UNIT VI

CHAPTER 6

Testing Framework

University Prescribed Syllabus for the Academic Year 2022-2023

Testing Framework : Software Quality, Software Quality Dilemma, Achieving Software Quality, Software Quality Assurance. Elements of SQA, SQA Tasks, Goals and Metrics, Formal Approaches to SQA, Statistical Software Quality Assurance, Six Sigma for Software Engineering, ISO 9000 Quality Standards, SQA Plan, Total Quality Management, Product Quality Metrics, In process Quality Metrics, Software maintenance, Ishikawa's 7 basic tools, Flow Chart, Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. Defect Removal Effectiveness and Process.

6.1 INTRODUCTION TO SOFTWARE QUALITY

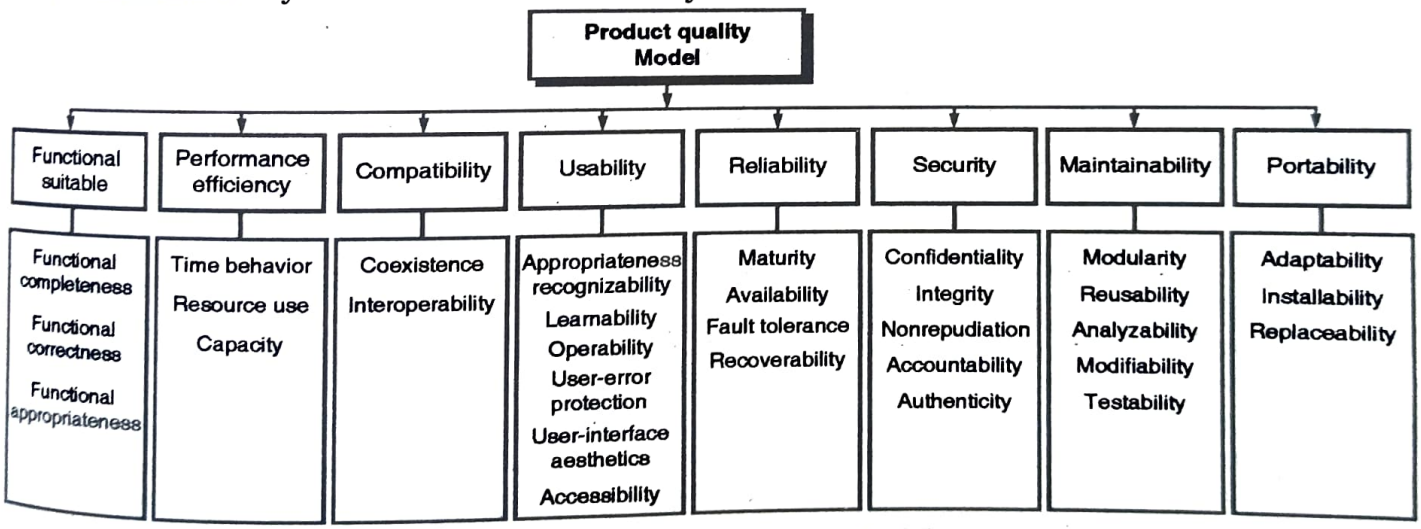
UQ. What Is Software Quality ?
GQ. Explain Software Quality Management ?

SPPU - Aug.19(Insem), May 18 (Endsem)

- **Software quality** is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality: defect management and quality attributes.
- A software defect can be regarded as any failure to address end-user requirements. Common defects include missed or misunderstood requirements and errors in design, functional logic, data relationships, process timing, validity checking, and coding errors.
- The software defect management approach is based on counting and managing defects. Defects are commonly categorized by severity, and the numbers in each category are used for planning. More mature software development organizations use tools, such as defect leakage matrices (for counting the numbers of defects that pass through development phases prior to detection) and control charts, to measure and improve development process capability.

Software Quality characteristics

- This approach to software quality is best exemplified by fixed quality models, such as ISO/IEC 25010:2011. This standard describes a hierarchy of eight quality characteristics, each composed of sub-characteristics :
  1. Functional suitability
  2. Reliability
  3. Operability
  4. Performance efficiency
  5. Security
  6. Compatibility
  7. Maintainability
  8. Transferability



(1E1)**Fig. 6.1.1 : Software Quality Model**

- Additionally, the standard defines a quality-in-use model composed of five characteristics:
  1. Effectiveness
  2. Efficiency
  3. Satisfaction
  4. Safety
  5. Usability
- A fixed software quality model is often helpful for considering an overall understanding of software quality. In practice, the relative importance of particular software characteristics typically depends on software domain, product type, and intended usage. Thus, software characteristics should be defined for, and used to guide the development of, each product.

- Quality function deployment provides a process for developing products based on characteristics derived from user needs.

6.1.1 Brief History of Software Quality Dilemma

UQ. Brief History of Software Quality Dilemma ? SPPU - Aug.18 (Insem)

UQ. What is Software Quality Dilemma ? SPPU - Nov./Dec. 19 (Endsem)

- Software quality dilemma is a situation where there is confusion regarding what should we prioritize: a good quality work or a fast paced work.
- In software development, many a times there will be deadlines to achieve, in such cases software quality dilemma is bound to occur. A developer would have to choose between writing and optimized and well commented code or just get the job done without proper optimized or reviews.
- In same lines, many companies have to decide between regular reviews and expert opinions of a product for good software quality or bypass them to meet budgets and deadlines.
- Quality is a complex term comprising a lot of meanings in a single word. Saying the quality, we mean the product lacks defects, is vulnerable-free, it meets our requirements.
- The good is made durable and reliable enough to serve or perform the whole lifecycle smoothly. Yet, it is not the end, the quality also ensures that the same product is effective and efficient. Quite an exciting scope of senses, indeed.
- As you may guess, in this post, we are going to write about quality, not in its general sense, but about quality assurance in software development area.
- Software quality management is intended to ensure the business value of delivered software, meaning an application conforms to requirements for its reliability, efficiency, security, maintainability, and size. Moreover, all the parameters correlate with risk and cost management.
- Here come three services to achieve the software quality; they are quality assurance, quality control, and testing.
- These processes are often mixed up and used to substitute each other in general talks, but they are not the same. Let us try to find the clear definition.
- **Quality assurance (QA)** is the process that starts along with the software development. It is intended to prevent issues, and it describes what customer requirements are testable and how. Quality assurance defines the standards, methods and sometimes even models for each stage of a development project.
- It deals with every phase: software design, coding, user experience, configuration, testing, release and even product integration.
- As well as everything else that is not mentioned in the list above. The idea is that quality assurance guides the whole project development projects to guard against defects and prevent issues.
- **Quality control (QC)** is the scope of means to check the product, find defects, fix bugs and correct the issues, improve a performance of the software. Usually, quality control service is applied to a ready product. It is also quality control that is aimed to check whether a software performs the way it should and does not execute the functions it is not supposed to.
- Testing is aimed to investigate how software works. Engineers set up a testing plan, perform tests, analyse their results, and confirm tests' validity and their verification. Of course, experts create a lot of reports on the work they have done. The test can be automated, or manual; analysis - quantitative and qualitative, both software structure and its functions can be inspected to get the product that meets customer's requirements and users' expectations.
- To some extent, quality assurance overlaps with quality control and testing, but they should do. They are all software quality management services, which are not about breaking a software or bug finding.

