ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
June 2017,
Vol. 10, No. (2):
Pgs. 480-490

Research Summary of A Study for the Estimation of Legacy Programs for Effective Reengineering

HARMEET KAUR[1], SHAHANAWAJ AHAMAD[2] and GOVINDER N. VERMA[3]

[1]Research Scholar PhD.Computer Applications I.K.G.PTU Jalandhar
[2]DCScSWE college of ComputerScience and Engg University of Ha'il KSA
[3]Sri Sukhmani Institute o f Engg. & Technology Derabassi Punjab
Corresponding author Email: hrmt01@yahoo.com

http://dx.doi.org/10.13005/ojcst/10.02.32

(Received: May 10, 2017; Accepted: May 29, 2017)

ABSTRACT

The present research estimates the efficacy of a legacy program and the areas of its development. The research also intends to put forward as to what extent reengineering of a legacy program has to be done on the basis of the estimation approach. The study has tried to outline the current issues and trends in reengineering of a legacy program from various perspectives. An all-inclusive literature review reveals that a lot of work has already been piled up with legacy system estimation and the reengineering domain, yet the basic assumptions of Complexity, Quality and Effort have not been worked out collectively. Hence the present research underlines this very maxim and studies the reengineering of a legacy program on the paradigms of Quality, Complexity, and Effort Estimation collectively. The findings put forward an equation and reengineering scale which would be highly compatible with present technology for the feasibility of an effective reengineering.

Keyword: Legacy, CQE, COBOL, Softgoal, Reengineering

INTRODUCTION

The working terminology of a legacy system was created some 20 to 30 years ago by legacy software engineers which was typically written in COBOL, FORTRAN, C or C++ and was called as a legacy program. Its relevance has never lost its grounds. Legacy frameworks are thought to be conceivably risky by numerous software engineers. These programs face certain difficulties with the latest in technology as legacy software runs on obsolete hardware. The expenses of keeping such legacy programs may inevitably exceed as the expenses of replacing both the software and the hardware are way high. Thus, Reengineering offers a way to deal with migration of legacy program towards an evolvable system in a disciplined way. The reengineering process might be seen as

applying engineering principles to a current legacy program so as to meet new requirements. So, in order to be successful, reengineering requires insight from various point of view.

The research is focused on complexity,Quality and effort(CQE) estimation of legacy program for effective reengineering. While zeroing down the research problem, a plethora of literature was explored including journals, research papers, books etc. With the advancement in the field of technology, it is important that one should enhance the coding of the software systems so as to meet the requirements of the ever growing software market. As a matter of fact, the legacy system is an important asset of an organization that cannot be discarded as it holds plenty of the resources of the organization. The research was carried out so as to decide whether the reengineering of the legacy systems is to be undertaken or not.

Literature Review

Complexity

The software complexity has been measured by many researchers using various affecting attributes such as control flow paths McCabe [1976]. Munson & Khoshgoftaar[1989] have examined recent investigations in the area of software complexity. Munson & Khoshgoftaar[1992] done study on measuring dynamic program complexity. Kim et.al.,[1995] presented a new framework for analyzing the scope of metrics to evaluate complexity of object-oriented programs. Munson & Hall [1995] conducted a research on dynamic program complexity and software testing. Kim et.al.,[1996] have proposed new metrics for computing the program complexity of object-oriented program. Dantsin[1997] surveyed various complexity results on different forms of logic programming.. Halstead [1977] worked on the volume of operands and operators. Wang and Shao[2003] conducted study on cognitive complexity. Cherkaskyy & Sadek [2004] have studied the various levels of program complexity. Cardoso et.al., [2006] have surveyed findings from neighboring disciplines on how complexity can be measured. Misra[2007] proposed an improved cognitive complexity measure. Gupta & Chhabra[2009] proposed new cognitive-spatial complexity measures. Srivastav et.al.,[2010] have proposed a new technique to calculate the complexity of faulty program slices. In an another study Kumar & Kaur[2011] have compared the complexity in accordance with object oriented metrics proposed in 90's. Debbarma et.al.,[2013] carried a review and analysis of software complexity metrics in structural testing.

