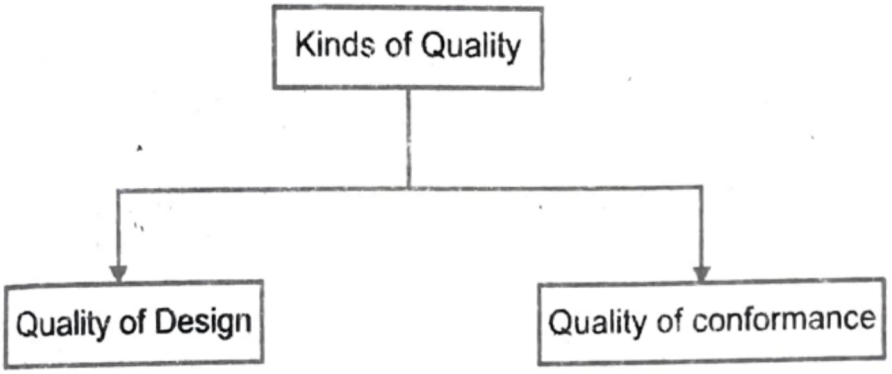
SOFTWARE QUALITY

1. *What is Quality? What are the different types of quality?*

Software quality can be defined as any characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability. All these characteristics are measurable.

There are Two Kinds of Quality :



(Figure)

Quality of Design : Quality of Design refers to the all characteristics which designers specify for any particular item. Here all the grade of materials, tolerances, and performance specifications are included which can contribute to the quality of design.

Quality of Conformance : The meaning of conformance is to tell how the quality of software is good and well maintained. Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.

Quality Software : Quality software can be defined as software which is bug or defect free (reasonably), the software which is delivered in time and within the specified budget, which can meets the requirements of the customers, and is maintainable. Here, software quality includes both functional quality as well as structural quality.

Quality Control : This is main aspect in Software Quality Engineering to control the Quality. This process involves a

series of inspections, reviews, and tests used throughout the software process. Entire this process ensures that each product meets the requirements of the customers. Quality control also includes a feedback to the process that created the work product.

Quality Assurance : It is important to give assurance about quality of the software. It can be the preventive measure which includes set of activities to get more the customer confidence about successful completion the project. In this case, more focus is on how the engineering and management activities can be performed. In fact everyone is interested in the quality of the final product, it should be assured that we are building the right product This is possible only when we can perform inspection & review of all the intermediate products, if there is any bug found, and then it could be removed as soon as possible Hence quality of the software can be enhanced.

Importance of Quality

The expectation of the quality is the concern about o the production of goods and services. However, there are also some special demands like intangibility and complexity.

To Increase Quality of Software Critically : The customer or end user is concerned about the genera quality of software, especially its reliability and maintainability. Software criticality means when the organizations become more dependent on their computer systems and software is used more and more in safety-critical areas. While checking quality of software, i should check in every aspect critically (by considerin worst case).

To Check Intangibility of Software : It is ver challenging to know that a particular task in a project ha been completed satisfactorily. The results of these task can be made tangible by demanding that the developer produce 'deliverables' that can be examined for quality.

To Accumulate Errors During Softwar Development : When there is development of compute system, there are several steps where the output from on level is given as input to the next, then what we have t check whether errors or deliverables? In this case, ther are lots of chances to accumulate the errors in the late

stages leading to accumulated determinable effects. In general, finding the error in the later stage of the project is more expensive to fix. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly challenging to control.

2. *What is Software Quality? What are the different attributes of Software Quality?*

The quality of software can be defined as the ability of the software to function as per user requirement. When it comes to software products it must satisfy all the functionalities written down in the SRS document. Software quality product can be defined in term of its overall quality and fitness of the product. In other words, a quality product does everything what the users want it to do; it meets with every expectation of customer. For every software development, the use is generally explained in terms of satisfaction of the requirements which is in the form SRS document. The fitness of purpose is a satisfactory interpretation of quality for many devices such as any electronic gadget as similar with the software products; "fitness of purpose" is not a wholly satisfactory definition of quality.

