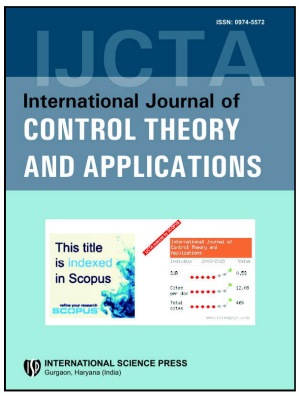

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 19 • 2017

A formal analysis of Statistical Method to improving Software Quality Cost Control based on Weyuker's Properties

Arun Kumar Marandi[1] and Danish Ali Khan[1]

[1] *Department of computer applications Institute of Technology, Jamshedpur-831014, India,*
Emails: arunmarandi.ca@nitjsr.ac.in , dakhan.ca@nitjsr.ac.in

Abstract: Software companies maintain their own existent in business world to facing lots of challenges like defect origins, defect tracking, defect removal, and finding the injected bugs or defects into the software development life cycle. These defects or bugs have breakdowns their economic business growth of software industries. In order to established the weyuker's properties to fulfill the nine theorems of software measures. Weyuker's Properties that permitted to improving the software quality and reducing their skyrocketing of software maintenance cost. In this paper, Cost model and Defect removal matrices to improve the software quality as well as minimizing the estimated cost. Phase by removal process and tracking the defects at each level which are injected during the software development life cycle process. In this methodology, it enables the predictions of software quality and reducing the estimated cost.

Keywords: defect injection, quality cost model, software quality, quality cost reduction, weyuker's properties.

1. INTRODUCTION

Software quality can achieve higher levels of quality by changing their development process or by product assessment where multitudes of different strategies are available [1-2]. Every decision has its own cost implications that must be taking into account. By reconciliation the challenging objectives of improve software quality and cost reduction, a quality of cost methods approach provides as a useful framework for comparing available software development process and estimation alternatives[3- 4-5]. The cost of quality strategies in software development and confirmation process re-inspect through statistical cost of quality model explored analytically using a sample data set of fundamental mathematical formulation models [4]. The paper deals, software quality of conformance plays a central role in developing process and defects removal effectiveness [6-7]. The goal of software industries is to identify produced defects before they are deliverer to the customer [8]. A wide range of practice alternative is available. These practices may differ in the choice of defect detection arrangement within a linear regression model, defect detection methods as well as defect tracking efficiencies or effectiveness. Statistical control process towards the goal of achieving high quality exist, defects detection and tracking method are most efficient and cost effective one can be a difficult task for software development process [9]. Phase wise defect detection process system where the interplay between development process they can become very complex,

software development companies facing the difficult task of defect removal process defect tracking methodology [10-11]. They combination of maximizes software quality of conformance at the lowest cost possible. Paper deals, to address the challenging objective of improving the software quality and cost reduction and, one must first understand the cost of quality of conformance [12]. In addition, metric that seeks to compare different options must reconcile the competing cost and quality of conformance objectives [13]. Statistical control process metric that captures both cost and quality implications of different development and verification options by measuring all costs associated with different levels of software quality of conformance [14]. Defect detection and defect removal metric will incorporate costs pertaining to prevention, curing defects as well as consequences of imperfect quality of conformance including rework, scrap and or field failure costs [15-16]. If the testing time is too short, the cost of the software remains reduced, but the end user may take a higher risk of buying unreliable software. In this paper, the cost model and statistical methods to predict the potential cost savings and defect reduction expected. The quality of software should have as few defects as possible [2-4-9]. It is accepts that defects will be injected and the objective is to deliver software with few defects within the estimated budget [3-17].In this paper, the mathematical formulations of Weyuker properties to satisfied nine axioms and try, to capture the density of composition[18]. The Weyuker properties nine compositions, which results the sum of all projects have same estimation of complexity greater than the sum of the complexities of individual projects that are collected.

The rest of the paper is organized as follows: section II present Weyuker properties and theirs axioms after section two proposed framework in section III. Section IV describes about conceptual economic model. Finally, analytical results and discussion are follows.