Quality

Stockman et.al., [1990] presented a framework for the measurement of software quality.. Wells et.al.,[1995] proposed customized tools for software quality assurance and reengineering. Basili et.al.,[1996] focused on a validation of object oriented design metrics as quality indicators. Khoshgoftaar & Allen[1997] studied the impact of costs of misclassification on software quality modeling. Chung et.al.,[2000] worked on non-functional requirements in software engineering. Cysneiros & Leite[2002] non-functional requirements from elicitation to modeling. Khoshgoftaar et.al.,[2002] worked on quality driven software re-engineering. Hill et.al.,[2004] worked on quantifying non-functional requirements a process oriented approach. Khoshgoftaar et.al.,[2004] undertook work on unsupervised learning for expert-based software quality estimation. Kassab et.al.,[2008] a metamodel for tracing non-functional requirements. Retna et.al., [2010] presented a study on quality of software and the metrics for evaluation. Xu.,[2010] presented an empirical study on the procedure to derive software quality estimation models. Fontana & Maggioni [2011] studied metrics and antipatterns for software quality evaluation. Sun[2011] presented knowledge for software quality control and measurement. Bajpai & Gorthi., [2012] on non-functional requirements a survey. Trivedi & Kumar.,[2012] focused on software metrics to estimate software quality using software component reusability..

Effort

Symons[1988] worked on function point analysis difficulties and improvements. Mukhopadhyay et.al.,[1992]examining the feasibility of a case-based reasoning model for software effort estimation. Clemons et.al.,[1995] proposed an integrative framework for identifying and managing risks associated with large scale reengineering efforts. Subramanian & Breslawski.,[1995] undertook

an empirical analysis of software effort estimate alterations. Shepperd et.al., [1996] studied effort estimation using analogy. Gray et.al.,[1999] studied factors systematically associated with errors in subjective estimates of software development effort the stability of expert judgment. Clark[2000] focused on quantifying the effects of process improvement on effort. Hill et.al., [2000] presented expert's estimates of task durations in software development projects. Jorgensen and Sjoberg[2001] studied the impact of effort estimates on software project work. Jorgensen[2004] presented a review of studies on expert estimation of software development effort. Kaczmarek & Kucharski[2004] worked on Size and effort estimation for applications written in java. Jorgensen[2005] worked on practical guidelines for expert-judgement-based software effort estimation. Grimstad et.al.,[2006] focused on software effort estimation terminology the tower of babel. Menzies et.al.,[2005] worked on validation methods for calibrating software effort models..Sandhu et.al.,[2009] proposed a model for estimation of efforts in development of software systems.

Research Problems and Issues

Since the introduction of the legacy software systems, they have proved quite useful to the organization not only catering to the business domain of the organization but also to keep the data intact and safe. Despite their relevance, being outdated these legacy systems are not easy to get discarded. Replacing legacy systems is a costly affair. Keeping these things in mind it is highly suggested to upgrade these systems so that they can easily meet the requirements of the ever-changing software industry. Since the organizations are concerned about the complexity, cost, effort and quality; it is necessary to have the knowledge of the same. From the literature survey and simple meta-analysis, it becomes evident as to what extent it is vital to consider CQE approach before taking decision whether the reengineering of the system will be beneficial or not.

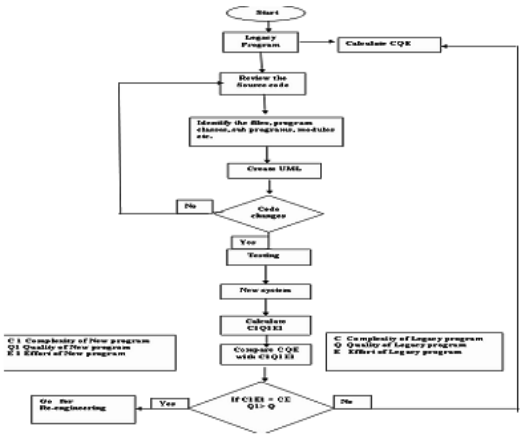

Fig. 1: Showing an overview of the methodology

ISSUES

- The legacy programs are written in archaic languages
- These are utilizing old software procedures for the development of the program
- Lack of documentation, maintenance and specialists
- Legacy program code is hard to re-module, extensive and complex
- It is difficult to comprehend the rationale of the legacy program
- Integration with newer systems may also be difficult because new software may use completely different technologies
- Legacy systems can be hard to maintain, improve, and expand because there is a general lack of understanding of the system

