Unit 4

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:-

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

Thus the maintenance is required to ensure that the system continues to satisfy user requirements.

Types of Software Maintenance

1. Corrective Maintenance

Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.

2. Adaptive Maintenance

It contains modifying the software to match changes in the ever-changing environment.

3. 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.

4. 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.
Causes of Software Maintenance Problems
Lack of Traceability
o Codes are rarely traceable to the requirements and design specifications.
o It makes it very difficult for a programmer to detect and correct a critical defect affecting customer operations.
o Like a detective, the programmer pores over the program looking for clues.
o Life Cycle documents are not always produced even as part of a development project.
Lack of Code Comments
o Most of the software system codes lack adequate comments. Lesser comments may not be helpful in certain situations.
Obsolete Legacy Systems
o In most of the countries worldwide, the legacy system that provides the backbone of the nation's critical industries, e.g., telecommunications, medical, transportation utility services, were not designed with maintenance in mind.
o They were not expected to last for a quarter of a century or more!
o As a consequence, the code supporting these systems is devoid of traceability to the requirements, compliance to design and programming standards and often includes dead, extra and uncommented code, which all make the maintenance task next to the impossible.
Software Maintenance Process

Program Understanding
The first step consists of analyzing the program to understand.
Generating a Particular maintenance problem
The second phase consists of creating a particular maintenance proposal to accomplish the implementation of the maintenance goals.
Ripple Effect
The third step consists of accounting for all of the ripple effects as a consequence of program modifications.
Modified Program Testing
The fourth step consists of testing the modified program to

ensure that the revised application has at least the same reliability level as prior.

Maintainability

Each of these four steps and their associated software quality attributes is critical to the maintenance process. All of these methods must be combined to form maintainability.

Software Maintenance Cost Factors

There are two types of cost factors involved in software maintenance.

These are

o Non-Technical Factors

o Technical Factors

Non-Technical Factors

1. Application Domain

o If the application of the program is defined and well understood, the system requirements may be definitive and maintenance due to changing needs minimized.

o If the form is entirely new, it is likely that the initial conditions will be modified frequently, as user gain experience with the system.

2. Staff Stability

o It is simple for the original writer of a program to understand and change an application rather than some other person who must understand the program by the study of the reports and code listing.

o If the implementation of a system also maintains that systems, maintenance costs will reduce.

o In practice, the feature of the programming profession is such that persons change jobs regularly. It is unusual for one user to develop and maintain an application throughout its useful life.

3. Program Lifetime

o Programs become obsolete when the program becomes obsolete, or their original hardware is replaced, and conversion costs exceed rewriting costs.

4. Dependence on External Environment

o If an application is dependent on its external environment, it must be modified as the climate changes.

o For example:

o Changes in a taxation system might need payroll, accounting,

and stock control programs to be modified.
o Taxation changes are nearly frequent, and maintenance costs for these programs are associated with the frequency of these changes.
o A program used in mathematical applications does not typically depend on humans changing the assumptions on which the program is based.

5. Hardware Stability

o If an application is designed to operate on a specific hardware configuration and that configuration does not changes during the program's lifetime, no maintenance costs due to hardware changes will be incurred.
o Hardware developments are so increased that this situation is rare.
o The application must be changed to use new hardware that replaces obsolete equipment.

Technical Factors

Technical Factors include the following:

Module Independence

It should be possible to change one program unit of a system without affecting any other unit.

Programming Language

Programs written in a high-level programming language are generally easier to understand than programs written in a low-level language.

Programming Style

The method in which a program is written contributes to its understandability and hence, the ease with which it can be modified.

