International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 6 (2017) pp. 1054-1061

An Approach of Statistical Methods for Improve Software Quality and Cost Minimization

Arun Kumar Marandi[1*] and Danish Ali Khan[2]

[1, 2] Dept. of computer applications, National Institute of Technology, Jamshedpur-831014, India.

Abstract

High quality software products play an important role for economic business growth of any organization. Software companies need to 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 there economic business growth. In this article, the paper discuses Phase by analysis of software project have the highest probability for finding the errors and bugs during their development time and re-inspected the software products. Software engineers have great competitive pressure to improving the software quality and reducing their skyrocketing of software cost. Cost of quality has big challenges to minimizing failure cost and improving the prevention cost, cost of poor quality is affected by internal and external failure cost. In this article, it describes the approach of cost Models to improve the software quality and statistical process control to minimizing the failure cost. Defect removal matrices to improve the software quality as well as minimizing the internal or external failure 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, fault detection, quality cost model, software quality, statistical methods, quality cost reduction.

INTRODUCTION

The current scenario, very highly competitive markets software development must provide high quality software products to produce. 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 also be taken 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]. In software industries, the general term "quality" refers to what quality plane literature divides into the two corresponding category of

software quality models and quality of conformance. However, the quality of design focuses on how the software product design meets consumer requirements satisfaction, quality of conformance is concerned about the quality formed and provide to the end user meets the intended design[4][5]. Quality of design is an integral part of software product quality it only has a minor impact on the substitution between development process and defect tracking strategies [6]. 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].

The rest of the paper is as follows related work and literature survey, follows by the next describes about the proposed framework, then after next section analytical results and discussion are follows.

RELATED WORK AND LITERATURE SURVEY

There are several studies conducted by different researchers for producing high quality software and minimizing the estimated cost. In this dimension, most of the studies and research works are done [9], Quality control and minimizing the estimated cost through the function point and defects injection into the requirement phase. Wolverton et al 1974 module estimated the cost of development of software products, in this module he focus on historical cost data and analysis between new observer cost data [18]. The project estimation model modifies and re-created, cost estimated the models Kemerer et al 1987. The models focused on function point, SLIM, COCOMO, and ESTIMACS. In this models it analysis of large (average of 200 KLOC) software products, Knafl et al 1986 and Kafura et al 1987 both are in-lighted software metrics and function points. Boehm et al 1981 cost model and calculation of effort estimation for software life cycle development process [19]. Mohanty et al 1981 and Conte both are focus on software effort estimation and cost of quality. In this research Mohanty has proposed cost model, estimation of project size, project development time, and the cost of development has been an insightful process [17]. David Binkley et al 1995 focused on cost effectiveness of the regression testing, in this process improving the software quality and cost effectiveness. Cost model, in this methods finding the root cause analysis and improving the software quality. Ganapathy et al 2014 has proposed cost of quality, in this approach software cost failure rate observed and finding the root cause defects analysis [20]. The proposed method provides the approach for minimization of cost and poor quality of software products. There are several studies conducted by different researchers for producing high quality software and reducing the estimated cost, A. Schiffaucrova and V. Thomson et al 2006 propose quality cost models, in this model practical use of cost of quality suggests that even though quality is consider an important issue [21]. Dale and Wan et al 2002, it focuses on quality costing method policies for industrial level. Most of the researcher only focuses on finding the defects on software developing process for these study Fang Chengbin (2008) was introduce a tool called bug tracing system (BTS), for defect tracing, has the advantage of popularity and low cost, and also improves the accurate tracking and identify the defects where it is located on software developing lifecycle (SDLC) phases. Stefan Wagner (2008) finding the defects and summarizes the work on defects classifications approaches that have been proposed by two companies IBM and HP. The IBM approach is called Orthogonal Defect Classification (ODC) and the HP approach is based on three dimensions – Defect Origin, Types and models. Pankaj Jalote et al and Naresh Agarwal et al 2007 stressed on how analysis of defects found in first iteration, letter on how to retrieve the feedback for defect prevention to next iterations, leading to quality and productivity improvement. To Improving, the quality of software can be approaches using the same basic principles support by quality leads to W. Edwards Deming, Philip B, Crosby, and Harold F. Dodge. Show that, it will be possible to predict the potential cost savings and defect reduction expected [22]. The processes which can provide to managers/ practitioners to devote effort in minimization of failure cost and optimize the prevention cost. Liang-Hsuan chen, Ming-Chu Weng et al 2002, proposed a fuzzy method is introduced in cost of quality, Liang and Weng suggested the quality of control to preventing the defect ratio and lower the failure cost [21][22]. In this methodology is applies and developed by Ming-Chu et al 2010, Lin Zhang et al 2010 extended quality function deployment (QFD) approach to assess quality cost. The behavioral method of various cost categories are studies by chopra et al 2011. The quality of software is heavily influence by proper attention to every phase of development. However, high quality software should have as few defects as possible. It is accepts that defects will be injected and the objective is to deliver software with few defects within the estimated budget. Cost models the specific equations of the model are given and are use subsequently in estimating the cost of an imaginary software system [22].

