An Insight upon the Effect of Quality Assurance on the Cost of Software Development

Deepika Manchanda
Graphic Era University
Dehradun

Akashdeep Singh
Graphic Era University
Dehradun

Neha Garg
Graphic Era University
Dehradun

ABSTRACT

The requirement of a software to be free from any kind of error,defect or fault is still a very big challenge to the IT industry.Software system's size, lifetime and complexity is continuously growing but there is often not much flexibility to deadlines and budget**.**Also,cost is the main factor which should be considered before developing a software project.A possible software failure may lead to millions of breakdown costs, loss of reputation, or even injure people.But,when this is done correctly,it helps in the successful completion of the project. This paper focusses on providing an insight upon finding the right balance between quality and quality assurance costs during different phases of software development life cycle which in turn would increase organizational economic status, as well as conceptual and economic perspective.

General Terms

Quality assurance, phases of software development, cost estimating models.

Keywords

Cost, Quality , SDLC(Software Development Life Cycle).

1. INTRODUCTION

Software nature is considered to be intangible,and thus to ensure improved software quality a strong commitment of time,resources and effort is required by each individual involved in the software development process.It is highly important to remain focussed on the user as well as the system requirements throughout the software development life cycle for quality assurance within given cost and time constraint. A lot of attention should be paid to requirements specification to deliver reliable and quality software.The importance in developing a high quality software system is in satisfying both the user requirements and the developer's budget.Thus a well-defined set of quality requirements are essential for preventing software projects failure.

According to a study conducted by the U.S National Institute Of Standards And Technology, the impact of software failures and software errors cost the American Economy $59.5 billion yearly(National Institute Of Standards And Technology , 2002).
In layman language,Quality in terms of software means having an essential property or behavior.However,a more comprehensive definition of quality is given by ISO that divides quality into six attributes:functionality, reliability, effeciency, usability, maintainabilty and portability.These are discussed in detail in the later section of this paper.

Quality assurance(QA) is ensuring that a software meets all its intended requirements such as functional, time, staff, budget ,etc. However, in order to make this sure various software validation and verification activities are being carried out.This in turn would require investment,contributing to the overall cost of the software.Interchangeably,QA and software testing is used as this paper explores their inclusion in the system development life cycle (SDLC).

Developing a software is indeed a lengthy process involving a software to undergo through various stages of development to ensure a reliable quality software in the end.For this, periodic reviews are being carried out in all stages of the development process for detecting and correcting errors.But in many cases, defects are not detected immediately after when they occur, rather they are noticed much later in the life cycle. So, once a defect is detected one has to go back to the phase where it was introduced and rework those phases-possibly change the design or change the code and so on.

Schenk, Vitalari and Davis et al.[20] indicated that discovery of errors early in the design process could have rippling effects as these errors were expensive and time consuming to correct after software project completion.Ewusi-mensah et al.[6] stated 52.7% of IT projects completed are 189% over budget with an additional cost of $59 billion. Hardgrave et al.[10] emphasized the need to integrate people's effort in the developing software by referring to Pfleeger's comments et al.[19], to understand the role of people in the adoption process, and how it related to drawing upon social science models.

2. PROBLEM AND PURPOSE OF STUDY

Software systems are expensive products because their construction involves a lot of skilled people. Companies which develop software are bound to invest excessive amounts of money to get a high quality software, which overcomes the firms actual quality needs. On the other hand some companies do not take quality assurance seriously enough or do not spent enough money, or do not use the right techniques for the quality assurance of their software production.

Charette et al.[2] reported that organizations and governments spent an estimated $1 trillion on IT hardware, software, and services worldwide. According to the Standish Group study conducted in 1995 [as cited in 11] the U.S. government and businesses spent approximately $81 billion on canceled software projects, and another $59 billion for budget overruns. In another related study, Michaels et al.[18] indicated that it was hard to determine the real cost of failed software projects; however, in the United States alone it was estimated to be upwards of $75 billion a year in rework costs and abandoned systems. The survey claimed that in the United States, only about one sixth of all projects were completed on time and within budget, nearly one third of all projects were canceled outright, and well over half were considered as a big challenge. Of these projects, the average project was 189% over budget, 222% behind schedule, and contained only 61% of the originally specified features.