Program Validation and Testing

o Generally, more the time and effort are spent on design validation and program testing, the fewer bugs in the program and, consequently, maintenance costs resulting from bugs correction are lower.
o Maintenance costs due to bug's correction are governed by the type of fault to be repaired.
o Coding errors are generally relatively cheap to correct, design errors are more expensive as they may include the rewriting of one or more program units.

o Bugs in the software requirements are usually the most expensive to correct because of the drastic design which is generally involved.
Documentation
o If a program is supported by clear, complete yet concise documentation, the functions of understanding the application can be associatively straight-forward.
o Program maintenance costs tends to be less for well-reported systems than for the system supplied with inadequate or incomplete documentation.
Configuration Management Techniques
o One of the essential costs of maintenance is keeping track of all system documents and ensuring that these are kept consistent.
o Effective configuration management can help control these costs.
Software Configuration Management
When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively.
Several individuals (programs) works together to achieve these common goals. This individual produces several work product (SC Items) e.g., Intermediate version of modules or test data used during debugging, parts of the final product.
The elements that comprise all information produced as a part of the software process are collectively called a software configuration.
As software development progresses, the number of Software Configuration elements (SCI's) grow rapidly.
These are handled and controlled by SCM. This is where we require software configuration management.
A configuration of the product refers not only to the product's constituent but also to a particular version of the component.
Therefore, SCM is the discipline which
o Identify change
o Monitor and control change
o Ensure the proper implementation of change made to the item.
o Auditing and reporting on the change made.
Configuration Management (CM) is a technic of identifying,

organizing, and controlling modification to software being built by a programming team.
The objective is to maximize productivity by minimizing mistakes (errors).
CM is used to essential due to the inventory management, library management, and updation management of the items essential for the project.
Why do we need Configuration Management?
Multiple people are working on software which is consistently updating. It may be a method where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently. It changes in user requirements, and policy, budget, schedules need to be accommodated.
Importance of SCM
It is practical in controlling and managing the access to various SCIs e.g., by preventing the two members of a team for checking out the same component for modification at the same time.
It provides the tool to ensure that changes are being properly implemented.
It has the capability of describing and storing the various constituent of software.
SCM is used in keeping a system in a consistent state by automatically producing derived version upon modification of the same component.
SCM Process
It uses the tools which keep that the necessary change has been implemented adequately to the appropriate component. The SCM process defines a number of tasks:
o Identification of objects in the software configuration
o Version Control
o Change Control
o Configuration Audit
o Status Reporting
Identification
Basic Object: Unit of Text created by a software engineer during analysis, design, code, or test.
Aggregate Object: A collection of essential objects and other aggregate objects. Design Specification is an aggregate object.

Each object has a set of distinct characteristics that identify it uniquely: a name, a description, a list of resources, and a "realization."

The interrelationships between configuration objects can be described with a Module Interconnection Language (MIL).

Version Control

Version Control combines procedures and tools to handle different version of configuration objects that are generated during the software process.

Clemm defines version control in the context of SCM: Configuration management allows a user to specify the alternative configuration of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes.

Change Control

James Bach describes change control in the context of SCM is: Change Control is Vital. But the forces that make it essential also make it annoying.

We worry about change because a small confusion in the code can create a big failure in the product. But it can also fix a significant failure or enable incredible new capabilities.

We worry about change because a single rogue developer could sink the project, yet brilliant ideas originate in the mind of those rogues, and

A burdensome change control process could effectively discourage them from doing creative work.

A change request is submitted and calculated to assess technical merit; potential side effects, the overall impact on other configuration objects and system functions, and projected cost of the change.

The results of the evaluations are presented as a change report, which is used by a change control authority (CCA) - a person or a group who makes a final decision on the status and priority of the change.

The "check-in" and "check-out" process implements two necessary elements of change control-**access control** and **synchronization control**.

Access Control governs which software engineers have the authority to access and modify a particular configuration object.
Synchronization Control helps to ensure that parallel changes, performed by two different people, don't overwrite one another.
Configuration Audit
SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.
SCM audits also ensure that traceability is maintained between all CIs and that all work requests are associated with one or more CI modification.
SCM audits are the "**watchdogs**" that ensures that the integrity of the project's scope is preserved.
Status Reporting
Configuration Status reporting (sometimes also called status accounting) providing accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.
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.
There are two kinds of Quality:

Quality of Design: Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.
Quality of conformance: 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.
Software Quality: Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all

professionally developed software.

Quality Control: Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

Quality Assurance: Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.