- The main goal of QA is to prevent errors and avoid outage and to save your budget, efforts, and resources at every moment of your software lifecycle.
- Protect your products, don't waste your money on unnecessary software maintenance or support, include quality assurance in the software development process.

6.2 ACHIEVING GOOD SOFTWARE QUALITY

UQ. What parameters required for achieving good Software Quality ? **SPPU - Nov. 19 (Endsem)**

- How you manage software quality has become a vital element of every stage of project management. Are you continually on the lookout for ways to improve your software quality that is not going to break the bank ? Excellent software quality will enable cost effectiveness and superior performance to deliver your projects.
- Finding ways to implement effective testing strategies at the earliest possible stage will help you detect and solve defects. Solving problems at the earliest stage of project management creates a win-win scenario. Increased efficiency results in better quality software and reduced costs. Conversely, poor software quality exacerbates problems and can become a time-consuming and expensive exercise.
- Instead of spending lengthy periods of time fire-fighting software issues, you can concentrate on delivering a quality project. To help you increase efficiency and excellence for your next project we will explain 11 effective methods to improve software quality. These methods are aimed to provide you with assistance so you can deliver your next project with peace of mind in how well your software will operate.
- This article will help you and your project team take a complete assured approach to software development.

1. **Test early and Test often with Automation**
   - To improve software quality, it is absolutely paramount to Test early and Test often. Early testing will ensure that any defects do not snowball into larger, more complicated issues. The bigger the defect, the more expensive it becomes to iron out any issues.
   - The earlier you get your testers involved, the better. It is recommended to involve testers early in the software design process to ensure that they remain on top of any problems or bugs as they crop up, and before the issues grow exponentially which generally makes it harder to debug.
   - Testing often requires a focus on early adoption of the right automated testing discipline. Start by automating non UI tests initially then slowly increasing coverage to UI based tests when the product stabilises. If your application utilises Webservices/APIs then automate these tests to ensure all your business rules and logic are tested.
   - This is an important time to work with your software developers to ensure automated testing is also introduced to your development teams, increasing testing coverage, accuracy and improving quality of the overall product.
   - A study published in the Journal of Information Technology Management has revealed that the cost to rectify a bug increases roughly 10 times with each passing stage of development.
   - **For Example :** An error that costs $100 to rectify in the business requirements stage would cost $1000 to rectify in the system requirements stage, $10,000 in the high-level design stage and $100,000 in the implementation stage.
2. **Implement quality controls from the beginning**
   - Testers can monitor quality controls and create awareness in partnership with developers to ensure standards are continually being met. Quality control starts from the beginning, which is an ongoing process throughout the delivery.

- A good relationship between testers and developers can help the project software strategy develop effectively. A systematic methodology in quality control can ensure that coding errors and bugs are dealt with effectively, following a structured process.

3. Echo the importance of quality assurance through the entire software development process

- We have identified how important testing is at the beginning of software development; however, the testing does not stop there. Quality assurance should be ever-present throughout the software development process.
- Quality Assurance is governance provided by the project team that instils confidence in the overall software quality. Assurance testing oversees and validates the processes used in order to deliver outcomes have been tracked and are functioning. Testing should be repeated as each development element is applied. Think of it as layering a cake. After every layer is added, the cake should be tasted and tested again.

4. Encourage innovations

- It is important that testing structures and quality measures are in place, however, there should always be room for innovation. A great way to allow for innovation is to automate testing where possible to minimise time spent on controls.
- Innovations are so important because they can lead to improvements in software quality that have the capability to transform how projects are delivered. Research and development (R&D) should be encouraged. Empower teams to explore, experiment and investigate continuously. Also, ensure that advancements in innovation are duly rewarded. They have the capacity to transcend your software quality and deliver projects with a competitive advantage over the competition.

5. Communication is key

- For any relationship to be successful whether it's personal or business, communication is key. To improve software quality it is important that all parties to the project have full information through fluid communication channels.
- Fluid communication can take many forms. It can be as simple as having clear, consistent KPIs that show how software quality is measured at every step of the development process. It is important that all team members, regardless of seniority have access to KPIs to keep the entire team on the same page. Another important aspect of fluid communication is that all parties have the opportunity to provide feedback to the team to ensure that all expectations are being met.
- It is also important to keep all stakeholders in the loop and not isolate team members from the vendors or end user of the software. Isolation can cause rifts and can often mean that the project is delayed or may not deliver on the goals expected by senior management.

6. Plan for a changeable environment

- Software contains so many variables and is in continuous evolution. It relies on several different external factors such as web browsers, hardware, libraries, and operating systems.
- These constant external factors mean that software development must be consistently monitored using checks and balances to certify that it remains in stride with its immediate environment.
- It is important to acknowledge that software is interdependent on these external factors. Accepting this interdependence means that you can plan ahead. It allows you to have the software quality tested, at each step of the process, against external variables, to see how it holds up. The end result is that you will prevent software dissonance and maintain software quality.

7. Take the attitude of creating products not projects

- This step is a reflection of the attitude of your team. Creating a project indicates to your team that you are producing a finite outcome. However, we are well aware that software is changeable. If you produce a finite outcome, before long the software quality will not stand up against its environment.