3. QUALITY

ISO 8402 provides the following definition cited in the ISO quality related documents:"The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs."

ISO-9126 (ISO, 2001)[12] provides a hierarchical framework for quality definition, being organized into quality characteristics and sub-characteristics.There are six quality characteristics,each associated with its exclusive (non-overlapping) sucharacteristics,as summarized in figure 1[12].

However,only considering a certain metric of a quality characteristic is not enough,we have to see the application environment of the software too. For instance 99.9% availability of an office application, thus corresponding to an average downtime of 8,76 hours/year, is fairly appropriate.In respect to availability this office application is of high quality. On the other hand, a power plant control software, having the same availability of 99.9% which stands for an average downtime of 8,76 hours/year too, is definately not acceptable. An unsafe failure might result in a disaster, polluting the environment and possibly injuring people.Additionally it is important to look at financial issues related with achieving desired quality of a software. Quality assurance is costly and the expenses to be made to achieve the right quality are of interest to the project management and the customer. Thus it becomes necessary to estimate and measure software quality costs.

Major Influencing Factors:
The major factors that affect the overall cost of software product are as follows:

1. The costs to set up the application of a specific technique.

2. The variable execution costs. These are mainly the personnel costs and hence dependent on the labour costs.

3. Cost for removing the defect after it's detection. Many factors have an effect on this such as labour costs, code inspection cost etc . Furthermore , it is dependent on the type of document and software development phase in which the defect is detected. A defect which is detected during requirements analysis phase involves only to change the requirements document. However, detecting the same defect during testing phase might require to change several documents, including the code and the design, and to re-inspect and re-test the software, thus adding up to the cost of the software.

4. The labour costs that have an indirect influence on several of the other quality cost factors.

5. Also it is not only important to detect defects but to detect the ones that would be most likely to cause a failure in the field.Measuring the probability of occurrence of failure also affects the cost of the software product.

6. Marketing Factors:Finally, there are also two important factors that are not directly related to the software development process.These are:
a)Time to deliver the product in the market :For some products an early market introduction with mo-re residual defects can be beneficial as the customers might prefer the first product on the market al-though the quality is not optimal.

b) Quality Requirements based on market demand:It might be beneficial to have differing (higher or lower) requirements on specific quality attributes such as some domains consider standards or even legal regulations that require specific safety or availability levels. Even so an economic analysis might suggest that testing is enough, those standards might introduce additional constraints. However, these the above two marketing factors are not taken much into consideration.

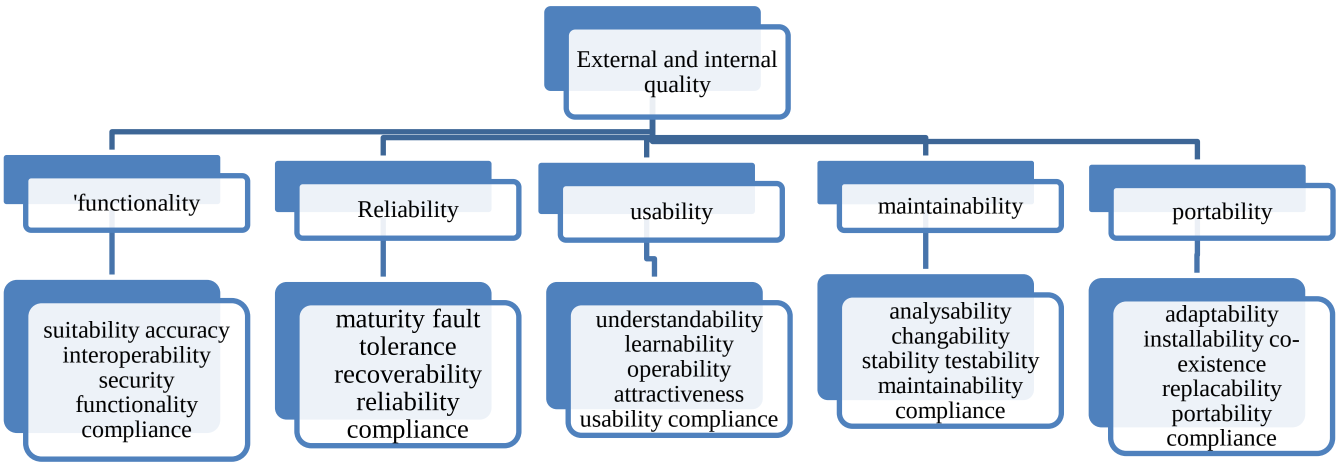
External and internal quality
'functionality
Reliability
usability
maintainability
portability
suitability accuracy interoperability security functionality compliance
maturity fault tolerance recoverability reliability compliance
understandability learnability operability attractiveness usability compliance
analysability changability stability testability maintainability compliance
adaptability installability co-existence replacability portability compliance

