Software Quality Measurements for Effective Software Products

B. Sheeba

***Abstract*— Software is needed by the user to solve a problem to achieve an objective. A quality of the software is depended on good management team, excellent plan, tools and knowledge of the programmers. First the team needs to understand the user requirements properly for to prepare qualitative software. The management should split the works according to the knowledge of the team members. Good environment should be provided for the software programmers and the tools also will lead to good software product.**

***Index Terms*— Software quality, quality factors, quality measurements**

I. Introduction

The goal of software engineering is to progress software quality of the products and to increase productivity. Software is indefinable. It has no volume, no mass, no color, no odor, no emotions and no physical properties. Source code is the power for the software. For development of the software there are five phases it needs to cross over.
The fundamental phases for the software development are analysis, design, implementation, testing and maintenance.

Analysis is the phase where the requirements for the software product are listed and documents will be prepared from this phase. In design phase the programming language, tools and algorithms, data flow diagrams will be decided. In implementation phase where the main coding will take place. In testing phase all the modules will undergo testing. In maintenance further maintenance and further establishment of peripheral devices will take place.

The time elapsed for maintenance is more, than the development time. Example the time taken to develop a software product is one year means the maintenance time will be approximately ten years. Example the time taken to develop a software product is one year means the maintenance time will be approximately ten years.

The most prevailing quality attribute is usefulness; the software product must accomplish user needs. This may seem obvious but sometimes delivered product habitually does not satisfy user needs. This may be poor communication among the user and software engineer. Vigilant planning, development and user involvement will guide to powerful software product.

Manuscript received May 17, 2016

B. Sheeba, Assistant Professor, Department of Computer Science and Applications, Nazareth College of Arts and Science, Chennai, India

II. WHAT IS SOFTWARE QUALITY?

Let's see fact about software. In olden days any work has been done by manpower. The disadvantage of manpower is need lots of materials and lots and lots of time and money. Especially accurate is questionable!

Instead of doing everything using manpower nowadays we have a special efficient tool is software which will do everything in systematic manner as well as less time and less money especially accuracy is predictable and extendable.

Example: where manpower can do a job in one hour but, by using software we can do the same job within a fraction of second with excellent precision.
When software will satisfy the user need? It should compete some of the quality attributes such as accuracy, reliability, portability, efficiency, error free, robustness and correctness. The quality assurance team should check these milestones for a good software product.

The quality assurance team is entirely different from the software developers. They need to undergo for special training to check for the quality of the software. Certified Quality Analyst (CQA) should check each and every module present in the software. For example each page consists of collection of objects (text boxes, labels, buttons, option button) every object should be validated by the Certified Quality Analyst. Because the users are not aware of the object and what should be the input for a particular object. So if the user by mistake enters any off beam value in the object software should show error message to ask them to enter the correct value.

To do these validations very successfully the CQA should go through modules very suspiciously to avoid error. CQA should know the every property of the object in the module, and how the data is flowing from one module to another module. This will ensure the software quality.

III. REVIEW OF LITERATURE:

IEEE states quality assurance is "a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements" [6].

The Institute of Electrical and Electronics Engineers Inc. (IEEE), in its standard 610.12-1990, defines software as 'computer programs, procedures, and possible associated documentation and data pertaining to the operation of a computer system' [4].

From the view of Dean Leffingwell and Don Widring, the purpose of a software quality assurance group is to provide assurance that the procedures, tools, and techniques used during product development and modification are adequate to provide the desired level of confidence in the work products [2].

The process of understanding real world problems of the user's needs the proposing solutions to meet those needs. The goal of problem analysis is to gain a better understanding of the problem being solved before development begins. Choosing the right technique will lead to qualitative software [2].

The knowledge will give the team the information it needs to make better decisions in the definition and implementation of the system. The requirements workshop may be the most powerful technique for eliciting requirements. It gathers all key stakeholders together for a short but intensely focused period.

The fundamental differences between software products (including firmware) and other products are caused by the higher product complexity, by the invisibility of software and by the nature of the product development and production process. These differences create the need for an SQA methodology and tools for SQA that will meet the special and different challenges for the development and operation of Quality assurance for software [5].

The reliability of the software depends on [2]

- Availability
- Mean time between failures (MTBF)
- Mean time to prepare (MTTR)
- Accuracy
- Maximum bugs, or defect rate
- Bugs per type

The performance of the software depends on [2]

- Response time for a transaction: average , maximum
- Throughput: transactions per second
- Capacity: the number of customers or transactions the system can accommodate
- Degradation modes: the acceptable mode of operation when the system has been degraded

The Rational Unified Process [Rational Software Corporation 2002] defines software quality as:

"*The characteristic of having demonstrated the achievement of producing a product that meets or exceeds agreed-on requirements – as measured by agreed-on measures and criteria-and that is produced by an agreed-on process*" [2]

According to this definition, quality is not simple "meeting requirements" or even producing a product or system that meets user needs and expectations. Rather, quality also includes indentifying the measures and criteria to demonstrate the achievement of quality, and the implementation of a process and execution of a project to ensure the product created by the process has achieved the desired result [2].

The additional aspects of quality include process and project measures such as time to market; overall budget adherence; and scope of team, project, and company investment. Quality is multidimensional concept, and these two primary dimensions- the end result of the application itself and the business impact on the producer and consumer – must each get primary consideration [2].

