IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 5, Ver. II (Sep.- Oct. 2017), PP 30-35
www.iosrjournals.org

An Enhancement of Software Quality – A Review

*Rekha Ahirwar
Lecturer Department of Computer Science & Engineering S.R. Govt. Polytechnic College, Sagar Madhya Pradesh (India)
*Corresponding Author: *Rekha Ahirwar*

Abstract: *In modern age, each and every users want to certain level of quality in software for achieving the desired performance. The sense of quality could not be limited but in general sense " it is fitness for use of software is much enough 'there was so many investigations were performed previously by the various research scholars in respect of quality of the software. in this papers we were focus on the quality aspects , steps associated to improving the quality of the software , software quality tools and phase of software developments There by we were achieved an improvement in software quality.*

Key word: *- software quality, quality aspect, quality tools (pareto-charts , fish-born diagram) , development phases*

Date of Submission: 08-09-2017 Date of acceptance: 22-09-2017

I. Introduction

All over the world each & every task – revolving around and depends upon the modern technology, each aspects of life fulfill with this technology. For example – application of computer, mobile phone and so many others. But a question arises here that's, what is the heart elements of such kind of devices, answer is very simple and easy i.e. Software of the devices. Again, we have a question, **is this fulfill the desired purpose efficiently or not**, thus we will directly have concentrated on software quality. The quality of that software really matters. Because it's so widely used and so important, low-quality software just isn't acceptable. In overall quality of any product is defined as in following way .Some important definition of quality of any product are given as, According to " **Juran**"– it is defined as **Quality is fitness for use ,** According to**" Cross-by "–**it is defined as **it is a conformance to requirements** According to**" W. Edwards Deming " –**it is defined as **Good quality means a predictable degree of uniformity and dependability with a quality standard suited to the customer , According to "American Society for Quality (ASQ) "** it is defined as **Quality denotes an excellence in goods and services, especially to the degree they conform to requirements and satisfy customers .** According to **Pressman's** "Software Quality" **Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.** A definition that is obtained from **IEEE "Software Quality" is** the degree to which a system, component, or process meets customer or user needs or expectations. Definitely Quality of the product or software is always on prime consideration, for any user of the product. Software quality is depending upon quality in a manufacturing process. In manufacturing, a primary goal is to minimize defects in products created through a repeatable process. There are so many tools are available for improving the software quality in previous literature, the most effective tools .in present we applied Six Sigma. In this paper we are trying to focus on quality of software these are **functional quality of the software, structural quality of the software, and process quality of the software.**

II. Literature Review

For any research related activity, literature review provided a basic platform upon which all the task depends. Actually literature review is a thought that were previously given by the various Scientists, Research scholars, Professors, other academic staff who is very curious regarding the subject. Following are some important thought that is related to software quality. What is software quality? There are so many different answers may be raised in mind because sense of quality has a complicated & critical concept — it means different things to different people. Garvin [18] has analyzed how software quality is perceived in different ways in different domains, such as philosophy, economics, marketing, and management. . Kitchenham and Pfleeger's article [60] on software quality gives a summary regarding the software quality. They discuss five views of quality in a comprehensive manner as follows

Transcendental View: It envisages quality as something that can be recognized but is difficult to define. The transcendental view is not specific to software quality alone but has been applied in other complex areas of everyday life.

User View: It perceives quality as fitness for purpose. According to this view, while evaluating the quality of a product, one must ask the key question: "Does the product satisfy user needs and expectations?

Manufacturing View: Here quality is understood as conformance to the specification. The quality level of a product is determined by the extent to which the product meets its specifications .

Product View: In this case, quality is viewed as tied to the inherent characteristics of the product. A product's inherent characteristics, that is, internal qualities, determine its external qualities.

Value-Based View: Quality, in this perspective, depends on the amount customer is willing to pay for itThe concept of software quality and the efforts to understand it in terms of measurable quantities date back to the mid-1970s. McCall, Richards, and Walters [19] were the first to study the concept of software quality in terms of quality factors and quality criteria. A quality factor represents a behavioral characteristic of system. Some examples of high-level quality factors are correctness,
reliability,efficiency,testability,maintainability,andreusability.