Research Objectives

The present research has put forth an extensive approach on the paradigms of reengineering as complexity, quality and effort as its basic estimation parameters. As a lot of work has already flooded the reengineering market in and around the globe, the present approach focuses on the advantages of complexity, quality and effort over other parameters of reengineering. As reengineering has earned its global repute for its usability, with the changing times and spread of technology the complexity, quality and effort approach has advantages of its own. The advantages of using complexity, quality and effort lies in the fact that at the customers end it increases the quality of the end product which at times suffer because of the above mentioned approaches. Also it must be added that the legacy code on to which a given organization works ensures that the reengineering has to take place in such a way that a legacy code has to be upgraded to meet the present

beside future's technical needs and also should not affect their business during such developments. The complexity, quality and effort approach also has another key feature of keeping the reengineered legacy code as simple as it can be which rules out the complexity factor to enhance its prospect. The objectives of the present study are summed up as under:

- To measure the complexity of the legacy program
- To measure the quality of the legacy program
- To measure the effort of the legacy program
- Development of framework
- Develop a model for CQE estimation
- To Develop an equation for CE estimation
- To develop an equation for quality estimation
- To develop a reengineering scale for legacy system
- To measure the feasibility of reengineering a legacy program

Research Methodology

It is a well known fact that organizations cannot progress without doing research activities. Hence, in the present problem, exploratory research is used which helps to diagnose a situation and thinking of alternatives to discover new ideas and concepts. In this research, mixed research methodology i.e. qualitative and quantitative is used.

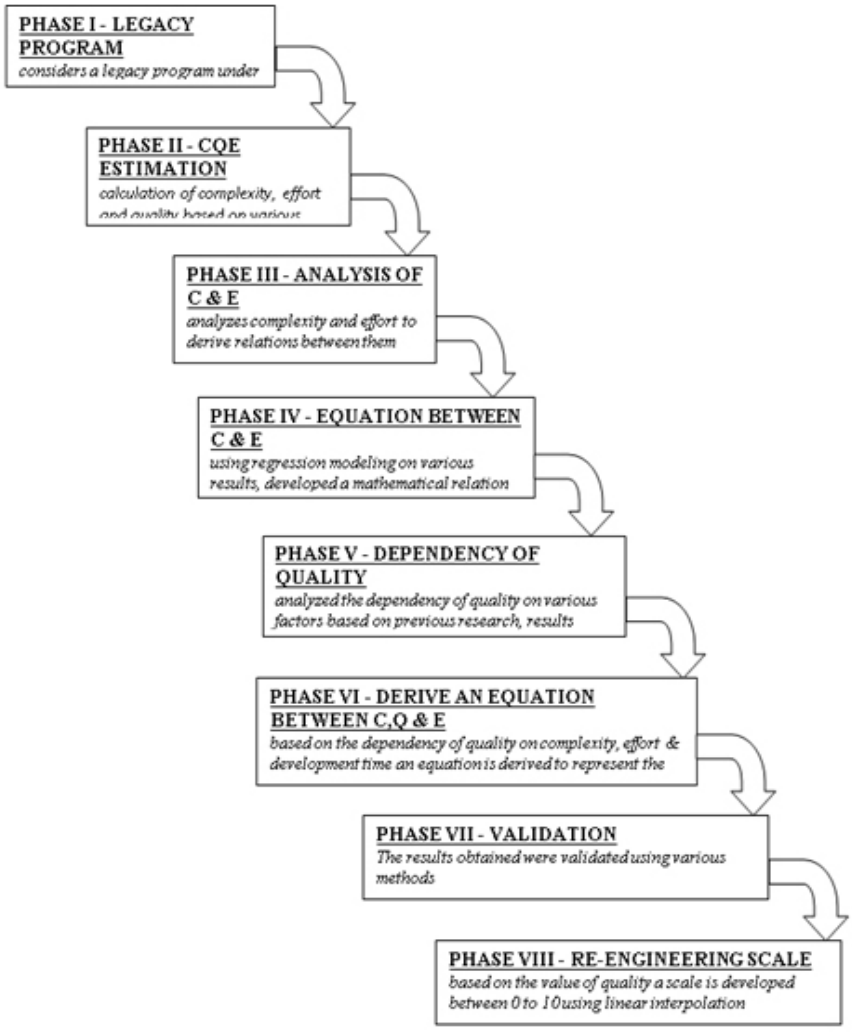

Fig. 2: Framework for the Estimation of Legacy Programs for Effective Re-engineering

Common type of qualitative research methodology is literature survey, experimental studies and others. In quantitative methodology common techniques are case studies, biographical studies etc.