Figure 1[12]: ISO/IEC 9126:2001 quality model [ISO9126]

3.1 Quality and Cost Estimation

Quality estimation is a bit more complex and requires more advanced models.Whereas costs can be measured, it is more difficult to measure quality.

Convinced with this measure of quality,now the task for software QA and quality engineering are to ensure software quality through the related validation and verification activities. These activities are carried out by the people and organizations responsible for developing and supporting these software systems in an overall quality engineering process that includes :

- quality planning;
- execution of selected QA or software validation and verification activities;
- measurement and analysis to provide convincing evidence to demonstrate software quality to all parties involved.

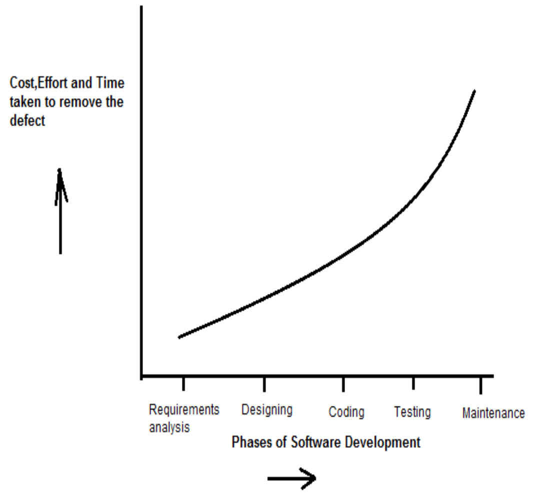

Figure 2:The change in the cost of quality assurance during various phases of software development over time.

Also there are several metrics used to help in qulity assurance of software.One of the first software cost estimation models has been developed by Barry Boehm in 1981.In his book Software Engineering Economics Boehm presents the COnstructive COst Model(COCOMO).According to Boehm, software cost estimation should be done through three basic stages:Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.The second version of COCOMO was developed in the 1990s.COCOMO II model make use of empirically collected data and has a new set of cost drivers.It focusses on process-driven quality estimation.The COnstructive QUALity MOdel COQUALMO is an extension to COCOMO II model.This model clearly relates cost, schedule and quality of software.

One of the estimating methods used to develop budget type estimates is known as parametric estimating.Parametric estimating can be defined as identifying the major task cost drivers,applying the relationships associated or factors to develop project costs.It's Primary Purpose is to provide a defensible budget estimate at early stages with little known design criteria.Another method is detailed estimating made from very defined engineering data.It is done to estimate labour, material and equipment prices. These are often referred to when dealing with cost estimates at certain stages of the project development.The Rough Order of Magnitude (ROM) Estimates and Factoring or Benchmark estimating are used at the very beginning of the project where there is little or no design.The Budget planning estimate is based on some of conceptual design and preliminary engineering data so they are a little more project specific and tighter than the other two types.All three of these types use parametric methods and tools to develop the cost estimatesAgain, Detailed or Definitive estimates are exactly what the name implies. It can usually be developed at about 60% to 70% design, however the more design,the better the accuracy of the estimate and less assumptions have to be made.

Table 1: Various cost estimating methods.

