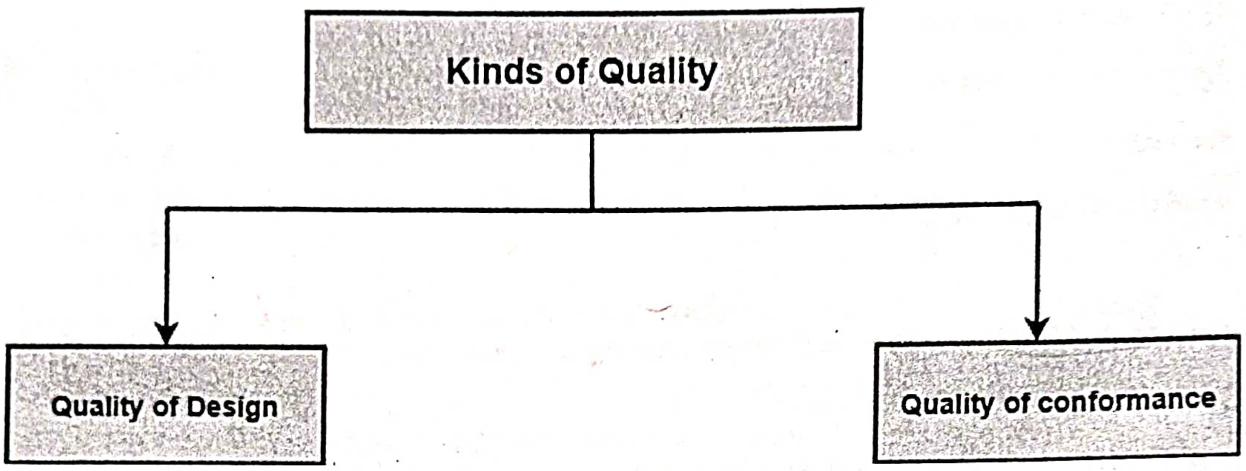
Software Quality Assurance

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

- A quality management approach
- Effective Software engineering technology (methods and tools)
- Formal technical reviews that are tested throughout the software process
- A multitier testing strategy
- Control of software documentation and the changes made to it.
- A procedure to ensure compliances with software development standards
- 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.

Software Engineering | Software Quality Framework

•

Software Quality Framework is a model for software quality that ensures quality by connecting and integrating the different views of software quality. This framework connects the customer view with the developer view of software quality and it treats software as a product. The software product view describes the characteristics of a product that bear on its ability to satisfy stated and implied needs.

This is a framework that describes all the different concepts relating to quality in a common way measured by qualitative scale that can be understood and interpreted in a common way. Therefore the most influential factor for the developers is the customer perception. This framework connects the developer with the customer to derive a common interpretation for quality.

1. **Developers View:**
   Validation and verification are two independent methods used together for checking that a software product meets the requirements and that it fulfills its intended purpose.Validation checks that the product design satisfies the purposeful usage and verification checks for errors in the software. The primary concern for developers is in the design and engineering processes involved in producing software. Quality can be measured by the degree of conformance to predetermined requirement and standards, and deviations from these standards can lead to poor quality and low reliability. While validation and verification are used by the developers to improve the software, the two methods don't represent a quantifiable quality measurement.

The developer view of software quality and customer view of software quality are both different things.
For example the customer understands or describes the quality of operation as meeting the requirement while the developers use different factors to describe the software quality.
The developer view of quality in the software is influenced by many factors. This model stresses on 3 primary ones:

1. **The code:**
   It is measured by its correctness and reliability.
2. **The data:**
   It is measured by the application integrity.
3. **Maintainability:**
   It has different measures the simplest is the mean time to change.

2. **Users View:**
   When the user acquires software, he/she always expect a high-quality software. When end users develop their software then quality is different. End-user programming, a phrase popularized by which is programming to achieve the result of a program primarily for personal, rather than public use. The important distinction here is that software itself is not primarily intended for use by a large number of users with varying needs.
   For example, a teacher may write a spreadsheet to track students'test scores. In these end-user programming situations, the program is a means to an end that could be used to accomplish a goal. In contradiction to end-user programming, professional programming has the goal of producing software for others to use.

   For example, the moment a novice Web developer moves from designing a web page for himself to designing a Web page for others, the nature of this activity has changed.
   Users find software quality as a fit between their goals and software's functionality. The better the quality, the more likely the user will be satisfied with the soft-ware. When the quality is bad, developers must meet user needs or face a diminishing demand for their software. Therefore, the user understands quality as fitness for purpose. Avoiding complexity and keeping software simple, considerably lessens the implementation risk of software.In some instances, users abandoned the implementation of a complex software because the software developers were expecting the users to change their business and to go with the way the software works.