Various estimation techniques, reengineering models and their application were studied. Also the terms viz. legacy code, legacy system and legacy program are used interchangeably in the present research. The present work was undertaken to effectively reengineer the legacy program by estimating CQE. The below mentioned flow chart illustrates the various stages of CQE estimation of a legacy program.

The various methodologies for CQE estimation are:

Complexity Estimation

The research has used Line of Code Method, Information Flow, Cyclomatic Complexity, Halstead's Method and Cognitive Weights Method.

Effort Estimation

Effort estimation is the way toward figuring the most practical utilization of effort required to create or keep up legacy software based on uncertain, incomplete and /or noisy input. The preferred outcomes can be utilized as input to planning a project, budget, speculation investigation, procedures and bidding rounds for better outcomes. Approved overviews on estimation practice propose that expert estimation is the prevailing tool while assessing the software effort.

Quality Estimation

The research used Non-Functional Requirement method (NFRs), Software Quality Goal and Logic Artifacts.

Considering the objectives of this research, numerical values obtained as output are used for the development of the equation using regression modeling which is known as CE equation. The values obtained from this equation gives the complexity and effort quotient of the legacy program under study and it fits in all the languages. Further, the values obtained from the CE equation were used for estimating the source code quality of legacy program. This further leads to the development of the reengineering scale. The values of the source code quality i.e. Q were fixed between 0 to 10 using linear interpolation method so that it becomes convenient to take decision whether the reengineering of the legacy code is feasible or not.

The research work was further carried out to estimate the quality of the legacy program. For measuring the quality of the legacy program, qualitative and quantitative methods were used. The NFR and softgoal interdependency graph method were used for measuring the quality of the legacy program. As the non functional requirements play an important role in the quality of the software, they cannot be ignored. The SIG method was applied on procedural as well as object oriented programming language. Also, the SIG method gave the results on the role of the NFR in measuring the quality of the software. To have more concrete results on the importance of NFRs in quality of the software, the study was further extended and field surveys were done to collect the data on the role of the NFRs. For the collection of the data, questionnaire was used. The data hence collected was further analyzed using total weight method to know the role of NFRs in the quality of the software.

Solution Approach

To have a solution of the research problem, a framework for the estimation of legacy programs for effective re-engineering has been suggested in fig.-1. This frame work consists of eight phases, the phase-I is for meta-analysis and to consider the legacy programs written in COBOL, C and C++. The meta-analysis helped to identify 68 variables out of these size, complexity and NFR are of the utmost importance. Phase-II is for the CQE estimation by applying few selected methods of estimation available in the literature. This phase gave the estimation results of complexity, quality and effort. Phase-III consisted of analysis of results of C and E and to derive a relation between them. In this phase various models were developed using tools used in model development. In phase-IV an equation between C and E was developed. For developing the equation regression modeling was applied on the results. During phase-V of the research, dependency of the quality on various

factors based on previous work and result obtained was studied mathematically. In this phase, to have more precise results on quality, the role of NFR using questionnaire, soft-goal interdependency graph method was also done. Quality of the legacy program was also analysed using expert judgement method. In phase VI – we have derived an equation between CQE. In this phase based on the dependency of quality on complexity, effort & development time an equation is derived to represent the source code quality. Phase-VII deals with the validation of the results obtained during the estimation. Phase - VIII dealt with developing re-engineering scale based on the value of quality.