Example : Consider there is fully developed software and we have to check its overall functionality. In short, we have check whether it can perform all tasks which are specified in the SRS document. But, in this stage there is no any user interface. Even though it may be functionally right, we cannot consider it to be a quality product, because it can't provide GUI to operate it to the customer.

Following are the some quality attributes :

(1) **Portability :** A software device can be defined as portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc. Such software is not machine or operating system dependent.

(2) **Usability :** A software product has better usability if various categories of users can easily invoke the functions of the product. Such software is more users friendly so that user can operate easily.

(3) **Reusability** : A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

(4) **Correctness** : A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

(5) **Maintainability** : A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

3. *Explain Cost of Quality.*

As there are various stages in the software development, every stage is associated with cost. Similarly there is cost of every activity in every project. This cost should have some business value and software testing and quality management is no exception. The main motto is to optimize cost of quality; testers (who perform testing) should optimize all testing functions appropriately. There should not be unnecessary testing otherwise it causes unnecessary delays and ends up with more costs. Also, there should not be incomplete or too less testing otherwise, there may be a chance of defective products to be handed to the end users. Therefore, software testing should be done appropriately.

Cost of Quality : To calculate the cost of quality, there is an effective measure of quantifying and calculating the business value of testing. There are four categories to measure cost of quality: Prevention costs, Detection costs, internal failure costs, and External failure costs.

These are explained as follows below.

(1) ***Prevention costs*** include cost of training developers on writing secure and easily maintainable code

(2) ***Detection costs*** include the cost of creating test cases, setting up testing environments, revisiting testing requirements.

(3) ***Internal failure costs*** include costs incurred in fixing defects just before delivery.

(4) ***External failure costs*** include product support costs incurred by delivering poor quality software.

Major parts of total cost are detecting defects and internal failure cost. But, these costs less than external failure costs.

How to calculate Cost (Business value) of software testing?

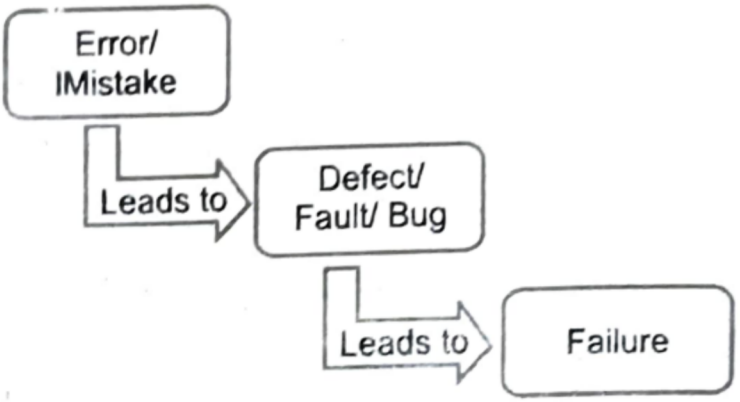
Test manager has responsibility to identify the business value and provide communications between teams and senior management. It concerns the cost of quality.

To calculate business value, these are some of the measurable ways as follows :

(1) To detect defects
(2) To document defects
(3) To prepare status reports
(4) To test metrics on project progress
(5) To implement and development products
(6) To increase confidence in product quality
(7) To check legal liabilities
(8) To ensure predictable product

4. *Explain Errors, Defects, Faults, and Failures in Software Engineering.*

The figure below shows the interrelation between Error, Defect, and Failure.



(Figure)

Errors

Basically we can define Error as a human mistake. An Error appears not only due to the logical mistake in the code made by the developer. Anyone in the team can make mistakes during the different phases of software development. For instance,

(1) BA (business analyst) may misinterpret or misunderstand requirements.

(2) The customer may provide insufficient or incorrec information.

(3) The architect may cause a flaw in software design.

(4) People on the team can also make mistakes due t unclear or insufficient requirements, time pressur lethargy, or other reasons.

Let us observe the basic types of errors in softwar testing :

Types of Error :

(1) **User Interface Error :** These are the errors tha generally appear during user interaction with th system. Such as missing or incorrect functionality the system, no backup function or reverse functio available, etc.

(2) **Error Handling Error** : Any error that occurs whil the user is interacting with the software need precise and meaningful handling. If not, it confuse Therefore, such errors are known as error handlin errors.

