A study on software quality

B.Reshma[1], B.Sathyavani[2]
Rama Devi Burri, Associate professor in IT, LBRCE, Mylavaram, Vijayawada

***Abstract*-Software is integrated in to our lives more frequently in each and every aspect of our lives. It grows rapidly in its size and functionality, so we need to develop more accurate, high-quality and reliable software to attain the software quality assurance more efficiently and proficiently. To estimate the quality of software artifacts and to stay behind its level far above the ground is much more complicated than to do them for the other developed goods.**

***Index Terms*-Software quality assurance, Software engineering.**

I. INTRODUCTION

The Software engineers have been tasked to develop large and composite programs in a cost efficient manner, so software engineers are facing many problems, without having a better knowledge in the field, such as late release of software, development teams beyond the budget, poor quality, user requirements are not completely supported by the software, difficult maintenance and unreliable software and lack of organized approach. Thus number of large size projects failed.

II. SOFTWARE QUALITY

The degree to which a software product meets established requirements is the software quality. The quality depends upon the degree to which those established requirements accurately represent stakeholders and users.

A software quality is defined based on the study of external and internal features of the software. The external quality is defined based on how software performs in real time circumstances in operational mode and how it is useful for its users. The internal quality focuses on the essential aspects that are reliant on the quality of the code which is developed. The user concentrates more on the software how it works at the external level, but the quality at external level can be maintained only if the coder has written a meaningful and good quality code. The quality system encompasses different activities like Staff development of personnel employed within the quality area. The development of standards procedure and guidelines.

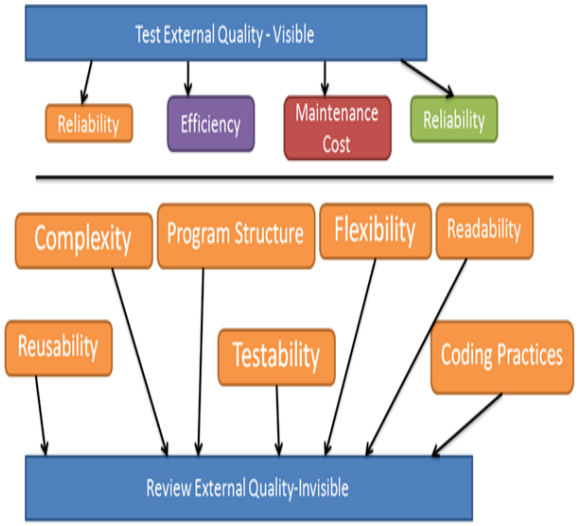


Fig*: Software quality*

There are two significant approaches that are used to establish the quality of the software:

1) Defect Management Approach
2) Quality Attributes approach

Due to lack of understanding of the requirements which are given by the customer the development team may leads to design error. The errors can be caused due to poor functional logic, poor coding and improper requirements gathering.

Error is defined as the measure of the estimated difference between the calculated value of the quantity and its true value .Defect is defined as anything that is not perfect in satisfying the customer requirements. Many times the development team may fail to give the perfect result due to lack of understanding of the problem which is given by the customer, this may leads to design errors. In order to keep track of the defects a defect Management Approach can be applied.

The number of defects is counted and actions are taken as per the severity.

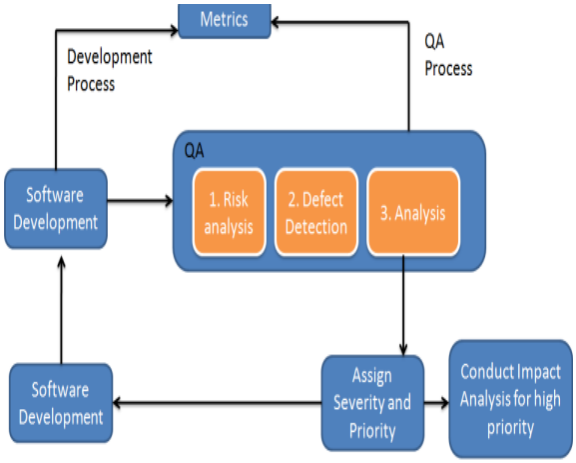


Fig: *Defect Management Approach*

III. THE QUALITY ATTRIBUTES

According to ISO 1926 software quality Attributes are as follows.