METHODOLOGY	**DESCRIPTION**	**TOOLS USED**	**END USAGE(Typical purpose of estimate)**	**EXPECTED ACCURACY RANGE**
1.Rough Order Of Magnitude (ROM)Estimates	It is a top down estimation approach,made without detailed engineering.	Scaling,factoring,or parametric tools.	Preliminary design-phase budget, costs for program level budget,General economic feasibility,Preliminary economic screening of alternatives.	Acccurate within +50% or -30% of actual costs.
2.Budget(planning) Estimates	It is a top down estimation approach based on preliminary or partial engineering data	Use the same techniques as ROM(however,the use of more detailed data increases the accuracy of the estimate)	Accurate design-phase budgets,Program-level budgets,Detail economic screening of alternatives, Updating existing ROM or Budget estimates with more detailed data	Accurate to within +30%or -15% of actual costs.
3.Detailed (definitive) estimating.	It is a bottom up approach ,based on high level of design.	There are many tools available in the market for detailed estimation,such as MCACES used by CORPES.	Independent Government Estimates (IGEs),Cost validation, negotiations – ensuring that project costs are in line with what should have been charged.	Accurate within +20% or -10% of actual costs.
4.Factor(Benchmark) estimating.	It utilizes costs from similar historical projects.	Cost curves, and factoring historical data.	It is used for the comparision of cost for similar projects.	It include cost curves, ratios and factoring the historical data.

3.2 Planning

Prior planning for any risk involved in quality assurance using the process approach will help in identifying opportunities throughout the system.

To plan for risk Konieczny (2012) et al.[15] and Ben-Jacob (2011) et al.[1] agreed that a method or process should be established.Also “80% of the product quality is determined in the early stage of product development” Kwai-Sang, Lian-Yu, & Li, 2003, p. 733 et al.[16] ,the focus of the planning should be at the early stage of development. The extent of the planning process should primarily be based on the extent of risk the organization is willing to assume or has the financial means to absorb.Contrary to this,poor planning can have a significant financial impact on the profitability of the organization and according to Lopez (1996)[17] it can be difficult to identify not only the severity of the loss but also the source of loss.

3.3 SOFTWARE PROCESS IMPROVEMENT

Indeed,the investment in the software process improvement is one strategy that any organisation might adopt to improve business performance. There have been significant gains in quality as a result of the SPI program. According to a study ,there has been about a 7-10% reduction per year in defects reported by customers .Thus a wide range of software process improvement activites are undertaken nowadays.Organisations often use these strategies to collect data economically.These include:

- forming process groups,
- training,
- performing peer reviews,
- devising forums for exchange of ideas ,and
- inserting new technology.

Eventually many of the organizations had formed a software engineering process group (SEPG) as the organizational focus of the process improvement efforts.Also, training is an important element of the improvement activities in many organizations,the most frequently offered courses being the project management, peer reviews, and instruction in the local development process and methods.Most of these organizations also conducted regular code and design inspections,requirements inspections**.**A few organizations had established forums for the exchange of ideas and for coordinating efforts among groups.Leadership support is required to establish an effective a quality policy, to set goals, objectives, and metrics. Regular management reviews by the leadership team are required to ensure the policy, goals, objectives, and metrics continue to remain relevant and effective to support customer requirements and the organizational strategy.Also, Several organizations had changed their processes not only for process improvement purpose but also to enable the insertion of new technology such as reuse of software.In order to get a more meaningful view of the costs, the following reports showed figures for some organizations in a more normalised way.Figure 4[14] shows the yearly reduction in the number of post release defect reports.The very large differences are because the organizations are very different in size, from division and sector level to small organizations.Two organizations,Q and R, had astonishing results within a very short time frame.The rates for P represent successive releases,with substantial amounts of new and modified code, all of which have gone through their entire life cycle. The last release had no defects reported in the new and modified code.The other two organizations, S and T, also had substantial reductions over a significant period.

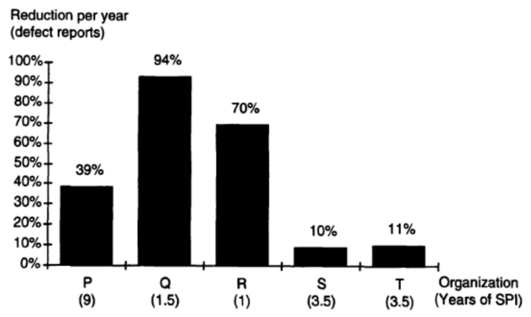

Figure 4[14]: Reduction per year in post release defect reports.