(3) **Syntactic Error :** Misspelled words grammatically incorrect sentences are Syntacti errors and are very evident when testing the softwar GUI.

(4) **Calculation Errors :** These errors occur due to ba logic, incorrect formulas, mismatched data type, etc

(5) **Flow control Error :** Errors concerning passing th control of the program in an incorrect direction whe the software program behaves unexpectedly are flo control errors. Such as the presence of an infini loop, reporting syntax error during run-time, etc.

(6) **Testing Errors :** It implies the errors that occurr when implementing and executing the test proces For example, bug scanning failure, inefficiency reporting an error or defect.

(7) **Hardware Errors :** Such errors are related to t hardware device. Such as no availability and compatibility with the device.

Defect

We can define Defect as a variance between expect and actual results. An Error that the tester finds is kno as Defect. A Defect in a software product reflects inability or inefficiency to comply with the specif

requirements and criteria and, subsequently, prevent the software application from performing the desired and expected work. The defect is also known as Fault.

Types of Defects : Defects can be classified on severity basis. Severity defines the degree of impact. Therefore, the severity of the defect reflects the degree or intensity of a particular defect to adversely impact a software product or its operation. Based on the severity metric, a defect falls under the following categories:

(1) **Critical** : Defects that are "critical" require immediate attention and treatment. A critical defect directly affects the essential functionalities which can otherwise affect a software product or its large-scale functionality. For instance, failure of a feature/functionality or collapse of the entire system, etc.

(2) **Major** : Defects, which are responsible for affecting the main functions of software products, are Major Defects. Although, these defects do not result in the complete failure of a system but may bring several primary functions of the software to rest.

(3) **Minor** : These defects produce less impact and have no significant influence on a software product. The results of these defects are visible in the operation of the product. However, it does not prevent users from executing the task. The task can be carried out using some other alternative.

(4) **Trivial** : These types of defects have no impact on the operation of a product. Hence, sometimes, we ignore and omit them. For example, spelling or grammatical errors.

In other way, software defects can be classified on probability basis. In this method we have to see a defect in a software application is the probability that it will occur, and chances that the user will find it. Depending on the likelihood or the possibility of a defect in software product in terms of percentage is classified in the following ways :

(1) **High** : Almost all users of the application can track the presence of defects. This indicates a high probability.

(2) **Medium** : Half of the users can trace the presence of defects in a software product.

(3) **Low** : In general, no user detects it, or only a users will be able to detect it.

In the third way, defects can be classified accordin the priority. Defects also have a business perspec comparison. The rectification of some defects must hap first. Likewise, some can solve at a later stage. Just lik business where everything happens according to current need and demand of the market. Just like probability base, priority classification also occurs in following ways :

(1) **High :** The high priority defines the most crit need of a company to correct a defect. This sho happen as soon as possible, in the same compilatio

(2) **Medium :** Medium priority defects are next to h priority. And any next version or release of a prod includes addressing them.

(3) **Low :** This type of defect does not need to corrected individually. Consideration of repai these types of defects, along with any other defect voluntary.

Failure

Failure is a Consequence of a Defect : It is observable incorrect behavior of the system. Failure occ when the software fails to perform in the real environm

In other words, after the creation & executior software code, if the system does not perform as expec due to the occurrence of any defect; then it is termed Failure. Not all Defects result in Failures; some rem inactive in the code, and we may never notice th Failures also occur due to the following reasons :

(1) Any physical damage or overheating in the hardw can cause the whole system to fail.

(2) If the software is not compatible with the hardw then also the system performs unexpectedly.

(3) Failures also happen by environmental condit like a radiation burst, a strong magnetic f electronic fields, or pollution could cause fault hardware or software.

There are some other factors which can be include the failure as follows :

(1) It may happen due to a human error in interac with the software, for example entering an incor input value which may give incorrect output.

(2) There may be error when someone tries to produce system failure or cause malicious damage.

(3) It may happen because of the mishandling of test data, test environment, etc.

(4) Sometimes, tests that result in undetected defects can also cause failure.