Quality attribute approach focuses on six quality attributes:

1) Functionality
2) Reliability
3) Usability
4) Efficiency
5) Maintainability
6) Portability

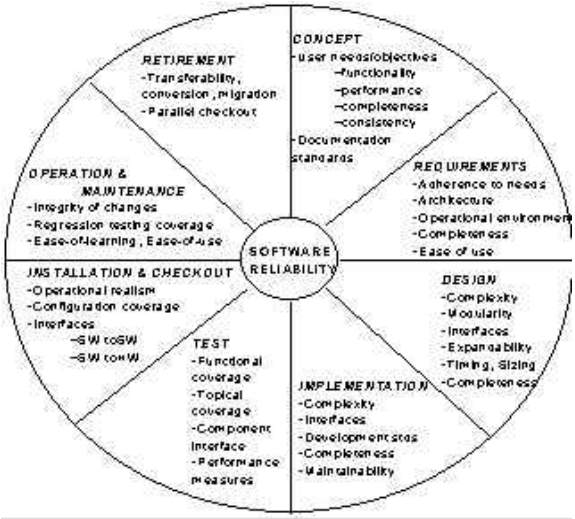
Functionality: It is the ability o the system to do some specific task. It is any necessary purpose of any product or service. But the functionality does not establish the architecture and there is no end for creating the architecture to gratify functionality.

Reliability: Reliability may be defined as the probability of an item to perform a required function under stated conditions for a specified period of time. Software Reliability is defined as the probability of the failure free software operation for a specified period of time in a specified environment. Software reliability also affects the system reliability. Unreliability of any product comes due to the failures or presence of faults in the system. The unreliability of software is primarily due to bugs or design faults in the software.

Usability: This exists with regard to the functionality and refers to the easiness of a given function. The product with more usability can help to differentiate products from those competitors. If two products are significantly equal in effectiveness, the usability will probably be regarded as superior.

Efficiency: This is concerned with the system resources, like the amount of memory space, disk space and network and so on. Efficiency is a measurable concept; Efficiency can often be expressed as a percentage of the result that could ideally be expected.

1) More efficient to use—takes less time to achieve a particular task.
2) Easier to learn—operation can be learn by observing the object.
3) More satisfying to use.



Maintainability: IEEE defines maintenance as 'a process of modifying a software system or component after delivery to correct faults, to improve performance or other attributes or to adapt the product to a changed environment.' Software maintainability is defined as the application is implicit and improved. Software maintenance is very important because 75% budget is dedicated to this. Learning from the past in order to improve the ability to maintain systems, or improve reliability of systems based on maintenance experience.

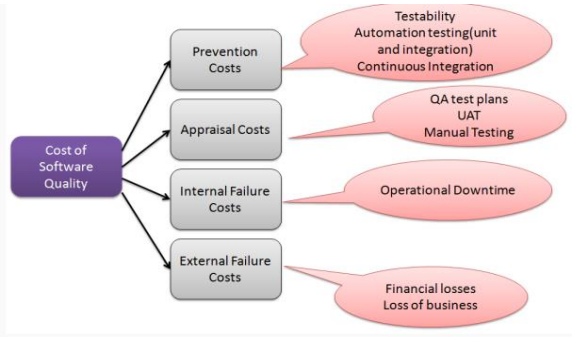
Portability: It is able to move software from one machine platform to another. Portability is a characteristic attributed to a computer program if it can be used in operating systems other than the one in which it was created without requiring major rework. Usb sticks can be used on any computer due to portability and we can store information in removable disks.

