Comparative Study of Object Oriented Measures for Internal Quality Attributes of Class Diagrams

Dr. Manju Kaushik[1] ,Bhawana Mathur[2]
[1]Associate Professor,Dept. of Computer Science &Engg ,JECRC University,Jaipur India
[2]Research Scholar, Dept. of Computer Science &Engg ,JECRC University,Jaipur India

Abstract
As a key early artifact in the development of Object Oriented software, the superiority of class diagrams is crucial for all later design work and could be a major determinant for the quality of the software product that is finally carried. Quantitative measurement instruments are useful to assess class diagram quality in an objective way, thus avoiding bias in the quality evaluation process. A set of metrics -based on UML relationships- which measure UML class diagram structural complexity following the idea that it is related to the maintainability of such diagrams. Also concise are two controlled experiments carried out in order to gather empirical evidence in this sense.As a result of all the experimental work, Researcher can conclude that most of the metrics researcher proposed (NAssoc, NAgg,NaggH, MaxHAgg, NGen, NgenH and MaxDIT) are good indicators of class diagram maintainability. Researcher cannot, however, draw such firm conclusions regarding the NDep metric.

Keywords*:* OO high-level metrics, structural complexity, class diagrams, maintainability, UML, empirical validation

I. Introduction

In the development of OO software, the class diagram is a key early artefact that lays the foundation of all later design and implementation work. The early focus on class diagram quality may help software designers build better OO software, without unnecessary revisions at later development stages when changes are more expensive and more difficult to perform. It is in this arena where software measurement plays an important role, since the early availability of metrics contributes to class diagram quality evaluation in an objective way avoiding bias in the quality evaluation process. Moreover, metrics provide a valuable and objective insight into specific ways of enhancing each of the software quality characteristics. Given that maintenance is (and will continue to be) the major resource consumer of the whole software life cycle, maintainability has develop one of the software product quality characteristics that software development organizations are more concerned about. However, researcher are aware that maintainability is an external quality attribute that can only be measured when the OO software product is (nearly) finished.

If such a relationship exists and is confirmed by empirical studies, researcher will have really obtained early indicators of class diagram maintainability. These indicators will allow OO software designers to take better decisions early in the OO software development life cycle, thus funding to the development of better quality OO software.

II. Review of Literature:

Ever since human beings started using computers, researches have been working to raise the abstraction level at which software engineers write programs[1]. Class cohesion is an important object-oriented software quality attribute. It designates how much the members in a class are related.[2] An improved hierarchical model for the assessment of high-level design quality attributes in object oriented designs. In this, structural and behavioral design properties of classes, objects, and their relationships are assessed using a suite of object-oriented design metrics.[3] A validation of object-oriented design metrics as quality indicators validation of that suite of object oriented metrics in object-oriented design[4].Experience from a dozen years of analyzing software engineering processes and products is summarized as a set of software engineering and measurement principles that argue for software engineering process models that integrate sound planning and analysis into the construction process[5]. Experimentation in software engineering is necessary but difficult. [6] So, after a thorough review and analysis of some of the current Object Oriented measures, applicable to class diagrams at high-level design stage [6-9] researcher have proposed [10-11] a set of UML class diagram structural complexity measures based on the use of UML relationships Such as generalizations, associations, aggregations and dependencies, where also, traditional metrics such as Number of Classes, Number of Methods and Number of Attributes are included. These metrics presented have been developed in a methodological way which consists of three main steps: metric definition, and theoretical and empirical validation [11-12].. As the proposal of metrics is of no value if their practical use is not demonstrated empirically [2,13-16] either by means of case studies taken from real projects or by controlled experiments, our main motivation is to examine, through research, if the metrics researcher proposed for UML class diagram structural complexity (internal quality attribute) are related to some class diagram maintainability (external quality attribute) sub characteristics: understandability, analyzability and modifiability[17].

III. Basic Concept of Maintenance in Object-Oriented Software Engineering (MOOSE)

The project performs empirical studies to better understand how one can improve the maintainability of evolving object-oriented software. Research questions addressed by this project include:

1. What content and level of detail of the Unified Modeling Language (UML) are required for efficient software maintenance, and how should the UML artifacts be developed and maintained?

2. What are the key product factors (e.g., design properties) and process factors (e.g., design activities) that affect (negative/positive trends in) maintainability?

3. How can we build, evaluate and apply prediction models to help focus quality improvement efforts (e.g., better UML documentation, refactoring, testing) on those parts of the software system with the highest potential of return on investment?

