Software Quality

Software Quality

- An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.
- Software quality determines how well the software is designed (quality of design), and how well the software conforms to that design (quality of conformance).
- Set of activities designed to evaluate the quality of developed or manufactured products
- A quality factor represents a behavioral characteristic of a system. Some examples of high-level quality factors are correctness, reliability, efficiency, testability, portability, and reusability.

Software Quality

- **Customers** may want an efficient and reliable software with less concern for portability.
- The **developers** strive to meet customer needs by making their system efficient and reliable, at the same time making the product portable and reusable to reduce the cost of software development
- The **software quality assurance** team is more interested in the testability of a system so that some other factors, such as correctness, reliability, and efficiency, can be easily verified through testing.

- The testability factor is important to developers and customers as well:
- (i) Developers want to test their product before delivering it to the software quality assurance team and
- (ii) customers want to perform acceptance tests before taking delivery of a product.

The Software Quality Dilemma

- If you produce a software system that has terrible quality, you lose because no one will want to buy it.

- If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.

- Either you missed the market window, or you simply exhausted all your resources. So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete. [Ven03]

McCall software quality model

What is Software Quality Model?

- Software quality models were proposed to measure the quality of any software model.
- There are three widely accepted models when it comes to measuring software quality

- McCall's Quality Model
- Boehm quality model
- Dromey's quality model

McCall software quality model was introduced in 1977.
This model is incorporated with many attributes, termed as software factors, which influence a software.
The main intention of this model was to maintain harmony between users and developers.

Mc call Quality Factors

Quality Factors	Definition
Correctness	Extent to which a program satisfies its specifications and fulfills the user's mission objectives
Reliability	Extent to which a program can be expected to perform its intended function with required precision
Efficiency	Amount of computing resources and code required by a program to perform a function
Integrity	Extent to which access to software or data by unauthorized persons can be controlled
Usability	Effort required to learn, operate, prepare input, and interpret output of a program
Maintainability	Effort required to locate and fix a defect in an operational program
Testability	Effort required to test a program to ensure that it performs its intended functions
Flexibility	Effort required to modify an operational program
Portability	Effort required to transfer a program from one hardware and/or software environment to another
Reusability	Extent to which parts of a software system can be reused in other applications
Interoperability	Effort required to couple one system with another

McCall software quality model

- **Correctness:** A software system is expected to meet the explicitly specified functional requirements and the implicitly expected nonfunctional requirements. If a software system satisfies all the functional requirements, the system is said to be correct. However, a correct software system may still be unacceptable to customers if the system fails to meet unstated requirements, such as stability, performance, and scalability. On the other hand, even an incorrect system may be accepted by users.

- **Reliability:** It is difficult to construct large software systems which are correct. A few functions may not work in all execution scenarios, and, therefore, the software is considered to be incorrect. However, the software may still be acceptable to customers because the execution scenarios causing the system to fail may not frequently occur when the system is deployed. Moreover, customers may accept software failures once in a while. Customers may still consider an incorrect system to be reliable if the failure rate is very small and it does not adversely affect their mission objectives. Reliability is a customer perception, and an incorrect software can still be considered to be reliable.

McCall software quality model

- **Efficiency:** Efficiency concerns to what extent a software system utilizes resources, such as computing power, memory, disk space, communication bandwidth, and energy. A software system must utilize as little resources as possible to perform its functionalities. For example, by utilizing less communication bandwidth a base station in a cellular telephone network can support more users.

- **Integrity:** A system's integrity refers to its ability to withstand attacks to its security. In other words, integrity refers to the extent to which access to software or data by unauthorized persons or programs can be controlled. Integrity has assumed a prominent role in today's network-based applications. Integrity is also an issue in multiuser systems.

McCall software quality model

- **Usability:** A software system is considered to be usable if human users find it easy to use. Users put much emphasis on the user interface of software systems. Without a good user interface a software system may fizzle out even if it possesses many desired qualities.

- However, it must be remembered that a good user interface alone cannot make a product successful—the product must also be reliable,

- for example. If a software fails too often, no good user interface can keep it in the market.

McCall software quality model

- **Maintainability**: In general, maintenance refers to the upkeep of products in response to deterioration of their components due to continued use of the products. Maintainability refers to how easily and inexpensively the maintenance tasks can be performed.

- For software products, there are three categories of maintenance activities: **corrective, adaptive, and perfective.** Corrective maintenance is a postrelease activity, and it refers to the removal of defects existing in an in-service software. The existing defects might have been known at the time of release of the product or might have been introduced during maintenance. Adaptive maintenance concerns adjusting software systems to changes in the execution environment. Perfective maintenance concerns modifying a software system to improve some of its qualities.

McCall software quality model

- **Testability:** It is important to be able to verify every requirement, both explicitly stated and simply expected. Testability means the ability to verify requirements. At every stage of software development, it is necessary to consider the testability aspect of a product. Specifically, for each requirement we try to answer the question: What procedure should one use to test the requirement, and how easily can one verify it? To make a product testable, designers may have to instrument a design with functionalities not available to the customer.

- **Flexibility:** Flexibility is reflected in the cost of modifying an operational system. As more and more changes are effected in a system throughout its operational phase, subsequent changes may cost more and more. If the initial design is not flexible, it is highly likely that subsequent changes are very expensive. In order to measure the flexibility of a system, one has to find an answer to the question: How easily can one add a new feature to a system?

McCall software quality model

- **Portability:** Portability of a software system refers to how easily it can be adapted to run in a different execution environment. An execution environment is a broad term encompassing hardware platform, operating system, distributedness, and heterogeneity of the hardware system, to name a few. Portability is important for developers because a minor adaptation of a system can increase its market potential. Moreover, portability gives customers an option to easily move from one execution environment to another to best utilize emerging technologies in furthering their business.

- Good design principles such as modularity facilitate portability.

- **Reusability:** Reusability means if a significant portion of one product can be reused, maybe with minor modification, in another product. It may not be economically viable to reuse small components. Reusability saves the cost and time to develop and test the component being reused. In the field of scientific computing, mathematical libraries are commonly reused. Reusability is not just limited to product parts, rather it can be applied to processes as

McCall software quality model

- **Interoperability:** In this age of computer networking, isolated software systems are turning into a rarity. Today's software systems are coupled at the input–output level with other software systems. Intuitively, interoperability means whether or not the output of one system is acceptable as input to another system; it is likely that the two systems run on different computers interconnected by a network.

- Example of interoperability is the ability to roam from one cellular phone network in one country to another cellular network in another country.

