Software Maintenance:

Software Maintenance:

- Software maintenance is a part of the Software Development Life Cycle.
- Its primary goal is to modify and update software application after delivery to correct errors and to improve performance.
- Software is a model of the real world. When the real-world changes, the software require alteration wherever possible.
- Software Maintenance is an inclusive activity that includes error corrections, enhancement of capabilities, deletion of obsolete capabilities, and optimization.

Need for Maintenance

Software Maintenance is needed for: -

- **Correct errors**
- **Change in user requirement with time**
- **Changing hardware/software requirements**
- **To improve system efficiency**
- **To optimize the code to run faster**
- **To modify the components**
- **To reduce any unwanted side effects.**

Types of Software Maintenance:

- **Corrective Software Maintenance**
- **Preventive Software Maintenance**
- **Perfective Software Maintenance**
- **Adaptive Software Maintenance**

Corrective Maintenance:

- Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.
- Corrective changes in the software are required when:

- Software is not working the way it is expected due to some acute issues, such as faulty logic flow, incorrect implementation, invalid or incomplete tests, etc.
- The issues in the software are affecting users after you have released the software.

Adaptive Maintenance;

- It contains modifying the software to match changes in the ever-changing environment.
- This type of software maintenance primarily focuses on software frameworks.
- Adaptive maintenance is made in response to new operating systems, platforms, and hardware to retain continuity with the software.

Preventive Maintenance:

- It is the process by which we prevent our system from being obsolete.
- It involves the concept of reengineering & reverse engineering in which an old system with old technology is re-engineered using new technology.
- This maintenance prevents the system from dying out.

Perfective Maintenance:

- It defines improving processing efficiency or performance or restricting the software to enhance changeability.
- This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.
- This process keeps software relevant as the market, and user needs, change.
- Perfective maintenance is important in order to improve a system efficiency, reliability or maintainability.

Software Reverse Engineering:

- Software Reverse Engineering is a process of recovering the design, requirement specifications and functions of a product from an analysis of its code.
- It builds a program database and generates information from this.
- Reverse engineering, also called as back engineering, is the process of extracting knowledge from anything man-made & reproducing something based on that information.

- Reverse Engineering Goals:
    - ✓ Cope with Complexity.
    - ✓ Recover lost information.
    - ✓ Detect side effects.
    - ✓ Synthesise higher abstraction.
    - ✓ Facilitate Reuse.3

Following are some important purposes of Reverse Engineering:

- ✓ Security auditing
- ✓ Enabling additional features
- ✓ Used as learning tools
- ✓ Developing compatible products cheaper than that are currently available in the market.

Following are three important parameters to be considered for of a reverse engineering process:

- ✓ Abstraction Level
- ✓ Completeness
- ✓ Directionality

Abstraction Level

- In the abstraction level of a reverse engineering process, the design information is extracted from the sou code.
- It is the highest level in the reverse engineering process.
- It is always expected that the abstraction level any reverse engineering process must be high.
- When the level of abstraction is high, then it becomes easy for the software developer to understand program.

Completeness

The completeness is nothing but the details available through the abstraction level of reverse engineer For example from the given source code it is very easy to develop the complete procedural design.

Directionality

- The directionality of a reverse engineering process is a method of extracting information from the source co and making it available to the software developers.
- The software developers can use this information during the maintenance activity.
- Following diagram exhibits the reverse engineering process.

Steps of Software Reverse Engineering:

1.0 Collection Information:

This step focuses on collecting all possible information (i.e., source design documents etc.) about the software.

2.0 Examining the information:

The information collected in step-1 as studied so as to get familiar with the system.

3.0 Extracting the structure:

This step concerns with identification of program structure in the form of structure chart where each node corresponds to some routine.