Much effort is spent within the software engineering community on proposing technologies (processes, methods, techniques, principles) that are believed to improve the efficacy of developing and maintaining object-oriented software.

Evolving object oriented Software with respect to maintainability:

Specifically, three themes related to the improvement, assessment and prediction of the maintainability of evolving object-oriented software.

1. **Improving maintainability with model-driven development (UML):** It is expected that many aspects of a software system need to be understood in order to correctly change it, with its functionality, architecture, and a countless of design details.

2. **Assessing maintainability of evolving object-oriented software:** To better understand how to design and maintain object-oriented software, the project will develop an assessment framework incorporating both qualitative and quantitative techniques.

3. **Predicting maintainability of evolving object-oriented software:** To improve the maintainability of object-oriented software, data-mining techniques can be used to build prediction models that model the relationships between various indicators of maintainability (e.g., fault proneness) and product and process data (e.g., structural properties of the software combined with historical change and fault data).

Measure Internal Quality Attributes of Class Diagrams

The primary aim of this work, therefore, is to present a survey and analysis, as complete as possible, of the existing relevant works regarding class diagram metrics. Supplementary aim of this work is to Aim of quality in software products is characterized by the presence of different External Attributes assistance reveal areas of research either lacking completion or yet to undertaken. Quality in software products is characterized by the presence of different external attributes[1].

Example: functionality, reliability, usability, efficiency, maintainability and portability [ISO01].
Therefore, it is necessary to find early indicators of such qualities based, for instance, on the structural properties of class diagrams [1]. This is the context where software measurement is essential, because measures can allow us to evaluate class diagram quality characteristics in an unbiased way, thus escaping a bias in the assessment process.

Objectives of quality attributes:

1. Concentrating on product metrics that can be applied to an advanced design or to code.
2. As the Unified Modeling Language (UML) [2] has emerged as a modelling standard, and in general has been widely accepted by most software development organizations, researcher will focus work on UML class diagrams.
3. A precise differentiation of analysis, design, and implementation activities is not informal, due to extensive adoption of iterative and cascade life cycles, which incline, occasionally deliberately, to blur their distinctions.
4. For our current purposes, researcher shall consider the UML class diagram, at its initial stages of development, to be self-possessed of the next UML constructs: Packages., Classes, Each class has attributes and operations, Attributes have their name, Operations only have their signature, i.e. their name and meaning of parameters, Relationships: Association, Aggregation[3], Generalization and Dependencies[4].

Analysis and Comparative Study of Quality Attributes

S.No	**Internal quality attributes(IQA)**	**External quality attributes(EQA)**
1.	IQA can be measured purely in terms of the product (e.g., complexity, coupling, cohesion, etc.) [1].	EQA can be measured only with respect to how the product relates to its environment.
2	An internal attribute can be measured by examining the product on its own, separate from its behavior.	The behavior of the product is more important than the product itself.
3	Internal quality attributes(IQA): Derived immediately from the product or process description (To derive, it is sufficient to have the description) Underlying assumption: internal quality leads to external quality (cfr. metaphor manufacturing lines) e.g. Efficiency	External quality attributes (EQA): Derived from the relationship between the environment and the system (or the process). (To derive, the system or process must run) e.g. Reliability, Robustness

1. *Maintainability* :How easy it is to *change* a system after its initial release
2. software entropy ⇒ maintainability gradually decreases over time.
3. Measuring class diagram quality allows OO software designers:
4. To recognize the weak design spots when it costs less to improve them, rather than repair consequent errors at later implementation phases.
5. To Selection of design alternatives in an objective way.

6. To predict external quality characteristics for example, reusability, maintainability, etc., and improve resource allocation based on these predictions.

Analysis of Existing Researcher Work :

S.No	Existing Researcher	Work
1.	Chidamber and Kemerer [Chida91; Chida94], Li and Henry [Li93b], Brito e Abreu and Carapuça [Brito94], Lorenz and Kidd [Loren94], Briand et al. [Brian97], Marchesi [March98], Harrison et al. [Harri98], Bansiya et al. [Bansi99; Bansi02]; Genero et al. [Gener00; Gener02].	The aim of this work is to present a broad survey and analysis of the existing literature of OO measures that can be applied to measure internal quality attributes of class diagrams.
2	El-Emam .	An interesting state-of-the-art of OO metrics and aspects related to their theoretical and empirical validity, but he only focused on one quality characteristic, fault-proneness.
3.	Card et al. [Card01]	A broad survey of the literature that assesses the state-of-the-art and practice in OO measurement and modeling, and maps the information collected onto the Practical Software Measurement framework (PSM), specially focuses on the view of quality as functional correctness.

