| Lack of Cohesion of Methods (LCOM) | Num. of disjoint sets of local methods, no two sets intersect, any two methods on same set at least one local variable (1998 definition) |

**Table2a:** Chidamber and Kemerer’s (CK) Metrics Suite (Class Metrics Only)

|  |  |
| --- | --- |
| Attribute Hiding Factor (AHF) | [1-total num. visible (can be accessed)) attributes in asset of classes] / total num. attributes in the set. Measures visibility of a class definition |
| Methods Hiding Factors (MHF) | [1- total num. visible (can be called) methods in a set of classes] / total num. methods in the set. Measures visibility of a class definition |
| Attributes Inheritance Factor (ALF) | The ratio of inherited attributes to the total number of attributes in a class |
| Method Inheritance Factors (MIF) | The ratio of inherited methods to the total number of the methods in a class |

**Table 2b**: Fernando Brito e Abreu’s (MOOD Metrics Only)

|  |  |
| --- | --- |
| Avg. Num. Ancestors (QMOOD\_ANA) | Average of DIT for all classes in the system. |
| Cohesion Among Methods (QMOOD\_CAM) | A measure of cohesion that is based on the similarity of method signatures in a class. Include for completeness; not implemented in this research. |
| Class Interface Size (QMOOD\_CIS) | The count of public methods in a class. |
| Data Access Metrics (QMOOD\_DAM) | The ratio of private or protected attributes to the total number of attributes declared in a class. |
| Direct Class Coupling (QMOOD\_DCC) | A count of classes that accept instance of a given class as a parameter plus classes including attributes of the given class type. |
| Measure of Aggregation(QMOOD\_MOA) | The percentage of data declaration in the system whose types ae of user defined classes, as opposed to those of system defined classes such as integers, real numbers, etc. |
| Measure of Fnctnl. Abstraction(QMOOD\_MFA) | Same as MOOD\_MIF. |
| Number of Methods(OMOOD\_NOM) | The number of methods in a class. Same as WMC when weights of the methods in the class equal unity. |

**Table 2c**: Bansiya and Davis’ QMOOD Metrics Suite (Class Metrics only)

CONCLUSION:

Our study reveals that CK metric suit is more effective in fault detection than MOOD and QMOOD metric suits. Though it is effective these are complex in their design and might require a scope for future work.

# Review Questions

The following are the research questions considered for the purpose of this review:

RQ1.) What is a software metric and what are the characteristics of good Software metrics?

A) Motivation: The main motivation behind this research question is to know about software metrics and their characteristics. Their role in detecting flaws and assessment of quality.

RQ2.) Why should we apply software metrics for Object Oriented Design?

A.)Motivation: As most of the industry and academia involves well established object oriented design, it requires software metrics to with stand certain challenges like risk and quality management.

Q3.)To what extent does the software metrics help to enhance Software quality?

A.) Motivation: This question is the motivation for choosing this notion as our field of study for this literature review.

# research methodology

According to kitchenham’s review guidelines [1] we designed the research protocol which defines identification of research, study selection, study quality assessment, data extraction and its contribution. We have consulted librarians and research experts in this field for required primary studies, articles, journals.

## **Pilot Study**

We formulated our search string as “Software metrics for software quality in Object oriented design”. From Kitchenham’s [1] guidelines and considering our research question we designed our search string for discrete databases.

|  |  |
| --- | --- |
| **Data Source** | **Documentation** |
| Digital Library | IEEE Xplorer |
| Search Strategy | ((“Software metrics” OR “design metrics”)AND “Software Quality”) AND (“Object Oriented programming”) |
| Date of search | 2015-02-22 |
| Years covered by search | 1990-2014 |
| Articles found | 433 |

**Table 2: Articles found before refinement of search string**

## **Refining the Search String and Research question:**

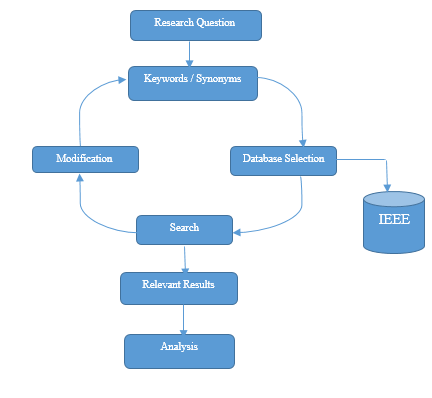
For our relevant literature we narrowed down the search string as shown in the table below.

|  |  |
| --- | --- |
| Data Source | Documentation |
| Digital library | IEEE Xplorer |
| Search String | (((software metrics) AND software quality) AND java programming) |
| Date of search | 2015-02-22 |
| Years covered by search | 1999-2013 |
| Articles found | 13 |

**Table 3: Results after refining search string.**

## **Search Strategy:**

Our search strategy is depicted in the form of figure below:



## **Inclusion / Exclusion Criteria**

**Inclusion Criteria:**

1. Articles which are in English
2. Recent journal articles.
3. Articles related to object oriented metrics, object oriented programming and relevant to our research questions.
4. Articles that have full text available
5. Articles in the field of software engineering.