4.0 Recording the functionality:

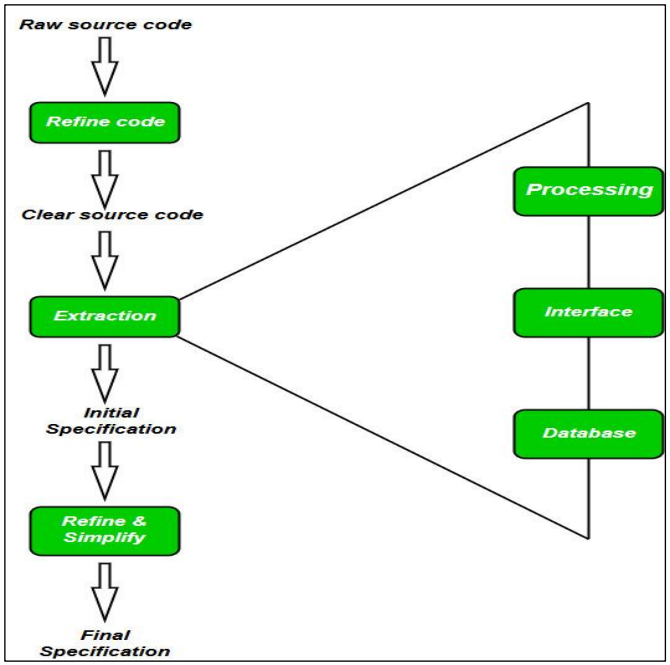
During this step processing details of each module of the structure, charts are recorded using structured language like decision table, etc.

4.0 Recording data flow:

From the information extracted in step-3 and step-4, set of data flow diagrams are derived to show the flow of data among the processes.

5.0 Recording control flow:

High level control structure of the software is recorded.



6.0 Review extracted design:

Design document extracted is reviewed several times to ensure consistency and correctness. It also ensures that the design represents the program.

7.0 Generate documentation:

Finally, in this step, the complete documentation including SRS, design document, history, overview, etc. are recorded for future use.

Reverse Engineering Tools:

Reverse engineering if done manually would consume lot of time and human labour and hence must be supported by automated tools. Some of tools are given below:

- CIAO and CIA: A graphical navigator for software and web repositories along with a collection of Reverse Engineering tools.
- Rigi: A visual software understanding tool.
- Bunch: A software clustering/modularization tool.
- GEN++: An application generator to support development of analysis tools for the C++ language.

Advantages of Reverse Engineering

The following are the advantages of reverse engineering:

Exploring Existing Designs and Maneuvers

We are able to observe what already exists thanks to reverse engineering. This includes any components, system, or procedures that might serve communities in other way. Through analysis of existing products, innovation and discovery are made possible.

Discovering Any Product Vulnerabilities

Reverse engineering helps in identifying product flaws, just as in the prior step. This is done to protect the users of the product's safety and wellbeing. An issue should ideally come up in the research stage rather than the distribution stage.

Inspiring Creative Minds with Old Ideas

Last but not least, reverse engineering provides a way for innovative design. An engineer may come upon a system during the process that could be valuable for a totally unrelated project. This demonstrates how engineering links tasks to prior knowledge.

Creative a Reliable CAD Model for Future Reference

The majority of reverse engineering procedures include creating a fully functional CAD file for future use. In order to inspect the part digitally in the event that problems develop later, a CAD file is made. This type of technology has improved product expressiveness and engineering productivity.

Risk

"Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project."

Whenever we start any business or any development process, we take into consideration the risks involve in accomplishing that task. Similarly, when software development process is started, it is main job of a software manager to look into all types of possible risks involved in the entire process.

It involves focusing on the possible risks that could affect the project development process: Schedule of the development process

Risk Management

A software project can be concerned with a large variety of risks. In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes. The project manager can then check which risks from each class are relevant to the project.

There are three main **classifications** of risks which can affect a software project:

- Project risks
- Technical risks
- Business risks

1. **Project risks**: Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems. A vital project risk is schedule slippage. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.

2**. Technical risks**: Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.