5. *What are the causes of Errors in Software?*

There are many factors which can cause the errors in software as follows :

(1) **Time Pressure** : As there is deadline for every software, the entire team is working under time pressure. The software developers do not have enough time to test their code before delivering it to the testing team. This may become main reason to introduce the errors.

(2) **Human Mistake** : By nature, human beings are prone to make mistakes. Basically, we have not yet discovered any other non-human agent that can develop software better than humans. Therefore, we continue to rely on human intelligence to develop software. Thereby, increasing the possibility of errors in it.

(3) **Software Developers Skill** : There may be lots of users included in the software development which may not be skillful and they are not having any previous experience of development. Allocation of correct work to the correct resource is fundamental for the success of any project. Team members should be assigned a task according to their skills and abilities. An inexperienced project participant may make mistakes if they don't have proper knowledge of the work. For example, a resource having a good understanding of the database but having limited knowledge of HTML/CSS is not suitable for designing a website.

(4) **Miscommunication** : Sometimes, there is miscommunication between the different participants involved in the software development. Improper coordination & poor communication between various departments in a project can result in disrupted

progress. There may be some conflictions betw
different peoples, which may lead misinterpretat
or misunderstanding the words or actions of anothe

(5) **The Complexity of the Code, Desig Architecture, or the Technology to be Used :** the complexity of the program, concerning co design or technology increases, the software becon more critical and more bugs appear. It is because brains can only deal with a reasonable amount complexity or change. Our minds may not be able process complex information like the form of desi architecture or technology, etc. Therefore, resulting low quality and erroneous coding.

(6) **Advanced Technologies :** In tod 's era, n technologies are arising day by day. Sometimes, t developers and testers need to develop an applicat using a technique that is unknown to them. Lack proper knowledge and understanding can lead errors. At that time, they require a certain level o & D, brainstorming, and training to reach a relia solution.

(7) **Environmental Conditions :** Natural disaste such as outbreaks of fires, floods, lightni earthquakes, etc. can affect the computers. example, a system may not work correctly if t software inside is affected by radiati electromagnetic, or pollution.

6. *What is defect rate? How to calculate it?*

We can define Defect rate as a system of counte the number of defects over the test cases performed. If reduce the number of test cases, then defect rate may decreased. It depends on the number of test ca performed. It might be the number of defects per thous lines of code, or perhaps it's the number of failures execution (that is, the amount of failures per request or test executed).

Decision to Observe Defects : It is important to find place where defects can be raised. It is the impor decision to choose the correct place to observe the defec there is any experiment, the try to track the defect f

initial stage. Through these phases and more in between, a defect takes on different as-states. It helps to know these because this becomes the first step to catching and weeding them from the product.

Decision about Coding and Reviews : Throughout the development, your team is going to add new code to your main software; some irrelevant code might also join your source. Consequently, senior developers/team leads take the mentorship role to read and highlight any errors before committing new code. The better talent you invest in, the fewer chances of this defect threat to take effect.

Decision about Integrating/Testing : At the time of execution testing, you can find more defects. There are more chances when you improve testing methods, you can switch to better tools, or run different tests than your previous efforts. The defects you can expect to encounter at this stage depend on the type of tests you execute. Integration tests will raise flags on incompatibility issues across your modules. Functional tests will show defects as improper execution of the application's logic.

Defects in the Production : The final defect frontier is when the software is in production. At this stage, most defects show up as performance events. However, they'd have to be severe or repeating for them to fit into a defect category. By extension, an infrastructure meltdown that renders applications unavailable doesn't count as a defect—not even if it becomes commonplace.

About User Feedback : You can recognize defects as user feedback that contains an application malfunction. Typically, such issues are traceable to a task once scheduled for execution in previous stages of the application's life cycle. This means developers can invest some technical effort into rectifying the defect.

How to Calculate a Software Defect Rate : You can calculate defects manually also. For that purpose, following formula is used :

$$\text{defect rate} = \frac{\text{total number of defects}}{\text{observed projects}}$$

The simplest equation for defect rate divides the total observed defects by the number of individual projects observed. For a single product, the latter would be equal to

one, leaving the rate as the number of defects dete
Some teams focus on defects detected by user
production. Then they can trace every instance back t
root or just log a bug in a tracking system for resolution

There is a traditional approach to keep track
defects. It is the detection of the defects in the earlier st
of life cycle of a software application.