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 [23]. 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][22]. 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. Secondly, defects removal part in this process software is inspection and removing the detected defects and unidentified defects are re-filtration process [24]. 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.

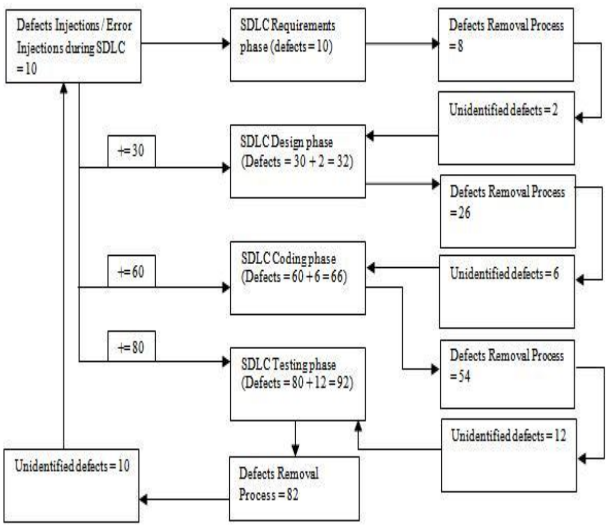

Figure 1: Phase-wise Defect injection Detection and Removal Process

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.

DEFECT INJECTION DURING SDLC

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
p2	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

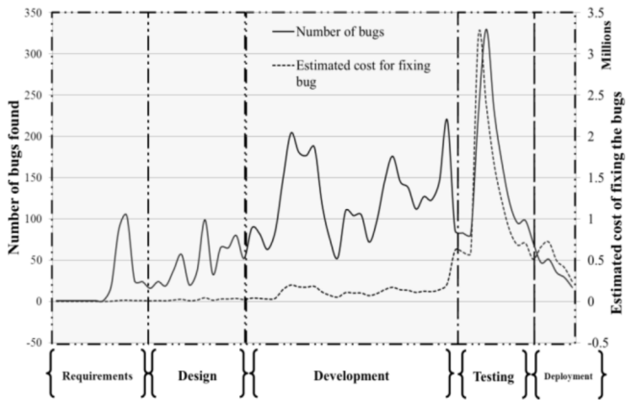

Figure 2: Phase-wise Defect injection Analysis with cost

DEFECT DETECTION AND REMOVAL PROCESS

In this section it, describes about the model gets the Defect Removal Efficiency [4], competitively analysis, and reducing cost; improve on software quality and time effectively in Iterative Phase Defect Removal Model (DRM). Defect detection, in this stage detected the defects and counts the error

percentage [27]. Next stage is Defect Removal Process, in this stage the model verify and analysis the defect or unidentified defect (Bad fixing) go to re-refine the Next Level. If the all defects are reduced it redirect to the next SDLC processes and iteratively [15]. The paper deals a simple model of the defect removal process to develop formulas for enhancing the quality and reducing the work effort. Phase-wise defect injection analysis with cost it clear that low level phase has less amount spent during the software development life cycle. The detection method defects are introduced throughout the software development life cycle and the art of testing is to find as many of them as possible when they are inserted. It is widely recognized that there is a parabolic curve of defect insertion [2][4][10]. The starting point is the requirement specification, which begins by inserting 60% of the defects [16]. The curve terminates with live or production, where the intended result is to find zero defects. The tester should report on completion of the project, the defect detection efficiency [28]. This looks at understanding for each defect, where it was inserted and where it was detected. A perfect test process would look to identify each defect, as it is inserting. This is highly unlikely and the reality is that some defects are finding in later phases of testing. It is therefore important that testing is involved in the project from the outset, not as something that is included if there is the time, the budget and the inclination.
The defect removal efficiency for each software development life cycle, relative to the defects present is the fraction found 'p_k 'times the fraction of good fixes $(1-\varepsilon_k)$ [29]. Hence the defect removal efficiency (DRE_{ck}) relative to the defects currently into the each step k is.