3. **Product View:**
   The product view describes quality as correlated to inherent characteristics of the product. Product quality is defined as the set of characteristics and features of a

product that gives contribution to its ability to fulfill given requirements. Product quality can be measured by the value-based view which sees the quality as dependent on the amount a customer is willing to pay for it. According the users, a high-quality product is one that satisfies their expectations and preferences while meeting their requirement. Satisfaction of end users of the product represents craft to learn, use, upgrade the product and when asked to participate in rating the product, a positive rating is given.

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.

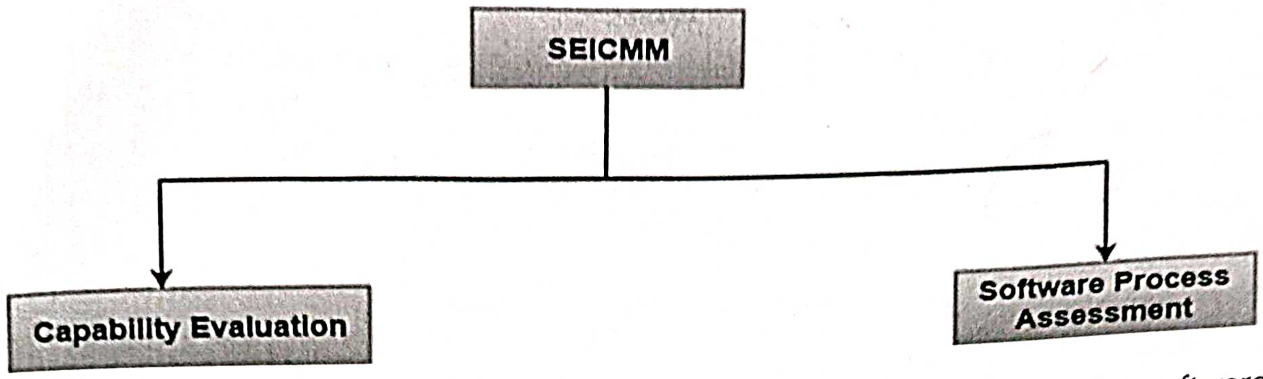
The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.

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

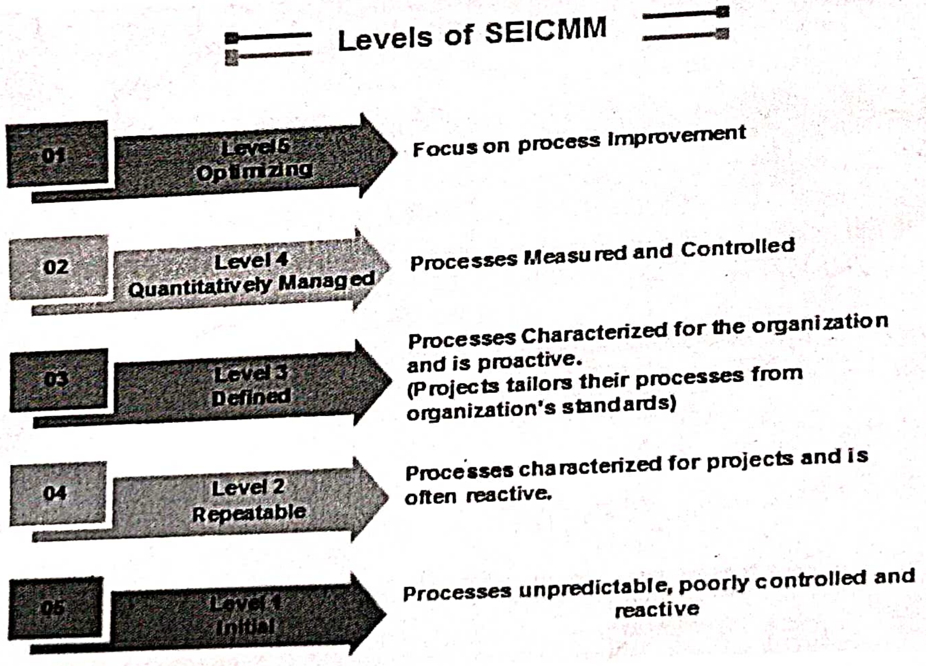
There are two methods of SEICMM:



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.


Levels of SEICMM
01
Level 5
Optimizing
Focus on process Improvement
02
Level 4
Quantitatively Managed
Processes Measured and Controlled
03
Level 3
Defined
Processes Characterized for the organization and is proactive.
(Projects tailors their processes from organization's standards)
04
Level 2
Repeatable
Processes characterized for projects and is often reactive.
05
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

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.

CMM Level	Focus	Key Process Areas
1. Initial	Competent People	NO KPA'S
2. Repeatable	Project Management	Software Project Planning software Configuration Management
3. Defined	Definition of Processes	Process definition Training Program Peer reviews
4. Managed	Product and Process quality	Quantitiative Process Metrics Software Quality Management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management

The focus of each SEI CMM level and the Corresponding Key process areas.

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.