- Instead, if your team takes the mind-set that they are creating a product it is more likely that they will deliver software quality that is adaptable to change and can stand the test of time. Focus on delivering continuous small progressions rather than one final end project and your team will deliver an increase in quality.

8. **Have a risk register**
   - A risk register is a fantastic management tool to manage risks. A risk register is more synonymous with financial auditing; however it is still a vital element in software development.
   - A risk register will provide everybody aligned on a software project a list of clearly identified risks and then assess them in regards to the importance of delivering the project. A risk register works well for software quality because its creation actively leads to risk mitigation.

A software development risk register must :

- Describe the risk.
- Recognise when the risk was identified.
- Acknowledge the chance of the risk occurring and its mitigation.
- Understand the severity of the impact of the risk.
- Identify who assesses and actions the risk.
- Relays the response if the risk materialises.
- Gives the status of each risk.
- Measures the negative impact of each risk.
- Prioritises the risks ranked on probability and gravity.

9. **Producing software quality requires long-term thinking and strategy**

Long-term thinking produces software quality because decisions are made to satisfy lasting issues. Here are the advantages of long-term thinking for producing software quality:

1. Doing it right first means you don't have to spend time doing it over.
2. Placing as much importance on architecture and design as coding ensures the veracity of your project.
3. Creating coding standards with long-term vision eliminates unnecessary mistakes.
4. Effective peer review ensures errors are minimised even though it may seem time-consuming at that particular moment.
5. Testing often allows you to plan further ahead with certainty that errors and bugs have been fixed.
6. Project leaders need to ensure that short-term gains and immediate gratification do not compromise long-term strategy. Effective planning will make sure all stakeholders prioritise software quality.

10. **Outline your deliverables**
    - From the outset of your project it is imperative that your team outline what they are going to deliver. A clear and concise plan of what the project will deliver helps ensure there is an emphasis on quality from the outset.
    - It also ensures that budgets, resources, and time are aligned correctly to deliver quality. Without clear deliverables it is likely shortcuts will be taken to meet budgets and deadlines. Ultimately, this will compromise the quality of the software delivered at the end of the project.

11. **Review, revise and remember**

Producing software quality is not a coincidence. This is why you must always do the following three things:

- **Review :** Testing often is a pillar of ensuring software quality. It ensures that standards are continuously met and bugs, errors and distractions can be fixed before they spiral out of control.

- **Revise :** Study what has worked throughout the software process. Utilise what is working and see if innovation can transcend your software quality even further.
- **Remember :** When you deliver quality remember what worked well and did not work well. Keep an updated record of both the positives and negatives of any given project and turn to it frequently when you start the next project from scratch.

6.3 SOFTWARE QUALITY ASSURANCE

UQ. How to maintain Software Quality Assurance ? SPPU - Nov. 18 (End-sem)

- Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.
- Software Quality Assurance is a process which works parallel to development of a software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of an Umbrella activity that is applied.
- There is no one universal definition of software quality. This is because of the complexity caused by the three or more participants affected by the quality of software, namely, customer, developer and stakeholders.
- The issue is whose views, expectations and aspirations are to be considered supreme. The majority hold that customer satisfaction should be the goal for measuring software quality.
- The customer may be satisfied though with software, the quality of which cannot be considered the best by other standards.
- The software quality definition is based on the following:
  1. Customer focus and customer satisfaction
  2. Functional and performance requirement
  3. Ease of learning, use and maintainability
  4. Adherence to development standards
- Customer satisfaction largely depends on meeting functional and performance requirements and ease of operations. Adhering to development standards ensures to a great extent the achievement of these goals.
- Software quality is defined as the quality that ensures customer satisfaction by offering all the customer deliverables on performance, standards and ease of operations. The definition is applicable for software as well as for a generic software product.

Software Quality Assurance (SQA)

- Software quality assurance is a planned effort to ensure that a software product fulfills these criteria and has additional attributes specific to the project, e.g., portability, efficiency, reusability, and flexibility.
- It is the collection of activities and functions used to monitor and control a software project so that specific objectives are achieved with the desired level of confidence.
- It is not the sole responsibility of the software quality assurance group but is determined by the consensus of the project manager, project leader, project personnel, and users.
- A formal definition of software quality assurance is that is 'the systematic activities providing evidence of the fitness for use of the total software product."
- Software quality assurance is achieved through the use of established guidelines throughout the software process.

Software Quality Assurance has :

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

6.3.1 Elements in SQA

UQ. What are different elements in SQA ? SPPU - Nov. 18 (Endsem)

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

6.3.2 Task Goal and Metric in SQA

UQ. What are different task goal and metric in SQA ? SPPU - May 19 (Endsem)

Goals, Attributes and Metrics

- **Requirement quality :** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow. SQA must ensure that the software team has properly reviewed the requirements model to achieve a high level of quality.
- **Design quality :** Every element of the design model should be assessed y the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality :** Source code and related work products must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness :** A software team should apply limited resources in a way that has the highest likelihood of achieving a high–quality result. SQA analyzes the allocation of resources for reviews and testing to assess whether they are being allocated in the most effective manner.

Sr. No.	Goal	Attribute	Metric
1.	Requirement quality	Ambiguilty	Number of ambiginous modifiers (e.g. many, large, human-friendly)
		Completeness	Number of TBA, TBD
		Understandability	Number of sections/subsections
		Volatility	Number of changes per requirement Time (by activity) when change is requested.
		Traceability	Number of requirements not traceable to design/code.
		Model clarity	Number of UML models. Number of descriptive pages per model. Number of UML errors.

Sr. No.	Goal	Attribute	Metric
2.	Design qualtiy	Architecutral integrty	Existance of architectural model
		Component completeness	Number of components that trace to architectural model
			Complextiy of procedural design
		Interface complexity	Average number of pick to get to a typical function or content
		Patterns	Layot appropriateness
			Number of patterns used.
3.	Code qualtiy	Complexity	Cyclomatic complexity
		Maintainability	
			Design factors
		Understanability	Percent internal comments
			Variable naming conventions
		Reusability	Percent reused components
		Documentation	Readability index
4.	QC effectiveness	Resource allocation	Staff hour percentage per activity
		Completion rate	Actual vs. budgeted completion time
		Review effectiveness	See review metrics
		Testing effectiveness	Number of errors found and criticality
			Effort required to connect an error
			Origin of error

6.3.3 Formal Approaches to SQA

UQ. Formal Approaches to SQA, Statistical Software Quality Assurance?
SPPU - May 19 (Endsem)

Abbreviated as SQAP, the software quality assurance plan comprises of 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).