McCall software quality model

TABLE 17.2 Categorization of McCall's Quality Factors

Quality Categories	Quality Factors	Broad Objectives
Product operation	Correctness	Does it do what the customer wants?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Does it quickly solve the intended problem?
	Integrity	Is it secure?
	Usability	Can I run it?
Product revision	Maintainability	Can it be fixed?
	Testability	Can it be tested?
	Flexibility	Can it be changed?
Product transition	Portability	Can it be used on another machine?
	Reusability	Can parts of it be reused?
	Interoperability	Can it interface with another system?

How do we achieve Software quality?

- Achieving quality will ensure **maximum profit** for your software business. But the biggest hurdle is to achieve quality and here are some of the ways.
- Define characteristics that define quality for a product
- Decide how to measure each of that quality characteristic
- Set standards for each quality characteristic
- Do quality control with respect to the standards
- Find out the reasons that are hindering quality
- Make necessary improvements

Software Quality Assurance

- The purpose of QA activity is to enforce standards and techniques to improve the development process and prevent the previous faults from ever occurring. A good QA activity enforces good software engineering practices which help to produce good quality software.

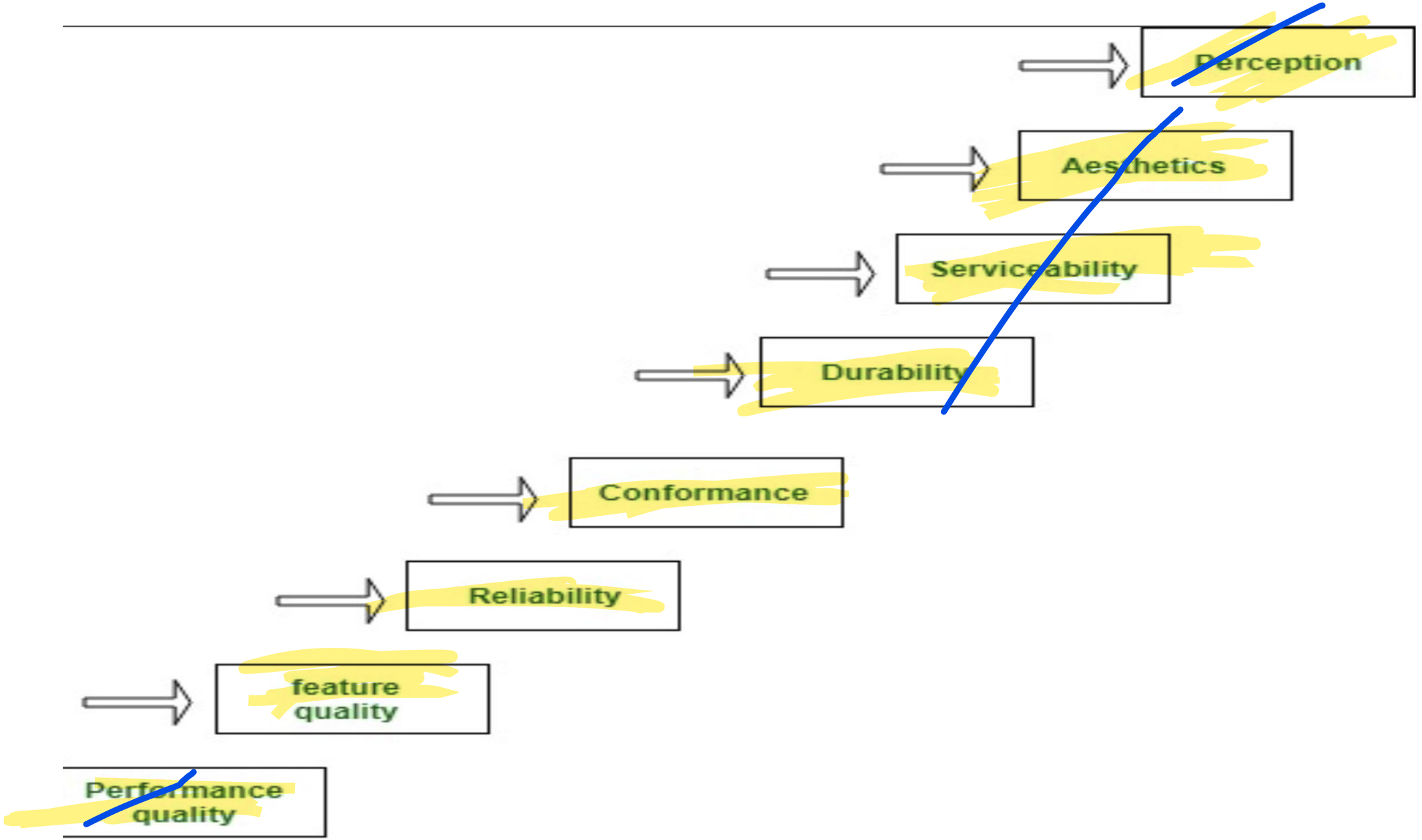
- The QA group monitors and guides throughout the software development life cycle. This is a defect prevention technique and concentrates on the process of the software development.

- Examples are reviews, audits, etc.

Garvin’s Dimensions Of Quality

- David Garvin suggests that quality ought to be thought-about by taking a third-dimensional read point that begins with an assessment of correspondence and terminates with a transcendental (aesthetic) view. Though Garvin’s 8 dimensions of quality weren’t developed specifically for the software system, they’ll be applied once software system quality is taken into account.

- Eight dimensions of product quality management will be used at a strategic level to investigate quality characteristics. The idea was outlined by David A. Garvin, formerly C. Roland Christensen academician of Business Administration at Harvard grad school (died thirty Gregorian calendar month 2017).



Garvin's Dimentions of Quality

Garvin’s Dimensions Of Quality

- **Performance Quality-**
  Will the software system deliver all content, functions, and options that are such as a part of the necessities model during a method that gives worth to the tip user?

  - basic operating characteristics

- **Feature Quality-**
  Does the software system offer options that surprise and delight first-time finish users?

    - Extra items added to basic features

Garvin’s Dimensions Of Quality

- **Reliability:**
  Will the software system deliver all options and capability while not failure?
  Is it obtainable once it’s needed?
  Will it deliver practicality that’s error-free?

  - Probability product will operate over time

- **Conformance:**
  Will the software system adjust to native and external software standards that are relevant to the application?
  Will it conform to the factual style and writing conventions? as an example, will the computer program conform to accepted style rules for menu choice or knowledge input?

  - Meeting pre established standards

Garvin’s Dimensions Of Quality

- **Durability:**
  Will the software system be maintained (changed) or corrected (debugged) while not the accidental generation of unintentional facet effects? can changes cause the error rate or responsibility to degrade with time?

- Lifespan before replacement

- **Serviceability:**
  Will the software system be maintained (changed) or corrected (debugged) in a tolerably short time period?
  Will support employees acquire all data they have to create changes or correct defects?Stephen A. Douglas Adams makes a wry comment that appears acceptable here: “The distinction between one thing that may get it wrong and something that can’t probably go wrong is that once something that can’t possibly go wrong goes wrong it always seems to be not possible to urge at or repair.”