A quality criterion is an attribute of a quality factor that is related to software development. For example, modularity is an attribute of the architecture of a software system. A highly modular software allows designers to put cohesive components in one module, thereby improving the maintainability of the system. Various software quality models have been proposed to define quality and its related attributes. The most influential ones are the ISO 9126 [20 – 22] and the CMM [14]. The ISO 9126 quality model was developed by an expert group under the aegis of the International Organization for Standardization (ISO). The document ISO 9126 defines six broad, independent categories of quality characteristics: functionality, reliability, usability, efficiency, maintainability, and portability .The CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. In the CMM framework, a development process is evaluated on a scale of 1 – 5, commonly known as level 1 through level 5. For example, level 1 is called the initial level, whereas level 5 — optimized — is the highest level of process maturity. In the field of software testing, there are two well-known process models ,namely, the test process improvement (TPI) model [23] and the test maturity Model(TMM) [24]. These two models allow an organization to assess the current state. Users of software solutions have been suffering from poor solution quality (Whittaker and Voas, 2002). Over the years, quality has emerged to be a key issue in software development (Prahalad and Krishnan ,1999). Software vendors have attempted to tackle this issue by adapting concepts from other engineering disciplines, such as manufacturing (Antony and Fergusson, 2004). There, approaches ranging from Total Quality Management over Six Sigma and Kaizen to Lean Production have led to significant gains in productivity and quality. To attain similar results in software development, many of these concepts have been adapted and tailored to its characteristics (Middleton and Sutton, 2005). In this quest for higher productivity and quality, the economics of software engineering are of particular interest (Boehm, 1981). While some economic aspects, such as software development effort estimation (Jorgensen and Shepperd, 2007) and software process improvement (Hansen et al., 2004), have frequently been discussed, others have received less attention. Indeed, little research has specifically been devoted to those costs which are "incurred in the pursuit of [software] quality or in performing quality-related activities" (Pressman, 2010, p. 407).This is remarkable; because software vendors typically spend 30 to50 percent of their development budget on defect detection and correction (Ebert and Dumke, 2010). In most engineering disciplines, literature studies summarizing the latest research results on quality costs are regularly published (e.g., Williamset al., 1999; Schiffauerova and Thomson, 2006). To the best of our knowledge, this is not the case in software engineering; no review prior tours have in particular been devoted to software quality cost research. However, several studies published in the broader field of software quality and software economics have also covered some quality cost aspects. For instance, the survey of software quality assurance research by Rai et al. (1998) considers software quality costs among other economic aspects, and Jorgensen and Shepperd (2007) systematically review work on software development effort estimation including approaches applicable to software quality cost estimation. This article tries to close this research gap. It is exclusively devoted to software quality cost research. Our objective is to systematically review and structure the existing body of research on software quality costs and to identify areas for future research. We analyze 87 articles published between1980 and 2009 in 60 leading computing journals for answering eight research questions. These research questions are directed at the research domain in general as well as at specifics of the existing research, regarding the software quality cost categories examined, the scope of investigation, and the research approaches employed. As in other engineering disciplines (Dale, 2003), the understanding of software quality has gone through different phases proposing different approaches for coping with the challenge of low quality and high quality-related costs (Whittaker and Voas, 2002; Karg and Beckhaus, 2007). Nevertheless, software quality remains low, while quality-related costs are high. In recent decades, software engineering economics in general (Boehm, 1981; Biffl et al., 2006) and software quality costs in particular (RTI, 2002) have moved into the spotlight. These developments, together with the need to cope with the high quality related costs, motivate the assumption that the research intensity in the software quality cost domain may have increased in recent years.

III. Software Quality Aspects

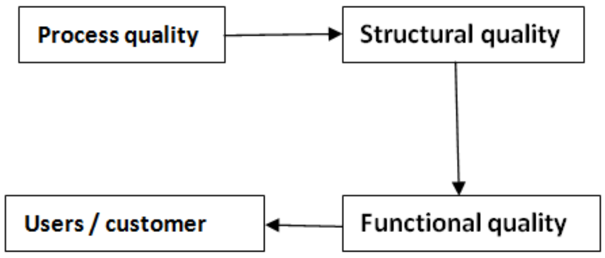
There are three aspects of software quality. These are functional quality, structural quality, and process quality. Structural quality: - Software structural quality refers to how software meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability; and the degree to which the software was produced correctly. Structural quality is evaluated through the analysis of the software inner structure, its source code; in effect, how its architecture adheres to sound principles of software architecture.