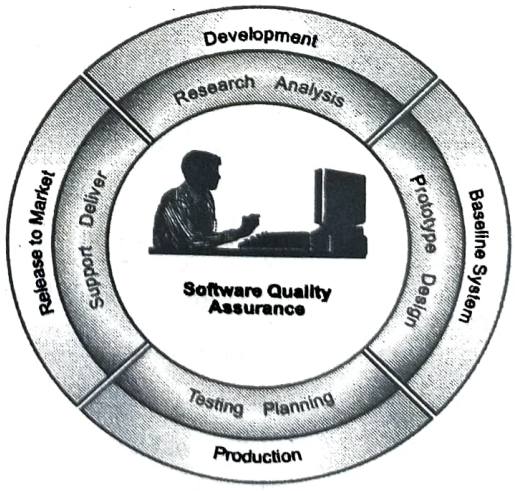


Fig. 6.3.1 : Software Quality Assurance Plan

Software Quality Assurance

- The plan identifies the SQA responsibilities of a team, 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 :
  - Purpose section
  - Reference section
  - Software configuration management section
  - Problem reporting and corrective action section
  - Tools, technologies and methodologies section
  - Code control section
  - Records : Collection, maintenance and retention section.

6.4 SIX SIGMA FOR SOFTWARE ENGINEERING

UQ. What is Six Sigma ? **SPPU - May 18, Nov. 19 (Insem)**

- Six Sigma is one of the most popular quality methods lately. It is the rating that signifies "best in class," with only 3.4 defects per million units or operations (DPMO). Its concept works and results in remarkable and tangible quality improvements when implemented wisely. Today, Six Sigma processes are being executed in a vast array of organization and in a wide variety of functions.
- Fuelled by its success at large companies such as Motorola, General electric, Sony, and Allied Signal, the methodology is proving to be much than just a quality initiative. Why are these large companies embracing Six Sigma? What makes this methodology different from the others?
- The goal of Six Sigma is not to achieve six sigma levels of quality, but to improve profitability. Prior to Six Sigma, improvements brought about by quality programs, such as Total Quality Management (TOM) and ISO 9000, usually had no visible impact on a company's net income. In general, the consequences of immeasurable improvement and invisible impact caused these quality programs gradually to be.
- Six Sigma was originally developed as a set of practices designed to improve manufacturing processes and eliminate defects, but its application was subsequently extended to other types of business processes as well In Six Sigma, a defect is defined as anything that lead to customer dissatisfaction.
- Six Sigma stands for six standard deviation from mean (sigma is three Greek letter used to represent standard deviation in statistics).
- Six Sigma methodologies provide the techniques and tools to improve the capability and reduce the defects in any process.
- Six Sigma strives for perfection. It allows for only 3.4 defects per million Opportunities (or 99.999666 percent accuracy)
- Six Sigma improves the process performance decrease variation and maintains consistent quality of the process output. This leads to defect reduction and improvements in profits, product quality and customer satisfaction.
- Six Sigma incorporates the basic principles and techniques used in business, statistics and engineering.
- The objective of Six Sigma principle is to achieve zero defects products/ process.
- It allows and engineering.
- The objective of Six Sigma principle is to achieve zero defects products/ process.
- It allows 4.4 defects per million opportunities.
- Features that St Six Sigma

- Apart from previous quality improvement initiatives include :
  - A clear focus on achieving measurable and quantifiable financial returns from any Six Sigma project.
  - An increased emphasis on strong and passionate management leadership and support.
  - A special infrastructure of "Champions, "Master black Belts, "black Belts", etc. to lead and implement the Six Sigma approach.
  - A clear commitment to making decisions on the basis of verifiable data, rather than assumptions and guesswork.
- Sigma Levels
  - 1 sigma = 690, 000 DPMO = 31% efficiency
  - 2 sigma = 308,000 DPMO= 69.2% efficiency
  - 3 sigma = 66,800 DPMO= 93.32% efficiency
  - 4 sigma = 6,210 DPMO = 99.379 % efficiency
  - 5 sigma = 230 DPMO = 99.977 % efficiency
  - 6 sigma = 3.4 DPMO = 99.9997 % efficiency

6.5 ISO 9000 STANDARDS

UQ. What are different ISO 9000 Standards ? SPPU - May 18 (Insem)

- ISO 9001: 2008 Quality management systems : Requirements is intended for use in any organization regardless of size, type or product (including service). It provides a number of requirements which an organization needs to fulfill to achieve customer satisfaction through consistent products and services which meet customer expectations. It includes a requirement for continual (i.e., planned) improvement of the Quality Management System, for which ISO 9004:2004 provides many hints.
- This is the only implementation for which third party auditors can grant certification. It should be noted that certification is not described as any of the 'needs' of an organization as a driver for using ISO 9001 but does recognize that it may be used for such a purpose.
- ISO 9004:2000 Quality management systems - Guidelines for performance improvements covers continual improvement. This gives you advice on what you could do to enhance a mature system. This document very specifically states that it is not intended as a guide to implementation.
- **There are many more standards in the ISO 9001 series, many of them not even carrying "ISO 9000" numbers. For example, some standards in the 10,000 range are considered part of the 9000 group: ISO 10007: 1995 discusses configuration management, which for most organizations is just one element of a complete management system. The emphasis on certification tends to overshadow the fact that there is an entire family of ISO 9000 standards..**
- **Organizations stand to obtain the greatest value when the standards in the new core series are used in an integrated manner, both with each other and with the other standards making up the ISO 9000 family as a whole.**

(1E3)**Fig. 6.5.1 : ISO Standard Principles**

- Note that the previous members of the ISO 9000 series 9002 and 9003 have been integrated into 9001. In most cases, an organization claiming to be "ISO 9000 registered" is referring to ISO 9001.