3. **Business risks**: This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

Other risk categories:

1. Known risks: Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)

2. Predictable risks: Those risks that are hypothesized from previous project experience (e.g., past turnover)

3. Unpredictable risks: Those risks that can and do occur, but are extremely tough to identify in advance.

Reactive Versus Proactive Risk Strategies

1 Reactive risk strategy:

- In this type of risk strategy, the project is monitored closely for the likely risks that may occur.
- Resources are monitored to guess the possible risks and deal with them so that they do not become the problems for budget slippage and cost slippage.
- Generally, in reactive risk strategy, the development team members are directly not involved in the risks occurring.
- This rapid action is called as "fire fighting mode".

2 Proactive risk strategy

- A proactive risk strategy is initiated when the actual technical work begins.
- The potential risks are identified and the probability and impact are assessed so that the development team members establish an appropriate plan to manage all these risk
- The main goal to avoid the risks initially.
- But all the risks cannot be avoided, therefore the development team creates a contingency plan that will control the risks in an effective manner.

Risk Assessment

The objective of risk assessment is to division the risks in the condition of their loss, causing potential. For risk assessment, first, every risk should be rated in two methods:

- The possibility of a risk coming true (denoted as r).
- The consequence of the issues relates to that risk (denoted as s).
- Based on these two methods, the priority of each risk can be estimated:

$$p = r * s$$

Where p is the priority with which the risk must be controlled, r is the probability of the risk becoming true, and s is the severity of loss caused due to the risk becoming true.

Risk Identification: The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

A project can be of use by a large variety of risk. To identify the significant risk, this might affect a project. It is necessary to categories into the different risk of classes.

There are different types of risks which can affect a software project:

Technology risks: Risks that assume from the software or hardware technologies that are used to develop the system.

People risks: Risks that are connected with the person in the development team.

Organizational risks: Risks that assume from the organizational environment where the software is being developed.

Tools risks: Risks that assume from the software tools and other support software used to create the system.

Requirement risks: Risks that assume from the changes to the customer requirement and the process of managing the requirements change.

Estimation risks: Risks that assume from the management estimates of the resources required to build the system.

Risk Control:

It is the process of managing risks to achieve desired outcomes. After all, the identified risks of a plan are determined; the project must be made to include the most harmful and the most likely risks. Different risks need different containment methods. In fact, most risks need ingenuity on the part of the project manager in tackling the risk.

There are three main methods to plan for risk management:

Avoid the risk: This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.

Transfer the risk: This method involves getting the risky element developed by a third party, buying insurance cover, etc.

Risk reduction: This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.

The 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

Risk management strategies should be included in the project plan itself. The risk management plan is usually a separate plan in the project plan. This is referred as risk mitigation, monitoring and management plan or RMMM plan.

The RMMM plan is executed as a separate plan to evaluate the risk. Each of the risks are documented separately.

The RMMM plan is well documented in the beginning of the project itself. Once the project begins, the mitigation and monitoring activities are started.

The risk monitoring has main three objectives:

- Check whether the predicted risks actually occur.
- All the risk assessment steps are properly followed, and
- Collect all the necessary information that can be useful for future risk analysis.

Risk Mitigation:

To mitigate this risk, you would develop a strategy for reducing turnover. Among the possible steps to be

taken are:

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and
- competitive job market).
- Mitigate those causes that are under your 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 work product standards and establish mechanisms to be sure that all models and documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is "up to speed").
- Assign a backup staff member for every critical technologist.

Risk Monitoring:

To mitigate this risk, you would develop a strategy for reducing turnover. Among the possible steps to be taken are:

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under your 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 work product standards and establish mechanisms to be sure that all models and documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is "up to speed").
- Assign a backup staff member for every critical technologist.

Risk Management and planning:

- Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality.
- If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team.
- In addition, you can 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 speed." Those individuals who are leaving are asked to stop all work and spend their last weeks in "knowledge transfer mode.".

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, in fact, risks may also come up after the project is delivered.

Ex. Risk: Computer Crash:

(a) Mitigation:

- The computer crash can cause the loss of important data and ultimately it will result in failure of the project.
- It is always recommended that software development organization should keep multiple copies of the data at multiple locations to avoid loss.

