UNIT- 5

Software Measurement:

A measurement is an manifestation of the size, quantity, amount or dimension of a particular attributes of a product or process. Software measurement is a titrate impute of a characteristic of a software product or the software process. It is an authority within software engineering. Software measurement process is defined and governed by ISO Standard.

Need of Software Measurement:

Software is measured to:

1. Create the quality of the current product or process.
2. Anticipate future qualities of the product or process.
3. Enhance the quality of a product or process.
4. Regulate the state of the project in relation to budget and schedule.

Classification of Software Measurement:

There are 2 types of software measurement:

1. **Direct Measurement:**
   In direct measurement the product, process or thing is measured directly using standard scale.
2. **Indirect Measurement:**
   In indirect measurement the quantity or quality to be measured is measured using related parameter i.e. by use of reference.

Metrics for Software Quality:

Software metrics can be classified into three categories:

Product metrics: Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.

- **Process metrics**: These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics**: This metrics describe the project characteristics and execution.
  - Number of software developer
  - Staffing pattern over the life cycle of software
  - Cost and schedule
  - Productivity

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Software quality metrics:

Software metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.

Software quality metrics can be further divided into three categories:

- Product quality metrics
- In-process quality metrics
- Maintenance quality metrics

Product Quality Metrics

This metrics include the following:

- Mean Time to Failure
- Defect Density
- Customer Problems
- Customer Satisfaction

Mean Time to Failure

It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

Defect Density

It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

Customer Problems

It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

The problems metric is usually expressed in terms of **Problems per User-Month (PUM)**.

PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period.

Customer Satisfaction:

Customer satisfaction is often measured by customer survey data through the five-point scale:

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example:

- Percent of completely satisfied customers
- Percent of satisfied customers
- Percent of dis-satisfied customers
- Percent of non-satisfied customers

Usually, this percent satisfaction is used.

In-process Quality Metrics

In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes:

- Defect density during machine testing
- Defect arrival pattern during machine testing
- Phase-based defect removal pattern
- Defect removal effectiveness

Defect density during machine testing

Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following:

- The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.
- The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.
- The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

Phase-based defect removal pattern

This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

Defect removal effectiveness

It can be defined as follows:

DRE=DefectremoveddduringadevelopmentphaseDefectslatentintheproduct×100%DRE=DefectremoveddduringadevelopmentphaseDefectslatentintheproduct×100%

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called **early defect removal** when used

for the front-end and **phase effectiveness** for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

Maintenance Quality Metrics

Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.

- Fix backlog and backlog management index
- Fix response time and fix responsiveness
- Percent delinquent fixes
- Fix quality

Fix backlog and backlog management index

Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.

BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarrivedduringthemonth×100%BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarrivedduringthemonth×100%

If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

Fix response time and fix responsiveness

The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.

The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

Percent delinquent fixes

It is calculated as follows:

PercentDelinquentFixes=PercentDelinquentFixes=
Numberoffixesthatexceededtheresponsetimecriteriabyceveritylevel Numberoffixesdeliveredinaspecifiedtime×100%Numberoffixesthatexceededtheresponsetimecriteriabyceveritylevel Numberoffixesdeliveredinaspecifiedtime×100%

Fix Quality

Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.

A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered. The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.

Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture. The quality goal for the maintenance process, of course, is zero defective fixes without delinquency.

Proactive and Reactive

The basics are simple. **Reactive** risk management tries to reduce the damage of potential threats and speed an organization's recovery from them, but assumes that those threats will happen eventually. **Proactive** risk management identifies threats and aims to prevent those events from ever happening in the first place.

Each strategy has its own activities, metrics, and behaviors that are useful in risk analysis.

Reactive Risk Management

One fundamental point about reactive risk management is that the disaster or threat must occur before management responds. Proactive risk management is all about taking preventative measures before the event to decrease its severity, and that's a good thing to do.

At the same time, however, organizations should develop reactive risk management plans that can be deployed ***after*** the event. Otherwise management is making decisions about how to respond as the event happens, which can be a costly and stressful ordeal.