Defect Rate Should be Kept at Minimum Lev

Defect rate must be kept at the minimum level. Beca
defect rate is lower, the higher is the confidence that y
software is performing well. On the other hand, it c
just be that you're not using sufficiently compe
testing/observing methods. If you have some log of
feedback pouring in with defects from your user that
significant starting point toward maintaining a low de
rate.

Once you establish a defect rate, the implied tas
keeping it as low as your team can manage. At the
least, your efforts will save you the embarrassmen
pushing error-studded code into production. W
application failures/defects capable of draining million
losses from companies, it's worth providing users with
best user experience.

7. *What is the importance of Software Reliability?*

It refers to the ability of a system or componen
perform its required functions under static conditions f
specified period of time.

Similarly, software reliability refers to the likeli
of a software system fulfilling its assigned task in a g
environment for a predetermined number of input c
assuming there are no errors in the input and hardwar

In terms of software quality, it includes functiona
usability, performance, serviceability, capability, abili
install, maintain, and documentation.

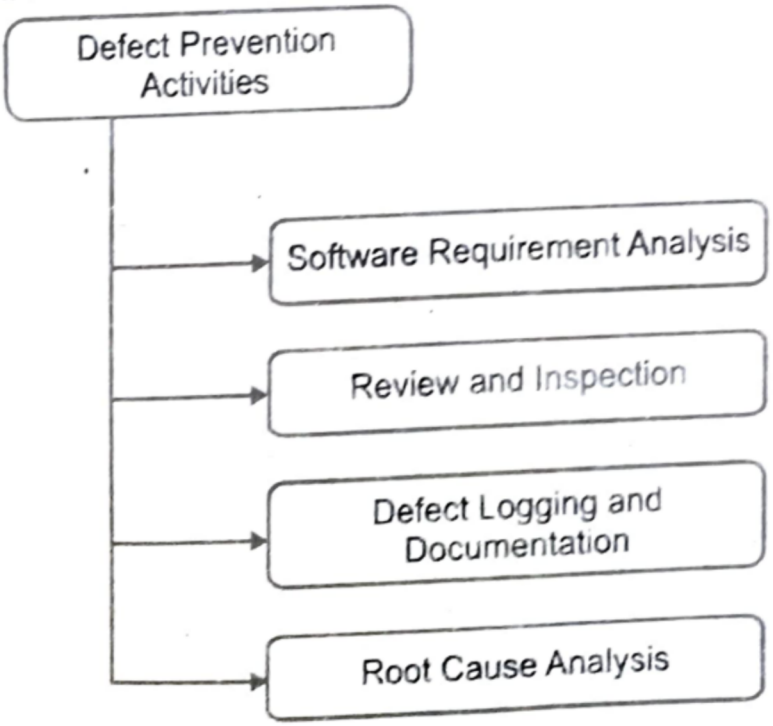
Software Reliability is hard to achieve because
complexity of software turn to be high. System develo
tend to push complexity into software in order to provi
certain level of reliability, as the speedy growth of sys
and ease of updating the software makes it difficu
reach a certain level of reliability.

8. *What is Defect Prevention?*

To prevent defects occurring frequently, defect prevention is considered. For that purpose, it's ensuring that defects that have been detected to date are not repeated. To achieve defect prevention there is facility to have proper communication among members of the team, designing defect prevention guidelines and so on.

To give overall attention on defect prevention, there is a special person assigned called as Defect Prevention Coordinator. He is primarily responsible to lead defect prevention efforts, to facilitate meetings between team members and management, etc. DP board generally has quarterly plan that sets some goals at organization level. To achieve these goals, various methods or activities are generally used and carried out to achieve and complete these goals.

Methods of Defect Prevention : For defect prevention, there are different methods that are generally used over a long period of time. These methods or activities are given below :



(Figure)

(1) **Software Requirement Analysis :** There is a lot of chance to occur defects during software requirement gathering. Generally, software defects occur as the result of faulty software requirements or designs.

