UNIT I

CHAPTER 1

Introduction to Software Testing

University Prescribed Syllabus for the Academic Year 2022-2023

Introduction : Historical perspective, Definition, Core Components, Customers suppliers and process, Objectives of Testing, Testing and Debugging, Need of Testing, Quality Assurance and Testing, Why Software has Errors, Defects and Failures and its Causes and Effects, Total Quality Management(TQM), Quality practices of TQM, Quality Management through- Statistical process Control, Cultural Changes, Continual Improvement cycle, Benchmarking and metrics, Problem Solving Techniques and Software Tools.

Software Quality : Constraints of Software product Quality assessment, Quality and Productivity Relationship, Requirements of Product, Software Development Process, Types of Products, Software Development Lifecycle Models, Software Quality Management, Processes related to Software Quality, Quality Management System's Structure, Pillars of Quality Management System, Important aspects of quality management.

1.1 INTRODUCTION

GQ. What Is Software Quality Assurance (SQA)? **OR** What is Software Quality Assurance?

Definition of SQA : Software quality assurance (SQA) is a process which guarantees that all software engineering progressions, methods, activities and work items are examined and comply in contradiction of the defined standards. These definite standards could be one or a grouping of any like ISO 9000, CMMI model, ISO15504.

SQA includes all software growth procedures starting from important necessities to coding until issue. Its prime goal is to ensure quality.

1.1.1 Software Quality Assurance Plan : Software Quality

UQ. Identify the software quality attributes? **SPPU - Aug 18 (In sem), May 19 (End sem)**

UQ. What is software quality? What is mean by quality attribute of the software ? **SPPU - Nov./Dec. 19 (End sem)**

Abbreviated as SQAP, the software quality assurance plan includes of the events, methods, and tools that are employed to type sure that a creation or service supports with the wants distinct in the SRS (software requirement specification).

1. **Quality :** The degree to which a module, system or process meets specified requirements and/or user/customer needs and expectations.
2. **Software Quality :** The totality of functionality and structures of a software product that bear on its ability to content stated or implicit needs.
3. **SOFTWARE QUALITY** is the degree of conformance to explicit or implicit requirements and prospects.
4. **Explicit :** clearly definite and familiar
5. **Implicit :** not clearly definite and recognized but ultimately suggested
6. **Requirements :** business/product/software necessities
7. **Expectations :** mainly end-user prospects

- Software testing is a process of performing a program or application with the determined of definition the software bugs.
- It can also be specified as the process of validating and verifying that a software program or application or product: Happens the business and technical supplies that guided it's design and development
- Works as expected. Can be executed with the same characteristic.
- The software system wants to be checked for its proposed behavior and direction of progress at each growth stage to avoid replica of efforts, time and cost overruns, and to assure achievement of the system within specified time.
- The software system wants to be tested for its planned performance and direction of growth at each development stage to avoid replication of efforts, time and cost overruns, and to assure completion of the system within stipulated time.
- System testing and quality assurance come to aid for checking the system. It includes
  1. Product level quality (Testing)
  2. Process level quality.

1.1.2 Characteristics of System Testing

1. Testing is the procedure or action that checks the functionality and precision of software according to definite user necessities in order to advance the quality and consistency of system.
2. It is an exclusive, time consuming, and critical approach in system development which requires proper planning of overall testing process.
3. System testing creates at the module level and earnings towards the addition of the entire software system.
4. Different testing methods are used at dissimilar times while difficult the system. It is directed by the designer for small projects and by autonomous difficult groups for large projects.

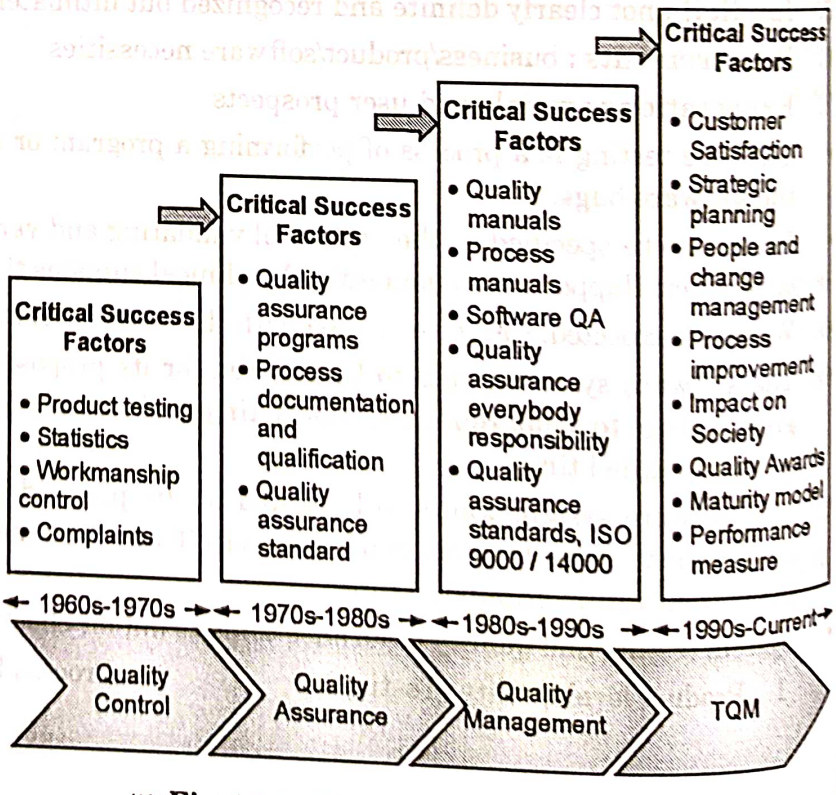
1.1.3 Why is Software Testing Necessary?

- Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous.
- We need to check everything and anything we produce because things can always go wrong – humans make mistakes all the time. Software testing is very important because of the succeeding reasons:
- Software testing is really required to point out the defects and errors that were made during the development phases.
- It's important since it types sure of the Customer's consistency and their approval in the request.
- It is very significant to confirm the Quality of the product. Quality product supplied to the customers benefits in ahead their confidence. (Know more about Software Quality)
- Testing is essential in order to offer the services to the customers like the distribution of high quality product or software request which needs lower conservation cost and hence results into more precise, reliable and consistent results. Testing is compulsory for an actual presentation of software application or product.
- It's significant to confirm that the application should not consequence into any disappointments because it can be very exclusive in the future or in the later phases of the development. It's required to stay in the business.

1.1.4 Software Testing Objectives and Purposes

Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

1. Definition defects which may get created by the programmer while evolving the software. Gaining confidence in and providing evidence about the level of quality.
2. To avoid defects. To type sure that the end results meets the business and user supplies.
3. To confirm that it contents the BRS that is Business Requirement Specification and SRS that is System Requisite Provisions.
4. To advantage the confidence of the customers by providing them an excellence product.



(1A1) **Fig. 1.1.1 : History of quality Assurance**

☞ **Definitions of Quality**

- **Customer Based Definition of Quality :** Quality must have fitness foe use and chance customer needs, requirements and help in attaining customer pleasure and customer delight.
- **Manufacturing Based Definition of Quality :** This approach gives conformance to requirements.
- **Product-Based Definition of Quality :** Product must have something that other related products do not have which help customer satisfy their needs in better way.
- **Value-Based Definition of Quality :** Customer must get value for his investment by buying product.
- **Transcendent Quality :** A product must have zero defects so that it organizes not prohibit normal usage by the users.

UQ. Identify the Software Quality Attributes for the following scenarios. **SPPU - May/June 19 (End Sem), Aug 18 (In Sem)**

(i) Any popular websites such as Google.com, Amazon.com and youTube.com is visited by thousands of users concurrently. What could be the top 2 architectural drivers (quality attributes) for this system? Justify your answer.

OR

(i) Company wants build a game for the children's which they should play from any device. Also numerous input devices (mouse, joystick, touch screen) may also integrated for playing a game.

(ii) While updating Anti-Virus software, it take lot of time so the updating tool displays a progress bar showing the percentages of completion of work and how many files are scanned so far. Which quality attribute is being addressed by this tactic? Justify your answer.

OR

(ii) Now a day's most of the peoples are using internet banking for online transaction. What could be the top 2 architectural drivers (quality attributes) for this system? Justify your answer.

(iii) A software makes use of adaptors to connect to SMS gateways of different service providers such as Airtel, Vodafone, BSNL, etc. All these adaptors implement the same interface. Depending on the service provider defined in the configuration file, the software instantiates (creates) an object of the appropriate adaptor class at run time and makes use of it to communicate with the SMS gateway.

OR

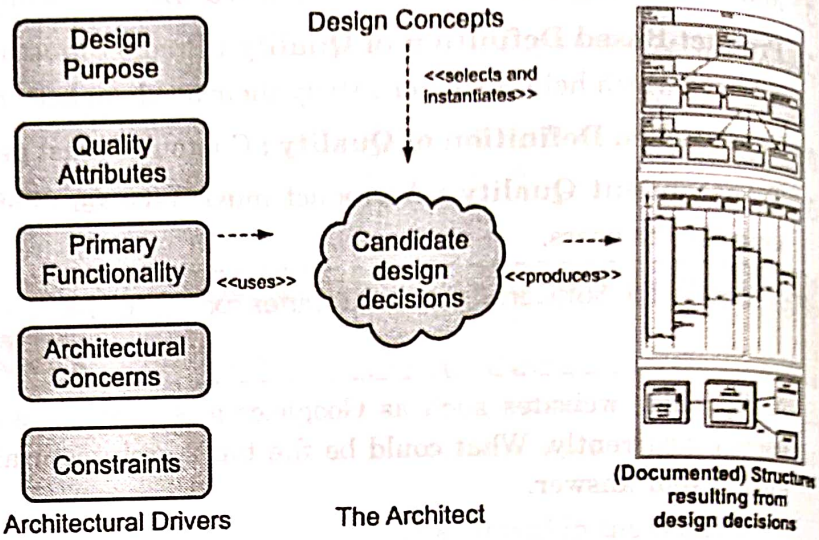
(iii) A software company is in a process of building social networking site which will have very large number of users in near future. Also company wish to add new features in this site and during addition of new features site should provide all the current features without any disturbance What top 2 quality attribute is being addressed by this tactic? Justify your answer.

1) Architectural Drivers (Answer for (i) question)

Already beginning design with ADD (or with any other enterprise method, for that matter), you essential to consider about what you are doing and why. While this statement may look extremely obvious, the devil is, as usual, in the details. We classify these what and why as architectural drivers questions.

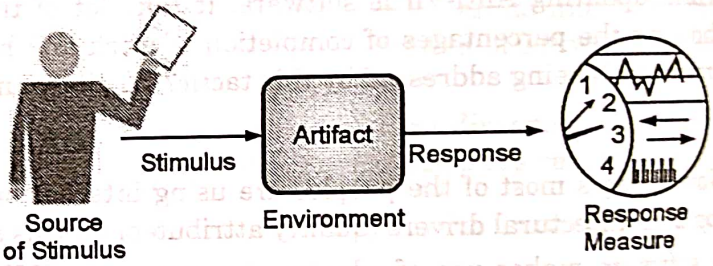
- As shown in Fig. 1.1.2, these drivers include a design purpose, excellence attributes, primary functionality, architectural disquiets, and limitations. These attentions are critical to the achievement of the system and, as such, they *drive* and shape the architecture.
- As with any other significant necessities, architectural driver's essential to be base ruled and achieved through the growth life cycle.

- Given their significance, you must concern about stimulating, requiring, arranging, and authorizing quality qualities. Given that so much is contingent on receiving these drivers right, this sounds like a discouraging task. Providentially, a numeral of glowing expected, widely scattered methods can support you here (The Quality Attribute Workshop and the Utility Tree)
- Quality Attribute Workshop (QAW) is a simplified suggesting assembly concerning a group of system investors that insurances the bulk of the events of generating, requiring, ordering, and realizing consensus on quality attributes.
- Mission Thread Workshop serves the same purpose as QAW, but for a system of systems.
- The Utility Tree can be used by the architect to prioritize quality attribute requirements according to their technical difficulty and risk.



(1A2)**Fig. 1.1.2 : Architectural Design Drivers (ADD)**

- Complete quality attribute scenario adds three other parts: the source of the motivation(in this case, the user), the artifact artificial (in this case, because we are production with end-to-end potential, the artifact is the complete system) and the atmosphere (are we in normal process, start up, degraded mode, or some extra mode?). In total, then, there stay six parts of a totally well detailed scenario, as shown in Fig. 1.1.3.

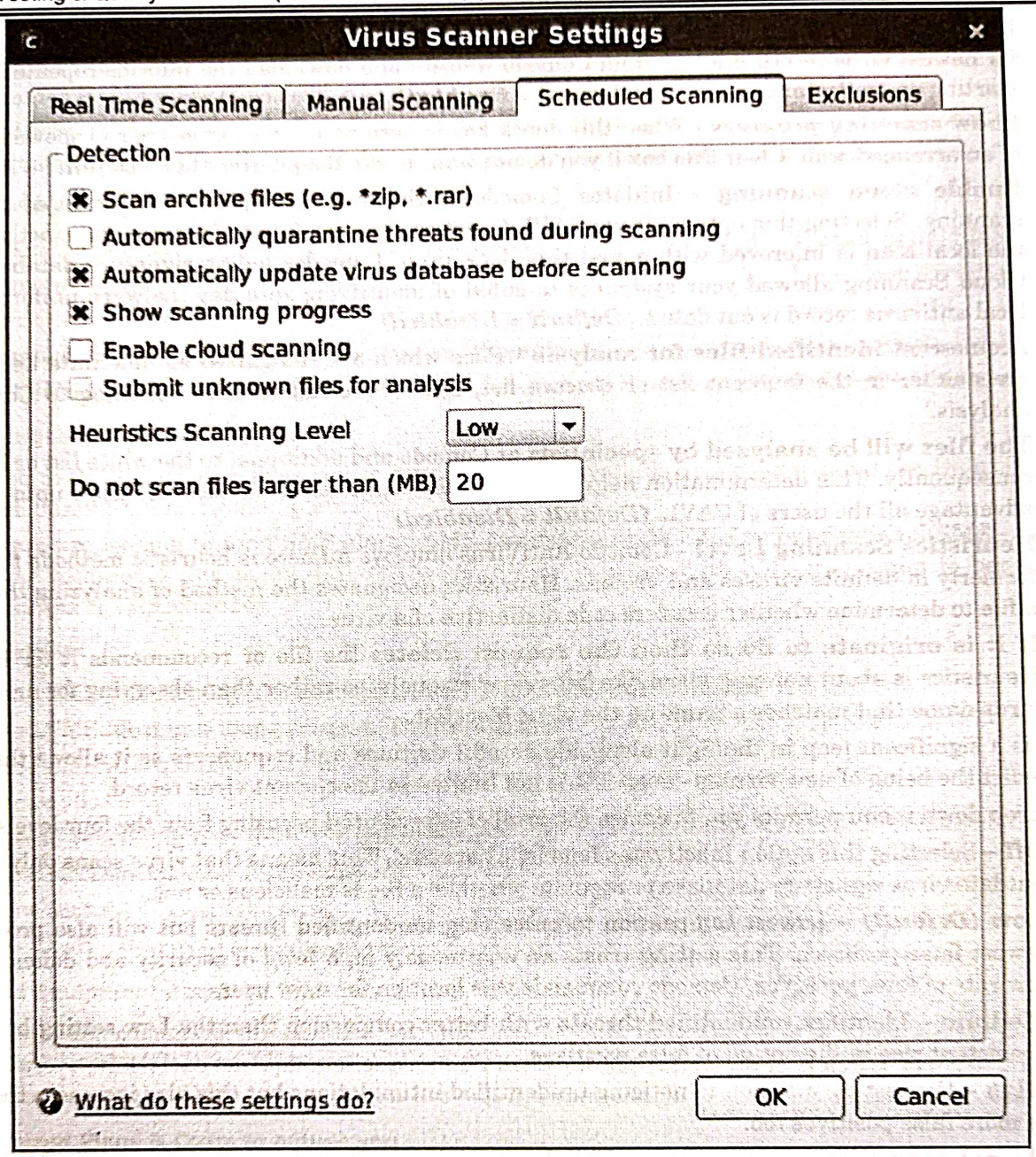


(1A3)**Fig. 1.1.3 : The six portions of a quality attribute situation**

- Scenarios are testable, ***falsifiable hypotheses*** about the quality attribute behaviour of the system below thought.
- Because they have obvious incentives and reactions, we can calculate a design in terms of how possible it is to provision the situation, and we can take quantities and test a prototype or fully fleshed out system for whether it contents the situation in repetition.
- If the analysis (or prototyping results) indicates that the scenario's response goal cannot be met, and then the hypothesis is deemed falsified.

2) Scheduled Scanning (Answer for ii question)

- The Scheduled Scanning settings area allows you to determine the scan limitations that will be applied when a planned scan takes place.



(1A4)**Fig. 1.1.4 : Virus Scanner Setting**

- You can take to run arranged tests at a certain time on a daily, weekly, monthly or custom interval basis. You can also choose which specific files, folders or drives are included in that scan by choosing the scan profiles.
- The detection settings are as follows:
  - **Scan archive files** - When this form box is designated, the Antivirus scans archive files such as .ZIP and .RAR files. You are notified to the occurrence of viruses in compacted files before you even open them. These include RAR, ZIP and CAB archives. (Default = Enabled)
  - **Automatically quarantine threats found during scanning** - When this check box is selected, the Antivirus moves the file detected to be containing the malware, to Quarantined Items. From the confined items the files can be reinstated or removed at your will. ***(Default = Disabled)***

- **Mechanically inform virus database earlier scanning** - Teaches Comodo Antivirus to check for newest virus record informs from Comodo website and download the informs repeatedly before starting an on-demand scanning. ***(Default = Enabled)***
- **Show scanning progress** - When this check box is designated, a progress bar is showed on start of an arranged scan. Clear this box if you do not want to see the progress bar. ***(Default = Enabled)***
- **Enable cloud scanning** - Initiates Comodo Antivirus to complete cloud created antivirus scanning. Selecting this option allows CAVL to notice the very latest viruses more exactly because the local scan is improved with a real time look up of Comodos online signature database. With Cloud Scanning allowed your system is talented of identifying zero day malware uniform if your local antivirus record is out dated. ***(Default = Disabled)***
- Acquiescent **identified files for analysis** - Files which are recognized as indefinite i.e. the files are neither in the innocent list or obscure list, from the cloud founded scanning to Comodo for analysis?
- **The files will be analyzed by specialists** at Comodo and additional to the white list or black list consequently. This determination help upholding the white list and black list more upto date and advantage all the users of CAVL. ***(Default = Disabled)***
- **Heuristics Scanning Level** - Comodo AntiVirus employs numerous heuristic methods to identify formerly in definite viruses and Trojans. Heuristics designates the method of analyzing the code of a file to determine whether it covers code distinctive of a virus.
- **If it is originate to do so then the request deletes** the file or recommends it for isolation. Heuristics is about noticing virus like behaviour or qualities rather than observing for an accurate virus name that matches a name on the virus blacklist.