- Ease of getting repaired, speed and competence of rapairs

Garvin’s Dimensions Of Quality

- **Aesthetics:**
  There’s no doubt that every folk includes a totally different and really subjective vision of what’s aesthetic.
  And yet, most folks would agree that an aesthetic entity includes a sure class, a novel flow, and a clear “presence” that are arduous to quantify however are evident still. The aesthetic software system has these characteristics.

- **Perception:**
  In some things, you’ve got a collection of prejudices which will influence your perception of quality. as an example, if you’re introduced to a software product that was engineered by a seller United Nations agency has created poor quality within the past, your guard is raised and your perception of the present software product quality may be influenced negatively.Similarly, if a seller has a wonderful name, you will understand quality, even once it doesn’t very exist.

What is Software Quality Assurance?

- **Software quality assurance (SQA)** is a process which assures that all software engineering processes, methods, activities and work items are monitored and comply against the defined standards. These defined standards could be one or a combination of any like ISO 9000, CMMI model, ISO15504, etc.

- SQA incorporates all software development processes starting from defining requirements to coding until release. Its prime goal is to ensure quality.

-

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 a 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.

SQA Activities

- **#3) Apply software Engineering Techniques:**

- Applying some software engineering techniques aids a software designer in achieving high-quality specification. 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.

SQA Activities

- **#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 regarding 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.

SQA Activities

- **#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 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.

SQA Activities

- **#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 which allows managers and developers to observe the activities and proposed changes from the beginning till the end of SDLC and initiate corrective action wherever required.

SQA Activities

- **#9) Performing SQA Audits:**
- The SQA audit inspects the entire actual SDLC process followed by comparing it against the established process.
- It also checks whatever reported by the team in the status reports were 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

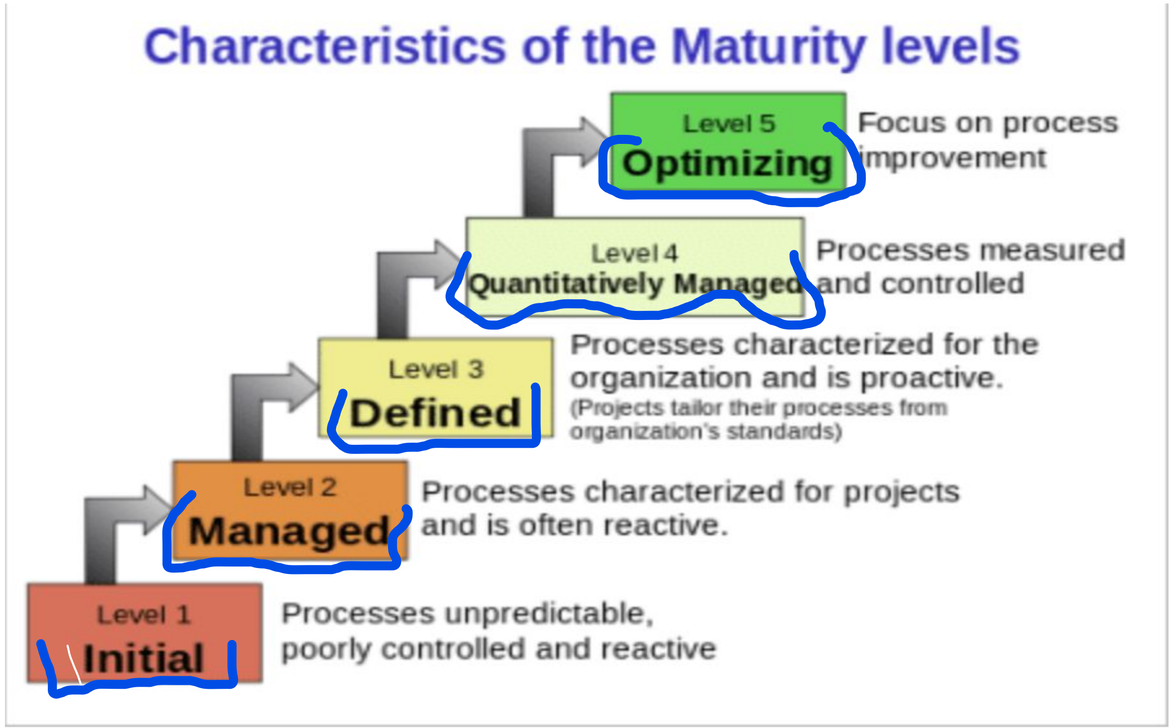
- In general, SQA may demand conformance to one or more standards.
- **ISO 9000:** This standard is based on seven quality management principles which help the organizations to ensure that their products or services are aligned with the customer needs'.

Software Quality Assurance Standards

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

Software Quality Assurance Standards

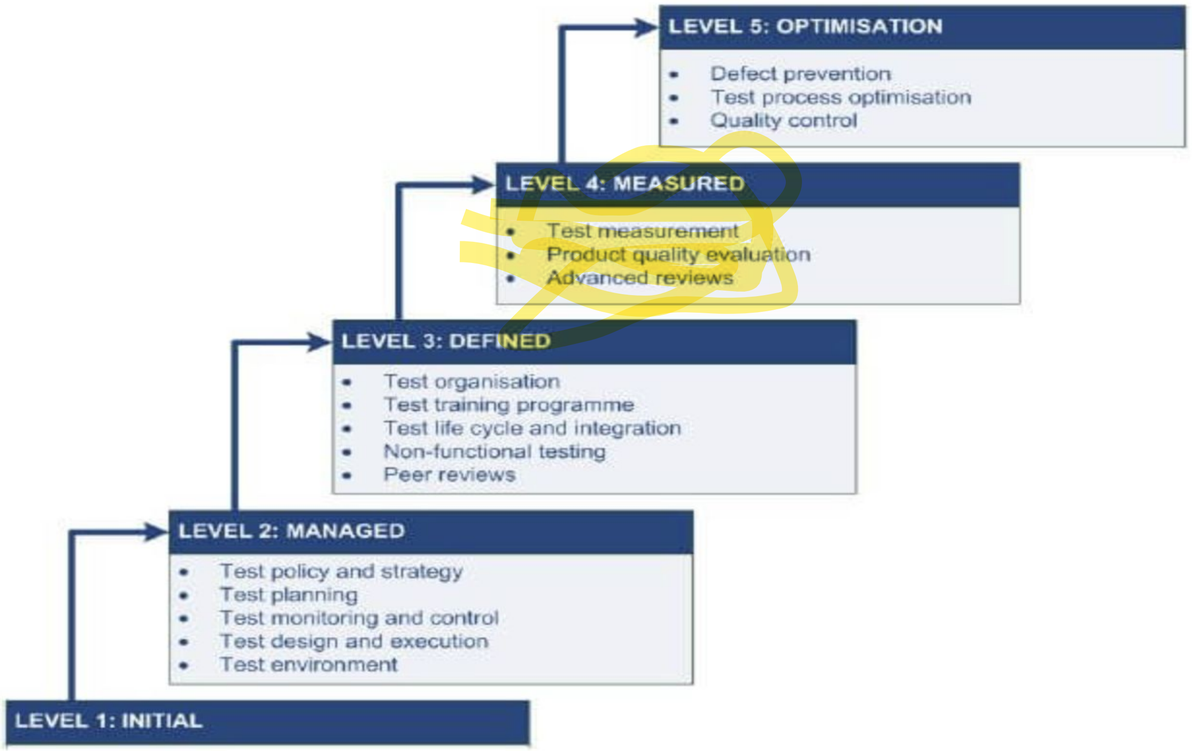
- **CMMI level:** CMMI stands for **Capability maturity model Integration**. This model was originated in software engineering. It can be employed to direct process improvement throughout a project, department, or an entire organization.
-