There's an obvious catch-22 with reactive risk management. Although this approach gives you time to understand the risk before acting, you're still always one step behind the unfolding threat. Other projects will lag as you attend to the problem at hand.

Helping to Withstand Future Risks

The reactive approach lear)the organization reacts ***after*** the threat has occurred and alters its measures to prevent future potential risks.

Proactive Risk Management

As the name suggests, proactive risk management means that you identify risks before they happen and figure out ways to avoid or alleviate the risk. It seeks to reduce the hazard's risk potential or, even better, prevent the threat altogether.

A good example here is vulnerability testing and remediation. Any organization of appreciable size is likely to have vulnerabilities in its software, which attackers could find an exploit. So regular testing (or, even better, continuous testing) can help to repair those vulnerabilities and eliminate that particular threat.

Allows for More Control Over Risk Management

Proactive management strategy gives you more control over your risk management generally. You can decide which issues should be top priorities, and what potential damage you're willing to accept.

Proactive management also involves constant monitoring of your systems, risk processes, cyber security, competition, business trends, and so forth. By understanding the level of risk prior to an event, you can educate and instruct your employees on how to mitigate them.

A truly proactive approach, however, does imply that each risk is constantly monitored. It also entails regular risk reviews to update the current risk and new risks affecting the company. This approach drives management to be always aware of the direction of those risks.

Software Risk:

Risk is uncertain events associated with future events which have a probability of occurrence but it may or may not occur and if occurs it brings loss to the project. Risk identification and management are very important task during software project development because success and failure of any software project depends on it.

Types of Risk:

1. **Schedule Risk :**
   Schedule related risks refers to time related risks or project delivery related planning risks. The wrong schedule affects the project development and delivery. These risks are mainly indicates to running behind time as a result project development doesn't progress timely and it directly impacts to delivery of project. Finally if schedule risks are not managed properly it gives rise to project failure and at last it affect to organization/company economy very badly.
   Some reasons for Schedule risks –
   - Time is not estimated perfectly
   - Improper resource allocation
   - Tracking of resources like system, skill, staff etc
   - Frequent project scope expansion
   - Failure in function identification and its' completion
2. **Budget Risk :**
   Budget related risks refers to the monetary risks mainly it occurs due to budget overruns. Always the financial aspect for the project should be managed as per decided but if financial aspect of project mismanaged then there budget concerns will arise by giving rise to budget risks. So proper finance distribution and management are required for the success of project otherwise it may lead to project failure.
   Some reasons for Budget risks:
   - Wrong/Improper budget estimation
   - Unexpected Project Scope expansion
   - Mismanagement in budget handling
   - Cost overruns
   - Improper tracking of Budget
3. **Operational Risks :**
   Operational risk refers to the procedural risks means these are the risks which happen in day-to-day operational activities during project development due to improper process implementation or some external operational risks.

Some reasons for Operational risks:

- Insufficient resources
- Conflict between tasks and employees
- Improper management of tasks
- No proper planning about project
- Less number of skilled people
- Lack of communication and cooperation
- Lack of clarity in roles and responsibilities
- Insufficient training

4. **Technical Risks :**
   Technical risks refers to the functional risk or performance risk which means this technical risk mainly associated with functionality of product or performance part of the software product.
   Some reasons for Technical risks:

   - Frequent changes in requirement
   - Less use of future technologies
   - Less number of skilled employee
   - High complexity in implementation
   - Improper integration of modules

5. **Programmatic Risks :**
   Programmatic risks refers to the external risk or other unavoidable risks. These are the external risks which are unavoidable in nature. These risks come from outside and it is out of control of programs.
   Some reasons for Programmatic risks:

   - Rapid development of market
   - Running out of fund / Limited fund for project development
   - Changes in Government rules/policy
   - Loss of contracts due to any reason

Methods for Identifying Risks

Identifying risk is one of most important or essential and initial steps in risk management process. By chance, if failure occurs in identifying any specific or particular risk, then all other steps that are involved in risk management will not be implemented for that particular risk. For identifying risk, project team should review scope of program, estimate cost, schedule, technical maturity, parameters of key performance, etc.