Research Contribution

The literature review of the present research revealed that the reengineering of a legacy program was previously done either on the basis of complexity or quality or effort. The previous researches were devoid of a collective approach which could have reengineered a legacy program on all the three parameters collectively. Hence the contribution of the present research lies in the fact that it serves as the basis for the reengineering of a legacy program based on CQE approach collectively. Also the contribution of the research eliminating the problems of the CQE estimation of a legacy program for effective reengineering can be summed up as follows:

- Develop software which will help in CQE estimation and to develop a framework that helps to improve legacy system/migrant system by following suggested measures.
- Develop and design collection of comprehensive soft-goal interdependency graphs as they pertain to various NFRs of large procedural and object-oriented legacy systems.
- The development of the equation for complexity and effort estimation known as CE equation of the legacy system for effective reengineering. The equation can be applied on legacy system written in any language i.e. procedural and object oriented language.
- The design and development of the Complexity, quality and effort estimation models to address specific estimation objectives. Develop a reengineering scale which serves as the basis to see the utility of a legacy program to be reengineered or not.
- The design and implementation of reengineering scale that allows the identification of levels to decide whether the effective reengineering of the legacy system is feasible or not.
- The design and implementation of prototype system/models that assists in effective reengineering process that pertains to enhancements of legacy system.

RESULT AND DISCUSSION

This section deals with the various outcomes of the result for the CQE estimation of the legacy programs for their effective reengineering.

CQE Estimation Framework

1. To know the factors impacting CQE, simple meta- analysis was done and eight major dimensions were identified and sixty eight resultants parameters were explored. The results of the same were published in (Scope of Exploring CQE Dimensions in Reengineering of Legacy Program Harmeet Kaur Shahanawaj Ahamad Gurvinder N. Verma *International Journal of Advanced Research in Computer Science and Software Engineering*, *Volume 3, Issue 7, July 2013*).
2. Factors responsible for the complexity of the legacy program were studied and identified the elements for the complexity of a leagacy program. (Elements of legacy program complexity Harmeet Kaur, Shahanawaj Ahamad, Gurvinder N. Verma *International Journal of Research in Engineering and Technology eISSN: 2319-1163*)
3. For complexity estimation, the programs written in COBOL, C and C++ were selected from open source and complexity was estimated using various methods e.g Mc Cabe method, Halstead method etc. The obtained results were published in (legacy program estimation Harmeet Kaur, Shahanawaj Ahamad, Gurvinder N. Verma *International Journal of Computer Science and Information Security, Volume 14, No. 02, February 2016*).
4. Role of quality parameters and sub-parameters in the development of the quality

software system were also scrutinized. The findings revealed that Quality of software is affected by a number of parameters e.g. conceptual integrity is influenced by variable name and coding style whereas availability is affected by malicious attacks, system load etc. (*Identification & Analysis of Parameters for Program Quality Improvement: A Reengineering Perspective Computer Engineering and Intelligent Systems, ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online))*

5. Quality estimation was done using conceptual and empirical methods. Role of NFRs were studied and the results revealed that in an online banking system client is more worried about the security followed by performance, usability and availability of the online banking services. It was also observed that reliability, visibility and confidentiality are closely related with total weightage score. The results were published in (A case study upon non-functional requirements of online banking system Harmeet Kaur, Shahanawaj Ahamad, Gurvinder N. Verma *International Journal of Computer Applications Technology and Research Volume 4– Issue 4, 220 - 225, 2015, ISSN:- 2319–8656*).
6. To enhance the accuracy of results an equation known as source code quality equation was developed. This equation is used to measure the source code quality of a legacy program.

$$Q = \frac{e * DeT}{C}$$

$$Q(Adjusted) = k^* \frac{e * DeT}{C}$$

The derived equation can be used to estimate the source code quality of almost all the languages. *Where K is some Constant and its value is taken in such a way that the resultant value lie between 1 to 10 using linear interpolation i.e k = 10/max (Q)*

7. Quality was estimated using soft goal interdependency graph method. Design patterns of the procedural and object oriented languages were studied and their interrelationship and role were observed using SIG.
8. Effort estimation was done using various methods e.g. Expert judgement, FP, COCOMO I & II etc. The average values of the results so obtained were taken and relation between complexity and effort was studied.
9. The research further proceeds to develop an equation using regression modeling method between complexity and effort. The equation is known as CE equation and can be used to estimate CE of almost all the programming languages.
   E=17.84+0.19C
   Where E is effort and C refers to complexity.
10. To check the feasibility of reengineering, a reengineering scale was developed. The scale was developed using the values of Q and these were interpolated between 0-10 using linear interpolation. The scale is used to make a decision whether the reengineering of the system is to be done or not.