2. WEYUKER PROPERTIES

Elaine J. Weyuker proposed a set of properties that permit us, formally compare and evaluating software complexity measures. They analysis showed that none of the evaluated measures fulfills all nine properties. The weyuker properties are follows as.

Property 1: A measure that rates all projects as equally complex is not really a measure.

There exists P, Q such that $|P| \neq |Q|$.

Property 2: Let C be a non-negative numbers. There is only finitely number of projects of complexity C. There should be more than just a few complexity classes. Software complexity does not distinguish between projects that perform little computation and those that do massive amounts if they have the same control structure.

Property 3: There are distinct programs P and Q such that $|P| = |Q|$. That means the measure do not assign to every program a unique complexity.

Property 4: There exists P, Q such that P is equivalent to Q and their complexities could be different. $(\exists P)(\exists Q)(P = Q)\ |P| \neq |Q|$.

Property 5: for every P, Q then $|P| \leq |P\ ;\ Q|$ and $|Q| \leq |P;\ Q|$ hold. These properties of monotonically as programs are collected.

Property 6: (i) There exists P, Q, R such that $|P| = |Q|$ and $|P : Q| \neq |Q : R|$. (ii) There exists P, Q, R such that $|P| = |Q|$ and $|R : P| \neq |R\ ;\ Q|$. In this property, concatenation of programs affects the complexity of the rustling program in uniformity.

Property 7: program complexity should be responsive to the order of the statements, and hence the potential interaction among statements there are P and Q such that Q is formed by permuting the order of the statements of P and $|P| \neq |Q|$.

Property 8: if P is renaming of Q, then |P| = |Q|. The renaming of various elements of the classes does not change the number of classes in the programs.

Property 9: At least in some cases, because of interaction, the complexity of concatenated program is greater than the sum of their complexities.

There exists P, Q such that |P| + |Q| < |P : Q|.

Above nine properties are not all satisfied their different software program complexity. These properties help us to identify the complexity during their software development process. This indicated enable the software reliability and effectiveness of the software product and enhancing the cost of software quality.

3. PROPOSED FRAMEWORK

The Proposed Framework illustrates the method used to eliminate phase-by-phase defects removal. To deploy high quality software, it is essential to develop a defect-free deliverable at each phase. A defect is any blemish, imperfection, or undesired behavior that occurs either in the deliverable or in the product [19]. Anything related to defect removal is a continual process and to removing defects, process and technique are use to improve the software quality. Defect analysis at early stages of software development reduces the time, cost and resources required for rework. Early defect detection prevents defect migration from requirement phase to design and from design phase into implementation phase [2-4]. There are several studies conducted by different researchers for producing reliable software through error removal in code lines and software testing. Few authors have given four way of detect defects a) Checklist Based Detection b) Scenario Based Detection c)Perspective Based Detection d)Traceability Based Detection by [7], some author depend upon Defect Density Model and Design Phase Analysis for defect detection [2-6].Defects impediment is a most important part of software quality. Defects injection and finding bugs in process development are the basic explanation for imperfect and software failure that imposes a direct impact on software quality and cost [4-20]. Therefore, the defects must take care of from the starting point of software development process. The proposed frameworks are follows three basic part. Firstly defects injection, in this part software are injected defects and bugs into the software development process.