Limitations of Object-Oriented Metrics

1. The success of object-oriented approach in the software development can be mainly attributed to its effectiveness in controlling the complexity & maintainability, and in increasing the reusability of its classes. But one needs to measure the strength/weakness of the classes and their dependencies in order to control and reuse them.
2. The intra-class weakness is computed with help of usage of various data members in the corresponding member functions. The inter-class weakness is found with help of calculating the dependencies (direct as well as indirect) of the class over remaining classes in the object-oriented software.
3. ISO/IEC international standard (14598) on software product quality states, "Internal metrics are of little value unless there is evidence that they are related to external quality." It need be noted that the validity of these metrics can sometimes be criticized [9]. Many things, including fatigue and mental and physical stress, can impact the performance of programmers with resultant impact on external metrics. "The only thing that can be reasonably stated is that the empirical relationship between software product metrics are not very likely to be strong because there are other effects that are not accounted for, but as has been demonstrated in a number of studies, they can still be useful in practice. [11]"

CK Metrics: Chidamber and Kemerer proposed a first version of these metrics and later the definition of some of them were improved and presented. Only three of the six CK metrics are available for a UML class diagram.

Table:1 Existing Metrics

S.No	Existing Metrics	**Metric name**
	CK metrics	WMC,DIT,NOC
	Li and Henry's metrics	DAC, DAC',NOM,SIZE2
	MOOD metrics	MHF, AHF, MIF,AIF,PF
	Lorenz and Kidd 's metrics	Class size metrics—PIM,NIM,NIV,NCM,NCV; Class inheritance metrics-NMO,NMI,NMA,SIX; Class internals-APPM
	Briand et al.'s metrics	ACAIC,OCAIC,DCAEC,OCAEC,ACMIC,OCMIC,DCMEC,OCMEC
	Marchesi's metrics	CL1,CL2,PK1,PK2,PK3,OA1,OA2,OA3,OA4,OA5,OA6,OA7
	Bansiya et al.'s metrics	DAM,DCC,CAMC,MOA,MFA
	Genero et al. 's metrics	NAssoc,NAgg,NDep,NGen,NGenH,NAggH,MaxDIT,MaxHAgg

Methodology :

This work is organised as follows: The existing proposals of OO metrics that can be applied to UML class diagrams are presented. An overall analysis of all the proposals. Finally, presents some concluding remarks and highlights the future trends in the field of metrics for UML class diagrams.

Future work and Scope

Further work is also necessary towards measuring OO models which cover dynamic aspects of OO software, such as, sequence diagrams, state chart diagrams, etc. As a final reflection, researcher want to remark that software measurement suffers from typical symptoms of any relatively young disciplines. Despite all the efforts and new developments in research and international standardization during the last decade, there is not a consensus yet on the concepts and terminology used in this field. With the goal of contributing to the harmonization of the different software measurement standards and research proposals, Researcher have proposed a thorough and comparative analysis of the concepts and terms used within each. This, in turn, may serve as a basis for discussion from where the software measurement community can start paving the way to future agreements.

Conclusions

The main contribution of this work is a survey and analysis of most of the existing relevant works related to metrics for class diagrams at initial stages of development, providing practitioners with an overall view on what has been done in the field and which are the available metrics that can help them in making decisions in the early phases of OO development. This work will also help researchers to get a more comprehensive view of the direction that work in OO measurement is taking.

Although the number of existent measures that can be applied to UML class diagrams is low in comparison with the large number of those defined for code or advanced design, researcher believe there needs to be a shift in effort from defining new metrics to investigate their properties and applications in replicated studies. researcher need to better understand what measures are really capturing, whether they are really different, and whether they are useful indicators of external quality attributes such as maintainability, productivity, etc. The need for new measures will then arise from, and be driven by, the results of such studies.

Researcher must be conscious that "associating with numeric ranges the qualifications good and bad is the hard part". This can contribute to metrics being useful for IS designers to make better decisions in their design tasks, which is the most important goal for any measurement proposal to pursue if it aims to be useful. Researcher suggests the contribution of aggregation relationships to design, evolution and reuse, have not been investigated at all. This is a topic that must be deeply investigated using some the metrics that have already been defined for this purpose or defining new ones if it is necessary.