Quality Assurance focuses on how the engineering and management activity will be done?

As anyone is interested in the quality of the final product, it should be assured that we are building the right product.

It can be assured only when we do inspection & review of intermediate products, if there are any bugs, then it is debugged. This quality can be enhanced.

Importance of Quality

We would expect the quality to be a concern of all producers of goods and services. However, the distinctive characteristics of software and in particular its intangibility and complexity, make special demands.

Increasing criticality of software: The final customer or user is naturally concerned about the general quality of software, especially its reliability. This is increasing in the case as organizations become more dependent on their computer systems and software is used more and more in safety-critical areas. For example, to control aircraft.

The intangibility of software: This makes it challenging to know that a particular task in a project has been completed satisfactorily. The results of these tasks can be made tangible by demanding that the developers produce 'deliverables' that can be examined for quality.

Accumulating errors during software development: As computer system development is made up of several steps where the output from one level is input to the next, the errors in the earlier ?deliverables? will be added to those in the later stages leading to accumulated determinable effects. In general the later in a project that an error is found, the more expensive it will be 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.
Software Quality Assurance
Software quality assurance is a planned and systematic plan of all actions necessary to provide adequate confidence that an item or product conforms to establish technical requirements.
A set of activities designed to calculate the process by which the products are developed or manufactured.
SQA Encompasses
o A quality management approach
o Effective Software engineering technology (methods and tools)
o Formal technical reviews that are tested throughout the software process
o A multitier testing strategy
o Control of software documentation and the changes made to it.
o A procedure to ensure compliances with software development standards
o Measuring and reporting mechanisms.
SQA Activities
Software quality assurance is composed of a variety of functions associated with two different constituencies ? the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, record keeping, analysis, and reporting.
Following activities are performed by an independent SQA group:
1. **Prepares an SQA plan for a project:** The program is developed during project planning and is reviewed by all stakeholders. The plan governs quality assurance activities performed by the software engineering team and the SQA group. The plan identifies calculation to be performed, audits and reviews to be performed, standards that apply to the project, techniques for error reporting and tracking, documents to be produced by the SQA team, and amount of feedback provided to the software project team.
2. **Participates in the development of the project's software process description:** The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal

software standards, externally imposed standards (e.g. ISO-9001), and other parts of the software project plan.
3. **Reviews software engineering activities to verify compliance with the defined software process:** The SQA group identifies, reports, and tracks deviations from the process and verifies that corrections have been made.
4. **Audits designated software work products to verify compliance with those defined as a part of the software process:** The SQA group reviews selected work products, identifies, documents and tracks deviations, verify that corrections have been made, and periodically reports the results of its work to the project manager.
5. **Ensures that deviations in software work and work products are documented and handled according to a documented procedure:** Deviations may be encountered in the project method, process description, applicable standards, or technical work products.
6. **Records any noncompliance and reports to senior management:** Non- compliance items are tracked until they are resolved.
Quality Assurance v/s Quality control
Quality Assurance
Quality Control
Quality Assurance (QA) is the set of actions including facilitation, training, measurement, and analysis needed to provide adequate confidence that processes are established and continuously improved to produce products or services that conform to specifications and are fit for use.
Quality Control (QC) is described as the processes and methods used to compare product quality to requirements and applicable standards, and the actions are taken when a nonconformance is detected.
QA is an activity that establishes and calculates the processes that produce the product. If there is no process, there is no role for QA.
QC is an activity that demonstrates whether or not the product produced met standards.
QA helps establish process
QC relates to a particular product or service
QA sets up a measurement program to evaluate processes

QC verified whether particular attributes exist, or do not exist, in a explicit product or service.
QA identifies weakness in processes and improves them
QC identifies defects for the primary goals of correcting errors.
Quality Assurance is a managerial tool.
Quality Control is a corrective tool.
Verification is an example of QA.
Validation is an example of QC.

Software Quality Assurance Plan

Abbreviated as SQAP, the software quality assurance plan comprises the procedures, techniques, and tools that are employed to make sure that a product or service aligns with the requirements defined in the SRS(software requirement specification).