To manage risk, project team or organization are needed to know about what risks it faces, and then to evaluate them. Generally, identification of risk is an iterative process. It basically includes generating or creating comprehensive list of threats and opportunities that are based on events that can enhance, prevent, degrade, accelerate, or might delay successful achievement of objectives. In simple words, if you don't find or identify risk, you won't be able to manage it.

Methods for Identifying Risks :
Earlier, there were no easy methods available that will surely identify all risks. But nowadays, there are some additional approaches available for identifying risks. Some of approaches for risk identification are given below:

1. **Checklist Analysis:**
   Checklist Analysis is type of technique generally used to identify or find risks and manage it. The checklist is basically developed by listing items, steps, or even tasks and is then further analyzed against criteria to just identify and determine if procedure is completed correctly or not. It is list of risk that is just found to occur regularly in development of software project.
   Below is the list of software development risk by Barry Boehm- modified version.

Risk	Risk Reduction Technique
Personnel Shortfalls	Various techniques include training and career development, job-matching, teambuilding, etc.
Unrealistic time and cost estimates	Various techniques include incremental development, standardization of methods, recording, and analysis of the past project, etc.
Development of wrong software functions	Various techniques include formal specification methods, user surveys, etc.
Development of the wrong user interface	Various techniques include user involvement, prototyping, etc.

2. **Brainstorming:**
   This technique provides and gives free and open approach that usually encourages each and everyone on project team to participate. It also results in greater sense of ownership of project risk, and team generally committed to managing risk for given time period of project. It is creative and unique technique to gather risks spontaneously by team members. The team members identify and determine risks in 'no wrong answer' environment. This technique also provides opportunity for team members to always develop on each other's ideas. This technique is also used to determine best possible solution to problems and issue that arises and emerge.
3. **Casual Mapping:**
   Causal mapping is method that builds or develops on reflection and review of failure factors in cause and effect of the diagrams. It is very useful for facilitating learning with an organization or system simply as method of project-post evaluation. It is also key tool for risk assessment.
4. **SWOT Analysis:**
   Strengths-Weaknesses-Opportunities-Threat (SWOT) is very technique and helpful for identifying risks within greater organization context. It is generally used as planning tool for analyzing business, its resources, and also its environment simply by looking at internal strengths and weaknesses and opportunities and threats in external environment. It is technique often used in formulation of strategy. The appropriate time and effort should be spent on thinking seriously about weaknesses and threats of organization for SWOT analysis to more effective and successful in risk identification.
5. **Flowchart Method:**
   This method allows for dynamic process to be diagrammatically represented in paper. This method is generally used to represent activities of process graphically and sequentially to simply identify the risk.

Risk Projection:

Risk projection, also called *risk estimation*, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur.

The project planner, along with other managers and technical staff, performs four risk projection activities:

(1) Establish a scale that reflects the perceived likelihood of a risk.
(2) Delineate the consequences of the risk.
(3) Estimate the impact of the risk on the project and the product.
(4) Note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk Table

Risk table provides a project manager with a simple technique for risk projection.

Steps in Setting up Risk Table

(1) Project team begins by listing all risks in the first column of the table.
Accomplished with the help of the risk item checklists.

(2) Each risk is categorized in the second column.
(e.g. PS implies a project size risk, BU implies a business risk).

(3) The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually.

(4) Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

Assessing Risk Impact

Nature of the risk - the problems that are likely if it occurs.

e.g. a poorly defined external interface to customer hardware (a technical risk) will preclude early design and testing and will likely lead to system integration problems late in a project.

Scope of a risk - combines the severity with its overall distribution (how much of the project will be affected or how many customers are harmed?).

Timing of a risk - when and how long the impact will be felt.

Overall risk exposure, RE, determined using:

RE = P x C

P is the probability of occurrence for a risk.

C is the the cost to the project should the risk occur.

Risk Refinement:

During early stages of project planning, a risk may be stated quite generally. As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.

One way to do this is to represent the risk in condition-transition-consequence (CTC) format . That is, the risk is stated in the following form: Given that <condition> then there is concern that (possibly) <consequence>.

