EMPIRICAL RESEARCH PRESS

International Journal of Engineering and Applied Computer Science (IJEACS)

Software Complexity Measurement: A Critical Review

Harmeet Kaur
Ph.D. (Computer Applications)
Research Scholar
Punjab Technical University Jalandhar, Punjab, India

Gurvinder N. Verma
Professor & Hood-Applied Sciences
Shri Sukhmani Institute of Engg. & Tech.
Derabassi, Punjab, India.

Abstract **- It is considerably recognized that in software engineering, the utilization of metrics at the initial stages of the object oriented software can encourage designers to bring about a noticeable improvement decisions. In this paper, a literature review and classification scheme for software complexity measurement researches is presented. The study shows that an expanding volume of complexity measurement has been conducted in diverse range of areas. As software complexity is an important factor that ought to be recognized at different levels of software development and it requires in profundity study with comprehension. Examinations of the chosen scrutinizes are completed and holes in the exploration are recognized. Analyses of the selected researches are completed and crevices in the research are identified. A complete record of references is explored. This review is planned to furnish driving force in exploration and help simulate further interest.**

Keywords— LOC; Complexity; SDLC; MOOD; OOP.

I. INTRODUCTION

More than 200 papers, which were published between the time period 1974-2012 in international journals and conferences of IEEE were collected, analyzed and classified into a number of categories and subcategories. The study led to the identification of gap in software complexity measurement researches and enabled the authors to recommend the area where there is a lot of scope for future research.

A. What is Software Complexity?

In last decades, software complexity has created a new era in computer science. Software complexity could be defined as the principle driver of cost, reliability and performance of software. In any case, there is no common agreement on software complexity definition, yet the greater part of them is dependent upon Zeus perspective of software complexity (Zuse, 1993)," software complexity is the level of challenge in analyzing, maintaining, testing, designing and modifying software ". In other words, software complexity is an issue that is in the whole software development process and each phase of software development life cycle (SDLC).

Software complexity is a broad topic in Software Engineering and has attracted numerous workers since1976, and numerous metrics have been proposed to measure software complexity. This measurement is exceptionally imperative in the software management and assumes a major role in project success. Complexity strongly impacts the needed effort to analyze and portray requirements, design, code, test and debugging the system during the development phases of software. In maintenance phases, complexity indicates the trouble in error correction and the needed effort to change distinctive software module.

The expanding vitality of software measurement and metrics accelerated the growth of new software complexity measurement and in software engineering metrics are essential for estimations for project planning and project measurement. The increased demand for software quality has brought about higher quality software and these days quality is the fundamental differentiator between the software products. Due to this reason software designers and developers require substantial measures for the assessment, improvement and acceptance of software product from the initial stages. These days software measurement assumes an essential part for measuring complexity and quality of software. Since software complexity influences software development effort, cost, testability, maintainability and so on. Thus it is indispensible to measure the software complexity in every software development phase. A variety of metrics have been proposed for measuring software complexity.

II. IDENTIFIED SOFTWARE COMPLEXITY MEASUREMENT RESEARCH AND EXTENT OF RESEARCH CARRIED OUT

The research framework shown in fig.1 is based on the literature review and the nature of software complexity measurement researches, which meant to give an understanding of how the subject has evolved and progressing. Since software complexity measurement is an important area it is very essential to have clear and better understanding of it.