The plan identifies the SQA responsibilities of a team, and lists the areas that need to be reviewed and audited. It also identifies the SQA work products.

The SQA plan document consists of the below sections:

1. Purpose section
2. Reference section
3. Software configuration management section
4. Problem reporting and corrective action section
5. Tools, technologies, and methodologies section
6. Code control section
7. Records: Collection, maintenance, and retention section
8. Testing methodology

SQA Activities

Given below is the list of SQA activities:

#1) Creating an SQA Management Plan:

The foremost activity includes laying down a proper plan regarding how the SQA will be carried out in your project.
Along with what SQA approach you are going to follow, what engineering activities will be carried out, and it also includes ensuring that you have the right talent mix in your team.

#2) Setting the Checkpoints:

The SQA team sets up different checkpoints according to which it evaluates the quality of the project activities at each checkpoint/project stage. This ensures regular quality inspection and working as per the schedule.

#3) Apply software Engineering Techniques:

Applying some software engineering techniques aids a software designer in achieving high-quality specifications. For gathering information, a designer may use techniques such as interviews and FAST (Functional Analysis System Technique).

Later, based on the information gathered, the software designer can prepare the project estimation using techniques like WBS (work breakdown structure), SLOC (source line of codes), and FP(functional point) estimation.

#4) Executing Formal Technical Reviews:

An FTR is done to evaluate the quality and design of the prototype.

In this process, a meeting is conducted with the technical staff to discuss the actual quality requirements of the software and the design quality of the prototype. This activity helps in detecting errors in the early phase of SDLC and reduces rework effort in the later phases.

#5) Having a Multi-Testing Strategy:

By multi-testing strategy, we mean that one should not rely on any single testing approach, instead, multiple types of testing should be performed so that the software product can be tested well from all angles to ensure better quality.

#6) Enforcing Process Adherence:

This activity insists on the need for process adherence during the software development process. The development process should also stick to the defined procedures.

This activity is a blend of two sub-activities which are explained below in detail:

(i) Product Evaluation:

This activity confirms that the software product is meeting the requirements that were discovered in the project management plan. It ensures that the set standards for the project are followed correctly.

(ii) Process Monitoring:

This activity verifies if the correct steps were taken during software development. This is done by matching the actually taken steps against the documented steps.

#7) Controlling Change:

In this activity, we use a mix of manual procedures and automated tools to have a mechanism for change control.

By validating the change requests, evaluating the nature of change, and controlling the change effect, it is ensured that the software quality is maintained during the development and maintenance phases.

#8) Measure Change Impact:

If any defect is reported by the QA team, then the concerned team fixes the defect.

After this, the QA team should determine the impact of the change which is brought by this defect fix. They need to test not only if the change has fixed the defect, but also if the change is compatible with the whole project.

For this purpose, we use software quality metrics that allow managers and developers to observe the activities and proposed changes from the beginning till the end of SDLC and initiate corrective action wherever required.

#9) Performing SQA Audits:

The SQA audit inspects the entire actual SDLC process followed by comparing it against the established process.

It also checks whether whatever was reported by the team in the status reports was actually performed or not. This activity also exposes any non-compliance issues.

#10) Maintaining Records and Reports:

It is crucial to keep the necessary documentation related to SQA and share the required SQA information with the stakeholders. The test results, audit results, review reports, change requests documentation, etc. should be kept for future reference.

#11) Manage Good Relations:

In fact, it is very important to maintain harmony between the QA and the development team.

We often hear that testers and developers often feel superior to each other. This should be avoided as it can affect the overall project quality.

Software Quality Assurance Standards

In general, SQA may demand conformance to one or more standards.