Software Quality Assurance Standards

- **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:**



Elements of Software Quality Assurance

- **There are 10 essential elements of SQA which are enlisted below:**
- Software engineering Standards
- Technical reviews and audits
- Software Testing for quality control
- Error collection and analysis
- Change management
- Educational programs
- Vendor management
- Security management
- Safety
- Risk management

Software quality metrics

Software metrics can be classified into three categories –

- **Product metrics** – Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
- **Process metrics** – These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics** – This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Software quality metrics

- **Software quality metrics** are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.
- Software quality metrics can be further divided into three categories –
- Product quality metrics
- In-process quality metrics
- Maintenance quality metrics

Software quality metrics

- Product Quality Metrics
- This metrics include the following –
  - Mean Time to Failure
  - Defect Density
  - Customer Problems
  - Customer Satisfaction

Software quality metrics

Product Quality Metrics

- Mean Time to Failure
  - It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.
- Defect Density
  - It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.
- Customer Problems
  - It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

Software quality metrics

Product Quality Metrics

- The problems metric is usually expressed in terms of **Problems per User-Month (PUM)**.
- PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period
- Where,
- Number of license-month of the software = Number of install license of the software × Number of months in the calculation period
- PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

Software quality metrics

Product Quality Metrics

- Customer Satisfaction
  - Customer satisfaction is often measured by customer survey data through the five-point scale –
    - Very satisfied
    - Satisfied
    - Neutral
    - Dissatisfied
    - Very dissatisfied

Software quality metrics

Product Quality Metrics

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

Usually, this percent satisfaction is used.

Software quality metrics

In-process Quality Metrics

- In-process Quality Metrics
- In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes –
  - Defect density during machine testing
  - Defect arrival pattern during machine testing
  - Phase-based defect removal pattern
  - Defect removal effectiveness

Software quality metrics

In-process Quality Metrics

- Defect density during machine testing
  - Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.
  - This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.
- Defect arrival pattern during machine testing
  - The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field.

Software quality metrics

In-process Quality Metrics

- Phase-based defect removal pattern
  - This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.
  - Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.
  - With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

Software quality metrics

In-process Quality Metrics

- Defect removal effectiveness

- It can be defined as follows –

$$DRE = \frac{Defect\ removed\ during\ a\ development\ phase}{Defects\ latent\ in\ the\ product} \times 100\%$$

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called **early defect removal** when used for the front-end and **phase effectiveness** for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

Software quality metrics

Maintenance Quality Metrics

- Maintenance Quality Metrics

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

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

Software quality metrics

Maintenance Quality Metrics

- Fix backlog and backlog management index
  - Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.
  - Backlog Management Index (BMI) is used to manage the backlog of open and unresolved

$$BMI = \frac{Number\ of\ problems\ closed\ during\ the\ month}{Number\ of\ problems\ arrived\ during\ the\ month} \times 100\%$$

Software quality metrics

Maintenance Quality Metrics

- Fix response time and fix responsiveness
  - The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.
  - The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.
  - Percent delinquent fixes- It is calculated as follows

$$Percent\ Delinquent\ Fixes = \frac{Number\ of\ fixes\ that\ exceeded\ the\ response\ time\ criteria\ by\ ceverity\ level}{Number\ of\ fixes\ delivered\ in\ a\ specified\ time} \times 100\%$$

Software quality metrics

Maintenance Quality Metrics

- Fix Quality
  - Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.
  - A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered. The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.
-

What is Software Quality Management?

- Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

- Software Quality management should be independent of project management to ensure independence of cost and schedule adherences. It directly affects the process quality and indirectly affects the product quality.

Software Quality Management

- Activities of Software Quality Management:
- **Quality Assurance -** QA aims at developing Organizational procedures and standards for quality at Organizational level.
- **Quality Planning -** Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.
- **Quality Control -** Ensure that best practices and standards are followed by the software development team to produce quality products.

- What is Quality Assurance?

- Quality Assurance is defined as the auditing and reporting procedures used to provide the stakeholders with data needed to make well-informed decisions.

- It is the Degree to which a system meets specified requirements and customer expectations. It is also monitoring the processes and products throughout the SDLC.