ISBN: 978-0-9957075-0-4

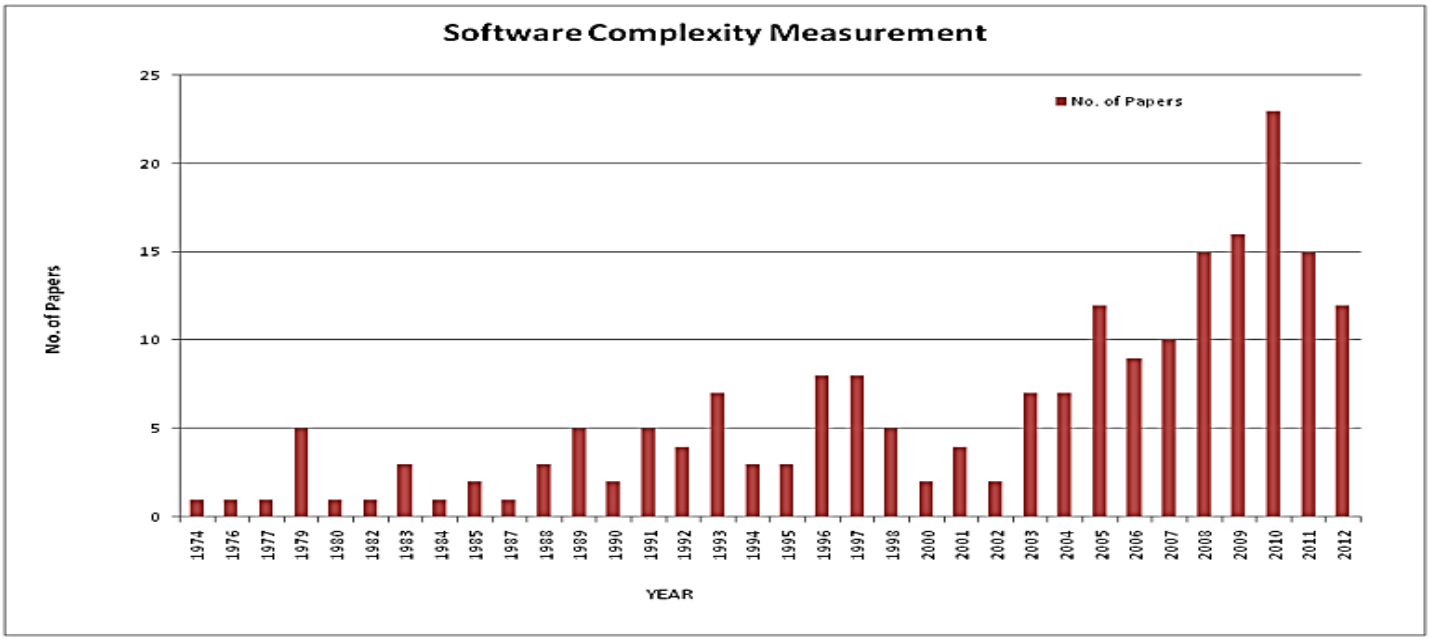

Figure.1 Number of papers published in the software complexity measurement based upon the year of publishing.

There are several researchers working in this area and trying to define and specify the universally accepted list of software complexity measurement methods and techniques and s trying to apply all these methods and techniques on a real project but until now they could not do that completely and perfectly and still they are working on it. Based on the literature survey software complexity can be classified as under:

TABLE 1: VARIOUS TYPES OF COMPLEXITY

Sr. No.	Type of Complexity
1	Architectural Complexity
2	Cognitive Complexity
3	Cyclomatic Complexity
4	Component and Time complexity
5	Control Flow Complexity
6	Computational Complexity
7	Data Scope Complexity
8	Functional Complexity
9	Inheritance Complexity
10	Program Complexity
11	Problem Complexity
12	Software Complexity
13	Syntactic Complexity
14	System complexity

In Table1 various types of complexities are shown and from the researches it was observed that cyclomatic complexity measures maximum numbers of independent paths in the program control graph whereas component and time complexity are related to the number of components and time whereas computational complexity is measure of computations involved in measuring the complexity of the program. It has been noticed that system complexity affects the reliability of the software while inheritance complexity measures the level of inheritance in the OOP and so on.

III. EXISTING SOFTWARE COMPLEXITY MEASURES

The predominant question is "What is Complexity?" IEEE outlines software complexity as "the degree to which a system or component has a design or execution that is challenging to comprehend and verify (IEEE, 1990). Through the years, research on measuring the software complexity has been carried out to comprehend, what makes computer programs difficult to understand. Few measures have indicated concern to propose the complexity measures whose calculation itself is not complex. A major force behind these efforts is to increment our capability to predict the effort, quality, coding efficiency, cost or all of these. Major complexity measures of software that refers to effort, time and memory expended have been utilized in the form of Halstead's software metric (Halstead, 1977), Mccabe's cyclomatic complexity (McCabe, 1976), Klemola's KLCID complexity Metric (Kelomola & Rilling, 2009), Wang's cognitive functional complexity (Shao wang, 2003) and many more.