Some of the most popular standards are discussed below:

ISO 9000: This standard is based on seven quality management principles which help the organizations to ensure that their products or services are aligned with the customer

needs.
7 principles of ISO 9000 are depicted in the below image:

CMMI level: CMMI stands for **Capability maturity model Integration**. This model originated in software engineering. It can be employed to direct process improvement throughout a project, department, or entire organization.
5 CMMI levels and their characteristics are described in the below image:

An organization is appraised and awarded a maturity level rating (1-5) based on the type of appraisal.
Test Maturity Model integration (TMMi): Based on CMMi, this model focuses on maturity levels in software quality management and testing.
5 TMMi levels are depicted in the below image:

As an organization moves to a higher maturity level, it achieves a higher capability for producing high-quality products with fewer defects and closely meets the business requirements.
Elements of Software Quality Assurance
There are 10 essential elements of SQA which are enlisted below for your reference:
1. Software engineering Standards
2. Technical reviews and audits
3. Software Testing for quality control
4. Error collection and analysis
5. Change management
6. Educational programs
7. Vendor management
8. Security management
9. Safety
10. Risk management

SQA Techniques
There are several techniques for SQA. Auditing is the chief technique that is widely adopted. However, we have a few other significant techniques as well.
Various SQA Techniques include:
- **Auditing:** Auditing involves inspection of the work products

and its related information to determine if the set of standard processes were followed or not.

• **Reviewing**: A meeting in which the software product is examined by both the internal and external stakeholders to seek their comments and approval.

• **Code Inspection:** It is the most formal kind of review that does static testing to find bugs and avoid defect growth in the later stages. It is done by a trained mediator/peer and is based on rules, checklist, entry and exit criteria. The reviewer should not be the author of the code.

• **Design Inspection:** Design inspection is done using a checklist that inspects the below areas of software design:

• General requirements and design
• Functional and Interface specifications
• Conventions
• Requirement traceability
• Structures and interfaces
• Logic
• Performance
• Error handling and recovery
• Testability, extensibility
• Coupling and cohesion

• **Simulation: A simulation** is a tool that models a real-life situation in order to virtually examine the behavior of the system under study.

• **Functional Testing:** It is a QA technique that verifies what the system does without considering how it does it. This type of black box testing mainly focuses on testing the system specifications or features.

• **Standardization:** Standardization plays a crucial role in quality assurance. It decreases the ambiguity and guesswork, thus ensuring quality.

• **Static Analysis:** It is a software analysis that is done by an automated tool without actually executing the program. This technique is highly used for quality assurance in medical, nuclear, and aviation software. Software metrics and reverse engineering are some popular forms of static analysis.

• **Walkthroughs:** A software walkthrough or code walkthrough is a kind of peer review where the developer guides the members of the development team to go through the product and raise queries, suggest alternatives, and make comments regarding possible errors, standard violations, or any other issues.

- **Path Testing:** It is a white box testing technique where the complete branch coverage is ensured by executing each independent path at least once.
- **Stress Testing:** This type of testing is done to check how robust a system is by testing it under heavy load i.e. beyond normal conditions.
- **Six Sigma:** Six Sigma is a quality assurance approach that aims at nearly perfect products or services. It is widely applied in many fields including software. The main objective of six sigma is process improvement so that the produced software is 99.76 % defect-free.

Software Quality

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.

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 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.
Software Quality Management System
A quality management system is the principal methods used by organizations to provide that the products they develop have the desired quality.
A quality system subsists of the following:
Managerial Structure and Individual Responsibilities: A quality system is the responsibility of the organization as a whole. However, every organization has a sever quality department to perform various quality system activities. The quality system of an arrangement should have the support of the top management. Without help for the quality system at a high level in a company, some members of staff will take the quality system seriously.
Quality System Activities: The quality system activities encompass the following:
Auditing of projects
Review of the quality system
Development of standards, methods, and guidelines, etc.
Production of documents for the top management summarizing the effectiveness of the quality system in the organization.
Evolution of Quality Management System
Quality systems have increasingly evolved over the last five decades. Before World War II, the usual function to produce quality products was to inspect the finished products to remove defective devices. Since that time, quality systems of organizations have undergone through four steps of evolution, as shown in the fig. The first product inspection task gave method to quality control (QC).
Quality control target not only on detecting the defective devices and removes them but also on determining the causes behind the defects. Thus, quality control aims at correcting the reasons for bugs and not just rejecting the products. The next breakthrough in quality methods was the development of