6.6 SOFTWARE QUALITY ASSURANCE PLAN

UQ. Explain in detail Software Quality Assurance Plan ? SPPU - May 18 (Insem)

- The software quality assurance (SQA) plan is an outline of quality measures to ensure quality levels within a software development effort. The plan is used as a baseline to compare the levels of quality during development with the planned levels of quality. If the levels of quality are not within the planned quality levels management will respond appropriately as documented within the plan.
- The plan provides the framework and guidelines for development of understandable and maintainable code. These Ingredients help ensure the quality sought in a software project, A SQA plan also provides the procedures for ensuring that quality software will be produced or maintained in house or under contract.
- These procedures affect planning. Designing, writing, testing, documenting, strong, and maintaining computer software. It should be organized in this way because the plan ensures the quality of the software rather than describing specific procedures for developing and maintaining the software.
- In the management approval process, management relinquishes tight control over software quality to the SQA plan administrator in exchange for improved software quality. Software quality is often left to software developers. Quality is desirable.
- But management may express concern as to the cost of a formal SQA plan. Staff should be aware that management views the program as a means of ensuring software quality, and not as an end in itself.
- To address management concerns, software life cycle costs should be formally estimated for projects implemented both with and without a formal SQA plan. In general, implementing a formal SQA plan makes economic and management sense.
- The SQA Plan helps to lay down the steps towards the quality goals of the organization. A standard for SQA plan gives details and templates on all activities, which have become part of the standard and which ensure quality standards implementation.
- Testing activity needs Test Plan likewise SQA activity also needs a plan which is called SQA plan.
- The goal of SQA plan is to craft planning processes and procedures to ensure products manufactured, or the service delivered by the organization are of exceptional quality.
- During project planning, Test Manager makes an SQA plan where SQA audit is scheduled periodically.
- The documentation of SQA plan includes
  - Project plan
  - Models of data, classes and objects, processes, design, architecture
  - Software Requirement Specifications (SRS)
  - Test plans for testing SRS
  - Users help documentation, manuals, online help, etc.
  - Reviews and audits
- The SQA process is made up of several activities. Some are organization specific, some are software specific and some are customer specific. It helps effective application of methods, tools, test plans and standards towards the goal of software quality. It also include measures, measurement and metric for building a quality in the organization.

6.7 INTRODUCTION TO TOTAL QUALITY MANAGEMENT

UQ. What Is Total Quality Management ? **SPPU - Oct. 19 (Insem)**

- Total quality management (TQM) is the continual process of detecting and reducing or eliminating errors in manufacturing, streamlining supply chain management, improving the customer experience, and ensuring that employees are up to speed with training.
- Total quality management aims to hold all parties involved in the production process accountable for the overall quality of the final product or service.
- TQM was developed by William Deming, a management consultant whose work had a great impact on Japanese manufacturing.
- While TQM shares much in common with the Six Sigma improvement process, it is not the same as Six Sigma. TQM focuses on ensuring that internal guidelines and process standards reduce errors, while Six Sigma looks to reduce defects.
- Total quality management (TQM) is a structured approach to overall organizational management. The focus of the process is to improve the quality of an organization's outputs, including goods and services, through continual improvement of internal practices.
- The standards set as part of the TQM approach can reflect both internal priorities and any industry standards currently in place.
- Industry standards can be defined at multiple levels and may include adherence to various laws and regulations governing the operation of the particular business.
- Industry standards can also include the production of items to an understood norm, even if the norm is not backed by official regulations.

6.8 PRODUCT QUALITY METRICS

UQ. Write in brief Product Quality Metrics ? **SPPU - Aug 17 (Insem), May 18 (Endsem)**

- Software metrics can be classified into three categories :
  1. **Product metrics :** Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
  2. **Process metrics :** These characteristics can be used to improve the development and maintenance activities of the software.
  3. **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.
- Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.
- Software quality metrics 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 :
  1. Product quality metrics
  2. In-process quality metrics
  3. Maintenance quality metrics

This metrics include the following :

(a) Mean Time to Failure	(b) Defect Density
(c) Customer Problems	(d) Customer Satisfaction

(a) **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.

(b) **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.

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

- The problems metric is usually expressed in terms of Problems per User-Month (PUM).
- PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

(d) **Customer Satisfaction :** Customer satisfaction is often measured by customer survey data through the five-point scale :

(i) Very satisfied (ii) Satisfied (iii) Neutral

(iv) Dissatisfied (v) Very dissatisfied

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

1. Percent of completely satisfied customers
2. Percent of satisfied customers
3. Percent of dis-satisfied customers
4. Percent of non-satisfied customers

Usually, this percent satisfaction is used.

6.9 IN PROCESS QUALITY METRICS

UQ. What is in process Quality Metrics ? **SPPU - Nov./Dec. 19 (Endsem)**

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

1. Defect density during machine testing
2. Defect arrival pattern during machine testing
3. Phase-based defect removal pattern
4. Defect removal effectiveness

1. 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.

2. Defect arrival pattern during machine testing

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

(i) The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.

(ii) The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.

(iii) The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

3. 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.

4. Defect removal effectiveness

- It can be defined as follows:

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

- 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.

6.10 SOFTWARE MAINTENANCE

UQ. What is Software Quality Maintenance Metrics? SPPU - Nov./Dec.18(Insem)

- 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.

1. Fix backlog and backlog management index

2. Fix response time and fix responsiveness
3. Percent delinquent fixes

☞ **Percent Delinquent Fixes**

$$\frac{\text{Number of fixes that exceeded the response time criteria by ceverity level}}{\text{Number of fixes delivered in a specified time}} \times 100\ \%\ \text{Fix quality}$$

(1) Fix backlog and backlog management index

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

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

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

(2) 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.

(3) Percent delinquent fixes

It is calculated as follows :

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

☞ **Fix Quality**

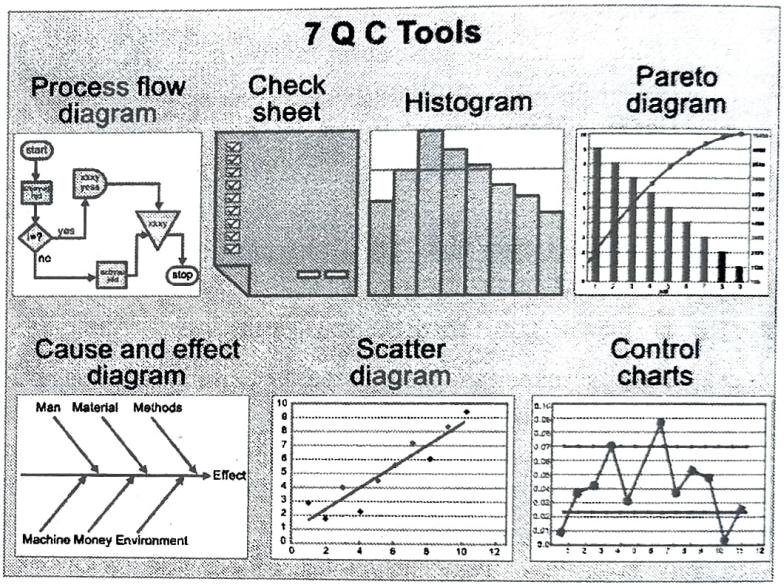
- Fix quality or the number of defective fixes is another important quality metric for the maintenance phase.
- A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction.
- The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.
- A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered.
- The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.
- Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture.
- The quality goal for the maintenance process, of course, is zero defective fixes without delinquency.