**Exclusion Criteria:**

1. Irrelevant articles are excluded
2. Resource and process metrics are excluded
3. Other object oriented languages like Smalltalk, .NET. are excluded

## **Quality Assessment Criteria:**

Based on the Articles that were studied for our literature review, we the authors present here the quality assessment in terms of questions which are answered by a reference of the article that justifies that question.

|  |  |  |
| --- | --- | --- |
| **Number** | **Question** | **Source** |
| **1.** | **How was knowledge extended by research?** | **[2],[3]** |
| **2.** | **How clear the assumptions / theoretical perspectives / values are shaped the form and output of evaluation?** | **[2],[3],[4],[6],**  **[7]** |
| **3.** | **How adequately are the research process been documented?** | **[2],[3],[4],[5]** |

## Data Extraction Process

The information related to the primary study and our review questions regarding the data extraction is depicted in thse data extraction form below.

## Validation of Protocol:

Our research proposal was validated by peer debriefing in which we have formulated research questions and generated keywords (Software metrics, Software Quality, Object oriented programming). Using this keywords and respective synonyms we performed an advanced search in the IEEE Xplorer database. We retrieved 433 results. After using inclusion exclusion criteria the articles obtained finally are 13. Taking our literature review into consideration we synthesized and analyzed our study and finally SLR is documented.

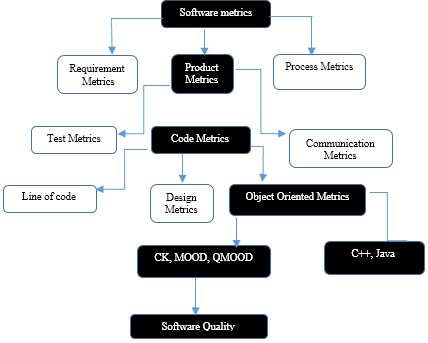
##### V. Included and exclided studies

By using the search string in discrete databases the following results were obtained from which only the articles relevant we included.

|  |  |
| --- | --- |
| **Database** | **Result** |
| Scopus | 141 |
| Web of science | 57 |
| Inspec | 45 |
| IEEE Xplorer | 13 |

Table no:

The following diagram represents the way in which different disciplines are included in our study.



**Diagram**

**Black Box**: *Included* **White Box**: *Excluded*

**Reasons for Exclusion:**

The Requirement Metrics deals with volatility and completeness and process metrics deals with human resource and time. Whereas the product metrics deals with object oriented metrics which is responsible for our literature where we assess the software quality of object oriented languages like c++, java.

VI. RESULTS:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Questions | Ref[1] | Ref[2] | Ref[3] | Ref[4] | Ref[5] | Ref[6] |
| What are the problems discussed? | Role of metrics in determining software defects. | .Detection of fault prone classes using three different OO metrics like CK, MOOD, QMOOD. | can class size be a threat in object oriented programming.  .should a class have an optimal or threshold size. | Design measures to assess software quality | Are the fault proneness models of one OO system applicable when applied to another OO system in an environment? If applicable are they viable decision making tools ? | Evaluation of quality of java program using qmood metric. |
| What are the limitations? | .Only subset of CK suit of metrics is considered.  .Design metrics can address only static aspects of OO Design. | .when the software is in its initial stages software metrics are not much effective.  .Class size can impact metric performance. | The threshold size for a class proposed is from experimental knowledge rather than a practical utility. | . the paper does not focus on encapsulation | .Research is confined to smaller data sets. | Qmood metrics are rather subjective than practical. |
| What is the Technique/method Used? | Weighted linear Regression | Binary logical regression. | Logistic regression | Statistical approach | Multivariate Adaptive Regression Splines[MARS] | Graphical method |
| What is the type of Study? | Empirical Analysis | Empirical  Analysis | Empirical  Analysis | Experimental analysis | Empirical study | Theoretical study |
| What is the scope for future studies? | .It can be extended to dynamic complexity measures such as Polymorphism.  .Can be extended to other metrics like time and effort to build a class. | .it can be extended to decision trees than using binary logical regression. | . Inoder to derive a relation between size and class defects we can use advanced threshold theories or strategies. | .There is a scope to use the inheritance property more accurately to enhance the quality of design. | .The study can be extended from linear models to more complex fault proneness models. | .can be extended for other object oriented languages. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| QUALITATIVE | [1] | [2] | [3] | [4] | [5] | [6] |
| Is the aim of the article clearly stated? | YES | YES | YES | YES | YES | YES |
| Does the article answer any of the review questions? | NO | YES | NO | YES | YES | YES |
| Does the results contribute to literature? | YES | YES | YES | NO | YES | YES |
| QUANTITATIVE | [1] | [2] | [3] | [4] | [5] | [6] |
| How credible are the findings? | STRONG | STRONG | WEAK | WEAK | STRONG | BIAS |
| How clear are the links between data, interpretation and conclusions? | STRONG | STRONG | STRONG | STRONG | STRONG | STRONG |
| How well are the data collection carried out? | STRONG | STRONG | STRONG | WEAK | STRONG | STRONG |