quality assurance methods.
The primary premise of modern quality assurance is that if an organization's processes are proper and are followed rigorously, then the products are obligated to be of good quality. The new quality functions include guidance for recognizing, defining, analyzing, and improving the production process.
Total quality management (TQM) advocates that the procedure followed by an organization must be continuously improved through process measurements. TQM goes stages further than quality assurance and aims at frequently process improvement. TQM goes beyond documenting steps to optimizing them through a redesign. A term linked to TQM is Business Process Reengineering (BPR).
BPR aims at reengineering the method business is carried out in an organization. From the above conversation, it can be stated that over the years, the quality paradigm has changed from product assurance to process assurance, as shown in fig.

ISO 9000 Certification
ISO (International Standards Organization) is a group or consortium of 63 countries established to plan and fosters standardization. ISO declared its 9000 series of standards in 1987. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the guidelines for maintaining a quality system. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.
Types of ISO 9000 Quality Standards
The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.
1. **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.

2. **ISO 9002:** This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.
3. **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

How to get ISO 9000 Certification?

An organization determines to obtain ISO 9000 certification applies to ISO registrar office for registration. The process consists of the following stages:

1. **Application:** Once an organization decided to go for ISO certification, it applies to the registrar for registration.
2. **Pre-Assessment:** During this stage, the registrar makes a rough assessment of the organization.
3. **Document review and Adequacy of Audit:** During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
4. **Compliance Audit:** During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
5. **Registration:** The Registrar awards the ISO certification after the successful completion of all the phases.
6. **Continued Inspection:** The registrar continued to monitor the organization time by time.

Software Engineering Institute Capability Maturity Model (SEICMM)

The Capability Maturity Model (CMM) is a procedure used to develop and refine an organization's software development process.

The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.

CMM was developed and is promoted by the Software Engineering Institute (SEI), a research and development center promote by the U.S. Department of Defense (DOD).

Capability Maturity Model is used as a benchmark to measure

the maturity of an organization's software process.
Methods of SEICMM
There are two methods of SEICMM:

Capability Evaluation: Capability evaluation provides a way to assess the software process capability of an organization. The results of capability evaluation indicate the likely contractor performance if the contractor is awarded a work. Therefore, the results of the software process capability assessment can be used to select a contractor.
Software Process Assessment: Software process assessment is used by an organization to improve its process capability. Thus, this type of evaluation is for purely internal use.
SEI CMM categorized software development industries into the following five maturity levels. The various levels of SEI CMM have been designed so that it is easy for an organization to build its quality system starting from scratch slowly.

Level 1: Initial
Ad hoc activities characterize a software development organization at this level. Very few or no processes are described and followed. Since software production processes are not limited, different engineers follow their process and as a result, development efforts become chaotic. Therefore, it is also called a chaotic level.
Level 2: Repeatable
At this level, the fundamental project management practices like tracking cost and schedule are established. Size and cost estimation methods, like function point analysis, COCOMO, etc. are used.
Level 3: Defined
At this level, the methods for both management and development activities are defined and documented. There is a common organization-wide understanding of operations, roles, and responsibilities. The ways through defined, the process and product qualities are not measured. ISO 9000 goals at achieving this level.
Level 4: Managed
At this level, the focus is on software metrics. Two kinds of

metrics are composed.
Product metrics measure the features of the product being developed, such as its size, reliability, time complexity, understandability, etc.
Process metrics follow the effectiveness of the process being used, such as average defect correction time, productivity, the average number of defects found per hour inspection, the average number of failures detected during testing per LOC, etc. The software process and product quality are measured, and quantitative quality requirements for the product are met. Various tools like Pareto charts, fishbone diagrams, etc. are used to measure the product and process quality. The process metrics are used to analyze if a project performed satisfactorily. Thus, the outcome of process measurements is used to calculate project performance rather than improve the process.
Level 5: Optimizing
At this phase, process and product metrics are collected. Process and product measurement data are evaluated for continuous process improvement.
Key Process Areas (KPA) of a software organization
Except for SEI CMM level 1, each maturity level is featured by several Key Process Areas (KPAs) that contains the areas an organization should focus on improving its software process to the next level. The focus of each level and the corresponding key process areas are shown in the fig.