Implementation

The programs written in COBOL, C and C++ were taken from the open source and the selected estimation methods were applied on these to calculate CQE factor. The results of the estimation were compared with the programs, case studies and industrial data. It was observed that the results of CE were nearly same as the data taken from the field study. Using linear regression modeling an equation between C & E was developed. The developed equation shows the relation between C and E. The factors affecting quality were analysed using theoretical and mathematical methods which derived the relationship between complexity, cost, effort and quality. The values so obtained measures the source code quality. The values of source code quality are fixed between 0 to 10 using linear interpolation and the same were used in developing reengineering scale using ordinal scale method as mentioned in the literature. The scale will be used in deciding whether the reengineering of the legacy program/software is to be done or not.

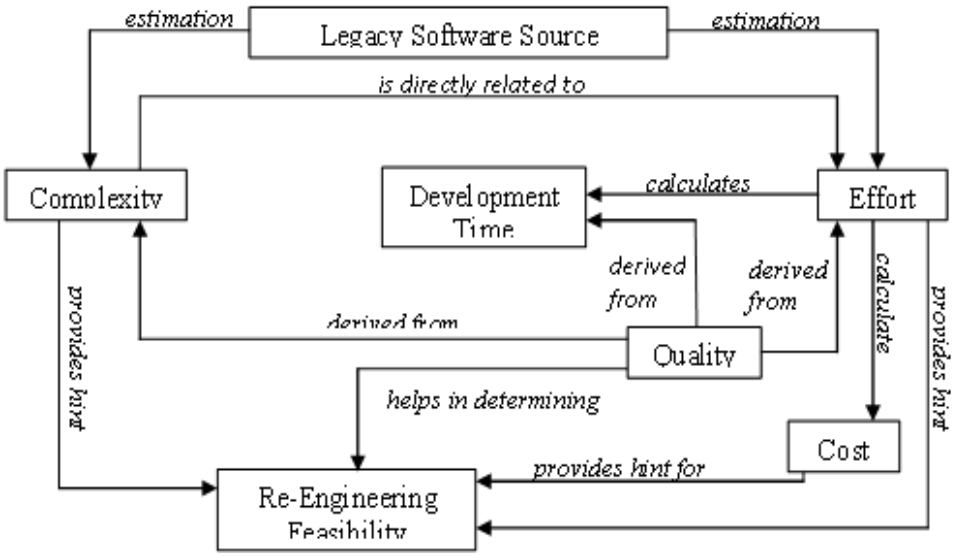

Fig. 3: Model for CQE estimation of legacy program

Validation

For validation various methods were used for estimating complexity effort and development time.

Empirical Validation

Empirical validation incorporates validation of different results on real projects. To validate the outcome, programs written in COBOL, C and C++ were taken from the industry dealing with reengineering and estimation. The projects incorporate modules, classes, libraries etc. Complexity, effort development time and so on were ascertained for every part of the three projects. Then overall complexity, effort development time etc. were computed. The outcomes so acquired were contrasted with the outcomes of the legacy programs taken from the open source. It was observed that the outcomes so obtained were at par with the outcomes acquired from open source.

Validation for Regression Modeling used in Quality Estimation

Average values of all the parameters calculated for complexity, effort development times etc. were used for the development of CE equation. The equation was developed using regression modeling and was validated using cross validation method. For the development of CE 15 results were randomly selected and were divided into two parts i.e. of 7 and 8 programs each. The first part of the sample is for exploration and model formulation and second part is for model validation, formal estimation and testing.

For the validation of the CQE estimation, the data were taken from the industries of repute which are dealing in the software estimation. The results which were hence gathered from the industry were of the same value as the results obtained during the CQE estimation. This indicates that the results validate for the construction of reengineering scale and CQE equation. Also the desired results validate the application of CQE estimation on other programming languages as well.