Hence it is easy to remove defects at early stage Software requirements analysis. It is a set requirements that describe the features functionalities of a target product, and also conv the expectations and needs of users from software

(2) **Review and Inspection :** On the second st defects can be prevented in testing phase. For t purpose reviews and inspections can takes pl Software testing, like inspection, is viewed as important and integral part of the developm process. They are considered powerful tools identifying and removing defects before they imp production. They are used in all softw development and maintenance methods. There two types of review i.e. self-review and peer-review

(3) **Defect Logging and Documentation :** There be chance to have defects in the various docume created during software development. Rec describing defects should be kept to easily maint the description of defects following succes analysis and review. These records can also be u to have a better understanding of defects

(4) **Root Cause Analysis :** An analysis of the root ca of a defect is called a root cause analysis. This pro analyzes the cause of the defect. By analyzing main cause of a defect, the best way to prevent s types of defects from occurring in the future ca identified.

9. *Explain Prevention, reduction and containment detail.*

There are many quality assurance measures u which we can decide the quality of any software. Qu assurance measures are typically classified into t categories:

(1) **Defect Prevention :** The main reason of incre software quality is to remove the defects in software. Hence for reducing the defects, the sou errors is reduces, so that it reduces numb failures. The main question is that how to find

source of errors? Even though each project is unique, there are some conventional sources that are usually addressed by these measures.

If there is any new project, then there will be a new source of errors detected, so the related measures are there to avoid those errors. A continuous refinement strategy is the best guarantee to maximize defect prevention.

The first measure to reduce defect prevention is Education and training. Education provides people with the right tools to detect error and avoid them. This training can be done at various levels:

One more measure is to develop People for a product for a specific area, so that they should be familiar with that area. There are design decisions that can be heavily influenced by how the product is supposed to behave in a domain. There is a possibility that these details are not captured by any of the software requirements, thus, training within the domain is a valuable asset.

Defect prevention can be increased by developing Software expertise. This is probably the most intuitive of all. The objective is to develop a software product, thus, expertise on development techniques are fundamental.

Defect prevention can be increased by giving knowledge about methodology, technology and tools for the particular software development. Using the right tools and methodology has an impact on the quality of a product. But some of these topics are not trivial and require formal training for people to use them effectively.

Also we have to provide development process knowledge of the software. The development process should be understood in detail by all team members. Misunderstanding in some of the procedures will sure increase the possibility of errors occurring and deriving into failures.

Additionally from training and education, there are other techniques that hold some promise at preventing errors. They are normally known as

"formal method" techniques. The approach consis on treating a system as a collection of unambiguo specifications, capture its behavior with a model, a then prove that the system behavior is formal derived from the specification and its behavior.

Other methods for defect prevention tackl directly the software development and provi support to detect ambiguous constructions, scan documentation, good programming practices, etc.

(2) **Defect Reduction** : Once a defect is detected, procedure is put into place to remove it. The ma difference with the previous one is that the faults a removed from the software product only after bei detected.

To implement defect reduction, a system nee to be conceived to search the faults as much possible. Detecting faults can be done automatical but also manually through conventional inspecti procedures. These inspections should include co architecture and design documents.

Defect reduction is a task that is having aim cover overall detection of all possible behavior, wh could make the defects. Thus, the strategy is to try explore as much as possible of the ent behaviour space and hope that most of the defects a detected

The defect prevention techniques can be furth sub-divided as follows : Inspections, Testing a reviews.

(3) **Defect Containment** : These techniques a designed to react after a failure has been detect and to reduce its effect to a minimum. Some softwa products include functionality specifically to tr failures and restore the product functionality much as possible.

In some systems, the containment of failu may require even the use of physical objects. example a product that controls traffic lights in street, upon failure may defect to all lights being to minimize damage.

Defect containment can be defined as the abi of a software product to detect a failure and maint

its effect within some reasonable boundary that still allows the product to proceed for the further development.

- Now in this stage the approach is different. It's but obvious that all faults cannot be detected. Hence, to minimize its impact, some additional measures need to be put in place for when a failure occurs. This technique is called as "fault tolerance".

When combining these three categories with the waterfall scheme for product development, we could assign focus preventing actions in the initial phases of software requirement and specification, Design and partially in coding. Defect remove would mostly focus on the coding and testing areas. Finally, defect containment would focus solely on the release and support phases of the project.