References

1. Atkinson C. and Kühne T.: “Model-Driven Development: A Meta modeling Foundation”, IEEE Software, 20(5), 36- 41, (2003).
2. Bansiya J., Etzkorn L., Davis C. and Li W.: “A Class Cohesion Metric For Object-Oriented Designs”, The Journal of Object-Oriented Programming, 11(8), 47-52, (1999).
3. Bansiya J. and Davis C.: “A Hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Transactions on Software Engineering, vol. 28(1), 4-17, 2002.
4. Basili V., Briand L. and Melo W.: “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Transactions of Software Engineering, 22(10), 751-761, (1996).
5. Basili V. and Rombach H.: “The TAME project: towards improvement-oriented software environments”, IEEE Transactions on Software Engineering, 14(6), 728-738, (1988).
6. Basili V., Shull F. and Lanubile F.: “Building Knowledge through Families of Experiments”, IEEE Transactions on Software Engineering, 25(4),435-437,(1999).
7. Basili V. and Weiss D.: “A Methodology for Collecting Valid Software Engineering Data”, IEEE Transactions on Software Engineering, 10, 728-738, 1984.
8. Boehm B.: Software Engineering Economics, Prentice-Hall, 1981.
9. Briand L., Morasca S. and Basili V.: “Property-Based Software Engineering Measurement”, IEEE Transactions on Software Engineering, 22(6),68-86, (1996).
10. Briand L., Devanbu W. and Melo W.: “An investigation into coupling measures for C++”, 19th International Conference on Software Engineering (ICSE 97), Boston, USA, 412-421,(1997).
11. Briand L., Wüst J. and LounisH.:”Investigating Quality Factors in Object-oriented Designs: An Industrial Case Study”, Technical report ISERN 98-29, version 2, (1998).
12. Briand, L., Daly J. and Wüst J.: “A Unified Framework for Coupling Measurement in Object-Oriented Systems”, IEEE Transactions on Software Engineering, 25(1),91-121,(1999).
13. Briand L., Arisholm S., Counsell F., Houdek F. and Thévenod-Fosse P.: “Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions”, Empirical Software Engineering, 4 (4),387-404, (2000).
14. Briand L., Wüst J., Daly J. and Porter V.: “Exploring the relationships between design measures and software quality in object-oriented systems”, The Journal of Systems and Software, 51,245-273, (2000).
15. Briand L., Bunse C. and Daly J.: “A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs”, IEEE Transactions on Software Engineering, 27(6), 513-530, (2001).
16. Briand L., Morasca S. and Basili V.: “An operational process for goal-driven definition of measures”, IEEE Transactions on Software Engineering, 28(12), 1106-1125, (2002).
17. Brito e Abreu F. and Carapuça R.: “Object-Oriented Software Engineering: Measuring and controlling the development process”, 4th International Conference on Software Quality, Mc Lean, VA, USA, (1994).
18. Kaushik M., Mathur B., “ An Experimental analysis of Parent Teacher Scale Involvement with help of K Mean Clustering Technique using Matlab” ,International Journal of Advance Research in Computer Science and Management Studies(IJARCSMS),2(9), 117-123, (2014)
19. Kaushik M., Mathur B., “Comparative Study of K-Means and Hierarchical Clustering Techniques”, International Journal of Software & Hardware Research in Engineering (IJSHRE),2(6),(2014).
20. Kaushik M., Mathur B. ,“Data Analysis of Students Marks with Descriptive Statistics”,International Journal on Recent and Innovation Trends in Computingand Communication(IJRITCC),2(5),1188-1191. (2014).
21. Kaushik M., Mathur B., “K-Means Cluster Analysis” International Conference (FTCEE-14) on Futuristic Trends in Computer and Electronics Engineering ,Jaipur (2014).
22. Kaushik M., Mathur B., “Evolution of Object Oriented Method for Library Management System,”4th National Conference on Computational and Mathematical Sciences (COMPUTATIA-IV,2014) on November 25th -26th , 2014 at Department of Computer Science and Mathematics ,VIT CAMPUS, Jaipur Sponsored by Department of Science and Technology ,Rajasthan, & Technically sponsored by Indian Society of Information Theory & Appplication & Rajasthan Academy of Physical Sciences and Dept. of Science & Technology ,Rajasthan.
23. Kaushik M., Mathur B ., “The Role and Issue of Clustering Techniques in Designing Maintainable Object Oriented System”, in International Journal of Computer Science and Software Engineering ,Volume 1, Number 1 (2014), pp. 17-24,December ,2014 ,Impact Factor 2.38.