6.11 ISHIKAWA'S 7 BASIC TOOLS

UQ. What are Ishikawa's 7 basic tools ? SPPU - May 18 (Endsem)

UQ. Explain the Terms Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. SPPU - Aug. 17 (Insem), May 18 (Endsem), May 19 (Insem)

- Dr. Kaoru Ishikawa was first total quality management guru, who has been associated with the development and advocacy of using the seven quality control (QC) tools in the organizations for problem solving and process improvements.
- Seven old quality control tools are a set of the QC tools that can be used for improving the performance of the production processes, from the first step of producing a product or service to the last stage of production. So, the general purpose of this paper was to introduce these 7 QC tools.
- This study found that these tools have the significant roles to monitor, obtain, analyze data for detecting and solving the problems of production processes, in order to facilitate the achievement of performance excellence in the organizations.
- There are seven basic quality tools, which can assist an organization for problem solving and process improvements.



(1F1)**Fig. 6.11.1 : Ishikawa 7 Basic Tool for Quality Control**

- The first guru who proposed seven basic tools was Dr. Kaoru Ishikawa in 1968, by publishing a book entitled "Gemba no QC Shuho" that was concerned managing quality through techniques and practices for Japanese firms.
- It was intended to be applied for "self-study, training of employees by foremen or in QC reading groups in Japan. It is in this book that the seven basic quality control tools were first proposed. valuable resource when applying the seven basic tools (Omachonu and Ross, 2004).
- These seven basic quality control tools, which introduced by Dr. Ishikawa, are :
  (1) Check sheets; (2) Graphs (Trend Analysis);
  (3) Histograms; (4) Pareto charts;
  (5) Cause-and-effect diagrams; (6) Scatter diagrams; (7) Control charts.
- Fig. 6.11.1 indicates the relationships among these seven tools and their utilizations for the identification and analysis of improvement of quality.

6.12 CHECK SHEET/CHECKLIST

- Check sheets are simple forms with certain formats that can aid the user to record data in an firm systematically.
- Data are "collected and tabulated" on the check sheet to record the frequency of specific events during a data collection period.

- They prepare a "consistent, effective, and economical approach" that can be applied in the auditing of quality assurance for reviewing and to follow the steps in a particular process. Also, they help the user to arrange the data for the utilization later.
- The main advantages of check sheets are to be very easily to apply and understand, and it can make a clear picture of the situation and condition of the organization.
- They are efficient and powerful tools to identify frequently problems, but they don't have effective ability to analyze the quality problem into the workplace. The chech sheets are in several, three major types are such as Defect-location check sheets; tally check sheets, and; defect-cause check sheets .

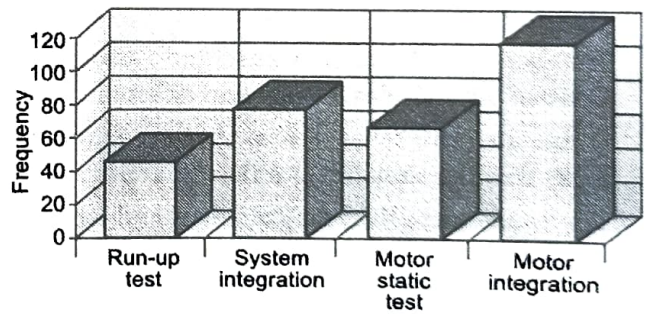
Telephone interruptions

Reason	Day					
	Mon	Tues	Wed	Thurs	Fri	Total
Wrong number	𝍸	II	I	𝍸	𝍸 II	20
Info request	II	II	II	II	I	10
boss	𝍸	II	𝍸 II	I	IIII	19
Total	12	6	10	8	13	49

(1F2)**Fig. 6.12.1 : Check sheet Tally for telephone interruptions**

6.13 HISTOGRAM

- Histogram is very useful tool to describe a sense of the frequency distribution of observed values of a variable. It is a type of bar chart that visualizes both attribute and variable data of a product or process, also assists users to show the distribution of data and the amount of variation within a process.
- It displays the different measures of central tendency (mean, mode, and average). It should be designed properly for those working into the operation process can easily utilize and understand them.
- Also, a histogram can be applied to investigate and identify the underlying distribution of the variable being explored. Fig. 6.13.1 illustrates a histogram of the frequency of defects in a manufacturing process.