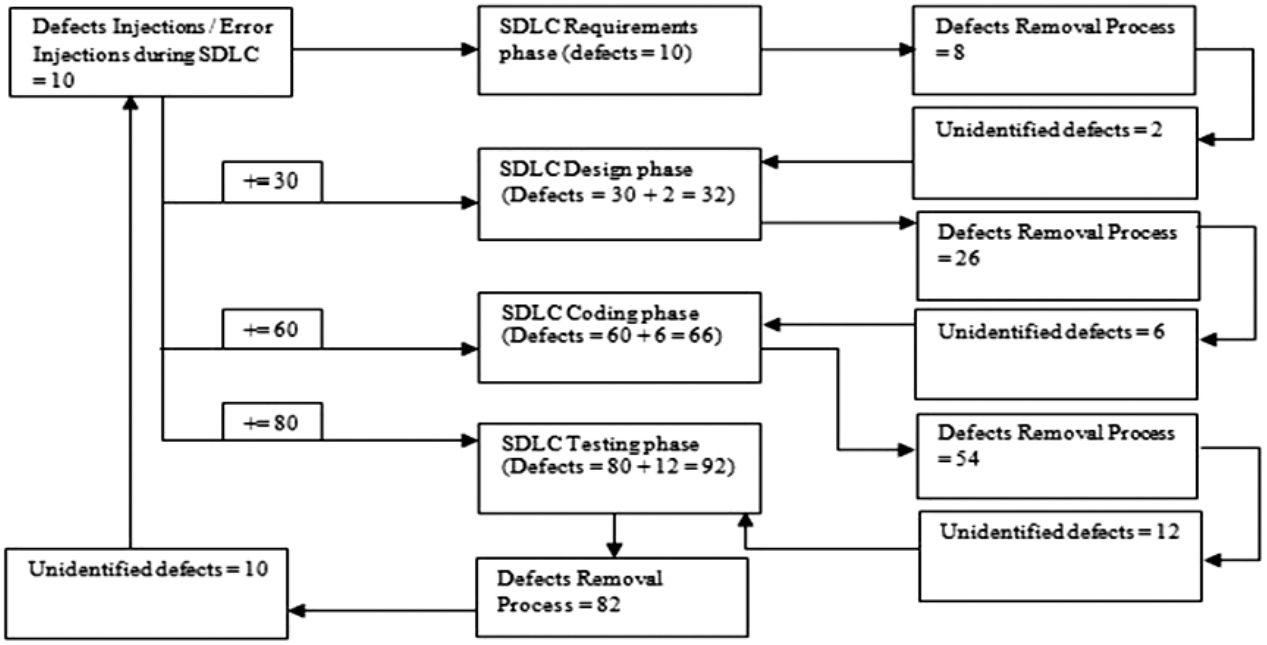

Figure 1: Phase-wise Defect injection Detection and Removal Processes

Secondly, defects removal part in this process software is inspection and removing the detected defects and unidentified defects are re-filtration process [21]. The paper first presents a simplified model of defect removal, and develops the details of the model at the level of individual defect removal at each step in software developing phase.

In most software industries, the project teams are focuses on internal failure and external failure. Failure cost is degrading software quality and improving estimated cost [4-22]. Thus, defects prevention regularly becomes an abandoned component. It is therefore wise to make events that prevent the defects from starting of software development cycle. While the cost of such measures are the nominal, the profit consequent due to overall cost saving are extensively higher compared to cost of prevention and non-conformance cost.

4. CONCEPTUAL ECONOMIC MODEL

In this section paper, describe a cost model introducing the risk level and the time to remove errors. The cost model formulated the mathematical formulas to minimize the expected total software cost and enhancing the software quality [4-13]. The cost of an inadequate infrastructure for software testing can also be express as the benefit of an improved infrastructure for software testing. These values (cost and benefit) are symmetrical. They are properly measured as either the minimum amount of money for all non-conformances would collectively require foregoing the improved infrastructure or as the maximum amount of money for conformance would collectively pay for the improved infrastructure [21-22]. An appropriate measure of the economic impact of an inadequate infrastructure for software testing is the profit differences of developers and users between conditions with the current testing infrastructure and conditions with the counterfactual infrastructure. This can be expressed by summing over all developers and users as follows

$$\Delta \text{ CQM} = \sum\Delta \text{ Conformance Cost} + \sum\Delta \text{ Non-Conformance Cost} \quad (1)$$

Cost of Quality (COQ) is a measure that quantities the cost of control/conformance and the cost of failure of control/non-conformance [4-22]. In this model the cost related to prevention and detection of defects and the costs due to occurrences of software defects.