- This is a significant leap in the fight alongside hateful writings and sequencers as it allows the engine to predict the being of new viruses - even if it is not limited in the current virus record.
- The drop down menu permits you to choice the level of experiential perusing from the four levels:
  - **Off** – Selecting this option inactivates heuristic perusing. This means that virus scans only uses the outdate virus signature database to regulate whether a file is malicious or not.
  - **Low *(Default)*** – Lowest compassion to perceiving unidentified threats but will also produce the fewest false positives. This setting trusts an enormously high level of security and defence with a low rate of false positives. Comodo commends this location for most users.
  - **Medium** – Identifies unidentified threats with better compassion than the Low setting but with a consistent rise in the option of false positives.
  - **High** – Highest compassion to noticing unidentified intimidations but this also increases the option of more false positives too.
  - **Do not scan files greater than** – This box permits you to set an extreme size (in MB) for the separate files to be perused during on admittance scanning. Files greater than the size quantified here, are not perused. ***(Default = 20 MB)***

3) Bulk SMS Service (Answer for iii questions)

- With the beginning of mobile technology, it's nonstop observing the increase in users having mobile devices. Without applying technical changes and advertising methods you cannot imagine growing in this inexpensive world.
- Sending Bulk SMSs is, therefore, measured the most expediency and fast way to transport material to multiple users within a blink of eyes. Online Bulk SMS facilities, providing by Bulk SMS Gateway are intentional to influence the maximum profits of this tech savvy world at most best cares.

☞ **Send Bulk SMS to any amount of People**

- Bulk SMS Gateway permits the fastest communication through bulk SMS without constraint you when it originates to a number of people.
- Group SMSs to your structural employees, Advertised SMS to your customers and devoted emails to your clients - all are difficulty free so that your business can bloom without difficulties.

☞ **Bulk SMS Service**

1. Bulk SMS Service
2. SMS Solutions
3. Transactional SMS
4. Promotional SMS
5. Group SMS
6. Long Code services
7. Your Own Sender ID
8. API for your Website
9. Schedule SMS
10. Dynamic SMS
11. Delivery Reports
12. Web to SMS
13. BulkSMS for iPhone
14. Bulksms for Android Phone
15. BulkSMS Mobi-grams

☞ **Advantages with Bulk SMS Gateway**

1. Web to SMS service enables you to send SMS to individuals and groups using the Bulk SMS Root web based SMS messaging platform.
2. If you are looking to send SMS alerts, SMS reminders, Bulk SMS or International SMS - here are some techniques and many different ways to help you use Bulk SMS messaging to benefit your business or organization.

☞ **Key Features**

1. Send SMS text messages to persons and collections from the internet
2. Send SMS Messages using program Date and Time
3. Send SMS in Dissimilar Languages
4. Personalise outgoing messages
5. Upload remaining lists from Tab enclosed files
6. View message transfer intelligences
7. View account of messages sent
8. Use Draft patterns to send messages

☞ **Minimal Equipment Requisite and Easy to Use Interface**

Transfer SMSs connected through the web interface of Bulk SMS Gateway is super simple due to the elegant interface which has multiple options and features to help users while sending the SMS. The only requirements to send the SMSs to your clients are - an Internet-enable device with a good connection.

☞ **Customized Plans & Costs to amuse you**

Our plans are variable as we understand the unique of your business. Beside with some pre decided plans, we also entertain modified plans so that small, medium as well large initiatives can use our Bulk SMS services without any hassles. Our assessing is faultless and intentional to suit your pockets.

☞ **Top Provision Facilities**

- For your enquiries, issues, and necessities, Bulk SMS Gateway has a faithful support team which looks into each single entity approaching from our customers. So, we are constantly there after you need us.
- Our website is a presumed portal to obtain the best Bulk SMS facilities and offers a safe way of SMS message. Except for economy Bulk SMS services, we are also providing every type of SMS communication facility to serve as a one stop SMS resolution to our users.

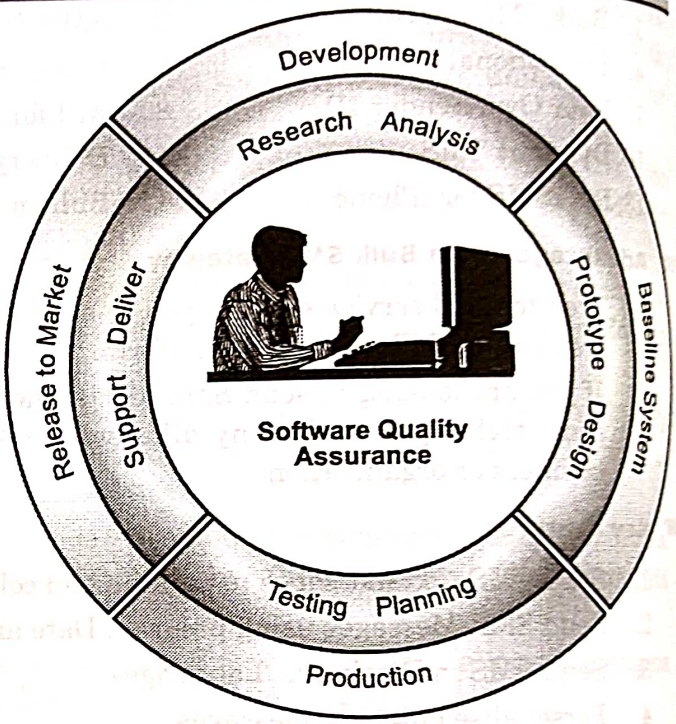
☞ **Getting started**

- Listing for your BulkSMS account is simple and easy and will only take a limited moments.
- To start transfer bulk SMS messages, Its Free Registration No Hidden controls, You can use bulk sms with free credits.

1.2 CORE COMPONENTS OF QUALITY

UQ. Define Software Quality. List and explain Core Components of Quality ?
SPPU - Oct. 19(In Sem)

- Quality is based on Customer satisfaction. Group by acquiring a product.
- The organization must define quality parameters before it can be achieved. The cycle of measurement include,
- Define, Measure, Monitor, Control, Measure.
- Management Must Lead Organization through improvement efforts.
- Continuous Process improvement is necessary.



(1A5)**Fig. 1.2.1 : Core Components SQA**

1.3 CUSTOMER, SUPPLIERS AND PROCESSES

- For any Organization, there are some suppliers supplying the inputs required and some customers who will be buying the output produced.
- Supplier and customers may be inside or outside to the organization.
- External supplier provides input to the organization and external customers receive the output of the organization.
- Suppliers may be customer for some organization and external customers accepts the output of the organization.

☞ **Internal Customer**

Internal customer is the functions and projects serviced and supported by some other functions / projects. System administration may have projects as their customer while purchasing may have system administration as their customer.

☞ **External Customer**

External clients are the external people to the organization who will be paying for the services offered by the organization.

1.4 OBJECTIVES OF TESTING,

The objectives of the testing are the reasons or purpose of the testing and the object of the testing is the work product to be tested.

Testing objectives can differ depending on few factors as,

(1) The context of the component
(2) System being tested
(3) The test level
(4) The software development life cycle model

Above distinctions may include, for example :

- **Example 1 :** One goal of component testing may be to find as many failures as possible so that the underlying defects can be identified and fixed as soon as possible. Another goal could be to increase the code coverage of the component tests.
- **Example 2 :** One goal of acceptance testing may be to confirm that the system works as expected and meets the requirements. Another goal of this testing could be to inform stakeholders about the risks of releasing the system at a specific time.

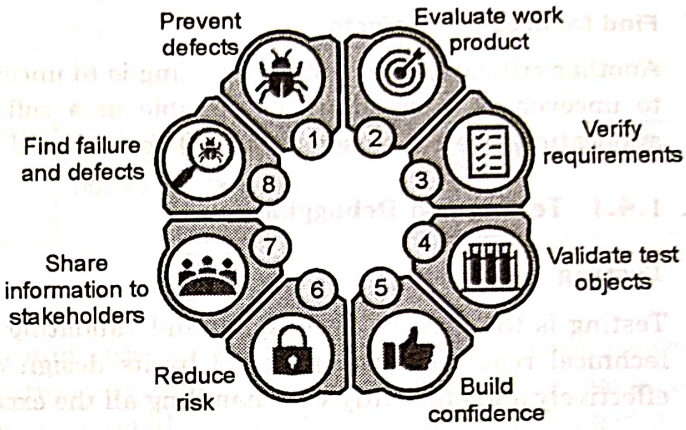


Fig. 1.4.1 : Objectives of testing

The facts may include in typical objectives of Testing

(1) **Prevent defects :** One of the objectives of software testing is to avoid mistakes throughout the development process. The cost and labor associated with error detection are considerably reduced when faults are detected early. It also saves time. Defect prevention entails conducting a root cause analysis of previously discovered flaws and then taking specific steps to prevent the recurrence of those types of faults in the future.

(2) **Evaluate work products :** The objectives are used to assess work items such as the requirement document, design, and user stories. Before the developer picks it up for development, it should be confirmed. Identifying any ambiguity or contradictory requirements at this stage saves a significant amount of development and testing time.

(3) **Verify Requirements :** This objective demonstrates that one of the most important aspects of testing should be to meet the needs of the client. Testers examine the product and ensure that all of the stipulated standards are met. Developing all test cases, independent of testing technique, ensures functionality confirmation for every executed test case.

(4) **Validate test objects :** Testing ensures that requirements are implemented as well as that they function as expected by users. This type of testing is known as validation. It is the process of testing a product after it has been developed. Validation can be done manually or automatically.

(5) **Build confidence :** One of the most important goals of software testing is to improve software quality. A lower number of flaws is associated with high-quality software.

(6) **Reduce risk :** The probability of loss is sometimes referred to as risk. The goal of software testing is to lower the likelihood of the risk occurring. Each software project is unique and has a substantial number of unknowns from several viewpoints. If we do not control these uncertainties, it will impose possible hazards not only during the development phases but also during the product's whole life cycle. As a result, the major goal of software testing is to incorporate the risk management process as early as possible in the development phase in order to identify any risks.

☞ **Share information to stakeholders**

- One of the most essential goals of testing is to provide stakeholders with enough information to allow them to make educated decisions, particularly on the degree of quality of the test object.
- The goal of testing is to offer complete information to stakeholders regarding technological or other constraints, risk factors, confusing requirements, and so on. It can take the shape of test coverage, testing reports that cover specifics such as what is missing and what went wrong. The goal is, to be honest, and ensure that stakeholders fully grasp the challenges influencing quality.

☞ **Find failures and defects**

- Another critical goal of software testing is to uncover all flaws in a product. The basic goal of testing is to uncover as many flaws as possible in a software product while confirming whether or not the application meets the user's needs. Defects should be found as early in the testing cycle as feasible.

1.4.1 Testing and Debugging

Testing

Testing is the process of verifying and validating that a software or application is bug free meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

Debugging

- Debugging is the process of fixing a bug in the software. It can define as the identifying, analysing and removing errors.
- This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software.
- It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

☞ **Differences between Testing and Debugging**

Sr. No.	Testing	Debugging
1.	Testing is the process to find bugs and errors.	Debugging is the process to correct the bugs found during testing.
2.	It is the process to identify the failure of implemented code.	It is the process to give the absolution to code failure.
3.	Testing is the display of errors.	Debugging is a deductive process.
4.	Testing is done by the tester.	Debugging is done by either programmer or developer.
5.	There is no need of design knowledge in the testing process.	Debugging can't be done without proper design knowledge.
6.	Testing can be done by insider as well as outsider.	Debugging is done only by insider. Outsider can't do debugging.
7.	Testing can be manual or automated.	Debugging is always manual. Debugging can't be automated.
8.	It is based on different testing levels i.e. unit testing, integration testing, system testing etc.	Debugging is based on different types of bugs.

Sr. No.	Testing	Debugging
9.	Testing is a stage of software development life cycle (SDLC).	Debugging is not an aspect of software development life cycle, it occurs as a consequence of testing.
10.	Testing is composed of validation and verification of software.	While debugging process seeks to match symptom with cause, by that it leads to the error correction.
11.	Testing is initiated after the code is written.	Debugging commences with the execution of a test case.
12.	Testing process based on various levels of testing-system testing, integration testing, unit testing, etc.	Debugging process based on various types of bugs is present in a system.

1.4.2 Need of Testing

The main benefit of testing is the **identification and subsequent removal of the errors**. However, testing also helps developers and testers to compare actual and expected results in order to improve quality. If the software production happens without testing it, it could be useless or sometimes dangerous for customers.

What is Software Testing ?

- **Software testing** is a process of checking software applications and products for bugs and errors to ensure their performance is efficient. Testing in software engineering is a fundamental process of creating reliable – and usable – software products.
- That's what makes it necessary in guiding effective software development. Understanding the different types of software testing is what the need for Quality Assurance in software development stems from.
- Usually, companies that attempt to create software have dedicated software testers – an in-house means of software testing help. By detecting errors and mistakes that affect the quality of software, these testers can ensure that software products are worthy of being sold in the market. Thus, they guarantee a software's usefulness, and help turning software applications into end-products that perform the way their designers intended them to.
- Although the various types of software application testing can be more than a little long-drawn to fully appreciate, it's important for companies to be comfortable with why software testing is so essential in the first place.
- It's also important to note that the process of software testing is generally only a company's responsibility if they are software developers creating software for the market. If they're purchasing off-the-shelf software, especially one that's already been reviewed and established, software testing has already taken place. In that case, it would make repeating the software testing process redundant.
- They would also need to consider the purpose of testing if they're developing bespoke software for their specific needs. There can be various advantages for a company to invest resources in developing their own software, and that's something every business should evaluate on its own terms.

For now, let's list the important reasons as to why software testing must be considered mandatory:

(1) To gain customer confidence
(2) To check software adaptability
(3) To identify errors
(4) To avoid extra costs
(5) To accelerate software development
(6) To avoid risks
(7) To optimize business

1. To Gain Customer Confidence

- Software testing makes sure that the software is user-friendly. That makes it capable of being used by the customers it is intended for.
- Those who specialize in software application testing are familiar with the needs of customers, and unless a software can satisfy a customer's needs, it would be a practically useless investment.
- Different kinds of software have different kinds of customers. For instance, a Human Resources department in a company that needs to track its employees and their performance would have an entirely different software package from a hospital administrator trying to evaluate which hospital departments need more resources. That's why just like software developers, software testers also tend to specialize in certain kinds of software designs.

2. To Check Software Adaptability

- Given that there are many devices, operating systems, and browsers available today, it is necessary for software to be compatible with all platforms in order to offer users a smooth user experience.
- If the functionality of software is affected by the change of devices, it can count towards a negative user experience.
- Testing eliminates such errors in the performance while adding to the compatibility and adaptability of the software.
- Of course, one can design software that's exclusively limited for use on a desktop, but given that so much of networking and work is now also conducted on smartphones, how useful would that software really be? Likewise, fewer customers will choose a software that only efficiently operates on an operating system such as that of the Apple Mac.

3. To Identify Errors

- While it seems intuitive, it's important to remember that software testing is what helps rid products of errors before the product goes into the hands of a client.
- Regardless of how competent software developers and engineers may be, the possibility of glitches and bugs is always present in untested software.
- Testing will lead to better functioning of the product as hidden errors will be exposed and fixed.
- Even a single bug can be damaging for the reputation of a software developing house. It can take a long time to regain customer confidence, so it's much better and ultimately more convenient to ensure testing is being achieved.

4. To Avoid Extra Costs

- Tied to the problem of errors is the issue of the costs of reimbursing clients who have experienced glitchy software. These additional expenses can amount to a significant amount of damages, as not only has the client been dissatisfied with the product for which they have paid, but a client's time that they could have invested elsewhere was also rendered worthless.
- That's why it's a misconception to believe software testing costs a lot of stress, and that testers themselves "break" software.
- Testers do feed large amounts of data to software applications, but that's what is necessary in software testing: if software cannot support large data-loads, what purpose does it achieve for businesses or individual customers ?

5. To Accelerate Software Development

- Some companies also neglect software testing as they have spent too much time on the software development process, and need to quickly deliver the product to the client. While this may be a

problem with time-management in how a software development team is considering its work delegations, it's also an issue of conceptualizing software testing.

- Software testing and software development if run in parallel can accelerate the software development process and make it more efficient. Staging the design process in a way that makes it certain that both software testing and software development are happening simultaneously takes care to avoid such pit-falls in software development.

6. To Avoid Risks

- The worst thing about bugs and glitches is that it indicates a software is not secure. Especially when it comes to software that is meant for organizations, errors or loopholes can lead to vulnerability.
- This can lead to huge losses of information to competitor businesses, and can also lead to a lot of communication errors within an organization.
- Thus, besides just being about a software developer's reputation or the annoyance of encouraging bugs in an application's usability, bugs can also lead to privacy leaks and data gaps that can cost more than either of those things.

7. To Optimize Business

- Testing allows the end-product to achieve a higher quality standard before being made live.
- It also adds to the company's brand image as well as its profitability through reduced support costs.
- Essentially, every software developer's goal is customer retention, and every customer's goal is finding a service that's reliable and worth their money. Providing effective software thus, allows a business to become entrenched in a software provider's reputation.

1.4.3 Quality Assurance and Testing

Quality assurance (QA) testing is **the process of ensuring that your product is of the highest possible quality for your customers**. QA is simply the techniques used to prevent issues with your software product or service and to ensure great user experience for your customers.

What is QA testing?

Definition : Quality assurance (QA) testing is the process of ensuring that your product is of the **highest** possible quality for your customers. QA is simply the techniques used to *prevent* issues with your software product or service and to *ensure* great user experience for your customers.

Fig. 1.4.2 : Quality Assurance / Software Testing

Combining test automation and manual testing

- Your quality assurance (QA) testing methodology should ideally combine both automated testing and manual testing. The key is to determine which type of test is most relevant for each aspect and stage of the product.
- Manual testing allows you to cover a wide range of conditions and scenarios. The feedback of the QA testers regarding the experience and feel of the app will prove invaluable. Manual tests are preferable

for exploratory testing, usability testing and ad hoc testing. Ideally, manual tests should be performed by highly skilled and experienced testers who represent different end user profiles and use a wide range of devices and operating systems.

- To save time whilst testing, manual testing can be supplemented with frequent automated tests. Automation is the most appropriate solution when performing white box testing, load tests and other performance testing.
- Any test that needs to be performed repeatedly should be automated. Automated tests are practical and reliable and will help you to make sure the app performs adequately from a technical standpoint.
- Automation won't be a good fit for all your testing needs. You can supplement the manual testing performed in-house with crowdtesting. With this approach, your product can be tested on a much larger scale in a time-efficient manner.

☞ **Incorporating agile methodologies into software testing**

- Adopting a methodology that incorporates testing into a series of short development cycles is another best QA practice worth considering.
- Agile methodologies make sense in the context of developing mobile apps, given that these products typically have short development cycles and mobile users have extremely high expectations regarding functionality, quality and frequent updates.
- With agile methodologies, QA testing is part of the design and development processes, rather than being a separate phase. Test results are directly incorporated into the design and development processes, and quality becomes a guiding principle.