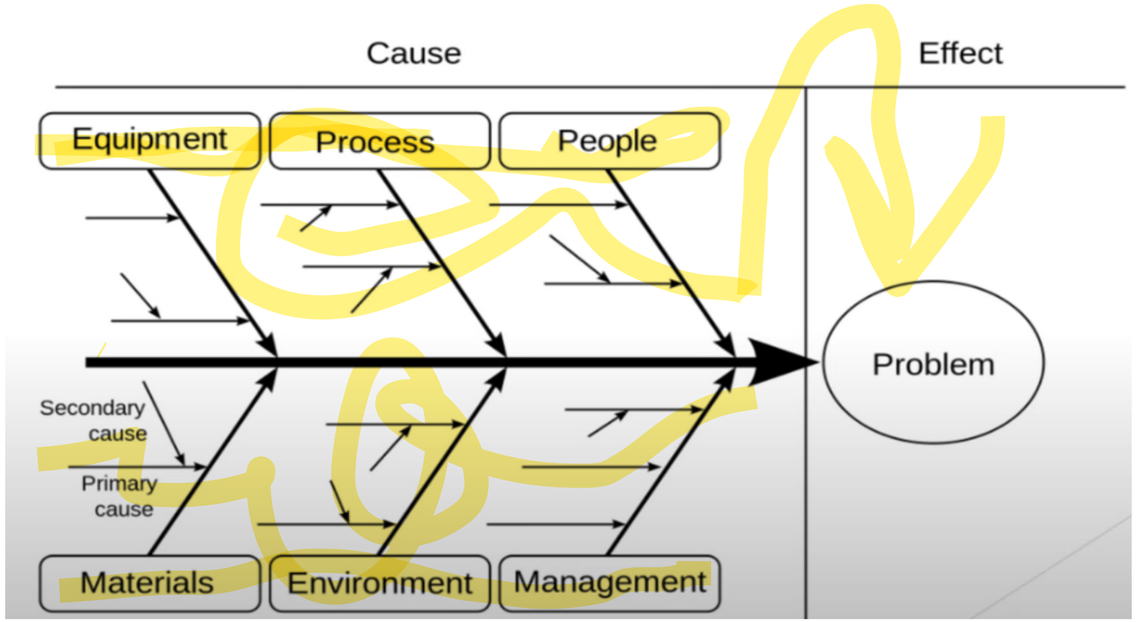
Quality Tools

- 7 QC Tools are basic Quality Control Tools which helps in solving quality issues through data collection, Analysis of data, identification of cause and measuring results.

Quality Tools

- **Cause-and-effect diagram** (also called Ishikawa or fishbone diagrams): Identifies many possible causes for an effect or problem and sorts ideas into useful categories.

- A fishbone diagram's causes and subcauses are usually grouped into six main groups, including measurements, materials, personnel, environment, methods, and machines. These categories can help you identify the probable source of your problem while keeping your diagram structured and orderly.



Quality Tools

- **Pareto chart**: A bar graph that shows which factors are more significant.

- As a quality control tool, the Pareto chart operates according to the 80-20 rule. This rule assumes that in any process, 80% of a process's or system's problems are caused by 20% of major factors, often referred to as the "vital few." The remaining 20% of problems are caused by 80% of minor factors.

- The goal of the Pareto chart is to highlight the relative importance of a variety of parameters, allowing you to identify and focus your efforts on the factors with the biggest impact on a specific part of a process or system.

Quality Tool

Pareto chart:

Quality Tools

- Scatter diagram: Graphs pairs of numerical data, one variable on each axis, to look for a relationship.

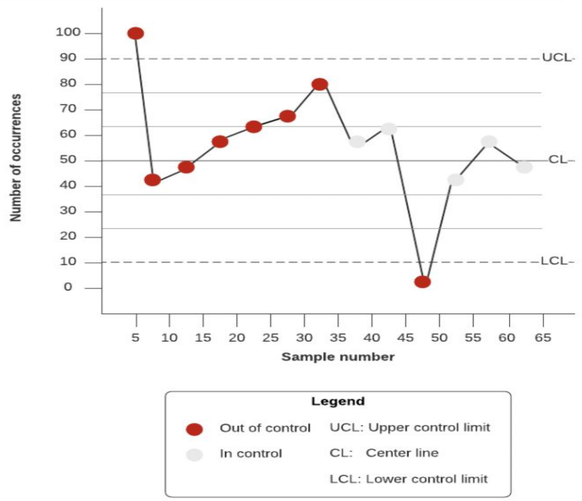
- Out of the seven quality tools, the scatter diagram is most useful in depicting the relationship between two variables, which is ideal for quality assurance professionals trying to identify cause and effect relationships.

- With dependent values on the diagram's Y-axis and independent values on the X-axis, each dot represents a common intersection point. When joined, these dots can highlight the relationship between the two variables. The stronger the correlation in your diagram, the stronger the relationship between variables.

- Scatter diagrams can prove useful as a quality control tool when used to define relationships between quality defects and possible causes such as environment, activity, personnel, and other variables. Once the relationship between a particular defect and its cause has been established, you can implement focused solutions with (hopefully) better outcomes.

Quality Tools

Control chart



Action plan

Action	Owner	Start date	End date	Required resources	Status
Analyze chart	Sam	3/15	3/20	Control chart, process analyst	In progress
Investigate the cause for the out-of-control metric	Mark	3/15	3/22	Documentation of expected process, process manger	In progress
Determine if changes need to be made to the process	Mark	3/16	3/22	Process manager	Waiting

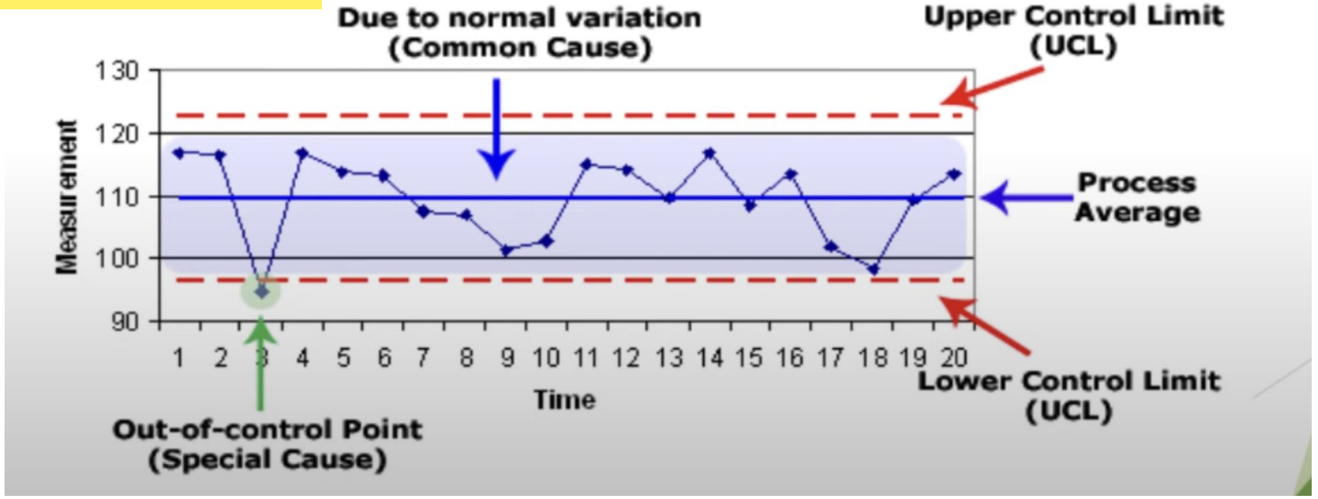
- **Control chart**: Graph used to study how a process changes over time. Comparing current data to historical control limits leads to conclusions about whether the process variation is consistent (in control) or is unpredictable (out of control, affected by special causes of variation).

- this quality improvement tool can help quality assurance professionals determine whether or not a process is stable and predictable, making it easy for you to identify factors that might lead to variations or defects.