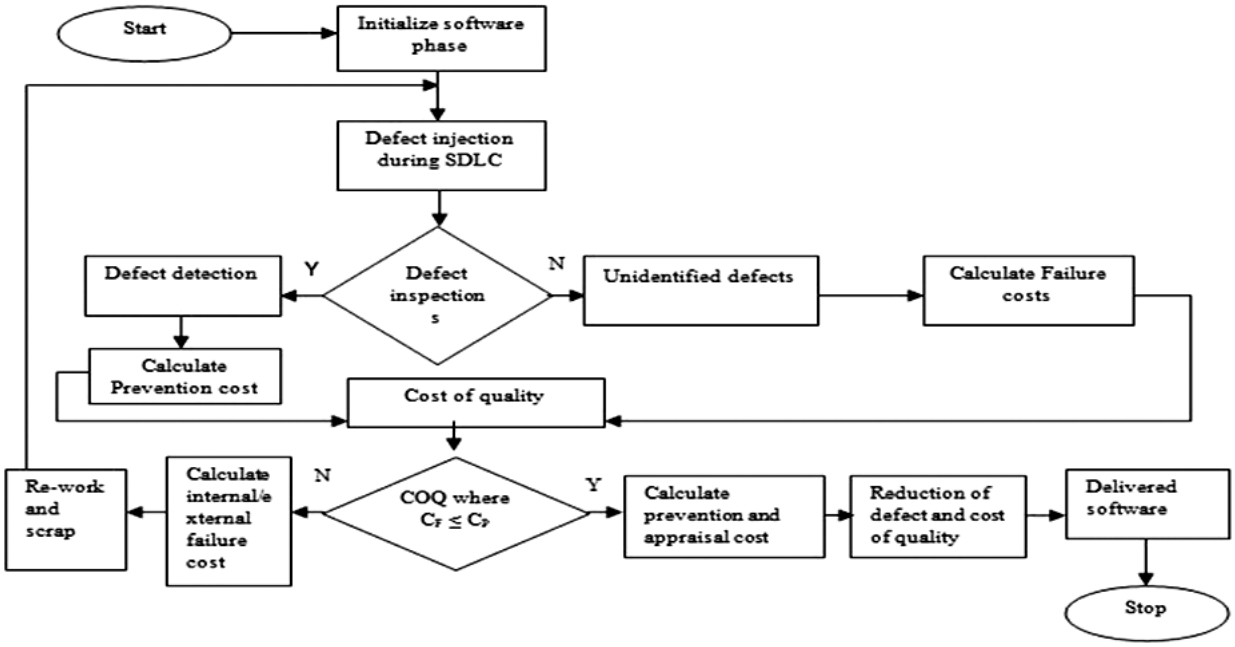

Figure 2: Cost Quality Model with Phase-wise Defect injection.

$$\Delta COQ = \sum\Delta \text{ Cost of Control} + \sum\Delta \text{ Cost of Failure Control} \quad (2)$$

Where,

$$\text{Cost of Control} = \text{Prevention Cost} + \text{Appraisal Cost} \quad (3)$$

And

$$\text{Cost of Failure Control} = \text{internal Failure Cost} + \text{External Failure Cost} \quad (4)$$

From equation (3) and equation (4),

$$\text{Cost Of Quality (COQ)} = \text{Cost Of Control (Prevention Cost} + \text{Appraisal Cost)} \quad (5)$$
$$+ \text{ Cost of Failure Control (Internal failure cost}$$
$$+ \text{ External Failure Cost)}$$

The Cost Quality Model is high risk due to the failure cost. If the non-conformance cost's is low the expected total cost of software development is minimize [23-24]. So the internal failure or external failure cost due to the effort of rework or unidentified defects or bugs into the software. These occur it calculates and formulated the mathematical formulas are as follows:

$$\text{Cost of Failure (COF)} = [\{\text{pre-delivery rework effort } (\sum\Delta \text{ applicable phase}) + \quad (6)$$
$$\text{Post-delivery rework effort } (\sum\Delta \text{ applicable phase})\} \times 100] / (\sum\Delta \text{ Actual Effort})$$

$$COQ = [(\text{cost of Appraisal}) + (\text{cost of Prevention}) + (\text{cost of Failure})] \times \text{Effort} \quad (7)$$