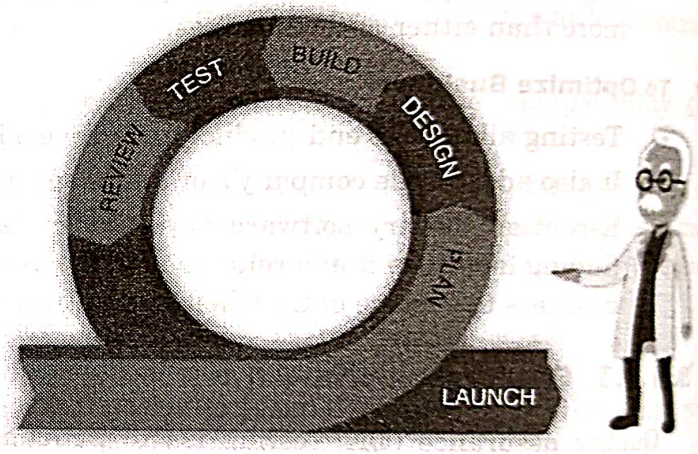


Fig. 1.4.3

- This is a collaborative approach that requires designers, developers, the QA team, and sometimes even users to communicate or work together. In order to facilitate collaboration, you can use a single repository for the app code.
- Your teams will go through a short design or development cycle, followed by a targeted quality control and testing phase for the new feature that was just added. Additional regression testing, security testing, and stress testing can be undertaken as needed. The outcome of this phase will determine what happens during the next design or development cycle.
- Leveraging automation will keep things moving once you adopt this approach. Test automation speeds up the targeted testing phases and helps you to move onto the next development cycle in a matter of hours or days. You will need to schedule manual tests after some key design or development cycles to incorporate feedback regarding the user experience and other key aspects of the app.
- You will need to create a framework for reviewing and using the data generated during the short testing phases. It's not enough to simply undergo functional testing - you need to incorporate feedback into the design and development process as early as possible.

☞ **Writing good test cases**

- Should developers write tests? On one hand, the agile approach is about ownership. Involving developers in the test case writing process will make QA one of their responsibilities.
- On the other hand, developers who create tests might become biased and write code that will pass the test without meeting other quality standards, or unconsciously create a test with limited coverage.

- For this reason, some teams create a test plan but then rely on outsourcing the process, or handing it over to dedicated QA engineers.
- Even though each test case should have a narrow focus, there should be cohesion in your test case suite. Your test case suite should have a scope that is adapted to the scale of your project.
- Customize and execute test cases in an environment that is different to the one used for development. Each test should be based on clear expectations and result in a measurable outcome.
- Break down each test case into a series of concise steps. Taking these steps will tell you whether or not a feature works. You can think of writing a test case as a series of actions associated with a question. When an action is taken, the automated test or human testers should answer a simple question to measure the success of the action.
- The instructions written for each test case should give testers a clear understanding of what they are expected to do. You can save time and get better results by providing test cases, instructions, and tutorials that aren't liable to misinterpretation. There are testing tools available to make this even easier.

☞ Continuous integration and continuous delivery

- Continuous integration (CI) and continuous delivery (CD) are strategies used in software development that complement the agile methodology. You can incorporate a continuous testing strategy to CI and CD.
- Without CI and CD, developers split up their work and assemble different segments of the code late in the development cycle. This can result in a lack of cohesion, compatibility, and issues with how the different segments of the code interact.

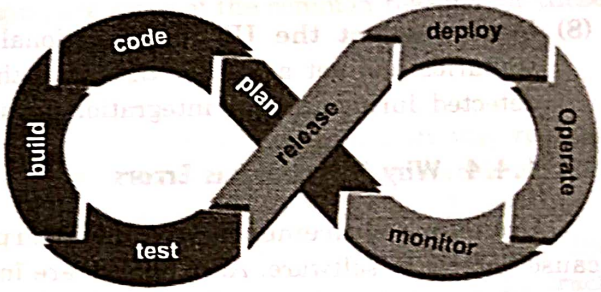


Fig. 1.4.4 : Continuous integration

- With continuous integration, the code is kept in a central repository. Developers work on making small changes to the code and upload small sections of code to the central repository regularly.
- You can incorporate quality management into this methodology by having a series of tests performed every time the code is updated. The new segments need to be tested, but you should also conduct regression testing to see how changes affect the main features of the product.
- Continuous delivery allows you to release new iterations of your product on a regular basis. This is a quick and efficient approach to addressing bugs and issues that affect the user experience.
- The key is to incorporate user feedback into your CI and CD processes so that issues can be quickly addressed and a new and improved version of your product can be released.
- Again, you will have to incorporate testing in your process, for instance by having crowd testers perform usability tests before a new major version of your product is made

☞ Developing your own QA testing strategies

- The right QA testing methodology will provide the information needed by your design and development teams to produce a quality app.
- Remember that software quality doesn't depend on testing but on the outcome of your QA tests and how you use this data.

Your approach to QA testing needs to be adapted to the product you are developing.

☞ QA testing best practices

(1) **Test one thing at a time :** tests should have clear objectives. Each test should focus on a feature or look at things like user interface or security.

(2) **Understand the types of testing on offer :** there are lots of different types of tests - from load testing to user acceptance testing (UAT) - so make sure you understand the differences and how to use them.

(3) **Use regression tests :** testing a main feature once isn't enough. New additions to the code repository can interfere with features that previously passed tests.

(4) **Report and track bugs :** determine how bugs will be reported and what kind of data is needed. Will you use an open-source bug tracking tool, or build one that's specifically suited to your workflow?

(5) **Leverage analytics :** decide which QA metrics to track. Keep records of every test conducted and use this data to determine where bugs are likely to occur. This data will help you to develop new tests that address problem areas.

(6) **Choose the right environment for tests :** try covering a wide range of scenarios, including different devices, OS and user profiles.

(7) **Use unit and integration tests :** unit testing will isolate each component of your app, while integration tests will assess how well each subsystem works. Run unit tests in parallel to save time, but don't move onto integration tests until you have ensured that individual components work like they should.

(8) **Don't neglect the UI :** use functional tests performed by human testers to perform end-to-end scenarios and get a feel for the UI of the app. It might be best to wait until you have fixed issues detected during unit and integration tests

1.4.4 Why Software has Errors

Unclear requirements and misinterpretation of requirements are the two major factors that cause defects in software. Also, defects are introduced in the development stage if the exact requirements are not communicated properly to the development teams.

Why Does Software Have Bugs?

One of the most prominent questions that almost all the Software Testers out there have in their mind is "Why does software have bugs?" and "How will these bugs occur?". This tutorial aims at answering these questions in simple terms.

In this tutorial, we will explore the top 20 reasons on "why Bugs occur in the Software".

What is a Software Bug?

A Software Bug is a failure or flaw in a program that produces undesired or incorrect results. It's an error that prevents the application from functioning as it should.

Why Does Software Have Bugs?

There are many reasons for the occurrence of Software Bugs. The most common reason is human mistakes in software design and coding.

Once you get to know the causes for Software Defects, then it will be easier for you to take corrective actions to minimize these defects.

Top 20 Reasons for Software Bugs

(1) Miscommunication or No Communication

- The success of any software application depends on the communication between stakeholders development, and testing teams. Unclear requirements and misinterpretation of requirements are the two major factors that cause defects in software.

- Also, defects are introduced in the development stage if the exact requirements are not communicated properly to the development teams.

(2) Software Complexity

- The complexity of the current software applications can be difficult for anyone with no experience in modern-day software development.
- Windows-type interfaces, Client-Server, and Distributed Applications, Data Communications, enormous relational databases, and sheer size of applications have all contributed to the exponential growth in software/system complexity.
- Using object-oriented techniques can complicate, instead of simplifying, a project unless it is well-engineered.

(3) Programming Errors

- Programmers, like anyone else, can make common programming mistakes. Not all developers are domain experts. Inexperienced programmers or programmers without proper domain knowledge can introduce simple mistakes while coding.
- Lack of simple coding practices, unit testing, debugging are some of the common reasons for these issues to get introduced at the development stage.

(4) Changing Requirements

- The customer may not understand the effects of changes or may understand and anyway request them to redesign, rescheduling of engineers, effects on the other projects, and the work already completed may have to be redone or thrown out, hardware requirements that may be affected, etc.
- If there are any minor changes or major changes, known and unknown dependencies, then the parts of the project are likely to interact and cause problems, and the complexity of keeping a track of changes may result in errors. The enthusiasm of engineering staff may be affected.
- In some fast-changing business environments, continuously changed requirements may be a fact of life.
- In these cases, the management must understand the resulting risks, and QA & test engineers must adapt and plan for continuous extensive testing to keep the inevitable bugs from running out of control.

(5) Time Pressures

- Scheduling software projects is difficult, often requiring a lot of guesswork. When deadlines loom and the crunch comes, mistakes will be made still.
- Unrealistic schedules, though not common, the major concern in small-scale projects/companies results in software bugs. If there is not enough time for proper design, coding, and testing, then it's quite obvious for defects to be introduced.

(6) Egotistical or Overconfident People

People prefer to say things like :

- 'no problem'
- 'piece of cake'
- 'I can whip that out in a few hours'
- 'It should be easy to update that old code'

Instead of :

- 'That adds a lot of complexity and we could end up making a lot of mistakes'.
- 'We do not know if we can do that; we'll wing it'.

- 'I can't estimate how long it will take until I take a closer look at it'.
- 'We can't figure out what that old spaghetti code did in the first place'.
- If there are too many unrealistic 'no problem's', then it results in software bugs.

(7) Poorly Documented C

- It's tough to maintain and modify the code that is badly written or poorly documented; the result is ***Software Bugs***. In many organizations, management provides no incentive for programmers to document their code or write clear, understandable code.
- In fact, it's usually the opposite: they get points mostly for quickly turning out code, and there's job security if nobody else can understand it ('if it was hard to write, it should be hard to read').
- Any new programmer working on this code may get confused because of the complexity of the project and the poorly documented code. Many times it takes a longer time to make even slight changes in poorly documented code, as there is a huge learning curve before making any code change.

(8) Software Development Tools

- Visual tools, class libraries, compilers, scripting tools, etc. often introduce their own bugs or are poorly documented, resulting in added bugs.
- Continuously changing software tools are used by software programmers. Keeping pace with the different versions and their compatibility is a major ongoing issue.

=> Read more on Software Development Tools

(9) Obsolete Automation Scripts

- Writing automation scripts takes a lot of time, especially for complex scenarios. If automation team's record/write any test script but forget to update it over a period, then that test could become obsolete.
- If the automation test is not validating the results properly, then it won't be able to catch the defects.

(10) Lack of Skilled Testers

- Having skilled testers with domain knowledge is extremely important for the success of any project. But appointing all experienced testers is not possible for all companies.
- Domain knowledge and the tester's ability to find defects can produce high-quality software. Compromise on any of this can result in buggy software.

Here are a few more reasons for Software Bugs. These reasons mostly apply for Software Testing Life Cycle :

(11) Not having a proper test setup (test environment) for testing all requirements.

(12) Writing code or test cases without understanding the requirements clearly.

(13) The incorrect design leads to issues being carried out in all phases of the Software Development Cycle.

(14) Releasing software patches frequently without completing the Software Testing Life Cycle.

(15) Not providing training to resources for the skills required for developing or testing the application properly.

(16) Giving very little or no time for Regression Testing.

(17) Not Automating Repetitive Test Cases and depending on the testers for manual verification every time.

(18) Not prioritizing test execution.

(19) Not tracking the development and test execution progress continuously. Last-minute changes are likely to introduce errors.

(20) Any wrong assumption made during coding and testing stages.

1.4.5 Defects and Failures and its Causes and Effects

We use the software in day-to-day life, such as at home, work, banking, shopping, etc. However, at times, the software does not work as expected. For instance, website not loading correctly, an error on a bill, a delay in credit card processing, are typical examples of problems that may happen because of errors, defects, and failures in the software. Today we will discuss the common terminologies that we use when software doesn't work as expected, i.e., ***Error, Defect, and Failure.***

- What is Error, Defect, and Failure?
- What are the causes of Errors in Software?
- Not all unexpected test results are failures
- What are Defects, Root Causes, and Their Effects?

Introduction to Error, Defect, and Failure

Let's try to understand the inter-relation between Error, Defect, and Failure:

It is well said by **Thomas Muller** *"A person can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or a document. If the execution of the defect in code happens, the system will fail to do what it should do (or something it shouldn't), which causes a failure".*

Let's take an example of an organization that developed a new application named "Staff Promotion System" for their annual appraisal. But employee satisfaction even after the appraisal was low. Because they considered it not up to the mark. The management, then, decided to analyse the root cause of this dissatisfaction.

Therefore, they backtracked the process and found that the software marked full day leave for the employees who reached office after 10 a.m. This was due to some coding errors. The tester also skipped these errors in coding. So, the software with this defect went to production. This, in turn, caused a general degradation and failure of the system.

Fig. 1.4.5 shows the interrelation between Error, Defect, and Failure.

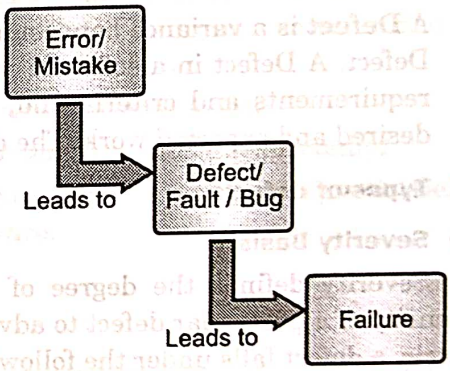


Fig. 1.4.5 : Error , Defect , and Failure

Errors

The **Error** is a human mistake. An Error appears not only due to the logical mistake in the code made by the developer. Anyone in the team can make mistakes during the different phases of software development. For instance,

- BA (business analyst) may misinterpret or misunderstand requirements.
- The customer may provide insufficient or incorrect information.
- The architect may cause a flaw in software design.
- People on the team can also make mistakes due to unclear or insufficient requirements, time pressure, lethargy, or other reasons.

Let us observe the basic types of errors in software testing :

☞ **Types of Error**

(1) **User Interface Error :** These are the errors that generally appear during user interaction with the system. Such as missing or incorrect functionality of the system, no backup function or reverse function available, etc.

(2) **Error handling error :** Any error that occurs while the user is interacting with the software needs precise and meaningful handling. If not, it confuses. Therefore, such errors are known as **error handling errors.**

(3) **Syntactic error :** Misspelled words or grammatically incorrect sentences are Syntactic errors and are very evident when testing the software GUI.

(4) **Calculation errors :** These errors occur due to bad logic, incorrect formulas, mismatched data type, etc.

(5) **Flow control error :** Errors concerning passing the control of the program in an incorrect direction where the software program behaves unexpectedly are flow control errors. Such as the presence of an infinite loop, reporting syntax error during run-time, etc.

(6) **Testing errors :** It implies the errors that occurred when implementing and executing the test process. For example, bug scanning failure, inefficiency in reporting an error or defect.

(7) **Hardware errors :** Such errors are related to the hardware device. Such as no availability and no compatibility with the device.

➡ Defect

A Defect is a variance between expected and actual results. An Error that the tester finds is known as Defect. A Defect in a software product reflects its inability or inefficiency to comply with the specified requirements and criteria and, subsequently, prevent the software application from performing the desired and expected work. The defect is also known as **Fault.**

☞ **Types of defects**

(I) Severity Basis

Severity defines the degree of impact. Therefore, the severity of the defect reflects the degree or intensity of a particular defect to adversely impact a software product or its operation. Based on the severity metric, a defect falls under the following categories:

(1) **Critical :** Defects that are **"critical"** require immediate attention and treatment. A critical defect directly affects the essential functionalities which can otherwise affect a software product or its large-scale functionality. For instance, failure of a feature/functionality or collapse of the entire system, etc.

(2) **Major :** Defects, which are responsible for affecting the main functions of a software product, are Major Defects. Although, these defects do not result in the complete failure of a system but may bring several primary functions of the software to rest.

(3) **Minor :** These defects produce less impact and have no significant influence on a software product. The results of these defects are visible in the operation of the product. However, it does not prevent users from executing the task. The task can be carried out using some other alternative.

(4) **Trivial :** These types of defects have no impact on the operation of a product. Hence, sometimes, we ignore and omit them. For example, spelling or grammatical errors.

(II) Probability Basis

One more angle to see a defect in a software application is the probability that it will occur, and chances that the user will find it. Depending on the likelihood or the possibility of a defect in a software product in terms of percentage is classified in the following ways:

- **High :** Almost all users of the application can track the presence of defects. This indicates a high probability.
- **Medium :** Half of the users can trace the presence of defects in a software product.
- **Low :** In general, no user detects it, or only a few users will be able to detect it.

(III) Priority Basis

Defects also have a business perspective comparison. The rectification of some defects must happen first. Likewise, some can solve at a later stage. Just like a business where everything happens according to the current need and demand of the market. Just like the probability base, priority classification also occurs in the following ways :

- **High :** The high priority defines the most critical need of a company to correct a defect. This should happen as soon as possible, in the same compilation.
- **Medium :** Medium priority defects are next to high priority. And any next version or release of a product includes addressing them.
- **Low :** This type of defect does not need to be corrected individually. Consideration of repairing these types of defects, along with any other defects is voluntary.

➡ Failure

- Failure is a consequence of a Defect. It is the observable incorrect behavior of the system. Failure occurs when the software fails to perform in the real environment.
- In other words, after the creation & execution of software code, if the system does not perform as expected, due to the occurrence of any defect; then it is termed as Failure. Not all Defects result in Failures; some remain inactive in the code, and we may never notice them. Failures also occur due to the following reasons:
- Any physical damage or overheating in the hardware can cause the whole system to fail.
- If the software is not compatible with the hardware, then also the system performs unexpectedly.
- Failures also happen by environmental conditions like a radiation burst, a strong magnetic field, electronic fields, or pollution could cause faults in hardware or software.

Not all unexpected test results are failures:

☞ Failures can also

(1) Happen due to a human error in interacting with the software, like entring an incorrect input value, or misinterpreting an output.

(2) Occur when someone deliberately tries to produce system failure or cause malicious damage.

(3) Happen because of the mishandling of test data, test environment, etc. Such conditions are known as defects, but they aren't actually a Defect.

(4) Sometimes, tests that result in undetected defects can also cause failure.

☞ What are the causes of Errors in Software?

Here we discuss some possible causes of these errors.

Time pressure : At times, software development happens under limited / insufficient resources with unrealistic deadlines. Developers do not have enough time to test their code before delivering it to the testing team. Which, in turn, introduces errors?

1. **Human fallibility :** Human beings are prone to make mistakes. It would be foolish to expect the software to be perfect, and without any flaws in it! Ironically, we have not yet discovered any other non-human agent that can develop software better than humans. Therefore, we continue to rely on human intelligence to develop software. Thereby, increasing the possibility of errors in it.

2. **Inexperienced or insufficiently skilled project participants :** Allocation of correct work to the correct resource is fundamental for the success of any project. Team members should be assigned a task according to their skills and abilities. An inexperienced project participant may make mistakes if they don't have proper knowledge of the work. For example, a resource having a good understanding of the database but having limited knowledge of HTML/CSS is not suitable for designing a website.
3. **Miscommunication between project participants :** Improper coordination & poor communication between various departments in a project can result in disrupted progress. Conflicts can arise each time a project participant misinterprets or misunderstands the words or actions of another. For example, the business analyst does the requirement gathering, and then some other team member does the documentation of the requirements. Any miscommunication between the two can lead to incomplete/wrong requirement document, which, in turn, affects the design of the project.
4. **The complexity of the code, design, architecture, or the technology to be used :** As the complexity of the program, concerning code, design or technology increases, the software becomes more critical and more bugs appear. It is because our brains can only deal with a reasonable amount of complexity or change. Our minds may not be able to process complex information like the form of design, architecture or technology, etc. Therefore, resulting in low quality and erroneous coding.
5. **Misunderstandings about intra-system and inter-system interfaces :** There are high chances of error while establishing an intra-system, and inter-system interfaces. Let's try to understand what is intra-system and inter-system interfaces mean:-
6. **Intra-system interface :** It implies the integration of different modules/features within a system/application. For example, in an online shopping portal, we have three modules: online order, shipping, and supply chain. These three modules form the intra-system for the online shopping portal. When these different modules are combined, there is a possibility of errors in the final product.
7. **The inter-system interface :** It is the compatibility of an application with other applications when operated together. For example, compatibility between smartphones and tablets while data transfer via Bluetooth.
8. **New, unfamiliar technologies :** Sometimes, the developers and testers need to develop an application using a technique that is unknown to them. Lack of proper knowledge and understanding can lead to errors. At that time, they require a certain level of R & D, brainstorming, and training to reach a reliable solution.
9. **Environmental conditions :** Natural disasters, such as outbreaks of fires, floods, lightning, earthquakes, etc. can affect the computers. For example, a system may not work correctly if the software inside is affected by radiation, electromagnetic, or pollution.