$$DRE_{ck} = (1-\varepsilon_k).p_k$$

Let us assume that ε and p are constants:

$$DRE_c = (1-\varepsilon).p \quad (1)$$

The fraction of defects removed relative to the total lifetime defects (DRE) is:

$$DRE_c = (1-\varepsilon).p \quad (1')$$

The fraction of defects removed in the review/inspection for each step k [29].

$$DRE_1 = MP/TD \quad (2)$$

Where Phase 1effectiveness (E_1) and Phase 2 effectiveness (E_2) [29],

$$E_1 = E_2 \quad (3)$$

(from Remus and Zilles "mathematical relationships of defect removal effectiveness)
From the equation (2) and equation (3), the result can be obtained directly.

$$DRE_1 = \frac{P_1.OD}{\left(\frac{1}{1-\varepsilon_1}\right).OD} \quad (4)$$

$= (1-\varepsilon_k).P_k$: (Let 'OD' is the original defects introduced into the SDLC Phases.)

$$TD = MP + (E_2 \times (TD - MP))$$

$$Q_k = TD\,(1-DRE_{ck})^n$$

or:

$$Q_2 = TD\,(1-DRE_{ck})^2 \quad (5)$$

Where,

$$Q = TD\,(1-E)\,(1-E2)$$
$$= TD\,(1-E)^2$$
$$= (TD-MP)\,E2 \times (TD-MP) \times \frac{1}{MP} \quad (6)$$

from equation (5) and equation (6),

$$Q_2 = \frac{TD}{\mu^2} \quad (7)$$

Note that Q_2 is inversely proportional to μ^2. The quality improves as the value of Q is decreases and the value of μ is increases.

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 [30]. 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

$$\Delta\, CQM = \sum\Delta\ \text{Conformance Cost} + \sum\Delta\ \text{Non-Conformance Cost} \quad (9)$$

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.

$$\Delta COQ = \sum\Delta\ \text{Cost of Control} + \sum\Delta\ \text{Cost of Failure Control} \quad (10)$$

Where,
Cost of Control = Prevention Cost + Appraisal Cost (11)

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 6 (2017) pp. 1054-1061

And
Cost of Failure Control = internal Failure Cost + External Failure Cost (12)

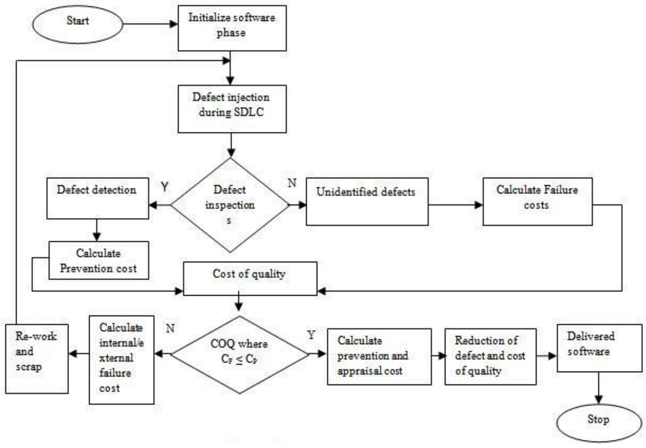

Figure 3: Cost Quality Model with Phase-wise Defect injection

From equation (11) and equation (12),

Cost Of Quality (COQ) = Cost Of Control (Prevention Cost + Appraisal Cost) + Cost of Failure Control (Internal failure cost + External Failure Cost) (13)

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 [31][32]. 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:

Cost of Failure (COF) = [{pre-delivery rework effort (∑Δ applicable phase) + (14)

Post-delivery rework effort (∑Δ applicable phase)} x 100]
/ (∑Δ Actual Effort)

COQ = [(cost of Appraisal) + (cost of Prevention) + (cost of Failure)] x Effort (15)

Where,
COA and COP are
1. COA = [{(Review effort + Testing Effort +UAT Effort)} x 100]
   / (∑Δ Actual Effort)

2. COP = [{(PQA effort +DP Effort + Training Effort+ KT Effort) }x100]
   / (∑Δ Actual Effort)

COST MODELS FORMULATION

In this section it's describe the statistical mathematical formula to minimize the expected total software cost. Let β and η are independent and identically distributed random variables belonging to weibull's distribution with density functions.

$$f(t)=\frac{\beta}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{\beta-1}.e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}} \quad (16)$$

Where:

$f(t)\geq 0, t\geq 0, \gamma$
$\beta > 0$; $\eta > 0$; $-\infty < \gamma < +\infty$