Where,

COA and COP are

1. $COA = [\{(\text{Review effort} + \text{Testing Effort} + \text{UAT Effort})\} \times 100] / (\sum\Delta \text{ Actual Effort})$
2. $COP = [\{(\text{ PQA effort} + \text{DP Effort} + \text{Training Effort} + \text{ KT Effort})\} \times 100] / (\sum\Delta \text{ Actual Effort})$

5. RESULT AND DISCUSSION

In this section it, describes about the model gets the defects injected into the different stage of software development processes. In the development processes that are related to defect injections [2-25]. Defects are injects into the product or in-between deliverables of the products at different phases. All the defects are injected at the beginning of the software development. In the requirement phase, information gathering processes and the development of programming function specifications at that time defects are injects. This inject defects are cumulatively added to the next developing phase. So at the time in testing phase, defects are incur large or in-performance failure rate is high. An empirical study conducted across several project from various service-based and product-based organizations reveals that requirement phase contain 50% to 60% of total defects are injected. 15% to 30% of defects are at design phase. Implementation phase contains 10% to 20% of defects [26]. Remaining are miscellaneous defects that occurs because of bad fixes. Bad fixes are injection of secondary defects due to bad repair of defects [2-25-26]. The common sources for defect injection at implementation phase are improper error handling, improper algorithm, programming language shortcomings and wrong data access and novice developers [3]. The defects are mostly injects into the software development process. In this below figure-2 describe the phase wise defect analysis with respect to cost and another figure shows that percentage of occurs into the phase wise development [14-23]. The phase wise defect injection analysis with cost, the first stage of SDLC cost should be low but the defect injection little bit higher than the cost level.

Table 1
Historical Project Data Set

PROJECTS/ Modules	*projects of Requirements*	*Requirement Review Defects*	*Design Review Defects*	*Code Review Defects*	*DIT Defects*	*SIT Defects*	*Post Release Defects*
P1	21	12	10	18	13	48	1
p3	28	25	11	24	19	70	1
p4	24	8	5	22	19	43	2
p5	19	15	5	17	10	30	1
p6	29	6	7	23	22	58	3
p7	17	6	3	15	8	28	1
p8	27	9	2	26	12	41	1
p9	23	10	5	23	8	52	1
p10	19	7	1	19	14	38	2
p11	13	11	3	15	6	33	0
p12	15	14	5	14	9	30	1
p13	24	5	8	22	20	60	1
p14	18	8	3	18	14	41	1
p15	21	9	5	18	11	35	2
p16	24	12	8	19	14	48	1
p17	16	8	2	15	13	32	0
p18	22	14	5	22	11	55	1
p19	13	6	8	13	10	22	1
p20	29	12	5	25	20	52	2
p21	12	15	8	12	9	18	1
p22	23	26	10	18	15	31	1
p23	19	11	6	13	6	17	1
p24	20	8	2	16	15	32	0
p25	27	7	11	25	12	36	1
p26	27	14	5	25	14	41	1
p27	15	7	4	15	8	27	2
p28	13	9	3	11	7	21	1
p29	23	13	10	20	12	37	2
p30	26	4	2	24	13	35	3