1.5 TOTAL QUALITY MANAGEMENT (TQM)

- Total Quality Management (TQM) struggles to location a corporation for superior customer gratification, effectiveness and effectiveness.
- TQM principle intends to view internal and external customers as well as internal and external suppliers for each process and project and entire organization as whole
- Conference Our Customer's Necessities
- Doing Effects Right the First Time; Freedom from Failure (Faults)
- Regularity (Reduction in Distinction)
- Continuous Improvement . Excellence in Everything We Do

☞ **Attention on Customer**

- Classify and meet customer wants
- Stay altered to altering needs, e.g. smartness styles

☞ **Nonstop Improvement**

- Nonstop knowledge and problematic solving, e.g. Kaizen, 6 sigma
- Plan D StudyAct (PDSA)

☞ **Benchmarking**

Operative Empowerment

- Allow all employees; external and internal clients
- Team Method
  - Teams designed around processes 8 to 10 people
  - Meet weekly to consider and solve problems

☞ **Use of Excellence Tools**

(1) Continuing exercise on investigation, valuation, and alteration, &operation tools

(2) Studying performs at "best in class" businesses

☞ **Other Quality Management through Stastical Process Control**

- **Numerical process control** is a group of tools that when used composed can result in procedure stability and alteration reduction
- **Statistical Process Control (SPC)** is an engineering standard organization for computing and governing quality through the engineering process.
- **Superiority data in the form** of Product or Process depths are attained in real time during developed.
- **This data is then plotted** on a graph with pre-determined control limits. Control limits are resolute by the competence of the procedure, whereas specification limits are determined by the client's needs.
- Quality Planning at all level
  - Quality Planning at Organization level.
  - Quality Planning at Unit level.

☞ **Statistical Process Control**

Monitoring production procedure to identify and avoid poor quality

☞ **Sample**

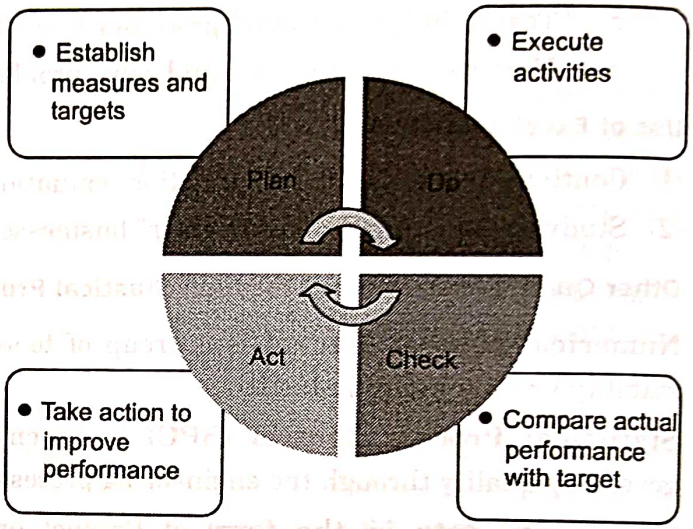
- Subset of items created to use for examination Quality Management through cultural change defines quality improvements as a cultural change driven by management. It involves
- Identifying areas in which quality can be improved depending upon process capability measurements and organizational priorities.
- Identifying teams representing different functions and areas for quality improvement can help in setting the change of culture.
- Setting measurable goals in each areas of an organization can help in improving processes at all levels.
- Giving recognition to achievers of quality goals will boost their morale and set positive competition among team leading to organizational improvements.
- Creating a quality culture within an organization is gradually known as one of the primary situations for the successful execution of Total Quality Management

1.5.1 Continual Improvement Cycle

UQ. Explain continual improvement cycle with neat labeled diagram of plan-Do- check-Act (PDCA) framework. **SPPU - Nov./Dec.19 (End Sem)**

- Continuous improvement is an ongoing effort to improve products, services or processes. These efforts can seek "incremental" development over time or "development" improvement all at once.
- Amongst the most extensively used tools for nonstop development is a four step quality model the **plan do check act (PDCA)** cycle, also recognized as Deming **Cycle or Shewhart Cycle:**

1. **Plan :** Identify and chance and idea for change.
2. **Do :** Implement the alteration on a trivial scale.
3. **Check :** Use data to investigate the outcomes of the change and regulate whether it made a alteration.
4. **Act :** If the alteration was effective, instrument it on a wider scale and nonstop assess your results. If the adjustment did not work, activate the cycle again.



(1A8)**Fig. 1.5.1 : Continual Improvement plan-do-check-act (PDCA) cycle**

Quality in Different Areas

- Quality attributes of various products in different areas..
- Different domains need different quality factors. They may be derived from customers/users of the domains. Some examples of some domains showing customer expectations in terms of quality of various products.
- Products and expected attributes

Table 1.5.1 : Products and expected attributes

Products/Service Category	Expected Attributes
Airline Industry	On time arrival/departure, low cost service, comfortable journey etc.
Healthcare Industry	Correct treatment, minimum wait time, safety and security.
Food Service Industry	Good Product, good taste, fast delivery, clean environment.
Military Services	Rapid deployment, Security
Automotive Industry	Clear communication, faster access, cheaper service

1.6 BENCHMARKING AND METRICS

UQ. Give an example quality in different areas of software growth, and Benchmarking and metrics for the same? SPPU - Aug .18(In Sem)

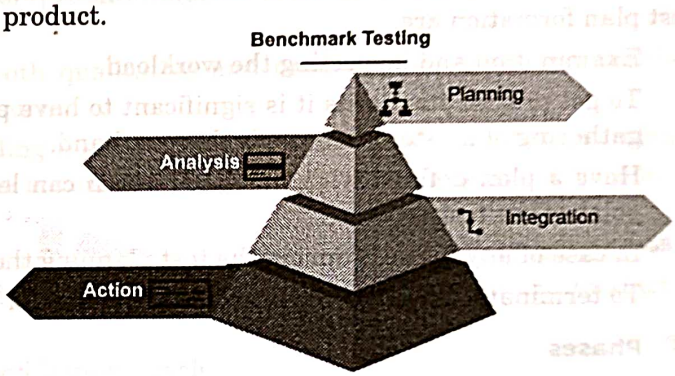
UQ. Explain following terms: (i) Benchmarking (ii) Metrics (iii) Defect SPPU - Oct.19 (In Sem)

Benchmarking

- It is important concept in Quality Function Deployment (QFD). It is concept of qualitative /quantitative metrics or measurable variables which can used to access product quality on several scales against benchmarks..
- Benchmarking include price of product paid by customer, time required to acquired to acquire it, customer satisfaction, defects, attributes and features of products.
- Benchmarking resources to amount the greatest performs of important productions, and learn and familiarize them for use in your business.

Metrics

- They are defined for collecting information about the products capabilities, process variability and outcome of the process in terms of attributes of the product.
- It is relative measurement of some parameters of a product which are related to the product and processes used to make it.



(1A9)**Fig. 1.6.1 : Benchmarking, Metrics, Defect**

What is Benchmark Defect ?

- Benchmark Testing is used to validate the product value of the system software application or the web-based software application, against the regulations set by the development team or the system architecture, in order to maintain or enhance the reputation of the application.
- This product value determination process involves various testing flow, depending on the regulations fixed for the application. In most cases, it is achieved by exposing the application for a round of performance testing.
- It essential be repeatable so that the presentation extents can be arrested, and the difference needs to be renowned and it would be only a few percents each time the test is existence run. This supports in variations to be made to the presentation in order to regulate if the performance can be better or corrupted.
- This testing can also be combined with security testing. For example, we can consider standard testing firewalls. These requirements that the system can be merged with different security violations simultaneously and executed so that the benchmark for performance can be determined.
- As a part of the Software Development Life Cycle, it can have both developers and database administrators involved which helps in getting the current performance and then it helps in improving the performance.

How it is performed ?

It must be performed in the same situation and the same situations as predictable so that a difference factor can be attained. It supports in setting up a standard and doing further processes therefore. The pre fundamentals for this involve:

- It should be confirmed that all software modules are working precisely.

- Before the difficult starts it would be check that all functioning scheme in forms and formations are occupied care of.
- The test bags should be well distinct and separated as essentials as per their dissimilar functionalities.
- While the difficult is being approved upon it should be chequered for its constancy and control events, as they are significant factors to achieve standard testing.
- Every time the tests are achieved it should be done in the same situation and under the same circumstances.
- The software and hardware components must always be in line with the requirements or specifications of the production environment as the benchmark should be set for the production. The challenging should be done as if it is done in manufacture.

After this, it is significant to find out which kind of standard test you would like to carry advancing. It can be whichever an organization standard which helps in outcome the quantity competences under positive quantified circumstances.

The second variety is the request benchmark which assistances in discovery the quantity capabilities of the database under conditions that look like the production.

☞ **Creating a Standard Test Plan**

When successful for it, this is the most significant step which needs to be moved correctly. The steps for test plan formation are:

- Examination and inspecting the workload.
- To get precise standards it is significant to have preceding standards and hence it is compulsory that a gathering of all stored events is there at hand.
- Have a plan defined with timelines which can let the user know the time required and the terminal point of the test process.
- In case of any failures during the test planning then a backup plan must be created.
- To terminate the last process an authority should be decided.

☞ **Phases**

It involves four phases :

1. Planning Phase	2. Analysis Phase
3. Integration Phase	4. Action Phase

1. Planning Phase

- In this phase, it is important to identify and prioritize different standards and requirements.
- It helps in deciding different benchmark criteria that help in setting up a standard and helps in delivering standard software in the least.

2. Analysis Phase

- The analysis phase assistances in receiving a quality product and helps in classifying the root cause of any subjects which were handled previous.
- By responsibility this you can naturally classify some alterations which are wanted and set areas for the difficult process. This supports the difficult process and helps in receiving quality.

3. Integration Phase

- Integration helps in getting outcomes from everyone where they share it and a concerned person helps in getting approval.

- Once everything is integrated the functionalities can be decided and accordingly function goals can be set.

4. Action Phase

- In this level, the actual work is done. All the above steps can lead to develop a test plan and document the changes that are needed.
- Once a plan is generated implementation changes can be made and once the work is started then the progress can be monitored and accordingly the plan can be executed till completion. The above points can be run continuously until the testing is completed.

1.7 PROBLEM SOLVING TECHNIQUES

- Techniques indicate more about a process used in measurement, analysis and decision making during problem solving.
- Improving quality of products and services offered to customers requires methods and techniques of solving problems associated with development and processes used during their lifecycle.
- An organization must use metrics approach of process improvement because it needs to make quantitative measurements.
- These measurements can be accomplished by both qualitative and numerical methods but problem definition becomes easier when we put some measures.
- Qualitative problem solving refers to understanding a problem solution using only qualitative indexes such as high, medium, low etc. depending on the something is improving from present status and so forth.
- Measurable problematic solving necessitates requirement of careful events of exact measures in numerical terms such as the cost of the 32.5% during last quarter or time required to one product is reduced by 32 minutes.
- It must follow define, measure, monitor, control and improve cycle.

1.8 PROBLEM SOLVING SOFTWARE TOOLS

- Tools are an organizations analytical asset that assist in understanding a problem through data and try to indicate possible solutions.
- Quality tools applied for solving problems face by projects and functional terms while improving quality in organization.
- Tools may be hardware/software and physical/logical tools.

☞ **Advantages of Using Software Tools for analysis and decision making**

(1) Accuracy and speed of the tools is much higher compared to performing all transactions and calculations manually.

(2) Decision support offered of the tool is independent of personal skills and there is least variation from instance to instance.

(3) Tools can be integrated with other systems to provide a systematic and highly integrated means of solving problems.

☞ **Disadvantage of Using Computer Tools for Analysis and Decision Making**

Tools may mean more cost and time to learn and implement.

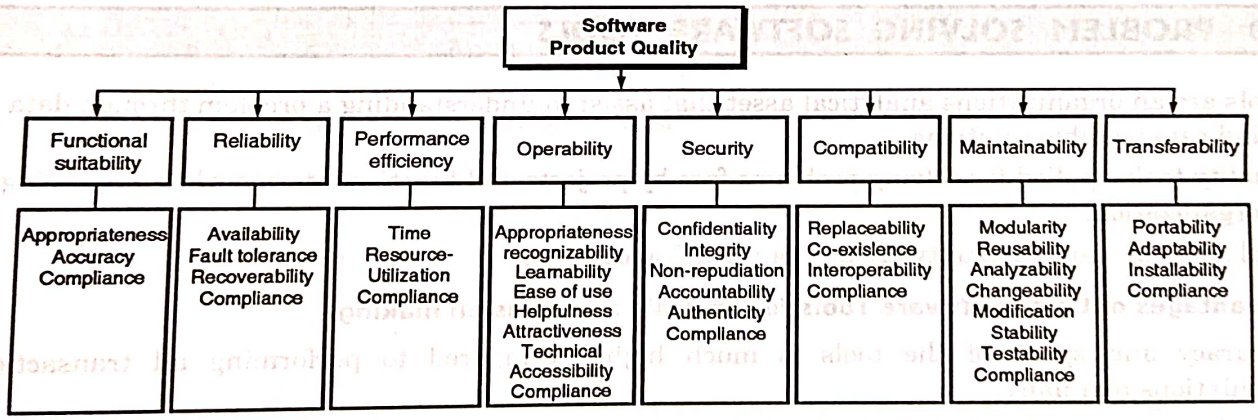
1.9 SOFTWARE QUALITY

The degree to which a scheme, component, or process chances definite requirements.

1. Quality is suitability for use.
2. Conformance to specification.
3. Transparency of service delivery
4. Achieving desired results
5. Continuous Improvement
6. Competitive advantage
7. Added value for society Best value for price
8. Cost effectiveness
9. Performance measurement
10. Satisfaction of stakeholders
11. Doing the correct things
12. Doing things right & doing the right effects right

1.9.1 Quality in Software Engineering

- In broader terms, the software quality definition of "fitness for purpose" refers to the satisfaction of requirements. But what are requirements ? Necessities, also called user levels in today's agile terms, can be considered as practical and non-functional. Functional requirements mention to precise functions that the software should be able to complete.
- For example, the facility to print on an HP Inkjet 2330 printer is a useful prerequisite. However, just since the software has a confident function or a user can comprehensive a task consuming the software, does not mean that the software is of respectable quality.
- There are probably many instances where you've used software and it did what it was supposed to do, such as find you a flight or make a hotel reservation, but you thought it was poor quality. This is because of how the function was executed. The displeasure with "how" signifies the non-functional necessities not being met.
- For this purpose the International Organization for Standardization (ISO) developed ISO 25010 as a model for specifying non-functional necessities. The classical shown below exemplifies the classification of non-functional requirements.



(1A10)**Fig.1.9.1 : Software product Quality Requirements and Evaluation (SQuaRE) Quality model and guide**

Why Non-Functional Quality Components Are Important?

Adequate non-functional necessities such as presentation, ease of use and learn aptitude first necessitates requiring and important. Only then can they be contented, and sufficient them can be even more problematic than substantial functional necessities. So, what does this mean for you? Let's examine the following non-functional characteristics using the same printing example :

- **Functional Suitability (functional appropriateness)** Does the function facilitate the completion of the user's task(s) and objectives? If the user doesn't want to print on that printer or wants to print a PDF but isn't given those options, then maybe not.
- **Performance Efficiency (time behaviour)** – Does the printer purpose return inside three seconds?
- **Compatibility(interoperability)** – Can the operator print over a variety of networks and printers and on processors with unlike operative schemes(Windows and Mac)?
- **Usability (learnability)** – Can the operative figure out in what way to print or will it income a rocket expert?
- **Consistency(recoverability)** – When the printer is released in the internal of printing a task, is the user learned?
- **Security (nonrepudiation)** – Is contiguous a record that the printer published the file efficiently?
- **Maintainability (testability)** – Can test events be measured for the print function?
- **Portability (adaptability)** – Can the software mechanically familiarize to new printer models, or an update in printer driver software? Can the print purpose deliver shortcuts for highly refined users?

Now that we have an accepting of non-functional requests, let's examine the excellence lifecycle in the diagram beneath. Looking at the three circles under, internal excellence represents quality that you wouldn't realize and is stately by internal possessions such as code feature. Exterior worth signifies what we have conversed above in the non-functional superiority model and is typically measured by actual execution of the code and examination of software behaviour.

1.10 CONSTRAINTS OF SOFTWARE PRODUCT QUALITY ASSESSMENT

Requirement specification is made by business analyst and system analyst. Tester may or may not have direct access to the customer and may get information though requirement statements, queries answered etc. either from customer or business analyst. There are few limitations of product quality assessment in this.

- Software is virtual in nature. Software products cannot be touched or heard.
- There is huge communication gap between users of software and developers/testers of the product.
- Software product is unique in nature. Similarities between any two products are superficial ones.
- All aspects of software cannot be tested fully as member of permutations and combinations for testing all possibilities tend to infinity.
- A software program in the same way every time when it is executing some instruction.

☞ **Quality Tool**

(1) Quality function deployment (QFD)
(2) Taguchi techniques
(3) Pareto charts
(4) Process charts
(5) Cause & effect diagrams
(6) Statistical process control (SPC)

1.11 CUSTOMER IS A KING

- External Customer - outside the organization (people who pay the bills.)
  - End-user customers
  - Manufacturer (OEM) for suppliers.
- Internal Customer - people within your organization who receive your work
- In many situations, producers have multiple customers and therefore find it useful to identify "core customers"
- A customer's perception is their reality.

- It's easier to keep your customers happy than attract new ones.
- Complaints spread like wildfire on the internet.
- Without customers we don't have a business.
- Brands win or lose by how well they wow customer.

TQM's Customer Approach

- the customer defines quality."
- "the customer is continuously right."
- "the customer always approaches first."
- "quality creates and trimmings with the customer."

1.12 QUALITY & PRODUCTIVITY RELATIONSHIP

- Productivity is the relationship between a given amount of output and the amount of input needed to produce it. Quality affects productivity.
- Productivity is tool of quantity that defines the competence of the association in relations of the ratio of output created with esteem to inputs used.
- Quality must improve productivity by reducing wastage.
- Improvement in quality directly leads to improved productivity.
- Cost reduction is possible by improved quality.
- Proper communication between management and employee is essential.
- Quality improvement leads to cost reduction.
- Employee involvement in quality improvement.