Quality Tools

- Control charts use a central line to depict an average or mean, as well as an upper and lower line to depict upper and lower control limits based on historical data. By comparing historical data to data collected from your current process, you can determine whether your current process is controlled or affected by specific variations.

- Using a control chart can save your organization time and money by predicting process performance, particularly in terms of what your customer or organization expects in your final product



Quality Tools

- **Flowcharts** (**Stratification**:) as one of the seven basic QC tools. Flowcharts are most commonly used to document organizational structures and process flows, making them ideal for identifying bottlenecks and unnecessary steps within your process or system.

- Mapping out your current process can help you to more effectively pinpoint which activities are completed when and by whom, how processes flow from one department or task to another, and which steps can be eliminated to streamline your process.

-

Quality Tools

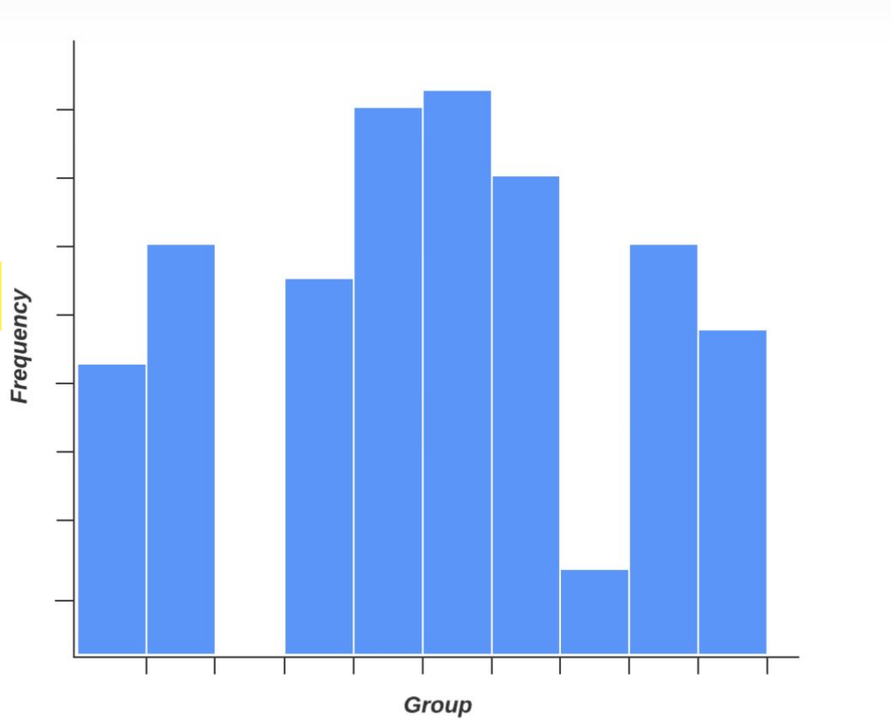
- **Histogram**: The most commonly used graph for showing frequency distributions, or how often each different value in a set of data occurs.

- Quality professionals are often tasked with analyzing and interpreting the behavior of different groups of data in an effort to manage quality. This is where quality control tools like the histogram come into play.

Quality Tools

- The histogram can help you represent frequency distribution of data clearly and concisely amongst different groups of a sample, allowing you to quickly and easily identify areas of improvement within your processes. With a structure similar to a bar graph, each bar within a histogram represents a group, while the height of the bar represents the frequency of data within that group.

- Histograms are particularly helpful when breaking down the frequency of your data into categories such as age, days of the week, physical measurements, or any other category that can be listed in chronological or numerical order.



Quality Tools

- **Check sheet**: A structured, prepared form for collecting and analyzing data; a generic tool that can be adapted for a wide variety of purposes.
- Check sheets can be used to collect quantitative or qualitative data. When used to collect quantitative data, they can be called a tally sheet.
- A check sheet collects data in the form of check or tally marks that indicate how many times a particular value has occurred, allowing you to quickly zero in on defects or errors within your process or product, defect patterns, and even causes of specific defects.

Quality Tools

- With its simple setup and easy-to-read graphics, check sheets make it easy to record preliminary frequency distribution data when measuring out processes. This particular graphic can be used as a preliminary data collection tool when creating histograms, bar graphs, and other quality tools.

Total Quality Management

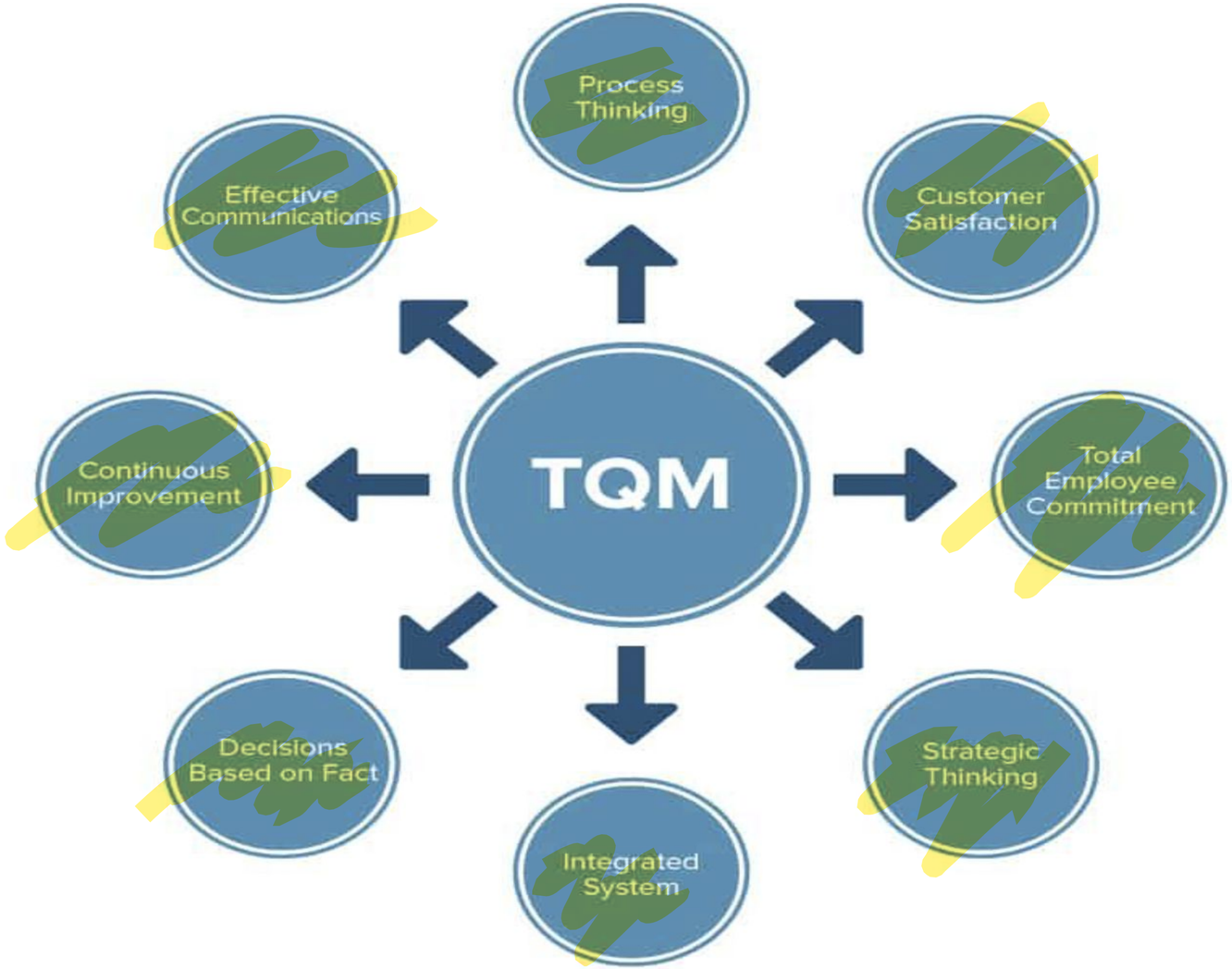
Total Quality Management (TQM) is a management approach that seeks to **provide long-term success** by providing unparalleled customer satisfaction through the constant delivery of quality IT services.