**VII DISCUSSION**

The results obtained in the SLR are presented below:

In the study to assess software quality using software metrics we have learnt different types of software metrics and their characteristics. Different metrics have different capabilities in finding the defects associated with software [3]. .Even after changing the size of software, the metrics are associated with defects. Effect of metrics on defects vary across the platforms [2]. Concepts like inheritance, coupling, and cohesion have effect on complexity [2]. These metrics can help in understanding design complexity, detecting the design flaws, software quality, testing and maintenance. Higher advance investment in design helps in controlling cost as well as improvement in quality [2].

There are many inconsistencies associated with different software metrics which need proper scientific proofs and further investigations. References [6] [7] make these arguments evident.

Moreover this systematic literature review helped us in pointing out a knowledge gap in this field. So it provides a good scope for future research.

**VIII LIMITATIONS**

Even though there are many metrics associated with software we have limited our study to just by considering 3 metrics.

There are many articles regarding software metrics which were not considered as the SLR might go out of scope.

**IX CONCLUSION**

In this study we present some results by validating the relation between different software metrics and defects associated with OO programming. This reveals the effect of OO design complexity metrics like no of methods (WMC), Coupling between the objects (CBO), inheritance depth (DIT) varies across different platforms [2]. Our analysis covers only some of the OO metrics. The study can be extended for other metrics as well on the measures like time, effort required to design a class. Also Coupling and complexity are inversely proportional to quality of software [7].

Some metrics like CK considers only static complexity measures and can be extended to dynamic complexity measures like polymorphism and encapsulation [2]. The CK metrics like NOM, LOC and CBO are strongly related with each other and are inversely proportional to software quality [6].

The MOOD metrics are not efficient in finding the error proneness of OO class quality. The effect of size on MOOD metrics is very evident and inversely proportional [3].

The QMOOD metrics are subjective in nature. By evaluating these metrics in early stages will reduce complexity in the later stages [7]. The individual metrics like CK-WMC, CK-RFC and QMOOD-NOM, QMOOD-CIS are consistent predictors of class quality but these are complex and highly correlated. QMOOD-NOM is analogues to CK-WMC [3].

MOOD metrics were not a good predictors of software quality when compared to QMOOD and CK [3]. Among the three metrics we considered CK, MOOD, QMOOD, the CK metric suit can efficiently predict software quality better than QMOOD, MOOD metrics.

X REFERENCES

1. [1] B. Kitchenham and S. Charters, “Guidelines for  
   performing systematic literature reviews in software  
   engineering,” Software Engineering Group, Keele  
   University and Department of ComputerScience,  
   University of Durham, UK, Technical Report EBSE-2007-  
   01, Jul. 2007.
2. [2] Subramanyam, R.; Krishnan, M.S., "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects," *Software Engineering, IEEE Transactions on* , vol.29, no.4, pp.297,310, April 2003.
3. Olague, H.M.; Etzkorn, L.H.; Gholston, S.; Quattlebaum, S., "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," *Software Engineering, IEEE Transactions on* , vol.33, no.6, pp.402,419, June 2007.
4. Briand, L.C.; Melo, W.L.; Wust, J., "Assessing the applicability of fault-proneness models across object-oriented software projects," *Software Engineering, IEEE Transactions on* , vol.28, no.7, pp.706,720, Jul 2002.
5. El Emam, K.; Benlarbi, Ph.D., Saida; Goel, N.; Melo, W.; Lounis, H.; Rai, S.N., "The optimal class size for object-oriented software," *Software Engineering, IEEE Transactions on* , vol.28, no.5, pp.494,509, May 2002.
6. Kulkarni, U.L.; Kalshetty, Y.R.; Arde, V.G., "Validation of CK Metrics for Object Oriented Design Measurement," *Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference on* , vol., no., pp.646,651, 19-21 Nov. 2010.
7. Goyal, P.K.; Joshi, G., "QMOOD metric sets to assess quality of Java program," *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on* , vol., no., pp.520,533, 7-8 Feb. 2014.