Figure 5[14] is of most interest to many practitioners and managers is the value returned on each dollar invested.This is often referred as the "return on investment" (ROI).The figures that are reported is the ratio of measured benefits to measured costs. In general the savings in business were calculated as follows. Benefits such as savings from productivity gains and savings from fewer defects,savings from earlier detection etc.The costs of SPI generally included the cost of the SEPG, assessments, and training, but did not included the indirect costs such as incidental staff time to put new procedures into place.There are very few investments one can make that returns a business value five or six times of their cost.

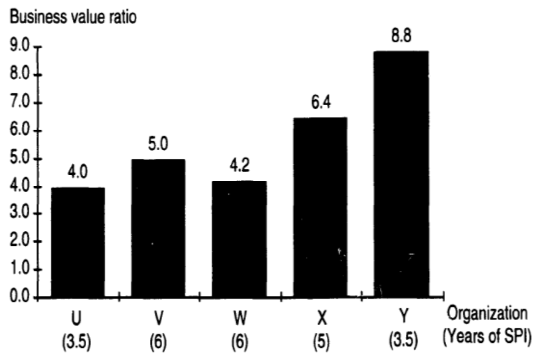

Figure 5[14]: Business Value Ratio Of SPI Efforts.

For the organizations that considered SPI efforts collecting and analyzing data is also important , in addition of making the business case for management for guiding the process improvement effort.

Code inspection:Inspections by themselves are not a silver bullet. They do not by themselves overcome serious flaws in the software engineering process. However it is important to inspect requirements as well as code and design documents.Otherwise, design defects will often not be discovered until system integration test.Most often organisations follow the strategy of "Inspect before unit test" since they expect to find defects,and smaller code segments can be inspected.In addition,it is also possible to skip unit test altogether, and inspections may uncover design defects that could slip through unit testing and therefore be much more expensive to detect and fix.Another example of basing an important management decision on an analysis of data from the current project was a case where unit test was skipped, at a savings of 1.5 person years.In this case, an inspection revealed a defect rate which was about half of what had been experienced in similar projects in the past.According to a recent study a high-level design defect costs twice as much to remove if it is not discovered until coding as it would have cost to remove had it been discovered in low-level design. If it is not discovered until unit test, the cost doubles again. The estimation procedure also makes the reasonable assumption that 50% of all defects that existed when a stage was entered are removed by inspections during that stage.This estimation procedure indicates that at least 1.2 million dollars are saved annually by inspections of requirements and design documents.Savings resulting from code inspections are much easier to quantify, and are estimated at 2.3 million dollars.

3.4 TESTING

Formal verification and validation of the quality requirements introduces both the added costs and potential benefits. During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module in isolation as this is the most efficient way to debug the errors identified at this stage. when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document.

The price of defects : Proper handling of defects is an important part of QA that requires involvement of many people altogether ie.the developers who fix discovered defects are typically not the same as the testers who observed and reported the problems in the first place.Also the effect of every defect is not the same.West et al.[21] classifies defects into levels based on the number of independent factors that are jointly required to cause their occurrence. In this classification, a defect due to a single cause is called a defect of level one. A defect of level two has two independent causes that must occur in a particular combination. For example,the first cause could be the failure of a standard routine , and the second cause could be the failure of the exception handling routine that is invoked to recover from the first failure.Similarly,a defect of level five would require five independent failures to occur.More precisely, the higher the level of a defect, the less likely its occurrence will be [4,7,9,21].Considering this one can assume that the defect level of potentially catastrophic failures is relatively high, requiring multiple things to fail in combination. As a result they tend to have a lower than average probability of occurrence.

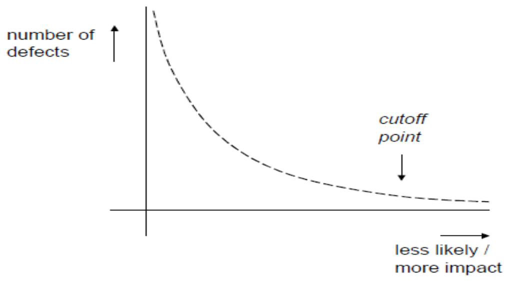

Figure 6[5].Risk and damage: low-impact defects tend to occur more frequently than high-impact defects.