The structural quality attributes includes following elements:-
Code testability. Is the code organized in a way that makes testing easy?
Code maintainability. How easy is it to add new code or change existing code without introducing bugs?
Code understandability. Is the code readable? Is it more complex than it needs to be? These have a large impact on how quickly new developers can begin working with an existing code base.
Code efficiency. Especially in resource-constrained situations, writing efficient code can be critically important.
Code security. Does the software allow common attacks such as buffer overruns and SQL injection? Is it insecure in other way?
Functional quality: - Functional quality means that the software correctly performs the tasks it's intended to do for its users.
The functional quality attributes includes following elements:-

- Meeting the specified requirements. Whether they come from the project's sponsors or the software's intended users, meeting requirements is the *sine qua non* of functional quality.
- Creating software that has few defects. Among these are bugs that reduce the software's reliability, compromise its security, or limit its functionality. Achieving zero defects is too much to ask for most projects, but users are rarely happy with software they perceive as buggy.
- Good enough performance. From a user's point of view, there's no such thing as a good, slow application
- Ease of learning and ease of use. To its users, the software's user interface *is* the application, and so these attributes of functional quality are most commonly provided by an effective interface and a well-thought-out user workflow. The aesthetics of the interface—how beautiful it is—can also be important, especially in consumer applications.

Process quality:-The quality of the development process significantly affects the value received by users, development teams, and sponsors, and so all three groups have a stake in improving this aspect of software quality.
The process quality attributes includes following elements:-

- Meeting delivery dates. Was the software delivered on time?
- Meeting budgets. Was the software delivered for the expected amount of money?
- A repeatable development process that reliably delivers quality software. If a process has the first two attributes software delivered on time and on budget but so stresses the development team that its best members quit, it isn't a quality process. True process quality means being consistent from one project to the next.



IV. Software Quality Improvement Steps

Following steps must be taken by software developers, in order to improve software quality.
Step - 1 Define Quality to Match Your Needs: - In this step we will involves Business stake-holders, Entire application development team for the analyses thereby we can achieve business requirements and satisfying user.
Step - 2 Broadcast Simple Quality Metrics: - In this step we will involve only Entire application development team for the analyses thereby we can eliminates or reduce the associated defects.

Step – 3 Fine-Tune Team/Individual Goals to Include Quality:- In this step we will involve only Management. For the analyses thereby we can eliminates or reduce the associated defects, meet business requirements, satisfying user.

Step - 4 Get the Requirements Right :- In this step we will involve managers, business analysts, user experience, designers, architects for the analyses thereby we can achieve business requirements, satisfying user

Step – **5** Tests **Smarter to Test less: -** In this step we will involve Quality assurance team and managers to improve quality and reduce defects.

Step - 6 Design Applications to Lessen Bug Risk: - In this step we will involve Architects and developers. To improve quality and reduce defects .

Step – 7 Optimize the Use of Testing Tools: - In this step we will involve Quality assurance and developers, to improve quality and reduce defects.

V. Software Quality Tools

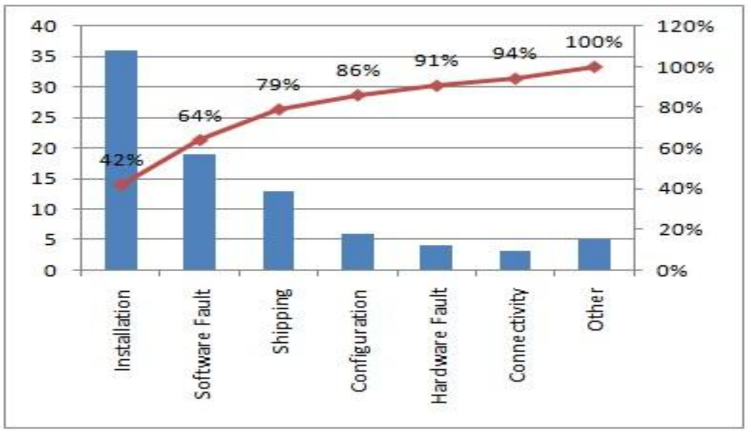
Software quality tools include static and dynamic analysis tools. Static analysis tools input source code, perform syntactical and semantic analysis without executing the code, and present results to users. There is a large variety in the depth, thoroughness, and scope of static analysis tools that can be applied to artifacts including models, in addition to source code. Categories of static analysis tools include the following:

- Tools that facilitate and partially automate reviews and inspections of documents and code. These tools can route work to different participants in order to partially automate and control a review process. They allow users to enter defects found during inspections and reviews for later removal.
- Some tools help organizations perform software safety hazard analysis. These tools provide, e.g., automated support for failure mode and effects analysis (FMEA) and fault tree analysis (FTA).
- Tools that support tracking of software problems provide for entry of anomalies discovered during software testing and subsequent analysis, disposition, and resolution. Some tools include support for workflow and for tracking the status of problem resolution.
- Tools that analyze data captured from software engineering environments and software test environments and produce visual displays of quantified data in the form of graphs, charts, and tables. These tools sometimes include the functionality to perform statistical analysis on data sets (for the purpose of discerning trends and making forecasts). Some of these tools provide defect and removal injection rates; defect densities; yields; distribution of defect injection and removal for each of the life cycle phases.

VI. Software Quality Measurement

Software quality measurements are used to support decision-making. With the increasing sophistication of software, questions of quality go beyond whether or not the software works to how well it achieves measurable quality goals. The techniques include followings Descriptive statistics-based techniques and tests often provide a snapshot of the more troublesome areas of the software product under examination. The resulting charts and graphs are visualization aids, which the decision makers can use to focus resources and conduct process improvements where they appear to be most needed. Software quality measurement includes measuring defect occurrences and applying statistical methods to understand the types of defects that occur most frequently. This information may be used by software process improvement for determining methods to prevent, reduce, or eliminate their recurrence.

Pareto Chart: - A Pareto chart is prepared to show the defect type with the highest frequency of occurrence of defects



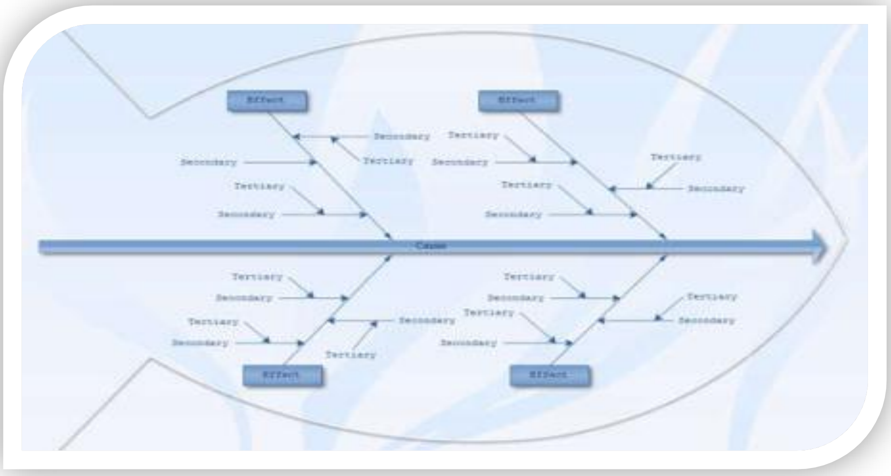
Pareto – charts

The Pareto Chart shows the frequencies of occurrences of the various categories of problems encountered, in order to determine which of the existing problems occur most frequently. The problem categories or causes are shown on the x-axis of the bar graph and the cumulative percentage is shown on the y axis of the graph. From the Pareto chart, it is understood that 80% of the defect are falling under the category Logical, Requirement, Design defect type. These defect types should be given higher priority and must be attended first.

Cause & Effect Diagram for a Software Defect: - Root-cause analysis is the process of finding the activity or process which causes the defects and find out ways of eliminating or reducing the effect of that by providing remedial measures. The root cause analysis of a defect is driven by two key principles:

➢ **Reducing the defects to improve the quality:** The analysis should lead to implementing changes in processes that help prevent defects in the formation stage itself and ensure their early detection in case its re-occurring.

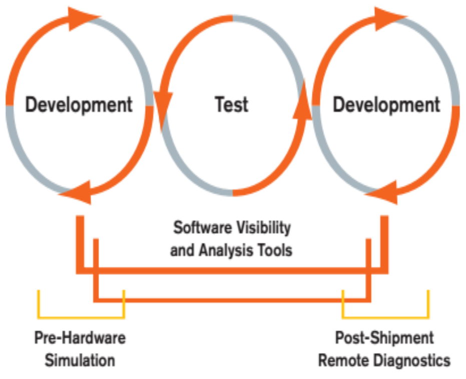
➢ **Utilizing local and third party expertise:** The people who really understand what went wrong should be present to analyze processes prevalent in that organization along with third party experts. A healthy debate ensures all possibilities are reviewed, analyzed and the best possible actions are arrived by consensus [5].With these guidelines, defects are analyzed to determine their origins. A collection of such causes will help in doing the root cause analysis. One of the tools used to facilitate root cause analysis is a simple graphical technique called cause-and-effect diagram/ fishbone diagram which is drawn for sorting and relating factors that contribute to a given situation.