The cost model function m(T) is given by.

$$m(T)=a(1-e^{-f(t)}) \quad (17)$$

The error detection rate function is

$$\lambda(T)=af(t)e^{-f(t)} \quad (18)$$

The expected software system cost, E(T) is defined as.(1) the cost to perform testing ,(2) the cost incurred in removing error during the testing phase; and (3) a risk cost due to software failure.
The cost to perform testing is given by

$$E_1(T)=C_1(T) \quad (19)$$

The expected total time to remove all N(T) errors is

$$E\left[\sum_{i=1}^{N(T)} yi\right]=E\left[N(T)\right]E\left[yi\right]=m(T).\mu_y \quad (20)$$

Where μ_y is

$$\mu_y=\mu_y(0)=\nu\Gamma\left(\frac{1}{\alpha}+1\right)\sum_{j=0}^{\nu}-1^j\binom{\nu-1}{j}/\left[(j+1)^{-1/\alpha}\right]$$

N(T) is number of errors to be detected by time T.
Hence, the expected cost to remove all errors detected by time T can be expressed as

$$E_2(T)=C_2E\left[\sum_{i=1}^{N(T)} yi\right]=C_2m(T).\mu_y \quad (21)$$

The risk cost due to software failure after releasing the software is

$$E_3(T)=C_3\left[1-e^{-f(t)}\right] \quad (22)$$

Where C_3 is the cost due to software failure.

$$\frac{\partial E}{\partial T}=E_1T+E_2T+E_3T$$

$$\therefore E(T)=C_1T+C_2(m)T.\mu_y+C_3(1-e^{-f(t)}) \quad (23)$$

So the expected software system cost E(T) is defined as (1) cost to do testing, E_1T ; (2) cost incurred in removing error during the testing phase, E_2T and (3) risk cost due to potential faults remaining(unidentified defects or bugs) in the uncovered software defects, E_3T .

RESULT AND DISCUSSION

In this section, the model relates the discussed economic factors and other technical factors with the aim to statistical formulation of defects factors at each level [9]. The defect-detection techniques used to manage the quality assurance plan in a development process. Later on, it used the models as a basis for reviewing the observed literature and hence describes only briefly the assumptions and equations [6]. A simplified version of this models is available that can be used to plan the quality assurance of a developing process using historical data [9][12]. It summaries the empirical knowledge available for the quality of defect-detection techniques introducing the approach in general and then describing the relevant studies and results for each of the models factors for different types of techniques and defects in general [4][14][33]. The field of quality assurance and defect-detection techniques in particular has been subject to a number of empirical studies over the last decades. These studies are use to assess specific techniques or to validate certain law and theories about defect-detection. However, the defect removal effectiveness during the software developing phases per unit cost of defects removal must be calculate for efficient quality planning [4][5][34]. Defect removal process at earlier phase it takes less expensive. The cost of defect removal are affected the software quality or software cost. Software cost model incorporating testing coverage and mean vale of failure time software [4][13][21]. In the earlier traditional cost items such as testing cost, error removal cost, risk cost and failure cost due to potential faults in the unidentified defects are include associated with the number demands from customers [35]. The optimal release policies that minimize the expected total cost subject to the reliability requirements [10][19]. The optimal software release time that minimizes the expected total software cost. The testing time required to attain the minimum cost is achieve. Thus, the marginal cost for further testing is an increasing the cost of the software [13]. If the testing time required attaining the minimum cost for further debugging is a decreasing functions for an interval of time [35].

(13) PROJECT DATA SET

Project N_NO	sumLOC_TOTAL	NUMDEFECTS	Defect_density	sumHALSTEAD_PROG_TIME	sumHALSTEAD_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
16	1255	23	0.018327	23407.2	421329.9	0.775401	326699.8
17	649	20	0.030817	12268.3	220828.7	0.763005	168493.4
18	384	0	0	1073.12	19315.65	0.752995	14544.59
19	55	0	0	453.98	8170.75	0.714993	5842.029
20	127	0	0	359.79	6476.65	0.73152	4737.798
21	159	2	0.012579	1058.13	19046.3	0.735919	14016.53
22	3	0	0	2.31	41.41	0.656284	27.17672
23	10	0	0	3.97	71.22	0.680761	48.48379
24	335	0	0	1519.07	27343.18	0.750371	20517.54
25	362	14	0.038674	9935.17	178833	0.751862	134457.7
26	256	8	0.03125	855.11	15391.82	0.74518	11469.68
27	267	7	0.026217	2130.77	38353.73	0.745994	28611.66
28	157	22	0.140127	1869.43	33649.7	0.735671	24755.12
29	795	5	0.006289	9619.81	173156.5	0.766842	132783.6
30	119	0	0	2134.8	38426.29	0.730243	28060.53