The report also suggested that there is a relation between perceived defect density and the average number of users of a product. Since estimates of residual defect density are typically based on the number of customer complaints post-release, successful products would appear to have a higher residual defect density than unsuccessful products.On the other hand,intoduction of a new functionality might have a negative impact on
residual defect density.

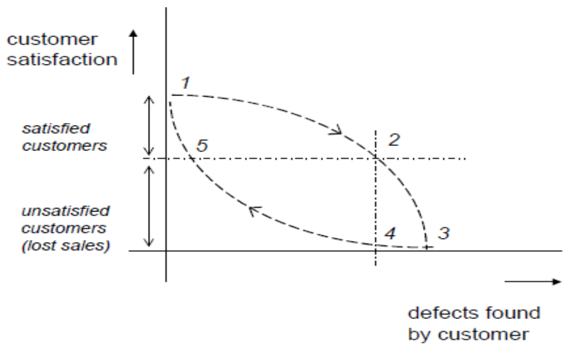

Figure 7[5]. Changes in customer satisfaction as a function of changes in residual defect density.

From the above figure 7 it is clear that starting from a point labeled '1' and moving towards point '2' in the curve, small changes in the residual defect density do not seem to affect customer satisfaction all that much. However ,after passing the point '*2'*, the effect will become very noticeable.Similarly,on starting from point '*3'* on the lower curve and moving towards point '*5'*,small changes in residual defect density do not cause easily observable effects, not even if we pass *4*,restoring the same residual defect density we had before we started losing users at point *2*. Any improvement in defect density will now have to be considerably greater, before it can restore customer confidence that the software product is worth using.

The price of failure **:** Failure costs can be categorized as prevention, appraisal, internal, and external failure costs (Feigenbaum, 2008)et al.[8] The purpose of dividing failure costs was to understand where the majority of the costs are generated and how to apply systems and resources to reduce costs.For instance, prevention costs can be determined by applying a cost to the initial product planning process. Internal failure costs include reworking product failures identified during appraisal.External failure costs are typically identified after the customer has the product or the service has been rendered.Appraisal costs include the cost of resources such as inspection involved with identifying failures to meeting customer product or service expectations.Developing metrics to monitor these costs will provide data for management to understand and allocate resources to reduce costs.The most effective resources applied to reducing internal failures are prevention costs.These costs can include the time and resources dedicated to development of the quality planning process for products. Activities such as developing customer profiles to identify customer contacts for clarification of product expectation and problem resolution, the contract review process, etc. can be undertaken under this.The benefit of prevention costs can be difficult to justify because it can be challenging to put a dollar value on a failure that didn't happen.It has been identified that "95% of quality cost is usually expended on the appraisal and failure" (Jaju & Lakhe, 2009b, p. 546) et al.[13] and only a small percent of failure costs are actually identified (Feigenbaum, 2008).
However,even now it is difficult to actually and precisely determine the cost of quality because of the hidden costs associated with appraisal and failures,such as,decreased team member moral, loss of potential sales, wasted talent(Crandall & Julien, 2010)et al.[3]. Thus what is required is proper identification and correction of all possible problems that can occur and allocating resources in order to reduce the costs associated with appraisal and failures

3.5 MAINTENANCE

Maintenance of a typical software product requires much more than the effort necessary to develop the product itself. Many studies carried out in the past clearly indicate that the relative effort of development of a typical software product to its maintenance effort is roughly as 40:60 ratio. Maintenance involves performing any one or more of the following three kinds of activities:

a. Correcting errors that were not discovered during the product development phase. This is called corrective maintenance.

b. Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. This is called perfective maintenance.

c. Porting the software to work in a new environment. This is called adaptive maintenance. Software maintenance work typically is much more expensive than what it should be and takes more time than required. It has a very poor image in industry since it is a very challenging task to understand someone else's work and then carry out the required modifications and extensions.