CONCLUSION

In this study CQE is estimated by using various methods as mentioned in the literature. A comparison was done between the parameters measured in COBOL, C and C++ and their interrelationship was also studied. In the first part, the research focused on most of the factors which affects the complexity of code. In the beginning five methods were selected for complexity estimation. To enhance the accuracy of the results few more method were later included so that it covers almost all the factors that contribute in complexity of the software. Based on the results, an equation known

as CE equation was proposed. The proposed equation was formulated to manifest the factors of complexity and effort and to show the interrelation between C & E. The equation can also be applied to codes which are written not only in a procedural language but also in an OO language since multi-paradigm encompasses both of them. The comparative case studies for CE have shown that CE has the following properties:

- combines many aspects in one measurement formula
- makes more sensitive calculation
- includes more complexity factors
- includes effort besides complexity
- gives interrelation between C and E

The higher the CQ, the better the code is. With the increments of C, E increases; with the decrement of C, E decreases. Higher FP indicates better functionality and merits.

Limitations and Future Scope

This research has some limitations which can regress the research work if the domain specific, development environment and tools are not found. The research findings includes verdict on reengineering, reuse of existing component and viability of a legacy program. The risks related to the research results are level of robustness of technology applied to carry out the research and accuracy of existing methods. One of the major challenges that a legacy program faces is the usability of its applications. In the absence of its documentation and experts, it becomes a daunting task to reengineer a legacy program. Also the users are at times reluctant to reengineer a legacy program in their organization.

All the existing metrics have their own merits. The goal of the thesis is not to criticize those metrics, equations and scales or to claim their inabilities but to understand their benefits and propose equations and scales based on some of them. There are also other metrics and methods used for estimating complexity, quality, effort, cost, development time etc. but they are out of research scope of the present study. Further research is required to add more CQE factors and simplify the equation and scale so that it becomes more practical. Although the study has tried to include most of factors, yet it has the possibility of adding few more factors.

ACKNOWLEDGEMENT

First I am thankful to the research supervisors (Dr.) Shahanawaj Ahamad,and (Dr.) Govinder Verma for their timely, scholarly and technical guidance and support. I am grateful for the support of the Department of CSE at the Sri Sukhmani institute of engineering and Technology, where I conducted this research.

REFERENCES

1. McCabe.A Complexity Measure. *IEEE Trans. Soft-ware Eng.*, 1976; 2: 308-320.
2. Munson, J.C., Koshgoftaar, T.M. Measuring Dynamic Program Complexity. *IEEE software.*1992; 9:48-55.
3. Kim, K., Shin, Y., Wu, C. Complexity measures for object-oriented program based on the entropy. *in Proceedings of the 2nd Asia-Pacific Software Engineering Conference* 1995; 127-136.
4. Munson**,** J.C. Hall, G.C.Dynamic Program Complexity and Software Testing. *Proceedings of the IEEE International Test Conference on Driving Down the Cost of Test*; 1995 730-737.
5. Kim, E.M. Heuristics for computing attribute values of C++ program complexity metrics.1996 IEEE.
6. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A. Complexity and expressive power of logic programming.*Proceedings., Twelfth Annual IEEE Conference on Computational Complexity*;1997: 82 – 101.
7. Halstead, M. Elements of Software Science, Elsevier North-Holland, New York,1977.
8. Shao, J., Wang, Y. A New Measure of Software Complexity Based on Cognitive Weights.

9. Canadian Journal of Electrical and ComputerEngineering.2003; 28(2): 69 –74.

10. Cherkaskyy ,M., Sadek, A.S.The levels of program complexity. *Proceedings of the International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science;2004.*

11. Cardoso,J. et.al., A Discourse on Complexity of Process Models. *International Conference on Business Process Management*;2006:117-128.

12. Misra ,S. Cognitive Program Complexity Measure *6th IEEE International Conference on Cognitive Informatics*;2007.

13. Srivasatav, M. et.al., An Optimized Approach of Fault Distribution for Debugging in Parallel. *Journal of Information Processing Systems*.2010;6(4):.537-552.

14. Kumar, R., Kaur G.Comparing Complexity in Accordance with ObjectOriented Metrics.*International Journal of Computer Applications*.2011;15(8):pp.42-45.

15. Sinclair G. Stockman et.al.,A Framework for Software Quality Measurement. *IEEE journal of selected areas in communication.*1990; vol.8(2):224 -233.

16. Wells,C.H. et.al.,Customized tools for software quality assurance and reengineering. *Proceedings of 2nd Working Conference on Reverse Engineering*.1995.

17. Basili, V. R. et.al.,A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE Transactions on Software Engineering*.1996;22(10):751-761.