10. *What are the different types of Software Review?*

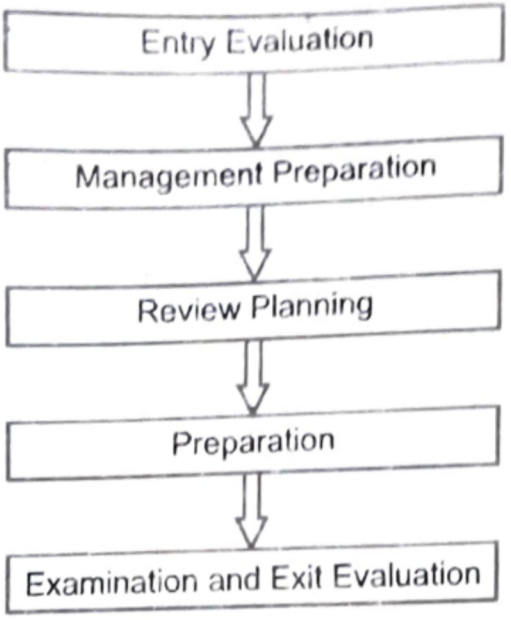
Software Review is systematic approach to inspect the overall development of software. For that purpose, one or more persons are involved to view every process of software development. Generally these people are working together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC). Software review is an essential part of Software Development Life Cycle (SDLC) that helps software engineers in validating the quality, functionality and other vital features and components of the software. It is a whole process that includes testing the software product and it makes sure that it meets the requirements stated by the client.

Usually performed manually, software review is used to verify various documents like requirements, system designs, codes, test plans and test cases.

Objectives of Software Review : The objective of software review is :

(1) To improve the productivity of the development team.
(2) To make the testing process time and cost effective.
(3) To make the final software with fewer defects.
(4) To eliminate the inadequacies.

Process of Software Review:



(Figure)

Types of Software Reviews : There are mainly 3 types of software reviews :

(1) **Software Peer Review :** Peer review is the process of checking the technical content and quality of the product. This is generally conducted by the author of the work product along with some other developers. Peer review is performed in order to examine or resolve the defects in the software, whose quality is also checked by other members of the team.

Peer Review has following types :

(a) **Code Review :** To check entire source code in a systematic way.

(b) **Pair Programming :** To review a code with the two developers working together on the same platform.

(c) **Walkthrough :** To ask the questions about software development by members of the development team, author and other interested parties and the participants

(d) **Technical Review :** A team of highly qualified individuals examines the software product for its client's use and identifies technical defects from specifications and standards.

(e) **Inspection :** In inspection the reviewers follow a well-defined process to find defects.

(2) **Software Management Review** : Software Management Review evaluates the work status. In this section decisions regarding downstream activities are taken.

(3) **Software Audit Review** : Software Audit Review is a type of external review in which one or more critics, who are not a part of the development team, organize an independent inspection of the software product and its processes to assess their compliance with stated specifications and standards. This is done by managerial level people.

Advantages of Software Review :

(1) Defects can be identified earlier stage of development.

(2) Earlier inspection also reduces the maintenance cost of software.

(3) It can be used to train technical authors.

(4) It can be used to remove process inadequacies that encourage defects.

11. *Give the brief introduction to Inspection Process in the Software Engineering. Also state advantages and disadvantages.*

It is very important to test and review and find the defects at early stage of software development. Inspection is used to determine the defects in the code and remove it efficiently. This prevents defects and enhances the quality of testing to remove defects. This software inspection method achieved the highest level for efficiently removing defects and improving software quality.

Following factors generate the high quality software :

(1) **Design Inspection and Code Inspections** : This factor refers to formal inspection of design and code.

(2) **Phrase Quality Assurance** : This factor refers to an active software quality assurance group, which joins a group of software developments to support them in the development of high quality software.

(3) **Formal Testing** : Formal testing is performed under various conditions such as black box testing, white box testing or grey testing.

Software Inspection Process : The Software Inspection process has an entry criterion that determines whether the

inspection process is ready to begin. This prevents incomplete products from entering the inspection process. For example, entry criteria may be as spelling checker of the document. There are some of the stages in the software inspection process such as :

(1) **Planning :** The moderator plan the inspection.