To properly execute on TQM methods, the entire organization needs to operate as a single unit in the pursuit of excellence.

- **Why Is Total Quality Management Important to an Organization?**
- Implementing a TQM philosophy can help an organization:
  - Ensure customer satisfaction and customer loyalty
  - Ensure increased revenues and higher productivity
  - Reduce waste and inventory
  - Improve design
  - Adapt to changing markets and regulatory environments
  - Increase productivity
  - Enhance market image
  - Eliminate defects and waste
  - Increase job security
  - Improve employee morale

Total Quality Management

Total Quality Management Principles



Total Quality Management

Total Quality Management Principles

- **Customer first.** TQM's first and foremost pillar of success is an unwavering focus on the customer's experience in all interactions with the organization. From first contact through purchase and continued support, the customer should always be the main priority.

- **Employee ownership.** TQM requires the involvement of every team member to ensure that complete quality control is offered at every level. TQM doesn't focus on a single department because the goal is to provide customers with a great experience from every level of the organization.

- **Process-based.** TQM focuses on the creation and implementation of processes that provide organizations with the ability to find success and repeat it. Quantifying success and defining the steps taken to get there are essential for successful implementation of TQM.

Total Quality Management

Total Quality Management Principles

- **System integration.** TQM strategies revolve around leveraging every asset available to the company. This is best achieved through system integrations that combine disparate parts of the organization into a single, well-oiled machine working in complete synergy.

- **Communication.** TQM requires every team member to be at their best and to function as a value-adding member of that team. This means communication and transparency is a core tenet of successful TQM practices.

Total Quality Management

- **Data-driven.** TQM doesn't employ guesswork. Instead, data is leveraged for the improvement of the organization and decisions are made based on quantifiable facts.

- **Constant improvement.** TQM isn't a one and done process. Perfection is impossible, so it must always be pursued to get the organization as close as possible to it.

- These pillars of TQM act as a framework for every decision made within the methodology. Whenever your organization feels lost, the TQM ideals are your guiding stars for righting course.

How to implement TQM

- The first step for implementing any new system is an honest assessment of the organization as it is today.
- Implementation of TQM is something that has to be applied to the current structure of the organization; there is no step-by-step guide that will tell you how to do it for your business. Each business is unique and requires its own approach, but the core tenets of TQM can guide each decision.
-

Total Quality Management

How Do You Implement Total Quality Management?

- PDCA lies at the core of many 20th century quality efforts. PDCA began in the 1920s as a conception by engineer and statistician Walter Shewhart. It was originally called PDSA (plan, do, study, act).

- Widely disseminated by Deming, who referred to it as the Shewhart cycle, it is now often referred to as the Deming cycle.

Total Quality Management

How Do You Implement Total Quality Management?

- **Plan:** The planning phase is the most important. That's where management, along with the associates, identify the problems to see what really needs to be addressed — the day-to-day things that may be happening on the productivity side that management is not aware of. So they're trying to determine a root cause. Sometimes, employees do research or high-level tracking to narrow down where an issue may originate.

- **Do:** The doing phase is the solution phase. Strategies are developed to try to fix those problems identified in the planning phase. Employees may implement solutions and if a solution doesn't appear to work, it's back to the drawing board. In contrast to Six Sigma, it's less about measuring gains and more about whether the employees judge the solution to be working.

-

Total Quality Management

How Do You Implement Total Quality Management?

- **Check:** The checking phase is the before and after. So after you've made these changes, you see how they're doing.

- **Act:** The acting phase is the presentation or the documentation of the results to let everybody know, 'Hey, here's how we were doing it. Here's how it is now. This is the new way, and this is what this should address going forward.'"

How to implement TQM

- **Emphasize customer satisfaction**

- Creating an emphasis on customer satisfaction will change the way departments think about their duties. If something they are doing isn't aiding in the improvement of the quality of the product or increasing the customer's experience, they aren't headed in the right direction.

- Each employee should take ownership of their role and be ready to consider ways in which they can improve their own department and outputs.

How to implement TQM

- Communicate with everyone

- Communication throughout the organization is essential for educating everyone about the changes that are coming while also providing an avenue for receiving feedback.

- As they say, communication is a two-way street. Employees will have a much easier time establishing a feeling of ownership over the process when they know their voice is heard and they had a hand in guiding the changes.

-

How to implement TQM

- Manage errors

- One of the most important aspects of delivering quality is managing errors. No matter how focused everyone is on driving quality, IT organizations will always run into one issue or another.

- Creating processes that mitigate issues is essential for TQM success. Errors should be addressed and dealt with as quickly as possible of course, but they should also be recorded and tracked.

Total Quality Management

- TQM is everyone’s responsibility

- TQM requires that all parties take ownership of the part they play and this applies equally to admitting fault and giving out praise. A focus on improving the quality of products and services requires accountability. Learning to snatch victory from the jaws of defeat is the pursuit of quality management when it comes to dealing with incidents and outages.

Total Quality Management

Key Players in Total Quality Management

- **The Key Players in Total Quality Management: Customers, Suppliers, and Employees**

- To achieve success with a total quality management program or any other improvement methodology, managers must understand the quality goals for their product or company. They must then communicate those goals, in addition to the benefits of TQM, to the company, as employees play a vital role by contributing their intimate, day-to-day knowledge of product creation and processes.