(b) Monitoring:

- The project manager must keep a watch on the working infrastructure and working environment before the actual development work starts.

(c) Management:

- Any inconvenience in working environment must be noticed immediately and a proper action should be taken to make the system stable.

Version Control

- Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.
- The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.
- Developers can compare earlier versions of the code with an older version to fix the mistakes.

✳Benefits of the Version Control System

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

Some key benefits of having a version control system are as follows.

- Complete change history of the file
- Simultaneously working
- Branching and merging
- Traceability

☢Types of Version Control System

1) Localized version Control System
2) Centralized version control systems
3) Distributed version control systems

❁Localized version Control System

The localized version control method is a common approach because of its simplicity. But this approach leads to a higher chance of error. In this approach, you may forget which directory you're in and accidentally write to the wrong file or copy over files you don't want to.

To deal with this issue, programmers developed local VCSs that had a simple database. Such databases kept all the changes to files under revision control. A local version control system keeps local copies of the files.

❁Centralized Version Control System:

The developers needed to collaborate with other developers on other systems. The localized version control system failed in this case. To deal with this problem, Centralized Version Control Systems were developed.

Centralized version control systems have many benefits, especially over local VCSs.

- Everyone on the system has information about the work what others are doing on the project.
- Administrators have control over other developers.
- It is easier to deal with a centralized version control system than a localized version control system.
- A local version control system facilitates with a server software component which stores and manages the different versions of the files.

Distributed Version Control System:

Centralized Version Control System uses a central server to store all the database and team collaboration. But due to single point failure, which means the failure of the central server, developers do not prefer it. Next, the Distributed Version Control System is developed.

What is Change Control?

Change Control is the process that a company uses to document, identify and authorize changes to an IT environment. It reduces the chances of unauthorized alterations, disruption and errors in the system.

Why Change Control?

Whenever any new or different changes are requested for the system, especially by stakeholders, it is neither optional nor ignorable. It has to be implemented without affecting other components of the system. This is when the change control comes handy. It helps project teams to modify the scope of the project using specified controls and policies. Change Control is practiced whenever a project is not progressing as planned.

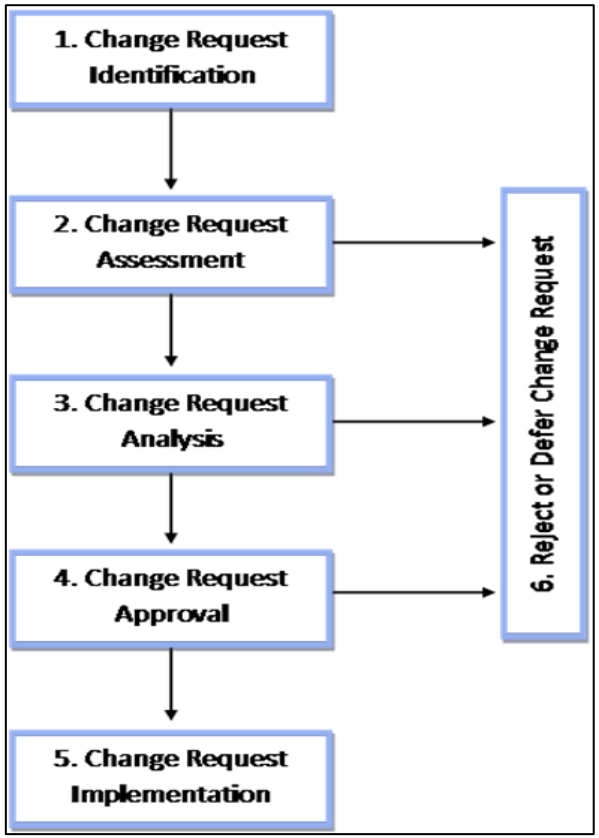
It is mandatory that a formal document for change request is completed and reviewed in order to keep control of change requests.