ISBN: 978-0-9957075-0-4

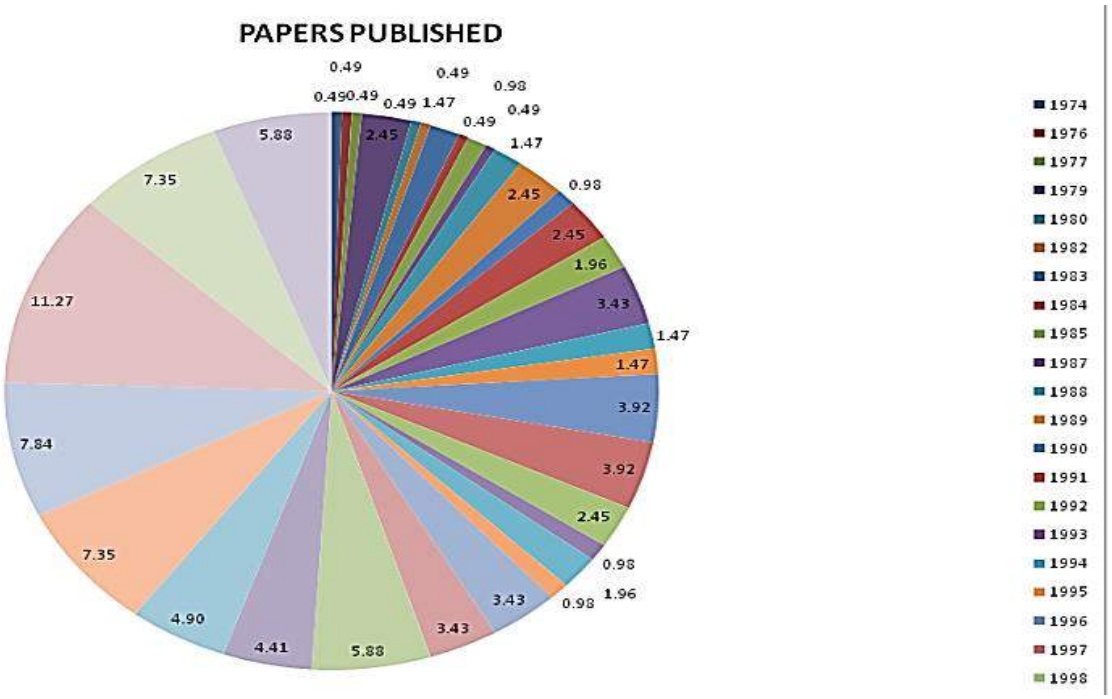

Figure 2. Percentage Distribution of the Papers Published in a Year.

The degree to which characteristics that hinder software maintenance are available is called software maintainability and is determined principally by software complexity, the measure of how demanding the program is to comprehend and work with. It has been evaluated that about 40 to 70% of the yearly software expenditure is spent on maintenance of software so if the complexity of the software is comprehended by the programmer than the maintenance procedure could be balanced. Maintenance characteristics that are influenced by complexity incorporate software understandability, software modifiability, and software testability. Different methodologies may be taken in measuring complexity characteristics, for example Baird and Noma's approach, in which scales of estimation are divided into four types.

In light of the fact that a great part of the software complexity measurement has been done in the last few years, numerous diverse techniques are being utilized. (Basili, 1975) has recommended that program size, data structures, dataflow, and flow of control can influence maintenance. Various measures have been developed to assess each of these aspects, and numerous hybrid measures have been created to acknowledge more than one concurrently. One of the central issues in software engineering is the inherent complexity. Since software is the consequence of human innovative activity, cognitive informatics assumes important role in comprehending its basic attributes.

(Shao & Wang, 2003) displays one of the principal aspects of software complexity, by inspecting the cognitive weights of basic software control structures. Taking into account this methodology another idea of cognitive functional size of software is developed.

The cognitive functional size furnishes an establishment for cross-stage examination of analysis of complexity, size, and comprehension effort in the design, execution, and maintenance phases of software engineering. Few of the methods and techniques are discussed here and plethora of literature is available on various methods and techniques used for software complexity measurement.