As per Swapna Kishore and Rajesh Naik , a tool published by Total Quality Management (TQM), benchmarking involves persons of an organization studying the processes and data of another organization to validate their approach and to gain insights on how they can improve their own processes [3].

According to QAS [3] ***Quality Assurance Institute (India):*** QAI (India) provides training, consulting and assessment services in the areas of software engineering and software management. It conducts conferences and provides an individual professional certification program called **"Certified Quality Analyst (CQA)".**

IV. FACTORS THAT INFLUENCE QUALITY:

- *Programmer ability* – lack of skill with the application area can result in poor quality. The modules developed by ineffectual programmers will guide to the poor quality and may lag in delivery time.
- *Technology Level* - such as programming language, programming practices (design notations, structured coding techniques), the machine environment (hardware and software) and the software tools.
- *Team Communication* – many programmers have small social needs they will not mingle with other team members they prefer work alone. These programmers rarely discuss their work details in a non systematic manner. This will guide them to the miscommunication with other team members and to make mistakes that may not be detected until sometime later. To avoid this recent innovation in software engineering is structured walkthroughs, design reviews, and code-reading exercises.
- *Reliability level* – the ability of software to perform required function under conditions in the declared time. Extreme reliability will be succeeded only with great care in analysis, design, implementation, system testing and maintenance of the software product.
- *Understanding the problem* – if the software engineer did not understand the nature of the problem it will lead to software failure. Sometimes the user will not be the end user. So the software engineer cannot communicate with end user.
- *Management skills* – the person who supervises the software programming team is called manager. The manager only needs to split the works for the team members. So manager supposed to know the personal skills of his/her team members. According to their skills the work will be allotted to them.

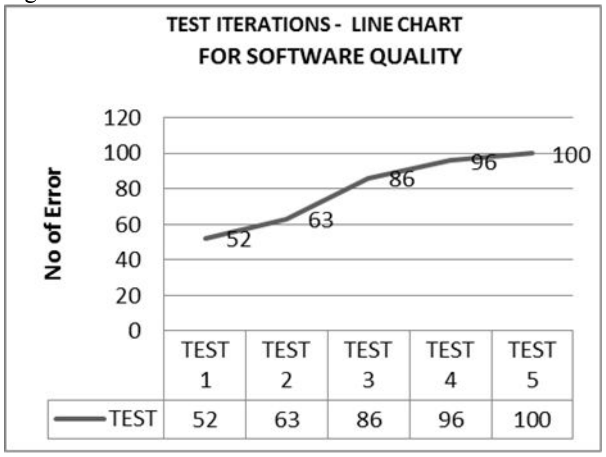
V. GUIDELINES FOR SOFTWARE QUALITY MEASUREMENTS

- Don't ignore the other side
- Understand the domain
- Communicate with the members and users
- Understand why they want it and how badly

- Don't be in a Hurry
- Do not consider anything as obvious
- Don't say "yes" when you may need to say "no"
- Work as a team
- Don't expect analysts to read your mind
- Don't estimate to match the budget
- Don't be over precise
- Don't be greedy
- Do test systematically
- Do everything document
- Correct the error properly Etc

Follow these guidelines and try to remember it in the entire software product, Implement it. First the best analysis team will lead to a good software product. A good management and good skilled programmers also will guide it to final target. The tools and the algorithm selected for the coding also should be a dominant one to channel us through the objective.

Figure 1



	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5
TEST	52	63	86	96	100

Figure 1 shows line chart for the testing iteration for the software quality. For example Project A undergoes for the first adversity process. Test1 epitomize reduced software quality of Project A. After rectifying inaccuracy in the test1 results, after miscalculation and validation the project undergoing for Test2 and vice versa. At last the quality of the Project A keep on increases and finally obtained the effective quality in Test5.

CONCLUSION

Software is fundamental need for life. It has many advantages. To produce a qualitative software product all the phases will be done suspiciously by all the team members. Every object in the form should be validated by the Analyst. Everything supposed to be documented for further maintenance use. The user manual also should be in the non-technical terms. Walk through, milestones and reviews are essential for the qualitative software product.
Systematic testing and recording errors, stable changes will guide us through the objective. Qualitative software for the user should satisfy all the stakeholders.

REFERENCE

[1] Richard Fairly – "Software engineering concepts" – 2010 The McGraw-Hill Edition (269)
[2] Dean Leffingwell , Don Widrig "Managing Software Requirements" A use case approach second edition 2007Pearson Education (81-82,299-300)
[3] Swapna Kishore, Rajesh Naik - "Software requirements and Estimation" 2006 Tata McGraw Hill Edition (269)
[4] The Institute of Electrical and Electro nics Engineers Inc. IEEE Standard Glossary of Software Engineering Terminology.
[5] IEEE Standard 610.12- 1990 (revision and re-designation of IEEE Std. 729-1983), The Institute of Electrical and Electronics Engineers Inc., New York, 1983.
[6] Daniel Galin – "Software Quality Assurance" From theory toimplementation - Pearson Education Limited 2004 (38)
[7] IEEE -83, IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 729-1983, published by the IEEE, NewYork, NY