Total Quality Management

Key Players in Total Quality Management

- TQM is a philosophy that values comprehensiveness. Therefore, suppliers are a crucial part of TQM execution. Companies must vet new suppliers and regularly audit existing suppliers to guarantee that materials meet standards. Communication with suppliers about TQM goals is also essential.

- Customers are the most significant part of the TQM equation. After all, they're the reason for TQM's existence. Aside from the obvious feedback the sales team provides, customers — product or service users — give information about what they want from the deliverable, whether that deliverable is tangible or a service.

Total Quality Management

- **Benefits of Total Quality Management**
- The benefits arising from the implementation of a Total Quality Management in an organization are:
  - This will increase the awareness of quality culture within the organization.
  - A special emphasis on teamwork will be achieved.
  - TQM will lead to a commitment towards continuous improvement.

Software Testing Tools

- Software testing tools are required for the betterment of the application or software.
- With the help of testing tools, we can improve our software performance, deliver a high-quality product, and reduce the duration of testing, which is spent on manual efforts.

- The software testing tools can be divided into the following:
- **Test management tool**
- **Bug tracking tool**
- **Automated testing tool**
- **Performance testing tool**
- **Cross-browser testing tool**
- **Integration testing tool**
- **Unit testing tool**
- **Mobile/android testing tool**
- **GUI testing tool**
- **Security testing tool**

Software testing tools

- **Test management tool**

- Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain manual testing as well.

- Example-1)TestLink -This is one of the very few open-source test management tools that are available for use in the market. It is a web-based tool with typical features like test case creation abd maintenance, test suite management, test runs, tracking bugs, reports, and integration with some common issue trackers.

- **Bug tracking tool**

- The defect tracking tool is used to keep track of the bug fixes and ensure the delivery of a quality product. This tool can help us to find the bugs in the testing stage so that we can get the defect-free data in the production server. With the help of these tools, the end-users can allow reporting the bugs and issues directly on their applications.

- Example-1) Backlog is a popular bug and project tracking tool in one platform. It's easy for anyone to report bugs and keep track of a full history of issue updates and status changes. Development teams use Backlog to work with other teams for enhanced team collaboration and high-quality project delivery.

Software testing tools

- **Automation testing tool**

  - This type of tool is used to enhance the productivity of the product and improve the accuracy. We can reduce the time and cost of the application by writing some test scripts in any programming language.

  - **Example- Selenium** is a very well-known tool when it comes to testing automation. It allows its users to write scripts in a lot of different languages, including Java, C#, Python, Perl, and Ruby. This tool also runs in several operating systems and browsers

Software testing tools

- **Performance testing tool**
  - Performance or Load testing tools are used to check the load, stability, and scalability of the application. When n-number of the users using the application at the same time, and if the application gets crashed because of the immense load, to get through this type of issue, we need load testing tools.

The attributes of Performance Testing include:

- **Speed** – It determines whether the application responds quickly.
- **Scalability** – It determines maximum user load the software application can handle.
- **Stability** – It determines if the application is stable under varying loads.

Software testing tools

- **Example-** 1) LoadNinja -It allows you to create scriptless sophisticated load tests and reduces testing time by half. It also replaces load emulators with real browsers and gets actionable, browser-based metrics at ninja speed. LoadNinja empowers teams to increase their test coverage without giving up on the quality by removing the tedious efforts of dynamic correlation, script translation, and script scrubbing.

Software testing tools

- **Cross-browser testing tool**
  - This type of tool is used when we need to compare a web application in the various web browser platforms. It is an important part when we are developing a project. With the help of these tools, we will ensure the consistent behavior of the application in multiple devices, browsers, and platforms.

- Example-LambdaTest is a cloud-based cross-browser testing platform that helps you perform compatibility testing on your web app or websites easily. You can run automated Selenium scripts on LambdaTest's scalable cloud grid, or can even perform live interactive testing on real browser environments.

Software testing tools

- **Integration testing tool-** This type of tool is used to test the interface between modules and find the critical bugs that are happened because of the different modules and ensuring that all the modules are working as per the client requirements.

Example-1) Citrus -: It is the most commonly used integration testing tool, which is a test framework and written in the Java programming language. It is used to take the request and respond to both server-side and client-side.

2) TESSY- It is an essential tool for integration testing that is used to execute the integration and unit testing for the embedded software. It will take care of the whole test organization along with the requirements, traceability, test management, and the coverage measurement.

Software testing tools

- **Unit testing tool**
  - This testing tool is used to help the programmers to improve their code quality, and with the help of these tools, they can reduce the time of code and the overall cost of the software.
- Example - 1) Junit: Junit is a free to use testing tool used for Java programming language. It provides assertions to identify test method. This tool test data first and then inserted in the piece of code.
- 2) NUnit: NUnit is widely used unit-testing framework use for all .net languages.

Software testing tools

- **Mobile/android testing tool**
  - We can use this type of tool when we are testing any mobile application. Some of the tools are open-source, and some of the tools are licensed. Each tool has its functionality and features.
  - Example-1) Calabash-Calabash is a mobile application testing framework that works with multiple languages
  - 2) Selendroid Selendroid is also known as selenium for mobile apps for Android. Testers can do native and hybrid mobile application testing using Selendroid.
  - 3) obotium Robotium is a popular open-source tool dedicated for testing android applications only.

Software testing tools

- **GUI testing tool**
  - GUI testing tool is used to test the User interface of the application because a proper GUI (graphical user interface) is always useful to grab the user's attention. These type of tools will help to find the loopholes in the application's design and makes its better.
  - Example-Eggplant is a GUI test automation tool, which is developed by Test Plant. It is a licensed tool. To execute the end-to-end testing process, eggplant can be integrated into the micro focus quality center, Jenkins, and IBM rotational quality manager. It will use the two-system model, where the first one contains the controller machine where scripts are written and executed, and another one is SUT (system under test) that runs on the VNC server.

Software testing tools

- **Security testing tool**

- The security testing tool is used to ensure the security of the software and check for the security leakage. If any security loophole is there, it could be fixed at the early stage of the product. We need this type of the tool when the software has encoded the security code which is not accessible by the unauthorized users.

- Example-1) Wfuzz- Developed in Python, Wfuzz is popularly used for brute-forcing web applications. The open-source security testing tool has no GUI interface and is usable only via command line.

- 2) Wapiti- One of the leading web application security testing tools, Wapiti is a free of cost, open source project from SourceForge and devloop. In order to check web applications for security vulnerabilities, Wapiti performs black box testing. As it is a command-line application, it is important to have a knowledge of various commands

Reference

- 1. S. Naik, P. Tripathy,” Software Testing and Quality Assurance”, Wiley, 2010/ Latest Edition
- 2. Yogesh singh, “Softwrae Testing”
- 3. Web resources