(2) **Overview Meeting :** The background of the work product is described by the author.

(3) **Preparation :** The examination of the work product is done by inspector to identify the possible defects.

(4) **Inspection Meeting :** The reader reads the work product part by part during this meeting and the inspectors the faults of each part.

(5) **Rework :** After the inspection meeting, the writer changes the work product according to the work plans.

(6) **Follow Up :** The changes done by the author are checked to make sure that everything is correct.

Advantages of Software Inspection :

(1) Helps in the Early removal of major defects.

(2) This inspection enables a numeric quality assessment of any technical document.

(3) Software inspection helps in process improvement.

(4) It helps in staff training on the job.

(5) Software inspection helps in gradual productivity improvement.

Disadvantages of Software Inspection :

(1) It is a time-consuming process.

(2) Software inspection requires discipline.

12. *Explain Software Metrics in detail. Also give types, advantages, disadvantages and Software Metrics.*

Software metric can be defined as a measure of software characteristics which can be measured or counted in specific amount. Software metrics are valuable as they can include measuring software performance, planning work items, measuring productivity, and many other uses.

There are different metrics in the software development process, which are connected with each other. Software metrics are as follows :

(1) Planning,
(2) Organization,
(3) Control
(4) Improvement.

Software metrics can be classified into two types as follows :

(1) **Product Metrics :** These are the measures of various characteristics of the software product. The two important software characteristics are :
   (a) Size and complexity of software.
   (b) Quality and reliability of software.
   These metrics can be computed for different stages of SDLC.

(2) **Process Metrics :** These are the measures of various characteristics of the software development process. For example, the efficiency of fault detection. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software.

Types of Metrics :

Internal Metrics : Used to measure Lines of Code (LOC).

External Metrics : Used to measure portability, reliability, functionality, usability, etc. of the software.

Hybrid Metrics : Used to combine product, process, and resource metrics.

Project Metrics : Used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost; these estimates are used as a base of new software.

Advantage of Software Metrics :

(1) It's possible to study various design methodology of software systems comparatively.
(2) Analysis, comparison, and critical study of different programming language can be done.
(3) It's possible to compare and evaluate the capabilities and productivity of people involved in software development.
(4) Preparation of software quality specifications can be done.

(5) To get an idea about the complexity of the code.
(6) In guiding resource manager for their proper utilization.

Disadvantage of Software Metrics :

(1) Difficult and costly process.
(2) Difficult to verify the validity of historical/empirical data for verification and justification of software metrics.
(3) This process is dependent on tools available and working environment.
(4) Most of the predictive models rely on estimates of certain variables which are often not known precisely.

13. *What are the different types of documents required in Software Testing?*

Testing documentation involves the various documents required in the development during the testing of Software.

Documentation for software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing, etc. During software testing following documents are required :

(1) Test Plan
(2) Test Scenario
(3) Test Case
(4) Traceability Matrix

Test Plan : A test plan gives brief idea about outline about the strategy which will be used to test an application, which resources are used, which will be the test environment in which testing will be performed, and also the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan includes the following :

(1) Introduction to the Test Plan document
(2) Assumptions while testing the application
(3) List of test cases included in testing the application
(4) List of features to be tested
(5) What sort of approach to use while testing the software

(6) List of deliverables that need to be tested
(7) The resources allocated for testing the application
(8) Any risks involved during the testing process
(9) A schedule of tasks and milestones to be achieved

Test Scenario : It is only one line statement which gives notification about the area in the application will be tested. Test scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

Test Case : Test cases involve a set of steps, conditions, and inputs that can be used while performing testing tasks. The main intent of this activity is to ensure whether software passes or fails in terms of its functionality and other aspects. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc.

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post-condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to

determine whether a software product is functioning as per the requirements of the customer.

Traceability Matrix : Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table that is used to trace the requirements during the Software Development Life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user-defined templates for RTM. Each requirement in the RTM document is linked with its associated test case so that testing can be done as per the mentioned requirements.

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship. It is used to track the requirements and to check the current project requirements are met.

❑❑