Using the CTC format for the reuse risk noted in Section 6.4.2, we can write:

Given that all reusable software components must conform to specific design standards and that some do not conform, then there is concern that (possibly) only 70 percent of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30 percent of components.

This general condition can be refined in the following manner:

Subcondition 1. Certain reusable components were developed by a third party with no knowledge of internal design standards.

Subcondition 2. The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.

Subcondition 3. Certain reusable components have been implemented in a language that is not supported on the target environment.

The consequences associated with these refined sub conditions remains the same (i.e., 30 percent of software components must be customer engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

RMMM:

A risk management strategy can be defined as a software project plan or the risk management steps. It can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

Teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet . In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:

(1) to assess whether predicted risks occur.

(2) to ensure that risk aversion steps defined for the risk are being properly applied; and

(3) to collect information that can be used for future risk analysis.

Effective strategy must consider three issues:

- risk avoidance
- risk monitoring
- risk management and contingency planning.

RMMM Plan:

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis. After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

Risk Mitigation :
It is an activity used to avoid problems (Risk Avoidance).
Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

Risk Monitoring :
It is an activity used for project tracking.
It has the following primary objectives as follows.

1. To check if predicted risks occur or not.
2. To ensure proper application of risk aversion steps defined for risk.
3. To collect data for future risk analysis.
4. To allocate what problems are caused by which risks throughout the project.

Risk Management and planning :

It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

Example:

Let us understand RMMM with the help of an example of high staff turnover.

Risk Mitigation:

To mitigate this risk, project management must develop a strategy for reducing turnover. The possible steps to be taken are:

- Meet the current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
- Assign a backup staff member for every critical technologist.

Risk Monitoring:

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

- General attitude of team members based on project pressures.
- Interpersonal relationships among team members.
- Potential problems with compensation and benefits.
- The availability of jobs within the company and outside it.

Risk Management:

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway, and a number of people announce that they will be

leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to the speed**".**

Drawbacks of RMMM:

- It incurs additional project costs.
- It takes additional time.
- For larger projects, implementing an RMMM may itself turn out to be another tedious project.
- RMMM does not guarantee a risk-free project, infact, risks may also come up after the project is delivered.

Software Quality Assurance (SQA):

Software Quality Assurance is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Software Quality Assurance has:

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

Major Software Quality Assurance Activities:

1. **SQA Management Plan:**
   Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.
2. **Set The Check Points:**
   SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

3. **Multi testing Strategy:**
   Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

4. **Measure Change Impact:**
   The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

5. **Manage Good Relations:**
   In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don't play politics.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating software.
7. Improves the quality of the software.

Disadvantage of SQA:
There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

Software Review:

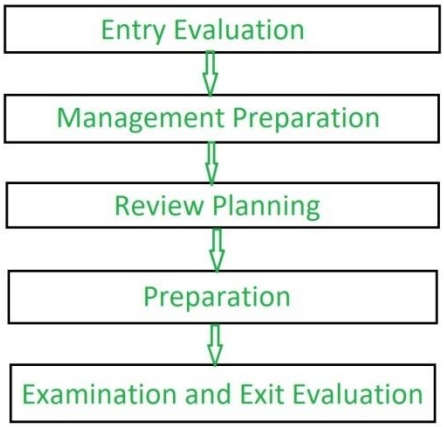
Software Review is systematic inspection of a software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC). Software review is an essential part of Software Development Life Cycle (SDLC) 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:



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

1. **Software Peer Review:**
   Peer review is the process of assessing the technical content and quality of the product and it is usually 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:

   - **(i) Code Review:**
     Computer source code is examined in a systematic way.
   - **(ii) Pair Programming:**
     It is a code review where two developers develop code together at the same platform.
   - **(iii) Walkthrough:**
     Members of the development team is guided bu author and other interested parties and the participants ask questions and make comments about defects.

- **(iv) 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.
- **(v) 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:

- Defects can be identified earlier stage of development (especially in formal review).
- Earlier inspection also reduces the maintenance cost of software.
- It can be used to train technical authors.
- It can be used to remove process inadequacies that encourage defects.

Formal Technical Review (FTR):

FTR is a software quality control activity performed by software engineers.