1.13 REQUIREMENTS OF PRODUCT

1. **Stated/Implied Requirements :** Functional and non functional requirements stated by customer.
2. **General/Specific Requirement :** Requirements are generic in nature.
3. **Present/Future Requirements :** Present when application is used and future for required after some time span.
4. **Primary Requirements :** Must /must not be requirements. Customer pay for this
5. **Secondary Requirements :** Should/Should not be requirements.
6. **Tertiary Requirements :** Could/could not be requirements.

1.14 ORGANIZATION CULTURE

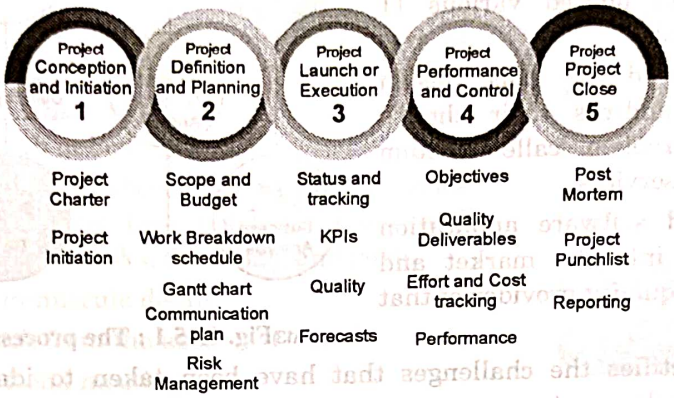
Quality culture is set of group standards that guide how developments are made to average working practices and resultant outputs. An organization's values can support entities at all levels make improved and more accountable choices involving issues of quality.

1.14.1 Following Features Emerged as Indicative of a Quality Culture Project Management

(1) Academic Ownership of quality.
(2) Quality culture is primarily about the behavior of stakeholders rather than the operation of a quality.
(3) A quality culture places students as center.
(4) Quality policy is more like a philosophy of any organization. Unless it is implemented, it is of no value.

(5) Quality policy is like a recipe book with a list of ingredients but no cooking instructions. Thus, any quality initiative needs to be managed as a project.

(6) A quality product project can use various tools, techniques, methodologies of project management. The success of a quality program depends the way it has been implemented.



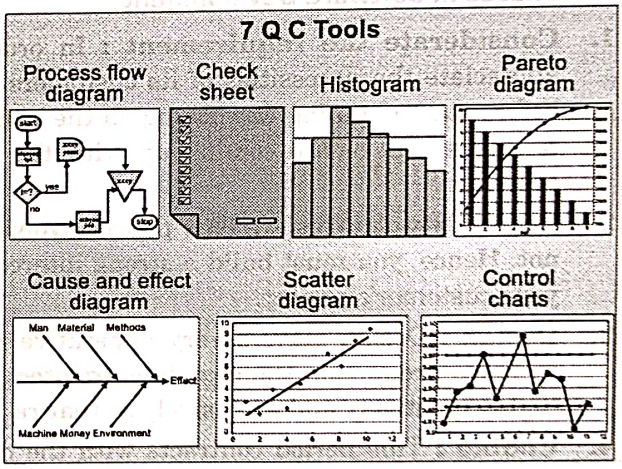
(1A11)**Fig. 1.14.1 : Steps in Project Management**

1.14.2 Sustainability and Quality Management

- Process efficiency, quality metrics, reduced waste etc. have started fascinating Sustainability practitioners nowadays. Sustainability teams are undergoing training & certifications to make themselves conversant with quality management.
- They are keen to use various tools & techniques of quality management in sustainability.

Seven Quality Control Tools

- Like Quality, Sustainability also has a strong attention on people. It takes into account quality of working life and employee satisfaction also.
- ISO 26000 brands a more thoughtful joining between people and excellence organization systems.
- Quality management is going to create value in Sustainability space in a big way.
- Thus, the trend of convergence between quality management and Sustainability in the offing.



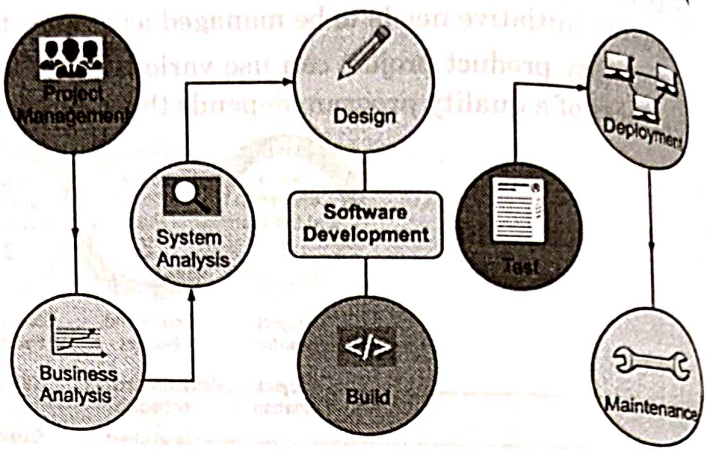
(1A12)**Fig. 1.14.2 : 7 QC Tools**

1.15 SOFTWARE DEVELPOMENT PRCOESS

UQ. What is impact of defect in different phases of software development ? **SPPU - Nov./Dec.19 (End Sem)**

- When we talk about the software development process, it requires establishing a connection between the idea and the approach to implement that idea.
- Thus, a proper process needs to be followed while implementing that idea for software development process. Over the years, the evolution of software development process came into existence for building customers business to reach success level.

- Since its evolution, the software industry is growing at a very fast pace because of the global level increase in market demand for software. Many players entered in the software industry and offered various IT solutions to their clients.
- Many companies around the globe design custom software to address their client's particular business problem, called custom software development services.
- Businesses introduced software application development services into the market and claimed to be the best quality provider of that service.



(1A13)**Fig. 1.15.1 : The process of Software Development**

- Software industry rectifies the challenges that have been taken to identify the gap and risk in managing software development.
- These steps are taken to confirm the quality of software processing at each level. It basically refers to attempt for creating the software application to meet the need of client's business.

1.15.1 Different Phases Included in Software Development Process

When it emanates to software development services, there are several approaches and models included in it. Here, we are conversing few major periods of software development methodologies:-

☞ **Phases in Software Development**

1. **Considerate the requirement :** In order to grow fully useful software, one necessity obviously appreciate the necessities of its customers. This is the most significant concern that should be taken, before start planning & working on the entire development process. A developer faces many challenges & alterations before coming up with the final plan as per the requirement & brief shared by the customer.
2. **Feasibility Analysis :** This phase involves scrutinizing the project whether it is feasible to work or not. Hence, you must build a strong interconnection between the project requirements and the need of your customer's project.
3. **Design :** Design plays a very imperative role in attracting visitors and generating more traffic. This includes production the complete architecture of the software. Henceforth, the design must be formed highly creatively and exclusively so that regulars can get best consequences.
4. **Coding :** This period contracts with the designer or programmers. It completely be contingent upon customer's choice in which programming language do they need to establish their project. It involves the transformation of design into coding through the help of a programmer.
5. **Software Testing :** Once the code is generated; it undergoes through various testing phases. This determines whether the product established is original or not. At this phase, any kind of bug or glitches found can be fixed.
6. **Maintenance :** Last but not the least, high maintenance is required in the project before & after it is being delivered to the user. The developer checks if the software is working fine before delivering it to the users & provides after sales service support & assistance in relations of the working of the software to the end users.

1.15.2 Seven Ways to Develop Software Development Process

- There are some ways through which the software development processes can be improved effectively such as shown in Fig. 1.15.2.
- Evolution in software development came with years of refined, tested and innovative processes. The demand for software development process has a view of the market and results in the growth of businesses.
- Life has never been better, therefore software provides human with the technology that helps in better and efficient accomplishment of tasks, functions, and goals.
  - No obligation for minuscule detail
  - Obtain the feedback from the user
  - Fewer people to attend meetings
  - Small projects and team
  - Empower your user
  - Implementation of features
  - Detain of price & flexibility
- Following Software Development Process Models are used:
  1. Waterfall Development Approach/Model.
  2. Iterative Development Approach/Model
  3. Incremental Development Approach/Model
  4. Spiral Development Approach/Model
  5. Prototyping Development Approach/Model
  6. Rapid Development Approach/Model
  7. Agile Development Approach/Model

(1A14)**Fig. 1.15.2 : Seven ways to develop Software Development Process**

1.15.3 Types of Product

- Life Affecting Products.
- Product Affecting Huge Some of Money.
- Products Which can be Tested only by Simulators: Example Space Research
- Other Products.

1.16 CRITICALITY DEFINATIONS

From User's Perspective :

Failure of product disrupts entire business. Products failure affects business.

From Developer's Perspective

This classification defines the complexity of the system on the basis of development capabilities required.

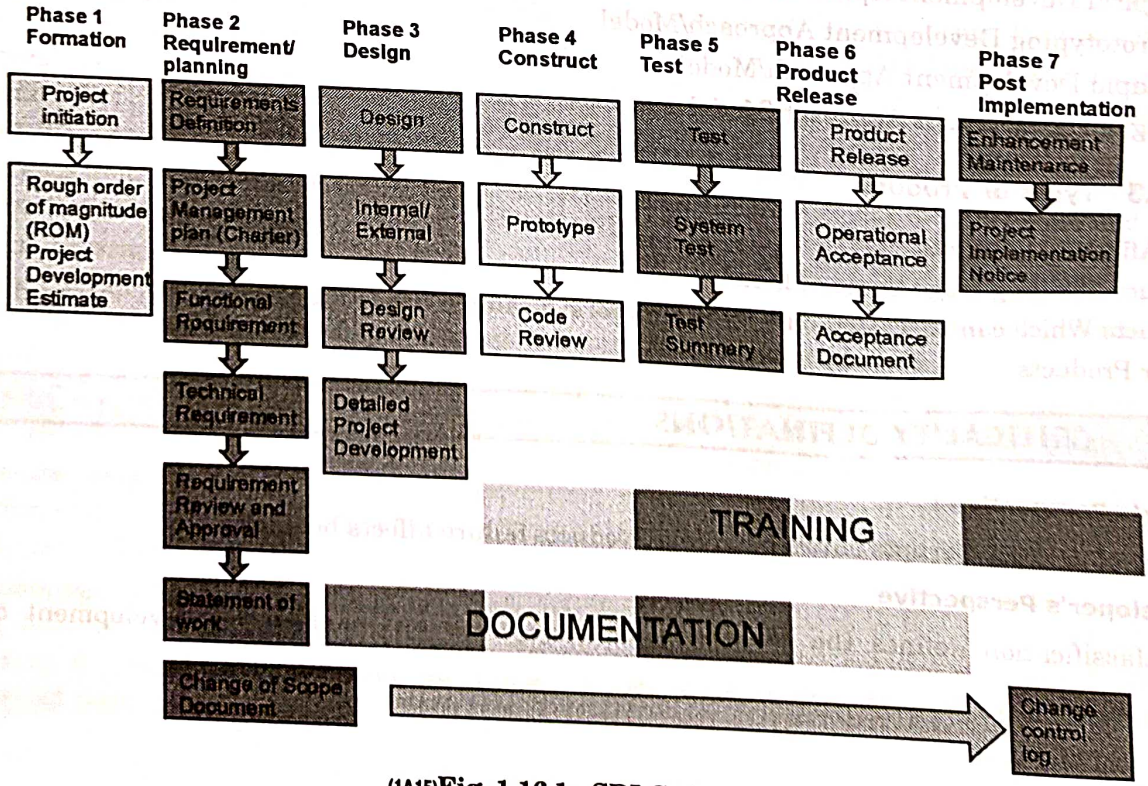
1.16.1 Problemstic Areas of SDLC

Software Improvement Life Cycle Organizations

What is SDLC ?

- Every day, software engineers and specialists similar have to submerge themselves into the subtleties of the best **Software Development Lifecycle (SDLC)** procedure and method to progress and transport software in optimal environments. But, what is SDLC?
- In the meekest positions, SDLC practices deliver a methodical context to initiative, progress and offer software requests, from start to end. It is a sequence of phases that proposal a basis for the software growth process.
- Having a organization to grow software is important, which is why there are numerous software development organizations obtainable to choose from.
- It is gradually significant for software engineers to excellent the accurate **SDLC** model that meets precise necessities and concerns of the scheme to drive achievement. In this article, we go into the particulars of **SDLC** organizations, their significance, their benefits, difficulties, and everything in between.
- To a definite extent, **SDLC** organizations can be assumed of like a checklist of the dissimilar stages that must be completed to progress and deliver effective software requests.
- All **SDLC** procedures share a common ground of separate stages that include development, analysis, strategy, structure, difficult, organizing, and conservation. These **SDLC** stages provide the outline of what a software request project requires.

In the succeeding section, we are going to travel how software development lifecycles impact the software growth process.



(1A15)**Fig. 1.16.1 : SDLC phases**

☞ The Software Development Process

- The software development method, as with all excessive projects, starts with an idea. It earnings planning, preparation, and management of stages and team members to influence a goal.
- SDLC is a mapped out, planned context that classically surveys the following universal stages to transport high quality software application.

☞ Formation phase

This basic, initial phase is the beginning of an idea for a resolution that advances an existing solution or develops an entirely new one. It helps define the magnitude of the project to plan resources.

☞ Requirement/Planning Phase

In this phase, supplies are gathered to formulate a design plan for the software application solution. This phase involves a systematic examination to evaluate user requests, feasibility, growth, enhancements, and more. It is very significant to contain documentation to refine necessities and keep a record of the solution's development. This phase includes the formation of a project charter which describes technical and functional necessities.

☞ Design Phase

This phase is intensive on the design feature of the software application resolution in relations of the designated technical and functional necessities and the effects of the exhaustive enquiry of the software's feasibility.

☞ Development Phase

This phase is the meat of the software growth process. In this phase, software engineers are exclusively attentive on building an example of the solution to attain a code evaluation and finally create the solution itself. The team everything on converting software conditions into a employed and dependable solution.

Testing Phase

This serious phase tests the software to confirm that entirety works as it planned. In the analysis phase, software engineers are able to identify deficiencies, bacteria, and errors in the software solution and eventually have a quality product that meets business prospects. **Quality Assurance (QA)** experts perform a series of tests to assess the position of the solution.

☞ Release Phase

Once the software application is entirely established and tested, it transfers to the release phase. In this phase, the software goes aware and is unconfined to the end operator for definite use of the product.

In essence, the software is fully operative in a live setting where end users operate it.

☞ Maintenance Phase

- This post issue phase is tasked with observance the software entirely operational, informing it to meet value standards, and improving it through its life to confirm it remains to entice and recall users.
- The software development process sets the tone and describes a goal from which developer's kick-start a project. Eventually, succeeding a software development process is proposed to develop software earlier and with a few hiccups as possible.
- Now that we've enclosed the universal **SDLC** phases, let's measure how imperative it is to follow software development procedures in an IT environment.
  - Problems with requirement phase.
  - Requirements change very frequently.
  - Unique product is built in any time.

- Product nature is intangible.
- Inspection can be exhaustive.
- Requirements are not easily communicated.
  (1) Technical Requirements.
  (2) Economical Requirements.
  (3) Operational Requirements.
  (4) System Requirements.

1.17 SOFTWARE QUALITY MANAGEMENT

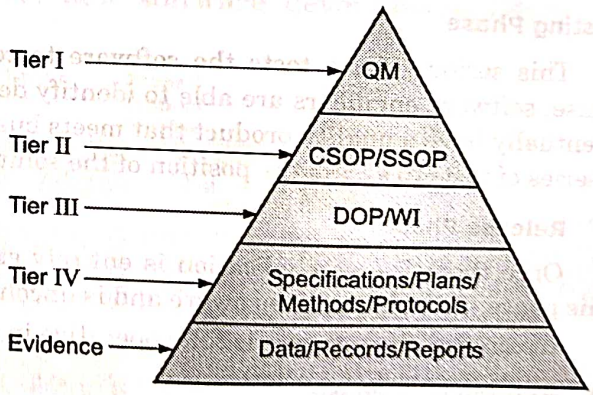
UQ. With respect Quality Management Systems explain the following?
(i) Quality Management Systems Structure
(ii) Pillars of Quality Management System
(iii) Important aspects of quality management
SPPU - Aug. 18(In Sem)

- Quality management involves management of all inputs and processing defined so that the output from the process as per defined criteria.
- It handles three levels of problems:
  (1) Correction.
  (2) Corrective Actions.
  (3) Preventive Actions.
- Quality Management System Structure
  (1) 1st Tier-Quality policy
  (2) 2nd Tier-Quality objectives
  (3) 3rd Tier-Quality Manual

The Importance of Hierarchical Organization

An organization is spirited when working with controlled documents. A suggested hierarchy for QMS documentation management is:

1. Quality Manual
2. Policies
3. Procedures
4. Work Instructions
5. Lists
6. Checklists
7. Forms



(1A16)**Fig. 1.17.1 : QMS Tier structure**

1.17.1 Pillars of Quality Management System

UQ. What are the pillars of Quality Management System ?
SPPU - Oct. 19 (In Sem)

- Quality processes/Quality Procedures/work instructions
- Guidelines and standards
- Formats and Templates

Steps for the Creation of an Effective QMS

The steps required for the conceptualization and implementation of a QMS include the following:

1. Define and Map Your Processes

- Process maps creation will force the organization to visualize and define their processes. In the process, they will define the interaction sequence of those processes.
- Process maps are vital for appreciating the responsible person. Define your main business process and converse the flow.

2. Define Your Quality Policy

- Your Quality Policy communicates the duty of the organization as it is about the quality. The mission may be what customers need, a quality mission.
- When constructing quality management system, consider the commitment towards customer focus. It may be Quality, Customer Satisfaction, and Continuous Improvement.

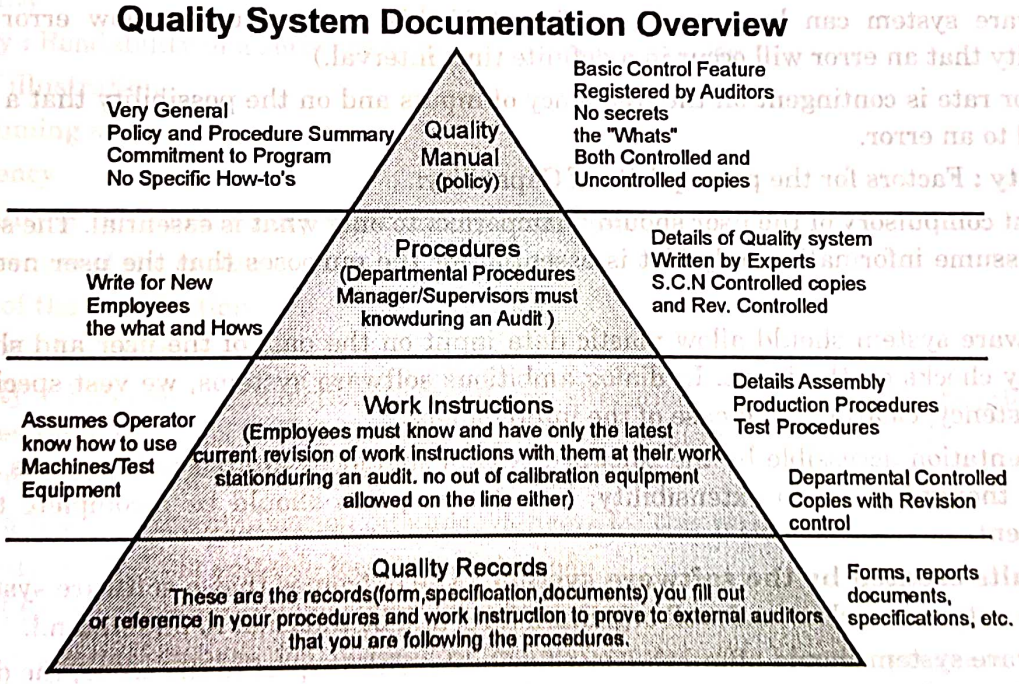
3. Define Your Quality Objectives

UQ. What are types of Requirements and Product? Also point out relationship between Quality and Productivity ? **SPPU - Aug 18(In Sem)**

All Quality management systems must have objectives. Each employee must appreciate their influence on quality. Quality objectives are derivative of your quality policy. It is measurable and set up throughout the organization. The objective may be in the form of critical success factors. This helps an organization in emphasizing the journey towards accomplishing its mission. These performance-based measures deliver a gauge to determine compliance with its objectives.