IV. Present Issues In Software Complexity Measurement

While going through the literature it was observed that many researchers have discussed about various types of software complexity for example structural complexity, functional complexity, psychological complexity etc. in the literature control flow complexity in terms of weyuker,s properties has been discussed. Functional complexity focuses complexity that results from factors related to system structure and connectivity. One of the studies aimed at finding the relation between complexity and security i.e. it was explored whether the more complex code is less secure or vice versa. Many of the studies have surveyed well known complexity measures like McCabe's cyclomatic complexity, halstead method, Loc etc. and few of the studies have given new ways of measuring LOC, Mc Cabe's cyclomatic complexity and Hallstead method. A new ways of measuring aspect oriented metrics and metrics to measure complexity of business process has also been discussed. A graph-theoretical complexity metric to measure object-oriented software complexity is also described. It shows that inheritance has a close relation with the object-oriented software complexity, and reveals that misuse of repeated (multiple) inheritance will increase software complexity and be prone to implicit software errors.

ISBN: 978-0-9957075-0-4

System complexity comprised of internal and external complexity it was examined that system complexity ordinarily influences characteristics for software reliability, maintainability, and testability of software systems which are recognized as of utter significance in composing an improved software product. For accomplishing these software qualities, system complexity must be regulated by modularizing the system into different modules of suitable complexities. Software complexity measures are regularly proposed as suitable indicators of diverse software quality traits.

An incredible deal of effort is currently being dedicated to the study, analyses, expectation, and minimization of expected software maintenance cost, much sooner than software is conveyed to users or stakeholders. It had been evaluated that, on an average, the effort spent on software maintenance is as expensive as the effort used on all other software stages. Ways to mitigate software maintenance complexity and increased cost may originate in software design.

In the past data complexity has been overlooked in measuring the software complexity however now few of the studies have done on data complexity and data scope complexity. The data scope complexity can demonstrate complexities of various characteristics of object oriented programming in the meantime and work to quantify object oriented and procedure oriented programming has been done. Many researchers have concentrated their work on measuring the cognitive weight unpredictability and the information flow complexity which is dependent upon the information held by the program. Much of work has been done on measuring the software complexity yet this field needs further research for the advancement of software complexity measurement techniques and strategies.

V. CONCLUSION

From the literature it has been concluded that software complexity measurement is a subject of concern for the researchers since 70's.From fig.1 it is clear that maximum work on software complexity measurement is done in the year 2010 followed by 2008-09, 2011 and 2012. From 1974-77 minimum work has been dealt with and slight increase in the software measurement complexity area has been seen in 1979.After 2003 it can be concluded that an increase in the work of software complexity measurement has been observed. The Maximum percentage of papers were published in the year 2010 followed by 2008-09 as depicted in the fig.2.Since complexity affects the quality attributes and cost of the software so it is an area of interest for the researchers working in the field of software development and maintenance and it requires further research.

VI. FUTURE SCOPE

Scientific investigation improvements are to be made in software productivity and quality. The development of new complexity metrics is required to refine the measures. Further studies are needed to fully resolve the question about the effectiveness of traditional metrics in measuring OOP software complexity. The work on only one object-oriented feature of complexity, that is, inheritance level has been done. Other object-oriented features like polymorphism, information hiding, and encapsulation require further study. As the work was concentrated on small object-oriented programs but it is expected that more complex object-oriented systems will be explored. It is expected to see continued use and further development of OO metrics is required in the years ahead In order to empirically validate the complexity metrics experiments need to be carried out it has been suggested that experimentation is a crucial part of evaluation of new metrics. Another important issue that needs to be investigated is what are both the meaning of complexity metrics and the precise number to use as a complexity limit in a process development. Future investigations are necessary to clarify how complexity of aspect-oriented programs depends on the internal structure of the code. The software complexity metric is becoming an extremely important part of the software engineering. And more work is required in this field in the future. In future an integrated approach to measure the software complexity is needed.

REFERENCES

[1] E. Brito, F. Abreu, W. Melo, "Evaluating the impact of Object-Oriented Design on Software Quality,"Proceedings of 3rd International Metric Symposium, 90-99, 1996.

[2] G. K. Gill and C. F. Kemerer, 1991, "Cyclomatic complexity density and software maintenance productivity," IEEE Transactions on Software Engineering, Vol. 17, No.12, pp. 1284–1288,1991.

[3] Halstead, M.H., Elements of Software Science, Elsevier North, New York, 1977.