Table 2
Project Data Set

ProjectN _NO	*sumLOC_ TOTAL*	*NUMD EFECTS*	*Defect_ density*	*sumHALSTEAD_ PROG_TIME (days)*	*sum HALSTEAD_ EFFORT*	*f(t)*	*E(T)*
1	2828	23	0.008133	50969.11	917443.8	0.790366	725116.4
2	1221	16	0.013104	22480.91	404656.2	0.774889	313563.8
3	1522	3	0.001971	7318.61	131734.5	0.778986	102619.3
4	1408	19	0.013494	22958.05	413245.4	0.777542	321315.5
5	504	6	0.011905	5723.73	103026.2	0.758197	78114.14
6	404	3	0.007426	2836.53	51057.77	0.753969	38495.96
7	293	3	0.010239	3844.61	69202.94	0.74779	51749.24
8	381	3	0.007874	3845.5	69219.5	0.752844	52111.51
9	242	4	0.016529	3026.06	54468.91	0.744091	40529.84
10	232	3	0.012931	2606.7	46920.19	0.743273	34874.52
11	268	5	0.018657	1919.8	34556.58	0.746067	25781.51
12	218	0	0	228.34	4110.2	0.742065	3050.037
13	132	4	0.030303	2391.13	43040.47	0.732277	31517.54
14	8	0	0	0.93	16.75	0.676238	11.32699
15	34	0	0	48.98	881.88	0.705402	622.0801

P1, p2,……p30 are the projects name and n table 2 project no are shown. These projects are taking from open source data set. These data are comparing or analysis based on weyuker's properties [27]. Property 1 is satisfies, the property is an essential requirement and states that there are projects P and Q such that the estimated fault removal cost of project P is not equal to the estimated fault removal cost of project. Property 2, a finite set of projects and let C is non-negative number. There exists a finite estimation fault removal cost of C. In property 3 the estimated project cost is equal to the other estimated cost of projects. Such that $E_{cost}|P| = E_{cost}|Q|$. Property 4 is satisfies such that their exists |P| and |Q|, $E_{cost}|P|$ not equal to $E_{Cost}|Q|$. in the above weyker's properties5 and 7 is not satisfies. Property 8 or 9 are satisfies the weyuker's axioms.

4. CONCLUSION

In this article, the formulation of statistical method to finding the defect injection into the software development phase and removing those defect using the reliable software metrics [10-24]. In this metrics software cost are minimize and improving the quality of software. Defect preventions and defect avoidance methods are improving the software quality and finding the defects where they located [3-26]. The paper presents application of the defects origin iterative method for finding the approximated solution of the Statistical mathematical formulation for Defect Removal Effectiveness problem [6-9-20]. The software companies spend more amount of expenditure on finding the defects and bad fixing. At these averages below 95% in cumulative Statistical Analysis of Defect Removal Effectiveness is not adequate in software quality methods and needs immediate improvements. Defect detection processes play a important role in between development processes and inspection strategies [26]. Software developing companies are focusing the difficult task of removal process and tracking methodology. In this method software quality or risk, analysis cost is affects to the software cost of quality. Statistical control process metric that captures both cost and quality implications of different development and verification option by measuring all costs associated with different types of software failure cost[2-14-21]. The companies that do not measure pre-defects are studied by the author during on-site benchmarks, they are almost always below 85% in Statistical Analysis of Defect Removal Effectiveness and usually lack adequate software quality methodologies;

inadequate defect prevention and inadequate defect removal effectiveness are strongly correlated with failure to measure phase defect removal efficiency [9-20]. For software quality is not only free but leads to shorter development schedules, lower development costs, and greatly reduced costs for maintenance and total costs for ownership.

REFERENCES

[1] Sakthi Kumaresh and R Baskaran., *"defect analysis and prevention for software process quality improvement",* international Journal of Computer Application (0975 – 8887) Volume 8 – no.7,October 2010.

[2] Kan, Stephen H., "Metrics and models in Software Quality Engineering," 2nd Edition, Addison Wesley Longman, Boston, 2003.

[3] Khoshgoftaar, T. M., Allen, E. B., Jones, W. D., & Hudepohl, J. P ,"*Cost-benefit analysis of software quality models".* Software Quality Journal, 9, 9–30. . (2001).

[4] T. M., Khoshgoftaar and E.B., Allen," *A practical classification rule for software quality models*",IEEE Trannsactions Reliability, 49,2:209-216, 2000.