Some Critical Success Factors are:

1. Financial Performance
2. Product Quality
3. Process Improvement
4. Customer Satisfaction
5. Market Share
6. Employee Satisfaction



(1A17)**Fig. 1.17.2 : Quality System Documents (QSD) Overview Levels**

4. Develop Metrics to Track and Monitor CSF Data

- Once critical success factors are known, measurements and metrics keep track of advancement.
- This can be complete complete a data reporting procedure used to collect specific data. Share the processed information with leaders. A process goal is to improve customer approval index score.
- There requests to be a goal and a quantity to inaugurate realization of that goal.

1.17.2 Important Aspects of Quality Management

(1) Quality planning at organisation level
(2) Quality planning at project level
(3) Resource management
(4) Work Environment
(5) Customer Related Processes
(6) Quality management system document and data control
(7) Verification and validation
(8) Software project management
(9) Software configuration management
(10) Software metrics and measurement
(11) Software Quality Audits

Software Quality Attributes are : Correctness, Reliability, Adequacy, Learn facility, Robustness, Maintainability, Readability, Extensibility, Testability, Competence, and Portability.

1. **Correctness :** The perfection of a software system mentions to:
   - Promise of program code with conditions
   - Individuality of the definite request of the software system.
   - The **correctness of a program** develops especially critical when it is implanted in a composite software system.
2. **Reliability :** Reliability of a software system develops from
   (a) Correctness (b) Accessibility
   - The conduct over time for the contentment of a given requirement be contingent on the consistency of the software system.
   - **Reliability** of a software system is definite as the prospect that this system achieves a function (resolute by the provisions) for a definite number of input trials under detailed input situations in a definite time interval (presumptuous that hardware and input are allowed of errors).
   - A software system can be seen as consistent if this test produces a low error rate (i.e., the possibility that an error will occur in a definite time interval.)
   - The error rate is contingent on the frequency of inputs and on the possibility that a separate input will lead to an error.
3. **Acceptability :** Factors for the prerequisite of Capability:
   - The input compulsory of the user should be imperfect to only what is essential. The software system should assume information only if it is essential for the purposes that the user needs to transmit out.
   - The software system should allow plastic data input on the part of the user and should carry out credibility checks on the input. In dialog ambitious software systems, we vest specific standing in the consistency, clearness and ease of the interchanges.
   - The presentation accessible by the software system should be altered to the needs of the operator with the thought given to extensibility; i.e., the purposes should be incomplete to these in the requirement.
   - **The results created by the software system :** The outcomes that a software system distributes should be output in a clear and well organized form and be informal to understand.
   - The software system should afford the user flexibility with respect to the scope, the degree of detail, and the form of presentation of the results. Error messages must be provided in a form that is comprehensible for the user.
4. **Learn ability :** Learn ability of a software system depends on:
   - The design of user interfaces.
   - The clarity and the simplicity of the user instructions (tutorial or user manual).

- The user interface should present information as close to reality as possible and permit efficient utilization of the software's failures.
- The user manual should be structured clearly and simply and be free of all dead weight.
- It should clarify to the user anything the software system should do, in what way the separate functions are motivated, what relations exist between functions, and which exclusions strength arise and how they can be modified.
- In adding, the user guide should serve as a orientation that maintenances the user in quickly and securely definition the correct responses to queries.

5. **Robustness :** Robustness decreases the impact of working mistakes, specious input data, and hardware errors.
   - A software system is robust if the significances of an error in its process, in the input, or in the hardware, in relative to a given application, is contrariwise comparative to the possibility of the amount of this error in the given request.
   - Frequent errors (e.g. erroneous commands, typing errors) must be touched with particular care.

   Less recurrent errors (e.g. power letdown) can be touched more laxly, but still must not lead to permanent consequences.
6. **Maintainability :** Maintainability = suitability for correcting (localization and correction of errors) and for modification and extension of functionality.

   The maintainability of a software system be contingent on its:
   - Readability
   - Extensibility
   - Testability
7. **Readability :** Readability of a software system be contingent on its:
   - Form of illustration
   - Programming style
   - Consistency
   - Readability of the application programming languages
   - Structureness of the system
   - Quality of the certification
   - Tools available for inspection
8. **Extensibility :** Extensibility sanctions compulsory alterations at the suitable locations to be made without undesirable side effects. Extensibility of a software system depends on its :
   - Structureness (modularity) of the software system
   - Opportunities that the application language delivers for this resolution
   - Readability (to find the suitable location) of the code
   - Availability of comprehensible program documentation
9. **Testability** : appropriateness for permitting the programmer to survey program implementation (runtime performance under given conditions) and for correcting.
10. **The testability** of a software system be contingent on its Modularity Structureness Modular; well-structured databases prove more appropriate for systematic, stepwise difficult than monumental, structured programs.
11. **Testing tools** and the opportunity of expressing constancy situations (declarations) in the source code decrease the testing effort and afford important fundamentals for the general, organized testing of all system mechanisms.

12 **Efficiency :** ability of a software system to accomplish its determination with the best possible operation of all essential properties (time, storage, transmission channels, and peripherals).

13. **Portability :** the easiness with which a software system can be altered to run on computers other than the one for which it was intended.

14. The movability of a software system be contingent on:

- Degree of hardware independence
- Implementation language
- Magnitude of manipulation of dedicated system functions
- Hardware possessions
- **Structureness :** System needy elements are composed in easily substitutable program mechanisms.
- A software system can be said to be transferable if the effort required for porting it shows meaningfully less than the effort essential for a new implementation.

...Chapter Ends

❑❑❑

UNIT II

CHAPTER 2

Test Planning and Quality Management

University Prescribed Syllabus for the Academic Year 2022-2023

Test Planning - Artifacts, Strategy, Test Organization – Test Manager & Tester Role, Test plan purpose & contents, Test Strategy and Approach, Test cases & Test Data, Test Entry-Exit criteria, Test Execution Schedule, Use case Testing, Scenario Testing, Test Monitoring & Control- Test Metrics – Test Case Productivity, Test case Coverage, Defect Acceptance & Rejection, Test Efficiency, Efforts and Schedule Variance, Test Efforts biasing Factors, Test Report & configuration Management, Quality Assurance Process, Documentation Risk & Issues. Software Quality, Quality Management Importance, Quality Best practices.

2.1 TEST PLANNING

GQ. What is a Test Plan ?

- A Test Plan refers to a detailed document that catalogues the test strategy, objectives, schedule, estimations, deadlines, and the resources required for completing that particular project. Think of it as a blueprint for running the tests needed to ensure the software is working properly – controlled by test managers.
- A well-crafted test plan is a dynamic document that changes according to progressions in the project and stays current at all times. It is the point of reference, based on which testing activities are executed and coordinated among a QA team.
- The test plan is also shared with Business Analysts, Project Managers, Dev teams, and anyone else associated with the project, This mainly offers transparency into QA activities so that all stakeholders know how the software will be tested.

GQ. Why are Test Plans important ?

- They help individuals outside the QA teams (developers, business managers, customer-facing teams) understand exactly how the website or app will be tested.
- They offer a clear guide for QA engineers to conduct their testing activities.
- They detail aspects such as test scope, test estimation, strategy, and so on. Collating all this information into a single document makes it easier to review by management personnel or to re-use for other projects.

☞ Components of a Test Plan

- **Scope :** Details the objectives of the particular project. Also, it details user scenarios to be used in tests. If necessary, the scope can specify what scenarios or issues the project will not cover.
- **Schedule :** Details start dates and deadlines for testers to deliver results.
- **Resource Allocation :** Details which tester will work on which test.
- **Environment :** Details the nature, configuration, and availability of the test environment.
- **Tools :** Details what tools are to be used for testing, bug reporting, and other relevant activities.
- **Defect Management :** Details how bugs will be reported, to whom and what each bug report needs to be accompanied by. For example, should bugs be reported with screenshots, text logs, or videos of their occurrence in the code ?
- **Risk Management :** Details what risks may occur during software testing, and what risks the software itself may suffer if released without sufficient testing.
- **Exit Parameters :** Details when testing activities must stop. This part describes the results that are expected from the QA operations, giving testers a benchmark to compare actual results to.

GQ. How to create a Test Plan ? Explain the steps?

Creating a Test Plan involves the following steps :

1. Product Analysis
2. Designing Test Strategy
3. Defining Objectives
4. Establish Test Criteria
5. Planning Resource Allocation
6. Planning Setup of Test Environment
7. Determine test schedule and estimation
8. Establish Test Deliverables

1. Product Analysis

- Start with learning more about the product being tested, the client, and the end-users of similar products. Ideally, this phase should focus on answering the following questions :
  - Who will use the product ?
  - What is the main purpose of this product ?
  - How does the product work ?
  - What are the software and hardware specifications ?
- In this stage, do the following :
  - Interview clients, designers, and developers
  - Review product and project documentation
  - Perform a product walkthrough

2. Designing Test Strategy

- The Test Strategy document is developed by the test manager and defines the following :
  - Project objectives and how to achieve them.
  - The amount of effort and cost required for testing.
- More specifically, the document must detail out :
  - **Scope of Testing :** Contains the software components (hardware, software, middleware) to be tested and also those that will not be tested.
  - **Type of Testing :** Describes the types of tests to be used in the project. This is necessary since each test identifies specific types of bugs.
  - **Risks and Issues :** Describes all possible risks that may occur during testing – tight deadlines, insufficient management, inadequate or erroneous budget estimate – as well as the effect of these risks on the product or business.
  - **Test Logistics :** Mentions the names of testers (or their skills) as well as the tests to be run by them. This section also includes the tools and the schedule laid out for testing.

3. Defining Objectives

This phase defines the goals and expected results of test execution. Since all testing intends to identify as many defects as possible, the objects must include:

- A list of all software features – functionality, GUI, performance standards- that must be tested.
- The ideal result or benchmark for every aspect of the software that needs testing. This is the benchmark to which all actual results will be compared.

4. Establish Test Criteria

Test Criteria refers to standards or rules governing all activities in a testing project. The two main test criteria are :

1. **Suspension Criteria :** Defines the benchmarks for suspending all tests. For example, if QA team members find that 50% of all test cases have failed, then all testing is suspended until the developers resolve all of the bugs that have been identified so far.
2. **Exit Criteria :** Defines the benchmarks that signify the successful completion of a test phase or project. The exit criteria are the expected results of tests and must be met before moving on to the next stage of development. For example, 80% of all test cases must be marked successful before a particular feature or portion of the software can be considered suitable for public use.

5. Planning Resource Allocation

- This phase creates a detailed breakdown of all resources required for project completion. Resources include human effort, equipment, and all infrastructures required for accurate and comprehensive testing.
- This part of the test plan decides the measure of resources (number of testers and equipment) the project requires. This also helps test managers formulate a correctly calculated schedule and estimation for the project.

6. Planning Setup of Test Environment

- The test environment refers to software and hardware setup on which QAs run their tests. Ideally, test environments should be real devices so that testers can monitor software behavior in real user conditions.
- Whether it is manual testing or automation testing, nothing beats real devices, installed with real browsers and operating systems are non-negotiable as test environments. Do not compromise your test results with emulators or simulators.

7. Determining Test Schedule and Estimation

- For test estimation, break down the project into smaller tasks and allocate time and effort required for each.
- Then, create a schedule to complete these tasks in the designated time with the specific amount of effort.
- Creating the schedule, however, does require input from multiple perspectives :
  - Employee availability, number of working days, project deadlines, daily resource availability.
  - Risks associated with the project which has been evaluated in an earlier stage.

8. Establish Test Deliverables

- Test Deliverables refer to a list of documents, tools, and other equipment that must be created, provided, and maintained to support testing activities in a project.
- A different set of deliverables is required before, during, and after testing.

Deliverables required before testing

Documentation on

- Test Plan
- Test Design

Deliverables required during testing

Documentation on

- Test Scripts
- Test Data
- Simulators or Emulators (in early stages)
- Error and execution logs

Deliverables required after testing

- Documentation on
  - Test Results
  - Defect Reports
  - Release Notes
- A test plan in software testing is the backbone on which the entire project is built. Without a sufficiently extensive and well-crafted plan, QAs are bound to get confused with vague, undefined goals and deadlines. This hinders fast and accurate testing unnecessarily, slowing down results, and delaying release cycles.

- The guidelines in this article are meant to help test managers and senior QA professionals with constructing a test plan that helps with executing cleaner, faster, and more result-oriented tests.

2.1.1 Artifacts, Strategy, and Test Organization

GQ. Explain Artifacts, Strategy, and Test Organization.

- **Test Artifacts** are simply integral part of software testing. These are generally set of documents which software project tester gets during STLC (Software Testing Life Cycle).
- Test artifacts are by-products that are generated or created while performing software testing. These generated test artifacts are then shared with clients and testing team or team managers, team leaders, stakeholders associated with project, and also with members of other team.
- A sign-off is taken from client simply so that there is no communication gap in what is required. These artifacts are communicated, engineered, and constructed or developed within same artifact sets as formed product. These artifacts are also implemented in programmable and repeatable formats such as software programs.
- Establishing transparency between members of team is main goal and purpose of test artifact. So, they are recorded in proper manner with exact and accurate data or information and details.
- Due to this, it is very easy and simple to identify and track changes and also be aware of recent progress of activities of testing from requirement, as everything is recorded properly.
- The developers of these artifacts use tools, techniques, and training same as software engineers use while developing or creating end product.

☞ **Types of Test Artifacts** (Refer Fig. 2.1.1)

GQ. Explain types of Test Artiacts.

1. Test Strategy

- Test strategy is generally prepared by Test or Project Manager at management level. It is outline of document that describes testing approach of development cycle of software which enlists how to achieve expected result using resources that are available.
- It simply provides easy understanding of targets, tools, techniques, infrastructure, and timing of test activities that are to be performed. It is also used to identify all risk factors that can arise during testing and appropriate solution to reduce or mitigate risk. It also clarifies major important challenges and approach to complete all testing process of project.
- Test strategy is usually derived from Business Requirement Specification Format. To develop this strategy, there are several points that are needed to be kept in mind. Some of them are given below :
  - What is main objective of testing that why you want to perform this testing ?
  - What are Guidelines that are needed to be followed for performing testing ?
  - What all are requirements that are needed for testing such as functional requirements, test scenarios, resources, etc. ?

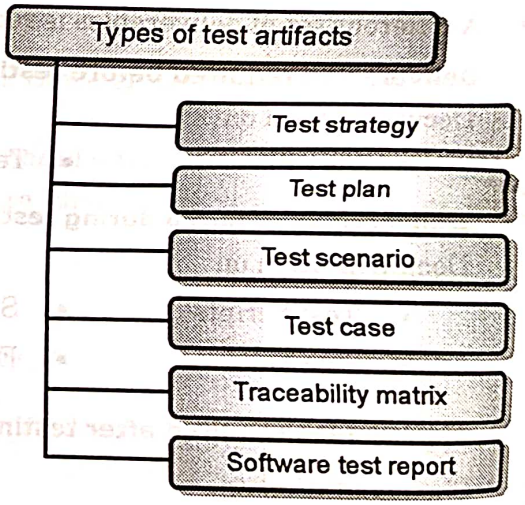


Fig. 2.1.1 : Types of Test Artifact

- What are roles and responsibilities of each and every functions and project manager to complete test ?
- What are different levels of testing ?
- What will be main deliverable of this testing ?
- What risks are there regarding testing along with project risks ?
- Is there any methods to resolve issues that might arise ?

2. Test Plan

- The test plan is detailed document that describes software testing scope, test strategy, test deliverables, risk, objectives, and activities. It is systemic approach generally used for software application testing. It is the most important and essential activity to simply ensure that there is initially list of tasks and milestones in baseline plan to track or identify project progress.
- It is dynamic document that generally acts as point of reference and only based on that testing which is carried out within QA (Quality Assurance) team. It is simply blueprint that explains how testing activity is going to take place in project. There are several points that are needed to be kept in mind to develop test strategy. Some of them are given below :
  - What is main purpose of testing activities ?
  - What is future scope of testing i.e., exact path that is needed to be followed or covered while performing testing ?
  - What is testing approach i.e., how testing would be carried out ?
  - What all are resources that are required for testing ?
  - What are exit criteria after completion of testing i.e., set of conditions and activities that are needed to be fulfilled in order to conclude testing ?
  - How will you manage risks that can arise ?

3. Test Scenario

- A test scenario is statement that is used to describe functionality of application that can be tested. It is simply used to make sure that end to end testing of feature or software is working well or not. It is derived from the use cases.
- It contains situation or condition in application form which many test cases can be developed.
- Test Scenario is also called as Test Condition or Test Possibility. One or more test cases can be accommodated in single test scenario.
- Due to this, test scenario has one-to-many relationship with test cases. It means talking and thinking about requirements in detailed manner.

4. Test Case

- The test case is detailed document that describes cases which will help execution while testing. It is document that consists of test case name, precondition, steps/input condition, and expected results.
- The development of test cases also helps in identifying or tracking problems or issues in requirement or design of software application.
- It is simply set of conditions or variables under which software tester will identify whether or not system under test satisfies requirements or work in proper and correct way.
- To write good test case, some of points that are needed to be included in test case :
  - Write Test case id i.e., unique identification number of test case.
  - Write Test case name i.e., strong title for test case.
  - Write full details and descriptions regarding test case.

- Write in steps to make it clear and concise i.e., simple.
- Write expected or actual outcome or result of test.

5. Traceability Matrix

- Traceability matrix is matrix that contains tables that shows and explains many to many relationships among requirements and test cases. It is actually document that co-relates any of two-baseline documents that require many to many relationships to check, trace, and maps relationship. It generally helps to ensure transparency and completeness of products of software testing.
- It is the tracing of all requirements of clients with test cases and identifying defects. The traceability matrix is of two types : Forward traceability matrix and Backward traceability matrix. Some of parameters that are included in Traceability Matrix are given below :
  - Requirement ID.
  - Requirement type along with description.
  - Status of test design along with execution of test status.
  - System and unit test cases.

6. Software Test Report

- A software test report is document that describes all testing activities. It provides detailed information about status of test cases, test suites, or test scripts for given scope.
- A test report is very much needed to represent test results in formal way that gives opportunity to find out testing results quickly.
- The test reports can be of many types: Individual test report, team report.
- The test reports can be generated on daily basis or it can be generated after completion of testing or at end of testing cycle.

2.1.2 Test Manager &amp

GQ. Explain Test Manager with example.