[4] IEEE Computer Society: IEEE Standard Glossary of Software Engineering Terminology, IEEE Standard 610.12 – 1990, IEEE.

[5] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," IEEE Transactions on Software Engineering, Vol. 18, No. 5, pp. 423–433,1992.

[6] McCabe, T.H., "A Complexity Measure", IEEE Transaction on Software Engineering, SE – 2, 6, pp. 308 – 320, 1976.

[7] Misra,S and Misra,A.K. ,"Evaluating Cognitive Complexity measure with Weyuker Properties,"

[8] Proceeding of the3rd IEEE International Conference on Cognitive Informatics, 2004.

[9] M. Marchesi,"OOA metrics for the United Modeling Languages," Proceedings of 2nd Euromicro Conference on Software Maintenance and Reengineering, Palazzo degli Affari, Italy, 67-73, 1998.

[10] M. Genero, M.E. Manso, M. Piattini, et al,"Early metrics for object oriented information systems," Proceedings of 6th International Conference on Object Oriented Information Systems, London, UK, 414-425,2000.

[11] M. R. Woodward, M. A. Hennell and D. A. Hedley, "A measure of control flow complexity in program text," IEEE Transactions on Software Engineering, Vol. 5, No. 1, pp. 45–50,1979.

[12] R. D. Banker, M. D. Srikant, C. F. Kemerer, and D. Zweig, "Software complexity and maintenance cost," Communications of the ACM, Vol. 36, No. 11, pp. 81–94, 1993.

[13] R. Subramanyam &M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects," IEEE Transactions on Software Engineering, Vol. 29, No. 4, pp. 297–310, 2003.

[14] S. Chidamber, C. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, 20(6), 476-493, 1994.

[15] Sheng, Yu., Shijie, Zh.,"A survey on metric of software complexity,"2nd IEEE International Conference on Information Management and Engineering, pp.352-356,2010.

[16] Tian, J., and Zelkowitz, M. V, "Complexity Measure Evaluation and Selection,"IEEE Transactions on Software Engineering, vol. 21, No. 8: 641—650, 1995.

[17] T. Menzies, J. Greenwald and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Transactions on Software Engineering, Vol. 33, No. 1, 2–13, 2007.

[18] Tuomas Klemola and Juergen Rilling, "A Cognitive Complexity Metric Based on Category Learning," IEEE International Conference on Cognitive Informatics, 2003.

[19] V. Basili and A. Turner, "Iterative Enhancement: A Practical Technique for Software Development," IEEE Trans. Software Eng., Vol. SE-I, pp. 390-396, 1975.

[20] Wang, Y., and Shao, J, "Measurement of the Cognitive Functional Complexity of Software," IEEE International Conference on Cognitive Informatics, 2003.

[21] Zuse, H, "Criteria for Program Comprehension Derived from Software Complexity Metrics," Proceedings of the Second International Workshop on Software Comprehension, IEEE, Capri/Italy: 8—16, 1993

AUTHOR PROFILE

Harmeet Kaur has seven years of experience in academic and research, she is currently pursuing Ph.D. in Computer Applications from Punjab Technical University, Jalandhar, India; after completing two years Master of Technology (I.T.) degree in year 2011; three years Master of Computer Applications (M.C.A.) degree in year 2008; she had the recipient of UNDP scholarship to conduct high quality research, is member of IEEE and has qualified National Eligibility Test (NET) conducted for lectureship. Presently she is working as Assistant Professor of Software Engineering in Green Hills Engineering College, Solan (HP). She has published three research articles; her area of research includes Program Analysis, Software Reengineering and Quality Assurance.

Prof. (Dr.) Govinder N. Verma has more than twenty years of wide experience in research, academics and administration, held various positions as Director, Professor, lecturer in universities and engineering colleges after completion of Ph.D. in Fluid Mechanics from Himachal Pradesh University, Shimla, with excellent grade; he has completed M.Phil, M.Sc., B.Sc. in Applied Mathematics all in first division from Himachal Pradesh University, Shimla; he has published five research papers in reputed journals and supervising Ph.D. thesis; currently he is designated as Professor and Head of Department of Applied Science of Shri Sukhmani Institute of Engineering and Technology, Derabassi, affiliated to Punjab, Technical University, Jalandhar, India.

ISBN: 978-0-9957075-0-4