Objectives of formal technical review (FTR):

Some of these are:

- Useful to uncover error in logic, function and implementation for any representation of the software.
- The purpose of FTR is to verify that the software meets specified requirements.
- To ensure that software is represented according to predefined standards.
- It helps to review the uniformity in software that is development in a uniform manner.
- To makes the project more manageable.

The review meeting:

Each review meeting should be held considering the following constraints-

Involvement of people:

1. Between 3, 4 and 5 people should be involve in the review.
2. Advance preparation should occur but it should be very short that is at the most 2 hours of work for every person.

3. The short duration of the review meeting should be less than two hour. Gives these constraints, it should be clear that an FTR focuses on specific (and small) part of the overall software.

At the end of the review, all attendees of FTR must decide what to do.

1. Accept the product without any modification.
2. Reject the project due to serious error (Once corrected, another app need to be reviewed), or
3. Accept the product provisional (minor errors are encountered and are should be corrected, but no additional review will be required).

Review reporting and record keeping:

1. During the FTR, the reviewer actively records all issues that have been raised.
2. At the end of the meeting all these issues raised are consolidated and a review list is prepared.
3. Finally, a formal technical review summary report is prepared.

It answers three questions:

1. What was reviewed ?
2. Who reviewed it ?
3. What were the findings and conclusions ?

Review guidelines:

Review the product, not the manufacture (producer).

1. Take written notes (record purpose)
2. Limit the number of participants and insists upon advance preparation.
3. Develop a checklist for each product that is likely to be reviewed.
4. Allocate resources and time schedule for FTRs in order to maintain time schedule.
5. Conduct meaningful training for all reviewers in order to make reviews effective.
6. Reviews earlier reviews which serve as the base for the current review being conducted.
7. Set an agenda and maintain it.
8. Separate the problem areas, but do not attempt to solve every problem notes.
9. Limit debate and rebuttal.

Statistical Quality Assurence:

Traditional compliance testing techniques can sometimes provide limited pass/fail information, which results in insufficient measurements on the batch's quality control, identification of the root cause of failure results and overall quality assurance (QA) in the production process.

SQA consists of three major methodologies:

1. **Force Diagram:** A Force Diagram describes how a product should be tested. Intertek engineers base the creation of Force Diagrams on our knowledge of foreseeable use, critical manufacturing process and critical components that have high potential to fail.
2. **Test-to-Failure (TTF):** Unlike any legal testing, TTF tells manufacturers on how many defects they are likely to find in every million units of output. This information is incorporated into the process and concludes if a product needs improvement in quality or if it is being over engineered, which will eventually lead to cost savings.
3. **Intervention:** Products are separated into groups according to the total production quantity and production lines. Each group then undergoes an intervention. The end result is measured by Z-value, which is the indicator of quality and consistency of a product to a specification. Intervention allows manufacturers to pinpoint a defect to a specific lot and production line; thus saving time and money in corrective actions.

Software Reliablity:

Software Reliability is an essential connect of software quality, composed with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software Reliability is hard to achieve because the complexity of software turn to be high. While any system with a high degree of complexity, containing software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the speedy growth of system size and ease of doing so by upgrading the software.

The ISO 9000 quality standards:

The International organization for Standardization is a world wide federation of national standard bodies. The **International standards organization (ISO)** is a standard which serves as a for contract between independent parties. It specifies guidelines for development of **quality system**.

Why ISO Certification required by Software Industry?

There are several reasons why software industry must get an ISO certification. Some of reasons are as follows :

- This certification has become a standards for international bidding.
- It helps in designing high-quality repeatable software products.
- It emphasis need for proper documentation.
- It facilitates development of optimal processes and totally quality measurements.

Features of ISO 9001 Requirements :

- **Document control:**
  All documents concerned with the development of a software product should be properly managed and controlled.
- **Planning:**
  Proper plans should be prepared and monitored.
- **Review:**
  For effectiveness and correctness all important documents across all phases should be independently checked and reviewed .
- **Testing:**
  The product should be tested against specification.
- **Organizational Aspects:**
  Various organizational aspects should be addressed e.g., management reporting of the quality team.