Cause & effect (fish-bone diagram)

VII. Software Analysis Throughout The Development Cycle

When selecting which software analysis tools to use, it is important to consider the entire software development process. From emulation and simulation in the pre-hardware phase to remote diagnostics after the product has shipped, thorough data streams are crucial. Software analysis tools can provide this data at every stage of the cycle.


Development
Test
Development
Software Visibility and Analysis Tools
Pre-Hardware Simulation
Post-Shipment Remote Diagnostics

➢ **Development: -** Software analysis tools augment a debugger when stop mode is not an option. An Instruction Trace tool can provide performance measurements while clarifying RTOS interactions, context switches and performance bottlenecks. The most effective tools not only trace function pathways, but provide visibility inside the functions as well.

➢ **Tuning Phase: -** Just because the software is up and running, that does not mean it is ready to ship. It has to be tuned for optimum performance. Software analysis tools are an invaluable aid during this phase, when the application's performance is being measured and tweaked. There are two software analysis methods that are commonly used:

➢ **Instrumentation: -** By applying tags to specific lines of code, the instrumentation method offers extremely accurate measurements of application performance. However, the method does increase the size of the code. It is important to find tools that permit a tight focus on specific problems

➢ **Sampling Method: -** The sampling method is non-intrusive to code size, but it cannot offer the 100 percent accuracy of instrumentation. Sampling also requires some form of monitor, which drains system resources and impacts its performance, affecting the accuracy of the test.

➢ **Check-In/Hand-Off to QA :-** Software analysis tools streamline the transitions from design to development, from development to test and from test to deployment by identifying resolved and unresolved issues as they occur in the development cycle. Unit tests must be developed to cover all new code. It is also important to check the code that calls new code to ensure it is functioning properly. Software analysis tools can provide objective metrics into code quality, giving project managers; engineers and marketing teams the data they need to feel confident in the quality of the system.

VIII. Conclusion

Software quality is on prime consideration, for any software manufacturing enterprises but it must have an adequate quality in all aspects like functional quality aspect, structural quality aspect, and process quality aspect. if the software is defective then we will take a corrective action thus applying or implementing the software quality control tool throughout the development of software in order to eradicate the root cause of defect in software , thereby we can manufactured a defect free software of desired quality standard

References

[1]. Ajit Ashok Shenvi,2009,"Defect Prevention with Orthogonal Defect Classification", In Proc-ISEC '09, February 23-26, 2009

[2]. Defect Prevention by SEI's CMM Model Version 1.1.,http://www.dfs.mil/technology/pal/cmm/vl/dp

[3]. Linda Westfall, Defect Density http://www.westfallteam.com/Papers/defect_density.pdf

[4]. Megan Graham, 2005, Software Defect Prevention using Orthogonal Defect Prevention. Http://twin spin.csumn.edufiles /ODC_TwinSPIN

[5]. Mukesh soni, 1997, Defect Prevention: Reducing cost and improving quality" published in IEEE Computer, (Volume 30, Issue 8), August 1997

[6]. (Art92) Arthur, L.J.(1992). *Improving Software Quality: an insider's guide to TQM*. John Wiley & Sons.

[7]. (Gal04b) Galin, D.(2004). *Software Quality Assurance*, Addison-Wesley: Harlow England

[8]. (Sla98) Slaughter, S.A., Harter, D.E., Krishnan, M.A.(1998). Evaluating the cost of software quality,*Communications of the ACM* , 41(8): 10–17.

[9]. Redman, T.C. (1992), Data Quality: Management and Technology, New York: Bantam. (1996), Data Quality for the Information Age, Boston: Artech House.

[10]. Naus, J.I. (1975), Data Quality Control and Editing, New York: Marcel Dekker

[11]. Sakthi Kumaresh ,R Baskaran "Defect Analysis and Prevention for Software Process Quality Improvement" International Journal of Computer Applications (0975 –8887) Volume 8–No.7, October 2010

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

Rekha Ahirwar. "An Enhancement of Software Quality – A Review." IOSR Journal of Computer Engineering (IOSR-JCE) , vol. 19, no. 5, 2017, pp. 30–35.