[5] H. Do, G. Rothermel, *"On the use of mutation faults in empirical assessments of test case prioritization techniques"*, IEEE Transactions on Software Engineering32 (9) (2006).

[6] Suma V and Gopalakrishnan Nair T.R. *"Defect Management Strategies in Software Development, Recent Advances in Technologies",* Maurizio A Strangio (ED.), 2009.

[7] Tom Walton., "*quality planning for software Development*", IEEE (7695-1372), 2001.

[8] T.R. Gopalakrishnan Nair and V. Suma., " *A paradigm for Metric Based Inspection Process for Enhancing Defect Management*", ACM SIGSOFT Software Engineering Notes Vol. 35 Number 3, May 2010.

[9] K.S. Jasmine, R. Vasantha, "*DRE- a quality metric for Component based Software Products*", proceeding of world Academy of Science, Engineering and Technology, Vol 23, ISSN 1307-6884, 2007.

[10] Singh, L.K., Tripathi, A.K., Vinod G., "*Software Reliability Early prediction in Architectural Design Phase*": Overview and Limitations: Journal on Software Engineering and Applications. 4,pp: 181-186,2011.

[11] Jayanthi. R and M Lilly Florence. "*A Study on Software Metrics and phase Based Defect Removal Pattern technique For Project Management*". International Journal of Soft Computing and Engineering. September 2013.

[12] S. Wagner. *"A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques".* In Proc. International Symposium on Software Testing and Analysis (ISSTA '06). ACM Press, 2006.

[13] Gopinath G. and Vijay D., " *Towards Reduction of cost of software quality by implementing Regression Automation*". Journal of Industrial and intelligent information, vol. 2, 2014

[14] Singh, B. and Kannojia, S.P., "*A Review on Software Quality Models*"., International Conference on Communication Systems and Network Technologies (CSNT), 2013

[15] T.G., Grbac and Z.D., Huljenic, " *A quality cost reduction model for large-scale software development"*, software quality Journal, Springer , 2014.

[16] B., Nikolik., "*Software quality Assurance Economics"* information and Software Technology,54 , 2012,pp1229-1238.

[17] S. N., Mohanty," *Software cost Estimation:Present and Future",* John Wiley & son Ltd. 1981.

[18] E. J. Weyuker. Evaluating software complexity measures. IEEE Trans. Software Eng., 14:1357-, 1988.

[19] L., Linda and Brennan, Carol M," *Software Measurment and Estimation: A Practical Approach*", John Wiley& Sons,2006.

[20] Abdul K. Khan. "*Software Excellence Augmentation through Defect Analysis and Avoidance, Science*", Technology and art research journal, Jan-Mar 2013.

[21] Guangtai Liang, Qian Wu, Qianxiang Wang, Hong Mei., *" An effective defect detection and warning prioritization approach for resource leaks*", ieee 30th international conference on computer software and applications (0730-3157), 2012.

[22] Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S.*" A systematic literature review on fault prediction performance in software engineering"*. IEEE Transactions on Software Engineering, 38(6),1276–1304. 2012.

[23] J.D. Musa, A. Iannino, K. Okumoto," *Software Reliability: Measurement,Prediction, Application"*, McGraw-Hill, New York, NY, 1987.

[24] Wellman, and Frank," *Software Costing: An objective Approach to Estimating and Controlling the Cost Of Computer Software*" Prentice Hall, Englewood Cliffs, NJ, ISBN 0-138184364, 1992.

[25] Sandeep Kumar Nayak, Raees Ahmad Khan and Md. Rizwan Beg., *"Requirement Defect identification An Early Stage Perspective",* IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 1, September 2012.

[26] Pankaj Jalote and Naresh Agarwal., *" using defect analysis feedback for improving quality and productivity in iterative software development"* , In proc- ITI 3rd international conference on information and communications technology, pp. 703-713. 2007.

[27] Jorge Cardoso,:Control-flow complexity Measurement of Processes and Weyuker's properties",International Journal of Mathematical, Coputational, Physical,Electrical and Computer Engineering., Vol:1 No8,2007.