Characteristics	Sub characteristics	Definitions
Functionality	Suitability	This is the important Functionality characteristic and refers to the suitability (to specification) of the functions of the software.
	Accurateness	This refers to the correctness of the functions; an ATM may provide a cash providing function but is the amount correct.
	Interoperability	A given software component or system does not typically function in isolation. This sub characteristic concerns the ability of a software component to interact with other components or systems.
	Compliance	Where appropriate industry (or government) laws and guidelines need to be complied with, i.e. SOX. This sub characteristic addresses the accommodating capacity of software.
	Security	This sub characteristic relates to unofficial access to the software functions.
Reliability	Maturity	This sub characteristic concerns frequency of failure of the software.
	Fault tolerance	The ability of software to withstand (and recover) from component, or environmental, failure.
	Recoverability	Ability to bring back a failed system to full operation, including data and network connections.
Usability	Understandability	Determines the ease of which the systems functions can be understood, relates to user models in Human Computer Interaction methods.
	Learn ability	Learning effort for different users, i.e. novice, expert, casual etc.
	Operability	Ability of the software to be easily operated by a given user in a given environment.
Efficiency	Time behavior	Characterizes response times for a given through put, i.e. transaction rate.
	Resource behavior	Characterizes resources used, i.e. memory, cpu, disk and network usage.
Maintainability	Analyzability	Characterizes the ability to identify the root cause of a failure within the software.
	Changeability	Characterizes the amount of effort to change a system.
	Stability	Characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes.
	Testability	Characterizes the effort needed to verify a system change.
Portability	Adaptability	Characterizes the ability of the system to change to new specifications or operating environments.
	Install ability	Characterizes the effort required to install the software.
	Conformance	Similar to agreement for functionality, but this characteristic relates to portability. One example would be Open SQL conformance which relates to portability of database used.
	Replace ability	Characterizes the plug and play feature of software components, that is how easy is it to exchange a given software component within a specified environment.

IV. THE COST OF THE SOFTWARE QUALITY

Cost of quality is calculated by analyzing the conformance costs and non-conformance costs. A conformance cost is related to:

1) *Prevention costs*: amount spent on ensuring that all quality assurance practices are followed correctly. This includes tasks like training the team, code reviews and any other Quality assurance related activity etc.
2) *Appraisal costs*: this is the amount of money spent on planning all the test activities and then carrying them out such as developing test cases and then executing them.



The non-conformance cost is the expense that arises due to:

1) *Internal failures*: It is an expenditure that arises when test cases are executed for the first time at interior level and some of them fail. The operating cost arise when the programmer has to correct all the defects uncovered from his piece of code at the time of unit or component testing.
2) *External failures*: It is the expense that occurs when the defect is found by the customer instead of the tester. These expenses are much more than what arise at internal level, particularly if the customer gets unconvinced or escalates the software failure.

V.COST OF SOFTWARE FAILURE

It displays lack of competence to keep up: this usually happens when the software starts aging. As it grows old the size increases because the easiest way of accumulating a feature is by adding new code without moving any part of code written earlier. Over a period of time it becomes bulky and it becomes difficult to identify the sections of code that need to be changed.

1) Performance drop is observed: Every application normally slows down with age and tends to reside in more and more computer memory therefore it is better to switch to other software.
2) It doesn't seem to be reliable: It is a known fact that every time when changes are made to the code of the software to fix an error, more defects are introduced in the system. Amazingly, this is one of the major reasons for increased failure rates and in order to save condition it is always better to through the project or give up bug fixing.

VI. SOFTWARE QUALITY IMPROVEMENT

The factors influence the quality of the software is:

1) Software architecture
2) Software reliability models
3) Software quality metrics
4) Root cause analysis

REFERENCES

[1] IEEE Standard for Software Quality Assurance Processes ISO/IEC/IEEE 24765:2010 [B42]).
[2] H. Pham, Software Reliability, Springer, Singapore, 2000.
[3] David Hooker, Seven Principle of Software Development.
[4] M. Jorgensen, Software quality measurement