SEI CMM provides a series of key areas on which to focus to take an organization from one level of maturity to the next. Thus, it provides a method for gradual quality improvement over various stages. Each step has been carefully designed such that one step enhances the capability already built up.
People Capability Maturity Model (PCMM)
PCMM is a maturity structure that focuses on continuously improving the management and development of the human assets of an organization.
It defines an evolutionary improvement path from Adhoc, inconsistently performed practices, to a mature, disciplined, and continuously improving the development of the knowledge, skills, and motivation of the workforce that enhances strategic business performance.

The People Capability Maturity Model (PCMM) is a framework that helps the organization successfully address their critical people issues. Based on the best current study in fields such as human resources, knowledge management, and organizational development, the PCMM guides organizations in improving their steps for managing and developing their workforces.
The People CMM defines an evolutionary improvement path from Adhoc, inconsistently performed workforce practices, to a mature infrastructure of practices for continuously elevating workforce capability.
The PCMM subsists of five maturity levels that lay successive foundations for continuously improving talent, developing effective methods, and successfully directing the people assets of the organization. Each maturity level is a well-defined evolutionary plateau that institutionalizes a level of capability for developing the talent within the organization
The five steps of the People CMM framework are:

Initial Level: Maturity Level 1
The Initial Level of maturity includes no process areas. Although workforce practices implement in Maturity Level, 1 organization tend to be inconsistent or ritualistic, virtually all of these organizations perform processes that are defined in the Maturity Level 2 process areas.
Managed Level: Maturity Level 2
To achieve the Managed Level, Maturity Level 2, managers starts to perform necessary people management practices such as staffing, operating performance, and adjusting compensation as a repeatable management discipline. The organization establishes a culture focused at the unit level for ensuring that person can meet their work commitments. In achieving Maturity Level 2, the organization develops the capability to handle skills and performance at the unit level. The process areas at Maturity Level 2 are Staffing, Communication and Coordination, Work Environment, Performance Management, Training and Development, and Compensation.
Defined Level: Maturity Level 3
The fundamental objective of the defined level is to help an organization gain a competitive benefit from developing the different competencies that must be combined in its workforce

to accomplish its business activities. These workforce competencies represent critical pillars supporting the strategic workforce competencies to current and future business objectives; the improved workforce practices for implemented at Maturity Level 3 become crucial enablers of business strategy.

Predictable Level: Maturity Level 4

At the Predictable Level, the organization handles and exploits the capability developed by its framework of workforce competencies. The organization is now able to handle its capacity and performance quantitatively. The organization can predict its capability for performing work because it can quantify the ability of its workforce and of the competency-based methods they use performing in their assignments.

Optimizing Level: Maturity Level 5

At the Optimizing Level, the integrated organization is focused on continual improvement. These improvements are made to the efficiency of individuals and workgroups, to the act of competency-based processes, and workforce practices and activities.

Next Topic

Quality Assurance	**Quality Control**
Quality Assurance (QA) is the set of actions including facilitation, training, measurement, and analysis needed to provide adequate confidence that processes are established and continuously improved to produce products or services that conform to specifications and are fit for use.	**Quality Control (QC)** is described as the processes and methods used to compare product quality to requirements and applicable standards, and the actions are taken when a nonconformance is detected.

QA is an activity that establishes and calculates the processes that produce the product. If there is no process, there is no role for QA.	**QC** is an activity that demonstrates whether or not the product produced met standards.
QA helps establish process	**QC** relates to a particular product or service
QA sets up a measurement program to evaluate processes	**QC** verified whether particular attributes exist, or do not exist, in a explicit product or service.
QA identifies weakness in processes and improves them	**QC** identifies defects for the primary goals of correcting errors.
Quality Assurance is a managerial tool.	Quality Control is a corrective tool.
Verification is an example of QA.	Validation is an example of QC.