Importance of Change Management:

- For improving performance
- For increasing engagement
- For enhancing innovation
- For including new technologies
- For implementing new requirements
- For reducing cost

Change Process Flow-Diagram

Change Process follows a specific pattern to implement the changes in the product or system. Here through flow-diagram we explained what are the steps involved in the Change Process.



Steps for Change Control

Steps for Change Control	Action
• Change request identification	• Identify the need for a change and describe it on the project change request form
• Change request assessment	• If the change is not valid, it has to be deferred or rejected • Determine appropriate resources required to analyze the change request • Perform quick assessment of the potential impact and update the change request form • At this stage, rejected change request should stopped
• Change request analysis	• For analysis assign the change request to an authorized member • Deferred change re-enter this analysis step • At this stage, rejected change request should stopped
• Change request approval	• Identify change risk and complexity level before approval • Identify the impact level of the change before approval • Review impact of Change Request to authorized person for approval • At this stage, rejected change request should stopped
• Change request implementation	• Update project procedure and management plans • Inform about the changes to the team • Monitor progress of change request • Record the completion of change request • Close change request

☯Software Configuration Management (SCM):

"In Software Engineering, Software Configuration Management (SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. "

The primary goal is to increase productivity with minimal mistakes.

SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Why do we need Configuration management?

The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating
- It may be a case where multiple version, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system.

SCM process:

1 Configuration Manager

- Configuration Manager is the head who is Responsible for identifying configuration items.
- CM ensures team follows the SCM process
- He/She needs to approve or reject change requests

2. Developer

- The developer needs to change the code as per standard development activities or change requests. He is responsible for maintaining configuration of code.
- The developer should check the changes and resolves conflicts

3. Auditor

- The auditor is responsible for SCM audits and reviews.
- Need to ensure the consistency and completeness of release.

4. Project Manager:

- Ensure that the product is developed within a certain time frame
- Monitors the progress of development and recognizes issues in the SCM process
- Generate reports about the status of the software system
- Make sure that processes and policies are followed for creating, changing, and testing

5. User

- The end user should understand the key SCM terms to ensure he has the latest version of the software

Software Configuration Management Tools

Any Change management software should have the following 3 Key features:

Concurrency Management:

When two or more tasks are happening at the same time, it is known as concurrent operation. Concurrency in context to SCM means that the same file being edited by multiple persons at the same time.

If concurrency is not managed correctly with SCM tools, then it may create many pressing issues.

Version Control:

SCM uses archiving method or saves every change made to file. With the help of archiving or save feature, it is possible to roll back to the previous version in case of issues.

Synchronization:

Users can checkout more than one files or an entire copy of the repository. The user then works on the needed file and checks in the changes back to the repository. They can synchronize their local copy to stay updated with the changes made by other team members.

Following are popular tools

1. Git: Git is a free and open source tool which helps version control. It is designed to handle all types of projects with speed and efficiency.

2. Team Foundation Server: Team Foundation is a group of tools and technologies that enable the team to collaborate and coordinate for building a product.

3. Ansible: It is an open source Software configuration management tool. Apart from configuration management it also offers application deployment & task automation.

Software Quality Assurance

What is Quality?

Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability.

- Software quality product is defined in term of its fitness of purpose.
- That is, a quality product does precisely what the users want it to do.
- For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document.
- Although "fitness of purpose" is a satisfactory interpretation of quality for many devices such as a car, a table fan, a grinding machine, etc.for software products, "fitness of purpose" is not a wholly satisfactory definition of quality.

Software Quality Assurance has:

- A quality management approach
- Formal technical reviews
- Multi testing strategy
- Effective software engineering technology
- Measurement and reporting mechanism

Example: Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.

The modern view of a quality associated with a software product several (measuring) quality methods such as the following:

Portability: A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.

Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.

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

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

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.

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.

Objectives of Software Review:

The objective of software review is:

- To improve the productivity of the development team.
- To make the testing process time and cost effective.
- To make the final software with fewer defects.
- To eliminate the inadequacies

formal technical review

The formal technical review or FTR is one of the quality assurance activities conducted by the software developers.