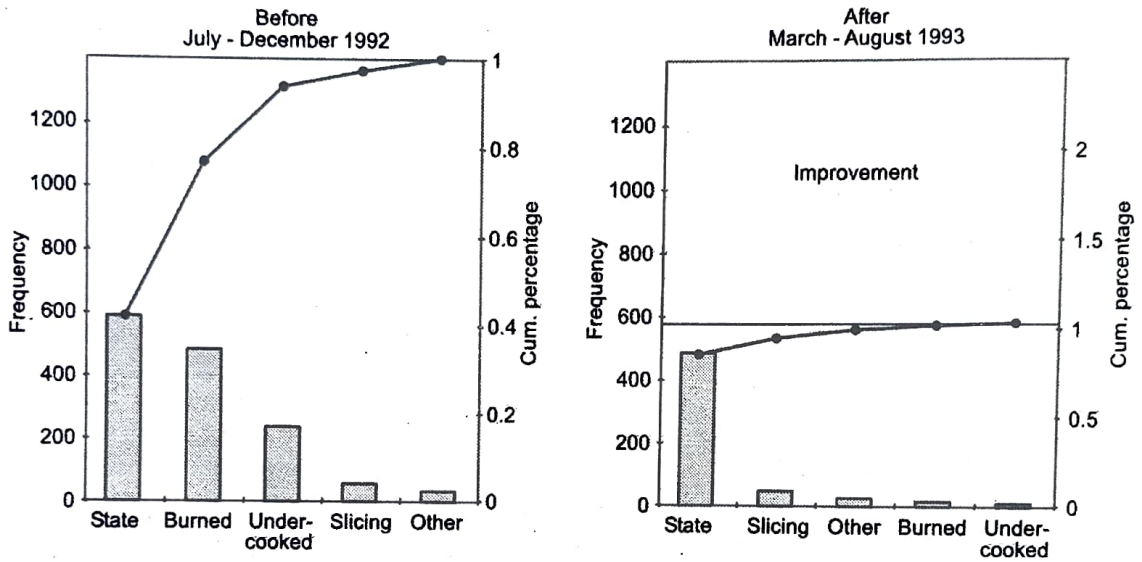


(1F3)**Fig. 6.13.1 : Histogram for variables**

6.14 PARETO ANALYSIS

- It introduced by an Italian economist, named Vilfredo Pareto, who worked with income and other unequal distributions in 19th century, he noticed that 80% of the wealth was owned by only 20% of the population. later,
- Pareto principle was developed by Juran in 1950. A Pareto chart is a special type of histogram that can easily be apply to find and prioritize quality problems, conditions, or their causes of in the organization (Juran and Godfrey, 1998).

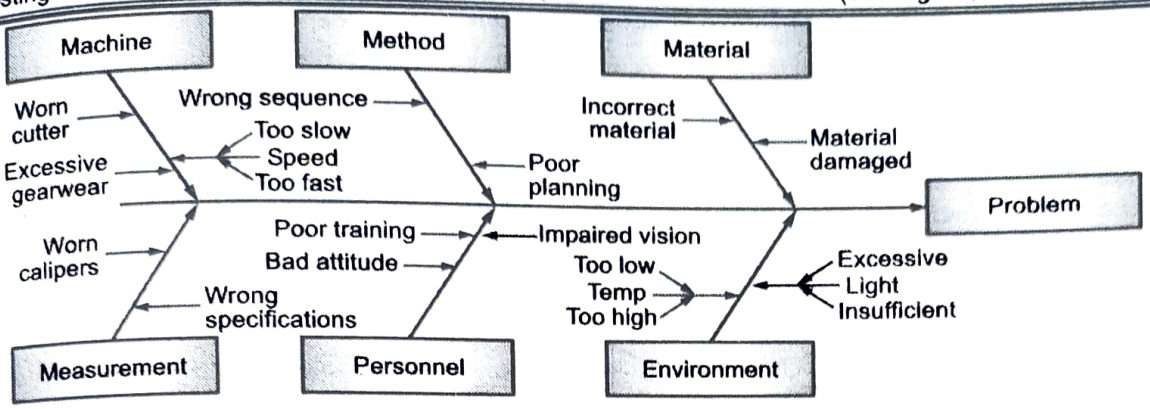
- On the other hand, it is a type of bar chart that shows the relative importance of variables, prioritized in descending order from left to right side of the chart.
- The aim of Pareto chart is to figure out the different kind of "nonconformity" from data figures, maintenance data, repair data, parts scrap rates, or other sources.
- Also, Pareto chart can generate a mean for investigating concerning quality improvement, and improving efficiency, "material waste, energy conservation, safety issues, cost reductions", etc., as
- Fig. 6.14.1 demonstrated concerning Pareto chart, it can able to improve the production before and after changes.



(1F4)**Fig. 6.14.1 : Pareto Charts**

6.15 FISHBONE DIAGRAM

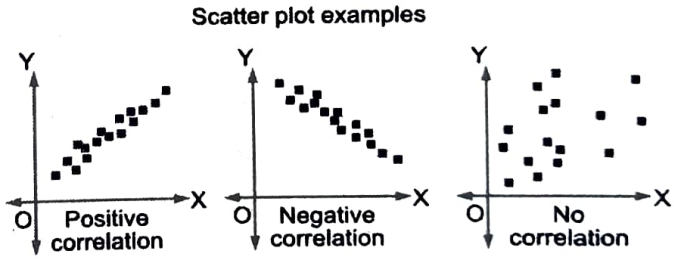
- Kaoru Ishikawa is considered by many researchers to be the founder and first promoter of the 'Fishbone' diagram (or Cause-and-Effect Diagram) for root cause analysis and the concept of Quality Control (QC) circles. Cause and effect diagram was developed by Dr. Kaoru Ishikawa in 1943.
- It has also two other names that are Ishikawa diagram and fishbone because the shape of the diagram looks like the skeleton of a fish to identify quality problems based on their degree of importance.
- The cause and effect diagram is a problem-solving tool that investigates and analizes systematically all the potential or real causes that result in a single effect.
- On the other hand, it is an efficient tool that equips the organization's management to explore for the possible causes of a problem.
- This diagram can provide the problem-solving efforts by "gathering and organizing the possible causes, reaching a common understanding of the problem, exposing gaps in existing knowledge, ranking the most probable causes, and studying each cause".
- The generic categories of the cause and effect diagram are usually six elements (causes) such as environment, materials, machine, measurement, man, and method, as indicated in Fig. 6.15.1. Furthermore, "potential causes" can be indicated by arrows entering the main cause arrow.