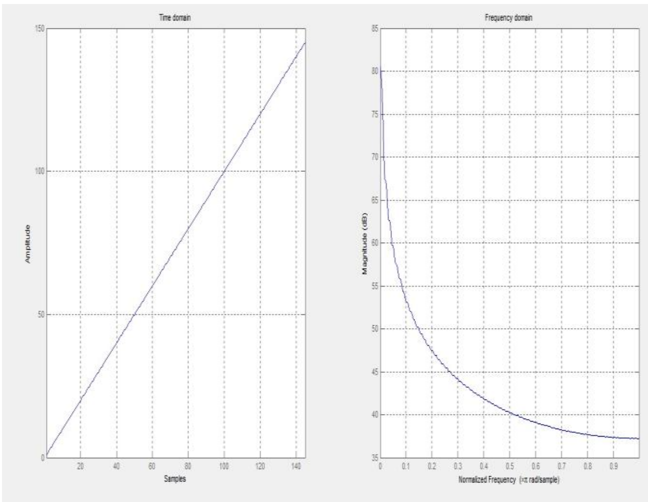

Figure 4: The Expected Total Cost Function vs Time

The testing cost coefficient, C_1 can be estimated to be about 600-700 staff units. It is estimate that there are 370 CPU test-execution units during testing with 1.9 staff unit per CPU unit. The error removal cost coefficient during testing time, C_2 is about 60 staff unit per error. The risk cost coefficient, C_3 is the cost due to potential faults in the unidentified defects. The value of this cost depends upon the nature of the software applications [3][15][21][29]. Let C_1 =600, C_2 =60, C_3 = 10,000, μ_y =0.8, x =4.0, D = 100 and assume that the reliability requirement R_0 is 0.90. The optimal release time (T) minimizes the expected total cost E(T). The testing coverage at this time is C(353.5) = 0.9435 or 94 %. This indicates that it should be need to continue testing the software until the software reliability exceeds 0.90 from the reliability analysis [35], it can be T =409.5 days the software reliability R(409.5) = 0.90006. This implies that it need to test that software for additional 56 days and the testing coverage at 409.5 is C(409.5) = 0.9692 or 97% . This indicates that it can stop testing the software. The statistical methodology improving the software quality and minimizing the software cost [13].

CONCLUSION AND FUTURE SCOPE

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.

REFERENCE

[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] 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.

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

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] Liang Tian and A. Noore., *" Modeling Distributed Software defect removal Effectiveness in the Presence of code Churn"* Mathematical and Computer Modelling 41, ISSN 379-389, 2005.

[27] Berry, A. and Milosevic, Z., "*Real-Time Analytics for Legacy Data Streams in Health: Monitoring* Health Data Quality"., 17th IEEE International on Enterprise Distributed Object Computing Conference (EDOC), 2013.

[28] Singh, A. and Singh, R." *Assuring Software Quality using data mining methodology: A literature study*"., International Conference on Information Systems and Computer Networks (ISCON), 2013

[29] Remus, H., and S. Zilles, "*prediction and Management of program Quality,"* Proceeding of the fourth international conference on software Engineering, Munich, IEEE computer society,1979,pp 341-350.

[30] Karg, L. M., Grottke, M., & Beckhaus, A, "*A systematic literature review of software quality cost research",*Journal of Systems and Software, 84(3), 415–427, 2011.

[31] Bhattacharjee, S. Bhadauria, A. and Kumar, I.K.G., "*Managing Product Delivery Quality in Distributed Software Development*"**.,** 8th International Conference on Global Software Engineering (ICGSE), 2013.

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

[33] L.L. Minku, D. Sudholt, x. Yao., " *Improved Evolutionnary Algorithm Design for the Projec Scheduling Problem Based On Runtime Analysis*"., IEEE Computer Society., 2014

[34] Lessmann, S., Baesens, B., Mues, C., & Pietsch, S.*" Benchmarking classification models for software defect prediction: A proposed framework and novel findings*". IEEE Transactions on Software Engineering, 34(4), 485–496. 2008.

[35] C., Andersson and P., Runeson," *A replicated quantitative analysis of fault distributions in complexsoftware systems",* IEEE Transactions on Software Engineering, 33(5), 273–286, 2007.