18. Koshgoftaar, T.M., Allen, E.B.The Impact of costs of misclassification on software quality modeling.*Proceedings Fourth International Software Metrics Symposium*,1997.

19. Chung, L.,Nixon, B., Yu, E., Mylopoulos, J. Non-Functional Requirements in Software Engineering. Norwell,Massachusetts: Kluwer Academic Publishers, 2000.

20. Cysneiros,L.M., Leite, J.C.S. do Prado.Non-functional requirements from elicitation to modelling languages.*Proceedings of the 24th International Conference on Software Engineering*,2002.

21. Hill, Raquel et.al.,Quantifying Non-Functional Requirements: A Process Oriented Approach"*Proceedings of the 12th IEEE International Requirements Engineering Conference,*2004.

22. Zhong, S., Khoshgoftaar, T. M., Seliya, N.Unsupervised Learning for Expert-Based Software Quality Estimation.*IEEE International Symposium on High Assurance Systems Engineering*,2004.

23. Kassab, M., Ormandjieva, O., Daneva, M.A Traceability Meta-model for Change Management of Non-FunctionalRequirements.*Sixth International Conference on Software Engineering Research, Management and Applications*.2008 :245-252.

24. Xu, Jie., Ho, D., Capretz, L. F. An Empirical Study on the Procedure to Derive Software Quality Estimation models. *International Journal of Computer Science & Information Technology*.2010;2(4):.1-16.

25. Arcelli,F.F., Maggioni,S.Metrics and Antipatterns for Software Quality Evaluation.*IEEE 34th Software Engineering Workshop*,2011.

26. Sun ,H. Knowledge for Software Quality Control and Measurement. *International Conference on Business Computing and Global Informatization*,2011

27. Bajpai,V., Gorthi,R.P.On Non-Functional Requirements: A Survey.*Students' Conference on Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE***.**

28. Trivedi, P., Kumar, R. Software Metrics to Estimate Software Quality using Software Component Reusability. International Journal of Computer Science.2012;9(2);pp.144-149.

29. Mukhopadhyay,T. et.al.,Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation.MIS Quarterly/June 1992: 155-171.

30. Clemons, E.K. et. al., An Integrated Framework for Identifying and Managing Risks Associated with Large Scale Reengineering Efforts. *Proceedings of the 28thHawaii International Conference on System Sciences.LosAlamitos, CA: IEEE Computer Society Press*, 1995.

31. Subramanian, G., S. Breslawski,S.An Empirical Analysis of Software Effort

Estimate Alterations. Journal of Systems and Software.1995; 31 (2) :135-141.

32. Shepperd M., SCHOFIELD, C., KITCHENHAM, B.Effort estimation using analogy.*Proceedings of the 18th International Conference on Software Engineering*, 1996:170-178
33. Clark,B.K. (2000),Quantifying the Effects of Process Improvement on Effort", *IEEE Software.*2000:pp. 65-70.
34. Hill,J., Thomas, L.C., Allen, D.E. (2000)"Experts' Estimates of Task Durations in Software Development Projects. *International Journal of Project Management.*2000;(18):13-21.
35. Jorgensen, M., Sjoberg, D. I. K. Impact of effort estimates on software project work. *Information and Software Technology.*2001; 43(15); 939-948.
36. Jørgensen, M. A Review of Studies on Expert Estimation of Software Development Effort. *The Journal of Systems and Software*. 2004;70:37–60.
37. Kaczmarek, J., Kucharski, M. Size and effort estimation for applications written in Java. *Information and Software Technology.* 2004;46(9):589-601.
38. Jørgensen, M. (2005). Practical Guidelines for Expert Judgement-Based Software Effort Estimation. IEEE Software 2005; 22(3), 57-63.
39. Grimstad,S., Jorgensen,M., Ostvold.,K.M. Software effort estimation terminology the tower of Babel. *Information and Software Technology.*2006; 48:302–310.
40. Menzies ,T., Port,D.,Chen,Z., and Karen,L. Validation Methods for Calibrating Software Effort Models. *IEEE transactions on Software Engineering*. 2006;32(11):1-13.
41. Sandhu,P.S., Prashar,M., Bassi,P., and Bisht,A. A Model for Estimation of Efforts in Development of Software Systems. *International Journal of Computer, Electrical, Automation, Control and Information Engineering.*2009;3,(8):1931-1935.