(1F5)**Fig. 6.15.1 : The cause and effect diagram (Fishbone Diagram)**

6.16 SCATTER DIAGRAM

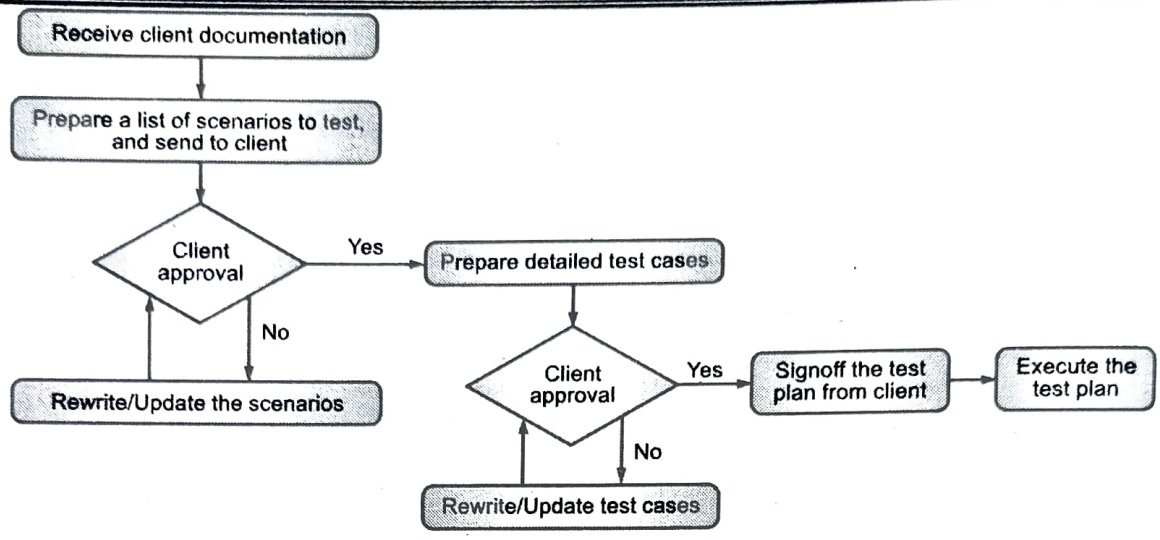
- Scatter diagram is a powerful tool to draw the distribution of information in two dimensions, which helps to detect and analyze a pattern relationships between two quality and compliance variables (as an independent variable and a dependent variable), and understanding if there is a relationship between them, so what kind of the relationship is (Weak or strong and positive or negative).
- The shape of the scatter diagram often shows the degree and direction of relationship between two variables, and the correlation may reveale the causes of a problem. Scatter diagrams are very useful in regression modeling.
- The scatter diagram can indicate that there is which one of these following correlation between two variables :
  (a) Positive correlation;
  (b) Negative correlation, and
  (c) No correlation, as demonstrated in Fig. 6.16.1. an operation, so it is very useful to find and improve quality into process.



(1F6)**Fig. 6.16.1 : Scatter Diagram**

6.17 FLOWCHART

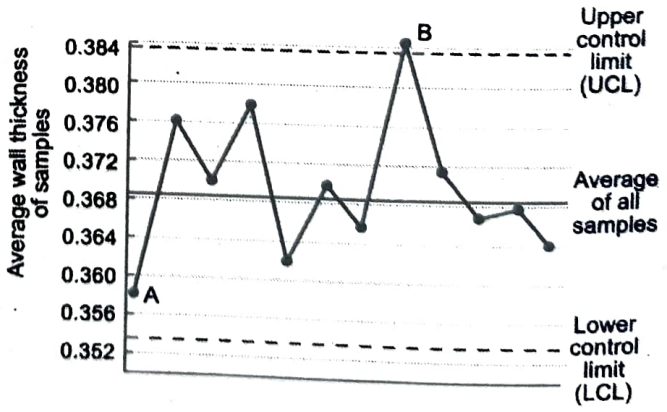
- Flowchart presents a diagrammatic picture that indicates a series of symbols to describe the sequence of steps exist in an operation or process.
- On the other hand, a flowchart visualize a picture including the inputs, activities, decision points, and outputs for using and understanding easily concerning the overall objective through process.
- This chart as a problem solving tool can apply methodically to detect and analyze the areas or points of process may have had potential problems by "documenting" and explaining an operation, so it is very useful to find and improve quality into process.



(1F7)**Fig. 6.17.1 : Flow Char Review Process**

6.18 CONTROL CHART

- Control chart or Shewhart control chart was introduced and developed by Walter A. Shewhart in the 1920s at the Bell Telephone Laboratories, and is likely the most "technically sophisticated" for quality management.
- Control charts are a special form of "run chart that it illustrates the amount and nature of variation in the process over time". Also, it can draw and describe what has been happening in the process.
- Therefore, it is very important to apply control chart, because it can observe and monitor process to study process that is in "statistical control".
- (No problem with quality) accordant to the samplings or samplings are between UCL and LCL (upper control limit (UCL) and the lower control limit (LCL)).
- "Statistical control" is not between UCL and LCL, so it means the process is out of control, then control can be applied to find causes of quality problem, as shown in Fig. 6.18.1 that A point is in control and B point is out of control. In addition, this chart can be utilized for estimating "the parameters" and "reducing the variability" in a process.
- The main aim of control chart is to prevent the defects in process. It is very essentiality for different businesses and industries, the reason is that unsatisfactory products or services are more coasted than spending expenses of prevention by some tools like control charts.



(1F8)**Fig. 6.18.1 : The Shewhart control chart**

6.19 DEFECT REMOVAL EFFECTIVENESS AND PROCESS MATURITY

UQ. Explain in brief Defect Removal Effectiveness and Process Maturity ? **SPPU - May 18 (Endsem)**

- CMM - capability maturity model is a process improvement model, a framework which is used as powerful tool for understanding and improving performance. There are five levels of which Level fifth is highest.
- The CMM level and number of defects present at various STAGES are based on data file of defect. The relationship between the variables is established using regression technique.
- CMM - capability maturity model is a process improvement model, a framework. A defect is nothing but a variance from the given specification, a hidden or coding error.
- Defects are undesirable, they cause increase in risk, revenue loss to the customer if they remain in the final product.
- Rectification may prove costly and results in risk and loss to the agency too. This is persistent from the innumerable surveys conducted by well recognized agencies.
- Defects are brought to the notice of project team by a process known as bug reporting. Defect reports are used to alert software programmers about the defect and gives them sufficient information to find root cause of problem and fix it. It provides information to technical writers and add test cases in the regression suite for next release. In this paper the relationship between CMM level and number of defect present in various STAGES like Requirement (REQ), Design (DSN), Coding (CDE), Document (DOC) is established.
- The relationship between CMM level and number of defects present in various STAGES like Requirement (REQ), Design (DSN), Coding (CDE), Document (DOC) are established in terms of prediction. Number of defects predicted are based on CMM level i.e. 0, 1, 3, 5.
- The number of defects present are associated with cost, time and quality. The lesser the number of defect indicate better the quality. Thus CMM level is associated with cost, time, improvement, quality and reduce rework. Based on it good or poor quality software can be rated.

...Chapter Ends

❑❑❑