Role and Responsibilities of Test Manager / Test Lead

- Software testing has gained importance over the years. It is more crucial for projects to under go rigorous testing by an independent testing team. Software is tested at every level of its development.
- The independent testing team is hired to test the software with a stringent plan, in an organized manner, following the standard techniques, using software testing tools. In this article we list the roles of test leader and test manager in software testing process of IT projects.
- The software testing team comprises of Test manager, Test leaders and Testers.
- The testers include entry-level software testers, senior testers, automation testers, performance testers, etc. There is a test manager who leads many testing groups, with each testing group led by a test leader. In some projects, if there are one or two testing groups, a test leader leads them and hiring a test manager is optional.
- The role of the test leader and test manager are similar. The testers and testing groups are hired based on testing work load in the team.
- The testing team works in collaboration with the other IT team members like business analysts, architects, developers and system administrators. Below is the organization structure for testing team in IT projects.

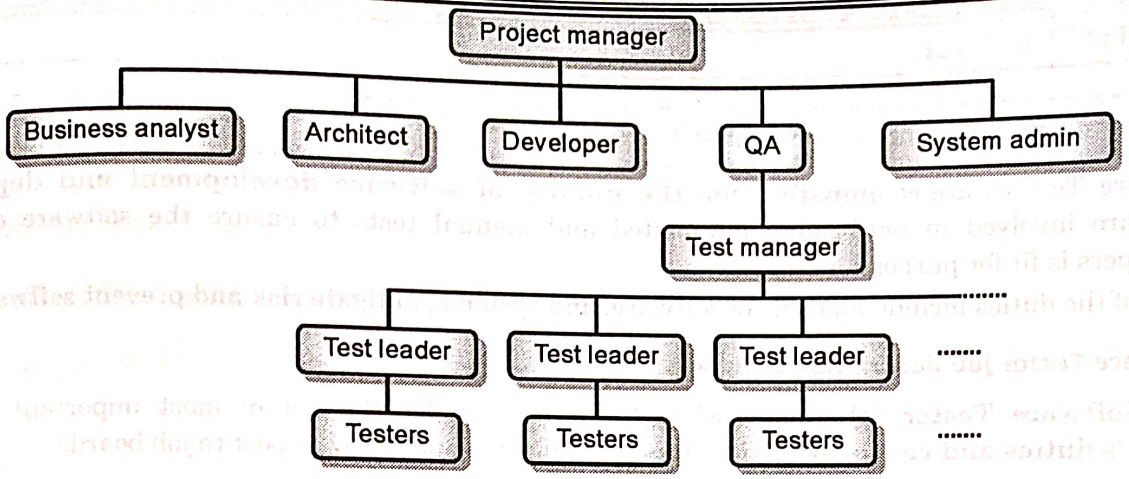


Fig. 2.1.2 : Project Test Manager task

- Before understanding the roles and responsibilities of a test lead or test manager, we will first know what is test management ? It is a important process to ensure software quality which involves the process of testing and validating the software.
- The test management practice includes organizing and controlling the testing process and also ensuring visibility, traceability and control of testing process which delivers a high quality software.
- The software development life cycle includes software testing as one of its phases, has advantages like, it improves quality, reliability and performance of the system and produces good quality product in the competitive market.
- To create an effective test process we need a good test manager. Test Manager or lead plays a central role in the team. Test manager or lead takes the full responsibility for the project's success.

The roles of Test leader and Test manager in software testing process of IT projects are listed below :

- Building and leading the testing team to the success of the project.
- Develop test strategy and test plans for projects
- Participate in developing and reviewing the test policies for organization.
- Defining the scope of testing within the context of every release and every software testing level or cycle.
- The use of resources in an effective way and managing the resources for software testing.
- Applying the appropriate test measurement and metrics for the software product and testing team.
- Identify and resolve the project risks in testing team like
  - No enough time to test
  - Not enough resources to test
  - The project budget is low
  - Testing teams are offshore
  - The requirements are too complex
- The test leader or test manager plays an important role at an outset of the project. They will in work collaboration with the stake holders, devise the test objectives, organizational test objectives, policies, test strategies and test plans. They decide when test automation is appropriate and they put effort and plan to select the tools and ensure training the testing team.
- During test implementation the test managers / test leaders make sure that the test environment is set up and verified before test execution and managed during the test execution.

2.2 TESTER ROLE

GQ. What is tester roles and responsibilities ?

- Software Testers are **responsible for the quality of software development and deployment.** They are involved in performing automated and manual tests to ensure the software created by developers is fit for purpose.
- Some of the duties include analysis of software, and systems, mitigate risk and prevent software issues.

☞ **Software Tester job description template**

- This **Software Tester job description template** includes the list of most important **Software Tester's duties and responsibilities.** It is customizable and ready to post to job boards.
- Use it to save time, attract qualified candidates and hire best employees.

☞ **Software Tester job profile**

- Software Testers are responsible for the quality of software development and deployment. They are involved in performing automated and manual tests to ensure the software created by developers is fit for purpose.
- Some of the duties include analysis of software, and systems, mitigate risk and prevent software issues.
- In order to attract a Software Tester that best matches your needs, it is very important to write a clear and precise Software Tester job description.
- Software Tester
- This position will offer the qualified candidate a great opportunity to work on new and exciting projects.
- Whether you have prior experience or not ? having the opportunity to join our team will be a great addition to your list of career accomplishments. If you meet the requirements of this position for a Software Tester and possess the ability to work in a team environment as well as independently please apply soon so that we can consider you for this great career opportunity.

☞ **Software Tester job description**

We are looking for an experienced and passionate Software Tester to join our team!. As a Software Tester at our company, you will have the opportunity to work on new and exciting projects and develop your career.

☞ **Software Tester duties and responsibilities**

- analyzing users stories and/use cases/requirements for validity and feasibility
- collaborate closely with other team members and departments
- execute all levels of testing (System, Integration, and Regression)
- Design and develop automation scripts when needed
- Detect and track software defects and inconsistencies
- Provide timely solutions
- Apply quality engineering principals throughout the Agile product lifecycle
- Provide support and documentation

☞ **Software Tester requirements and qualifications**

- X years of experience as a Software Tester or similar role
- Ability to handle multiple tasks simultaneously
- X years of experience in data analysis
- HHS EPLC experience preferred

- Ability to work in a fast-paced environment with minimal supervision
- Sense of ownership and pride in your performance and its impact on the company's success
- Critical thinker and problem-solving skills
- Team player
- Good time-management skills
- Great interpersonal and communication skills

2.2.1 Test Plan Purpose &amp

GQ. Explain Test Plan Purpose with example.

- A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan **helps us determine the effort needed to validate the quality of the application under test.**
- The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.
- As per ISTQB definition: "Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities."

Making Test Plan document has multiple benefits

GQ. What is the Importance of Test Plan ?

- Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
- Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
- Important aspects like test estimation, test scope, Test Strategy are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

GQ. How to write a Test Plan.

You already know that making a Test Plan is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

- Analyze the product
- Define Test Criteria
- Schedule and Estimation
- Design the Test Strategy
- Resource Planning
- Determine Test Deliverables
- Define the Test Objectives
- Plan Test Environment

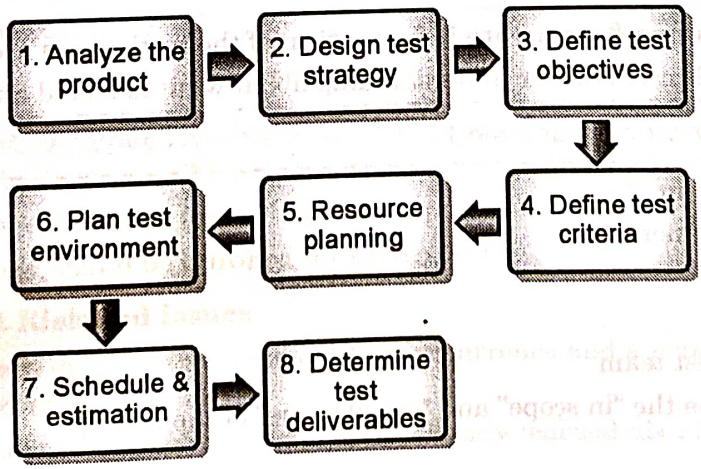


Fig. 2.2.1 : Test Plan Process

Step 1 : Analyze the product

- How can you test a product **without** any information about it ? The answer is **Impossible.** You must learn a product **thoroughly** before testing it.
- The product under test is Guru99 banking website. You should research clients and the end users to know their needs and expectations from the application
  - Who will use the website ?
  - What is it used for ?
  - How will it work ?
  - What are software/ hardware the product uses ?
- Use the following approach to analyze the site

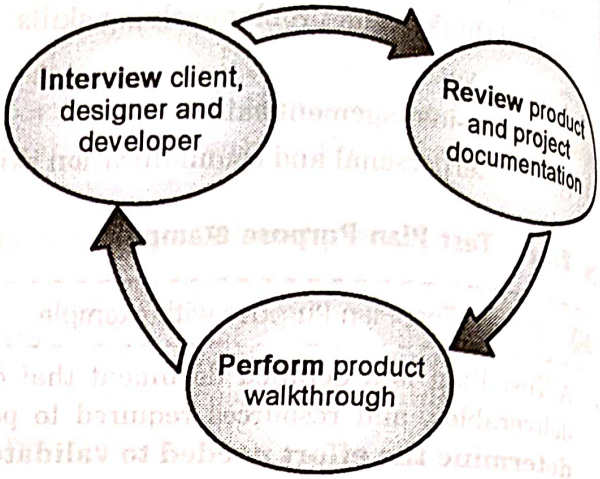


Fig. 2.2.2 : Product Analysis

Step 2 : Develop Test Strategy

- Test Strategy is a **critical step** in making a Test Plan in Software Testing.
- A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines :
  - The project's testing objectives and the means to achieve them
  - Determines testing **effort** and **costs**
- Back to your project, you need to develop Test Strategy for testing that banking website. You should follow steps below.

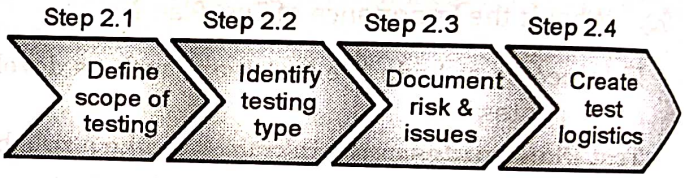


Fig. 2.2.3

Step 2 : 1) Define Scope of Testing

Before the start of any test activity, scope of the testing should be known. You must think hard about it.

- The components of the system to be tested (hardware, software, middleware, etc.) are defined as "in **scope**"
- The components of the system that will not be tested also need to be clearly defined as being "**out of scope.**"

Defining the scope of your testing project is very important for all stakeholders. A precise scope helps you

- Give everyone a **confidence & accurate information** of the testing you are doing
- All project members will have a **clear** understanding about what is tested and what is not

GQ. How do you determine scope your project ?

To determine scope, you must –

- Precise customer requirement
- Project Budget
- Product Specification
- Skills & talent of your test team

Now should clearly define the "in scope" and "out of scope" of the testing.

Problem Scenario

- The customer wants you to test his API. But the project budget does not permit to do so. In such a case what will you do ?
- Well, in such case you need to convince the customer that Api Testing is extra work and will consume significant resources. Give him data supporting your facts. Tell him if Api Testing is included in-scope the budget will increase by XYZ amount.
- The customer agrees and accordingly the new scopes, out of scope items are
  - In-scope items : Functional Testing, Api Testing
  - Out of scope items : Database Testing, hardware & any other external interfaces

Step 2 : 2) Identify Testing Type

- A Testing Type is a standard test procedure that gives an expected test outcome.
- Each testing type is formulated to identify a specific type of product bugs. But, all Testing Types are aimed at achieving one common goal "Early detection of all the defects before releasing the product to the customer"
- The commonly used testing types are described as following Fig. 2.2.4

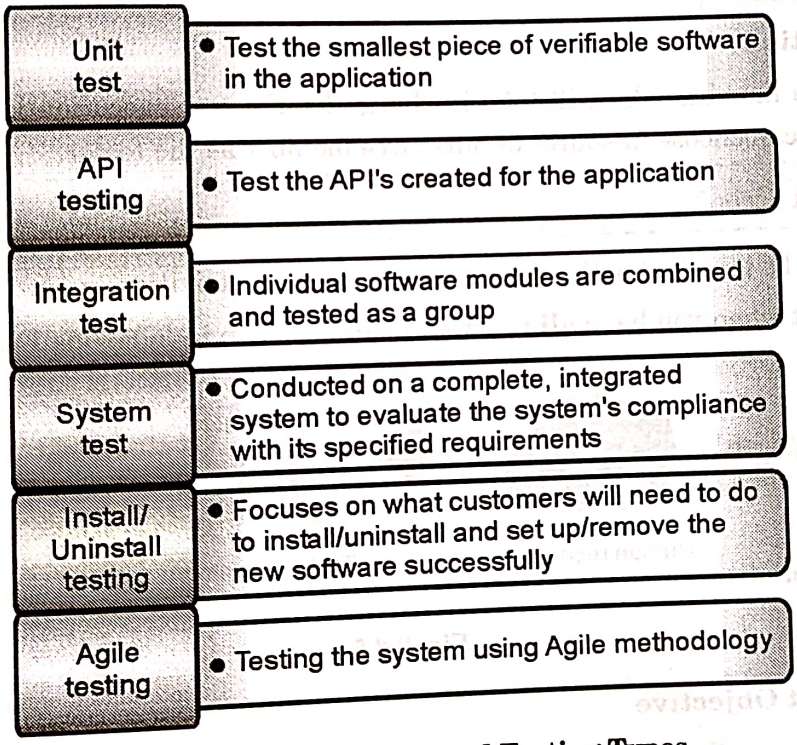


Fig. 2.2.4 : Commonly Used Testing Types

- There are **tons of Testing Types** for testing software product. Your team **cannot have** enough efforts to handle all kind of testing. As Test Manager, you must set **priority** of the Testing Types
  - Which Testing Types should be focused for web application testing ?
  - Which Testing Types should be **ignored** for saving cost ?

Step 2 : 3) Document Risk and Issues

- Risk is future's uncertain event with a probability of occurrence and a potential for loss. When the risk actually happens, it becomes the 'issue'.
- In the article Risk Analysis and Solution, you have already learned about the 'Risk' analysis in detail and identified potential risks in the project.

Step 2 : 4) Create Test Logistics

In Test Logistics, the Test Manager should answer the following questions :

- **Who** will test ?
- **When** will the test occur ?

GQ. Who will test ?

- You may not know exact names of the tester who will test, but the **type of tester** can be defined.
- To select the right member for specified task, you have to consider if his skill is qualified for the task or not, also estimate the project budget. Selecting wrong member for the task may cause the project to fail or **delay**.
- Person having the following skills is most ideal for performing software testing :
  - Ability to **understand** customers point of view
  - Strong **desire** for quality
  - **Attention** to detail
  - Good **cooperation**
- In your project, the member who will take in charge for the test execution is the **tester.** Base on the project budget, you can choose in-source or outsource member as the tester.

GQ. When will the test occur ?

- Test activities must be matched with associated development activities.
- You will start to test when you have **all required items** shown in following Fig. 2.2.5

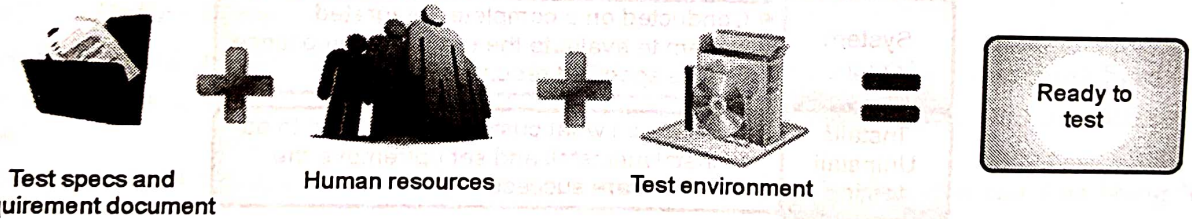


Fig. 2.2.5

Step 3 : Define Test Objective

- *Test Objective* is the overall goal and achievement of the test execution.
- The *objective of the testing* is finding as many software defects as possible; ensure that the software under test is **bug free** before release.
- To define the test objectives, you should do 2 following steps
  1. List all the software features (functionality, performance, GUI...) which may need to test.
  2. Define the **target** or the **goal** of the test based on above features
- You can choose the **'TOP-DOWN'** method to find the website's features which may need to test. In this method, you break down the application under test to **component** and **sub-component**.

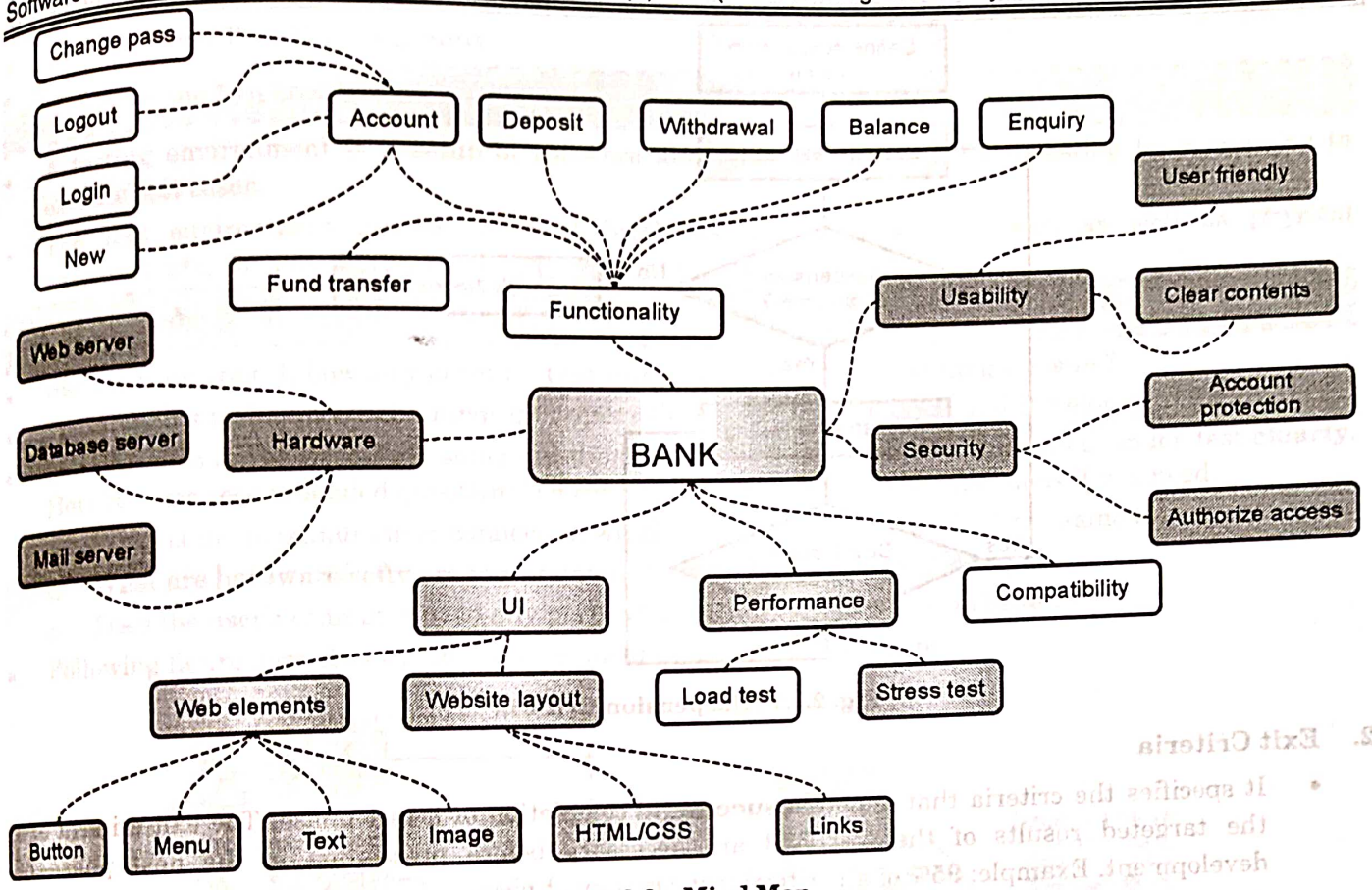


Fig. 2.2.6 : Mind Map

- This figure shows all the features which the website may have.
- Based on above features, you can define the Test Objective of the project as following
  - Check that whether website **functionality** (Account, Deposit...) is working as expected without any error or bugs in real business environment
  - Check that the external interface of the website such as **UI** is working as expected and & meet the customer need
  - Verify the **usability** of the website. Are those functionalities convenient for user or not ?