The FTR can be conducted by the other stakeholders of the project also at different point of times.

Following are some important objectives of the formal technical reviews

- Detect the possible errors in the program logic and uncover errors during implementation of the product.
- Evaluate that the product satisfy the clients' requirements.
- The product is built according to the standards defined earlier.
- The product design is uniform, and
- Ensure that the project is manageable.

The new developers can be benefitted a lot by FTR since they come across with different approaches for the software development activities like requirement analysis, design and implementation of the design.

The FTR is assumed as a separate class of software reviews and consists of

- Walkthroughs
- Inspections, and
- Reviews by individuals.

12.4.1 Review Meetings

The review meetings are conducted that abide by the following rules and constraints:

- In the review meeting, there should be at least three to five people involved for better understanding and adaptation of ideas and reviews.
- There must be prior preparation for the meeting.
- The review meeting should not exceed two hours.

By using these constraints all the stakeholders must focus on the specific parts of the product or the project.

At the end of the meeting all the members those who have attended meeting must take unanimous decisions for accepting the product or rejecting the product.

During the review meeting, the reviewer records all the issues on a paper and summarizes all the issues and present at the end of the meeting to all the attendees of the meeting.

The review summary has following three important questionnaires:

- Which parts of the product reviewed,
- Name the people who reviewed, and finally
- The conclusion of the meeting.

12.4.2 Review Guidelines

Following are some review guidelines for better software review meetings:

- During the review meeting, always review the product and do not review the producer who produced it.
- The review leader should ensure the healthy and friendly environment and there should not be any misconduct by any attendee of the review meeting
- The review leader should immediately call the meeting cancelled if situation goes out control.
- Always establish the agenda of the meeting and adhere to it. The FTR must be on the proper track and schedule.

- For any issue raised, there should be limit in the debate.
- List the problems and do not try to solve all the problems as it is impossible and people will not be agreed upon.
- Always take the note of all the issues discussed.
- There should be some limit for the number of participants. There should not be heavy crowd in the meeting as objective of the meeting will be failed.
- Develop a checklist for the product to be reviewed.
- Allocate proper resources and proper schedule for the review meetings.
- Finally all early reviews must be reviewed.

Software Reliability

- Unlike other quality factors, software reliability can be measured directly by using historical data. The software reliability is an important quality factor used by most of the developers.
- The software reliability is defined as the probability of failure free program in a specified environment.
- When we discuss failure, one question comes into mind. What is failure?

Measures of Reliability and Availability

If we talk about hardware reliability model then we predict failure due to wear rather than failure due to design defects i.e. physical wear due to effect temperature, corrosion, shock etc.

But when we discuss software reliability then the failure is related to design defects and the failure can be traced to implementation problems.

Consider a computer based system in that the measure of reliability is Mean-Time-Between-Fallure (MTRF)

where

$$MTBF = MTTF\ MTTR$$

Where.

- MTTF Mean-Time-To-Failure
- MTTR = Mean-Time-To-Repair

In addition the software availability is defined as the probability that a program is running as per the requirement at a given point of time.

$$Availability = [MTTF/(MTTF+MTTR)] \times 100\%$$

The availability measure is more sensitive to MTTR and it is an indirect measure of maintainability of software.

12.5.2 Software Safety

- Software safety is an activity of Software Quality Assurance (SQA) and it focuses on finding and assessing various
- hazards. These hazards can have the worst effect on the entire system and can halt the system from working. If these hazards are identified in the beginning of the process, then it is easy to eliminate and control the hazards.
- Once these hazards are identified at system level then analysis techniques are used to find the probability of occurrence and the safety measures can be taken.
- Even if software reliability and software safety are closely related it is very important to understand the
- significant differences between them.
- Software reliability uses historical data to find the probability of the occurrence of software failure.
- While software safety is a way in that also failures occurs but these failures lead to accidents that are dangerous to life.