4. RESULT AND FUTURE SCOPE

Improving the software quality simultaneously leads to improvement in the productivity.Throughout the paper we have seen that SPI efforts helps in boosting quality as well as productivity taking into consideration developer's indispensible time, efforts and cost.This paper focussed on an oppurtunity to decrease the financial losses by surveying and evaluating the effectiveness of the quality planning process.

5. ACKNOWLEDGMENTS

This study was supported by the Department of computer science of Graphic Era University , Dehradun - 248001,Uttarakhand,India.

6. REFERENCES

[1] Ben-Jacob, R., Kenett, R.S., Lum, S., Urkin, E., & Yu, L.W. (2011) Site seeing. Quality

[2] Charette, R. N. (2005). Why software fails. IEEE Spectrum, *42*(9), 42–49.

[3] Crandall, R. E., & Julien, O. (2010). Measuring the cost of quality. Industrial Management,*52*(4), 14-18.

[4] Eckhardt, D.E., Caglayan, A.K., Kelly, J.P.J., Knight, J.C.,Lee, L.D., McAllister, D.F., and Vouk, M.A. An experimental evaluation of software redundancy as a strategy for improving reliability, IEEE Transactions on Software Engineering, Vol. 17, No. 7, 1991, pp. 692-702.

[5] Economics of Software Verification,Gerard J. Holzmann,Bell Laboratories MH 2C-521,600,Mountain Avenue Murray Hill, NJ 07974.

[6] Ewusi-mensah, K. (1997). Critical issues in abandoned information systems development projects. Communications of the ACM, *4*0(9), 74–80.

[7] Fenton, N., and Neil, M., A critique of software prediction models. IEEE Trans. On Software Engineering, Vol., 25.

[8] Feigenbaum, A. V. (2008). Raising the bar. Quality Progress, *41*(7), 22-27.

[9] Hecht, H., and Wallace, D., Towards more effective testing for high assurance systems. Proc. High Assurance Systems Engineering Conf., Washington, DC, August 1997.

[10] Hardgrave, B. C., Davis, F. D., & Riemenschneider, C. K. (2003). Investigating determinants of software developers' intentions to follow methodologies. Journal of Management Information Systems, *20*(1), 123.

[11] Hutcheson, M. L. (2003). Software testing fundamentals: Methods and metrics. New York: Wiley.

[12] West, C.H., Protocol validation in complex systems. Proc. 8th ACM Symposium on Principles of Distributed Computing,1989, pp. 303-312.

[13] .ISO/IEC: ISO/IEC 9126-1:2001 Software engineering - Product quality - Part 1:Quality model. International Standards Organization,Geneva, Switzerland (2001).

[14] Jaju, S. B., & Lakhe, R. R. (2009b). Tracing quality cost in a luggage manufacturing industry.Proceedings of World Academy of Science: Engineering & Technology, *4*(9), 546-549.

[15] James Herbaleb,Anita Carleton, James Rozum, Jane Siegel ,David Zubrow"Benefits of CMM-Based Software Process Improvement:Initial Results", Software Engineering Institute,Technical Report,CMU/SEI-94-TR-13ESC-TR-94-013,(1994)August,pp.25-26..

[16] Konieczny,T.(2012).A closer look:Understanding risk management for medical device manufacturers. QualityMagazine,*51*(1),48-51.Progress,*44*(9),17-25.

[17] Kwai-Sang,C.,Lian-Yu, Z.,& Li,W.(2003).A hybrid rough-cut process planning for quality.International Journal of Advanced Manufactoring Technology,*22*(9/10),733-743. doi:10.1007/s00170-003-1618-x.

[18] Lopez, D.A. (1996). Quality planning in aerospace manufacturing. Total Quality Management,*7*(3), 249-256.

[19] Michaels, P. (2007). Calculating the cost of failed software projects.

[20] Pfleeger, S .L. (1999).Understanding and improving technology transfer in software engineering.Journal of Systems and Software, *47*(2–3), 111–124.

[21] Schenk, K. D., Vitalari, N. P., & Davis, K. S. (1998). Differences between novice and expert systems analysts: What do we know and what do we do? Journal of Management Information Systems, *15*(1), 9–50.