Step 4 : Define Test Criteria

Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There're 2 types of test criteria as following

1. **Suspension Criteria**
   - Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be suspended until the criteria are resolved.
   - Test Plan Example : If your team members report that there are 40% of test cases failed, you should suspend testing until the development team fixes all the failed cases.

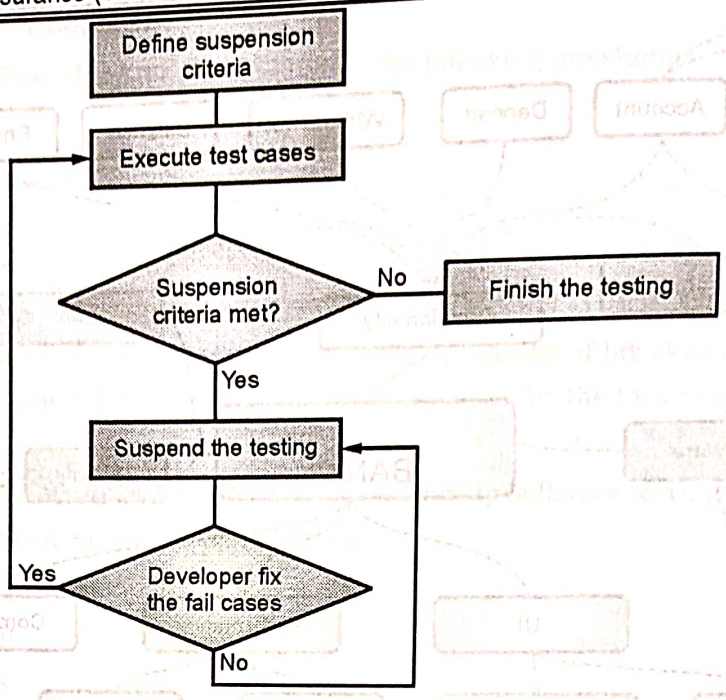


Fig. 2.2.7 : Suspension Criteria

2. Exit Criteria

- It specifies the criteria that denote a **successful** completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: **95%** of all critical test cases must pass.
- Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.
- Run rate is ratio between **number test cases executed/total test cases** of test specification. For example, the test specification has total 120 TCs, but the tester only executed 100 TCs, So the run rate is 100/120 = 0.83 (83%)
- Pass rate is ratio between **numbers test cases passed / test cases executed**. For example, in above 100 TCs executed, there're 80 TCs that passed, so the pass rate is 80/100 = 0.8 (80%)
- This data can be retrieved in Test Metric documents.
- Run rate is mandatory to be 100 % unless a clear reason is given.
- Pass rate is dependent on project scope, but achieving high pass rate is a goal.
- Test Plan Example : Your Team has already done the test executions. They report the test result to you, and they want you to confirm the Exit Criteria.
- In above case, the Run rate is mandatory is **100%,** but the test teaming only completed 90% of test cases. It means the Run rate is not satisfied, so do NOT confirm the Exit Criteria .

Step 5 : Resource Planning

- Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project
- The resource planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment...) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.
- This section represents the recommended resources for your project.

Step 6 : Plan Test Environment

GQ. What is the Test Environment ?

- A testing environment is a setup of software and hardware on which the testing team is going to execute test cases.
- The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front end running environment.

GQ. How to setup the Test Environment ?

- Back to your project, how do you set up **test environment** for this banking website ?
- To finish this task, you need **a strong cooperation** between Test Team and Development Team
- You should ask the developer some questions to understand the web application under test **clearly**. Here're some recommended questions. Of course, you can ask the other questions if you need.
  - What is the maximum user connection which this website can handle at the same time ?
  - What are hardware/software requirements to install this website ?
  - Does the user's computer need any particular setting to browse the website ?
- Following figure describes the test environment of the banking website

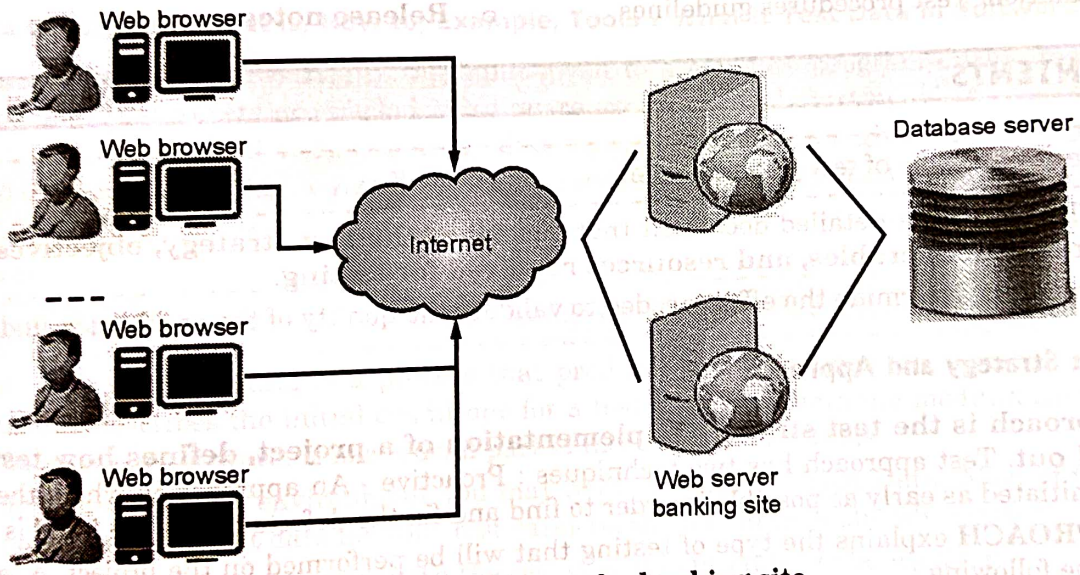


Fig. 2.2.8 : Web Server for banking site

Step 7 : Schedule and Estimation

- In the article Test estimation, you already used some techniques to estimate the effort to complete the project. Now you should include that estimation as well as the schedule to the Test Planning
- Making schedule is a common term in project management. By creating a solid schedule in the Test Planning, the Test Manager can use it as tool for monitoring the project progress, control the cost overruns.
- To create the project schedule, the Test Manager needs several types of input as below :
  1. **Employee and project deadline:** The working days, the project deadline, resource availability are the factors which affected to the schedule
  2. **Project estimation:** Base on the estimation, the Test Manager knows how long it takes to complete the project. So he can make the appropriate project schedule

3. **Project Risk :** Understanding the risk helps Test Manager add enough extra time to the project schedule to deal with the risks

Step 8 : Test Deliverables

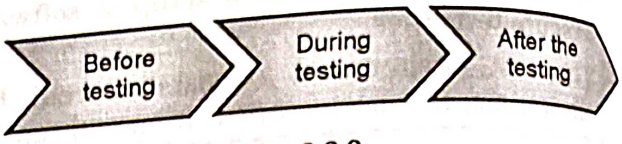


Fig. 2.2.9

- Test Deliverables is a list of all the documents, tools and other components that has to be developed and maintained in support of the testing effort.
- There are different test deliverables at every phase of the software development lifecycle.
- Test deliverables are provided **before** testing phase.
  - Test plans document.
  - Test cases documents
  - Test Design specifications.
- Test deliverables are provided **during** the testing
  - Test Scripts
  - Test Data
  - Error logs and execution logs.
  - Simulators.
  - Test Traceability Matrix
- Test deliverables are provided **after** the testing cycles is over.
  - **Test Results/reports**
  - Installation/ Test procedures guidelines
  - Defect Report
  - **Release notes**

2.3 CONTENTS

GQ. What are the contents of test plan template?

- Test Plan Template is a detailed document that **describes the test strategy, objectives, schedule, estimation and deliverables, and resources required for testing**.
- Test Plan helps us determine the effort needed to validate the quality of the application under test.

2.3.1 Test Strategy and Approach

- **A test approach is the test strategy implementation of a project, defines how testing would be carried out.** Test approach has two techniques : Proactive - An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- **TEST APPROACH** explains the type of testing that will be performed on the project. A test approach considers the following :
  1. Criteria for approach selection,
  2. Approach for organizing tests, and
  3. Approach for executing tests.
- **TEST STRATEGY** clarifies the major tasks and challenges of a test project. It is created to inform project managers, testers, and developers about key issues of the testing process. This includes the testing objective, methods of testing new functions, total time and resources required for the project, and the testing environment. The test approach is typically covered in the test strategy.

2.3.2 Test CASEs &amp

GQ. What is a Test Case ?

- A **Test Case** is a set of actions executed to verify a particular feature or functionality of your software application.

- A Test Case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement.
- The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

☞ Test Scenario Vs Test Case

- Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.
- For a Test Scenario : Check Login Functionality there many possible test cases are :
  - **Test Case 1 :** Check results on entering valid User Id & Password
  - **Test Case 2 :** Check results on entering Invalid User ID & Password
  - **Test Case 3 :** Check response when a User ID is Empty & Login Button is pressed, and many more

2.4 TEST DATA

GQ. Explain Test data with example.

☞ Test Data Generation: What is, How to, Example, Tools ? What is Test Data in Software Testing ?

- **Test Data in Software Testing** is the input given to a software program during test execution. It represents data that affects or affected by software execution while testing. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.
- Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

GQ. What is Test Data Generation? Why test data should be created before test execution ?

- Everybody knows that testing is a process that produces and consumes large amounts of data. Data used in testing describes the initial conditions for a test and represents the medium through which the tester influences the software. It is a crucial part of most Functional Tests.
- Depending on your testing environment you may need to CREATE Test Data (Most of the times) or at least identify a suitable test data for your test cases (is the test data is already created).
- Typically test data is created in-sync with the test case it is intended to be used for.

☞ Test Data can be Generated

- Manually
- Mass copy of data from production to testing environment
- Mass copy of test data from legacy client systems
- Automated Test Data Generation Tools

Typically sample data should be generated before you begin test execution because it is difficult to handle test data management otherwise. Since in **many testing environments creating test data takes multiple pre-steps or very time-consuming test environment configurations.** . Also if test data generation is done ***while*** you are in test execution phase you may exceed your testing deadline.

☞ Test Data for White Box Testing

In White Box Testing, test data Management is derived from direct examination of the code to be tested. Test data may be selected by taking into account the following things :

- It is desirable to cover as many branches as possible; testing data can be generated such that all branches in the program source code are tested at least once
- Path testing : all paths in the program source code are tested at least once – test data preparation can done to cover as many cases as possible
- Negative API Testing :
  - Testing data may contain invalid parameter types used to call different methods
  - Testing data may consist in invalid combinations of arguments which are used to call the program's methods

☞ Test Data for Performance Testing

- Performance Testing is the type of testing which is performed in order to determine how fast system responds under a particular workload.
- The goal of this type of testing is not to find bugs, but to eliminate bottlenecks. An important aspect of Performance Testing is that the set of sample data used must be very close to 'real' or 'live' data which is used on production.
- The following question arises: 'Ok, it's good to test with real data, but how do I obtain this data ?' The answer is pretty straightforward: from the people who know the best – the customers. They may be able to provide some data they already have or, if they don't have an existing set of data, they may help you by giving feedback regarding how the real-world data might look like.
- In case you are in a maintenance testing project you could copy data from the production environment into the testing bed. It is a good practice to anonymize (scramble) sensitive customer data like Social Security Number, Credit Card Numbers, Bank Details etc. while the copy is made.

☞ Test Data for Security Testing

- Security Testing is the process that determines if an information system protects data from malicious intent.
- The set of data that need to be designed in order to fully test a software security must cover the following topics :
  1. **Confidentiality:** All the information provided by clients is held in the strictest confidence and is not shared with any outside parties. As a short example, if an application uses SSL, you can design a set of test data which verifies that the encryption is done correctly.
  2. **Integrity :** Determine that the information provided by the system is correct. To design suitable test data you can start by taking an in-depth look at the design, code, databases and file structures.
  3. **Authentication :** Represents the process of establishing the identity of a user. Testing data can be designed as a different combination of usernames and passwords and its purpose is to check that only the authorized people are able to access the software system.
  4. **Authorization :** Tells what the rights of a specific user are. Testing data may contain a different combination of users, roles and ***operations*** in order to check only users with sufficient privileges are able to perform a particular operation.

☞ Test Data for Black Box Testing

In Black Box Testing the code is not visible to the tester. Your functional test cases can have test data meeting following criteria –

1. **No data :** Check system response when no data is submitted
2. **Valid data :** Check system response when Valid test data is submitted
3. **Invalid data :** Check system response when InValid test data is submitted
4. **Illegal data format :** Check system response when test data is in an invalid format

5. **Boundary Condition Dataset :** Test data meeting boundary value conditions
6. **Equivalence Partition Data Set :** Test data qualifying your equivalence partitions.
7. **Decision Table Data Set :** Test data qualifying your decision table testing strategy
8. **State Transition Test Data Set :** Test data meeting your state transition testing strategy
9. **Use Case Test Data :** Test Data in-sync with your use cases.

☞ **Automated Test Data Generation Tools**

- In order to generate various sets of data, you can use a gamut of automated test data generation tools. Below are some examples of such tools :
- DTM Test Data generator, is a fully customizable utility that generates data, tables (views, procedures etc) for database testing (performance testing, QA testing, load testing or usability testing) purposes.
- Datatect is a SQL data generator by Banner Software, generates a variety of realistic test data in ASCII flat files or directly generates test data for RDBMS including Oracle, Sybase, SQL Server, and Informix.

2.4.1 Test Entry Exit Criteria

GQ. Explain Test Entry Exit Criteria.

- Software testing, an essential part of software development life cycle, is quite a vast and complex process that requires ample time and efforts of testers to validate software product's quality and effectiveness. This process, though extensively helpful, often becomes tedious as it has to be executed a plethora of times across different platforms.
- Moreover, there are multifarious requirements that need to be considered and tested, which sometimes become a source of uncertainty for testers, mostly regarding where to commence & terminate testing.
- To avoid this confusion, specific conditions and requirements are established by the QA team, before the inception of testing that helps testers throughout the testing life cycle. These conditions are termed as **entry and exit criteria**, which play a crucial role in software testing life cycle.

☞ **What is An Entry Criteria in Software Testing ?**

- As the name specifies, entry criteria is a set of conditions or requirements, which are required to be fulfilled or achieved to create a suitable and favourable condition for testing.
- Finalized and decided upon after a thorough analysis of software and business requirements, entry criteria ensures the accuracy of the testing process and neglecting it can impact its quality.
- Some of the entry criteria, which are generally used to mark the beginning of the testing, are :
  - Complete or partially testable code is available.
  - Requirements are defined and approved.
  - Availability of sufficient and desired test data.
  - Test cases are developed and ready.
  - Test environment has been set-up and all other necessary resources such as tools and devices are available.

Both, development and testing phases are used as a source to define the entry criteria for software testing process, like :

- Development phase/process provides useful information pertaining to software, its design, functionalities, structure, and other relevant features, which offer assistance in deciding the accurate entry criteria like functional and technical requirement, system design, etc.
- From testing phase, following inputs are considered :
  - Test Plan.
  - Test Strategy.
  - Test data and testing tools.
  - Test Environment.

- The entry criteria is mainly determined for four specific test levels i.e., unit testing, integration testing, system testing and acceptance testing.
- Each of these test levels requires distinct entry criteria to validate the objective of test strategy and to ensure fulfillment of product requirements.

☞ **Unit Testing**

- Planning phase has been completed.
- System design, technical design and other relevant documents are properly reviewed, analyzed and approved.
- Business and functional requirements are defined and approved.
- Testable codes or units are available.
- Availability of test environment.

☞ **Integration Testing**

- Completion of unit testing phase.
- Priority bugs found during unit testing has been fixed and closed.
- Integration plan and test environment to carry out integration testing is ready.
- Each module has gone through unit testing before the integration process.

☞ **System Testing**

- Successful completion of integration testing process.
- Priority bugs found during previous testing activities has been fixed and closed.
- System testing environment is available.
- Test cases are available to execute.

☞ **Acceptance Testing**

- Successful completion of system testing phase.
- Priority bugs found during previous testing activities has been fixed and closed.
- Functional and Business requirement has been met.
- Acceptance testing environment is ready.
- Test cases are available.

GQ. What is an Exit Criteria in Software Testing ?

- Exit a criterion is an important document prepared by the QA team to adhere to the imposed deadlines and allocated budget. This document specifies the conditions and requirements that are required to be achieved or fulfilled before the end of software testing process. With the assistance of exit criteria, the team of testers is able to conclude the testing without compromising the quality and effectiveness of the software.
- Exit criteria highly depends on the by-product of the software testing phase i.e. test plan, test strategy, test cases, test logs, etc. and can be defined for each test level, right from test planning, specification, and till execution.
- The commonly considered exit criteria for terminating or concluding the process of testing are :
  - Deadlines meet or budget depleted.
  - Execution of all test cases.
  - Desired and sufficient coverage of the requirements and functionalities under the test.
  - All the identified defects are corrected and closed.
  - No high priority or severity or critical bug has been left out.

- Similar to entry criteria, exit criteria is also defined for all different levels of testing. Few of them are :

☞ **Unit Testing**

- Successful execution of the unit tests.
- Project code is complete.
- All the identified bugs have been fixed and closed.

☞ **Integration Testing**

- Successful execution of the integration tests.
- Satisfactory execution of stress, performance and load tests.
- Priority bugs have been fixed and closed.

☞ **System testing**

- Successful execution of the system tests.
- Priority bugs have been fixed and closed.
- All specified business and functional requirements has been met.
- System's compatibility with supported hardware and software.

☞ **Acceptance testing**

- Successful execution of the user acceptance tests.
- Business requirements got fulfilled.
- Signing off acceptance testing.
- Approval from management to stop UAT.
- No critical defects have been left out.

2.4.2 Test Execution Schedule

GQ. Write a note on Test Execution Schedule.

- An execution schedule contains steps that are run sequentially at a scheduled time or when they are triggered by a build completion.
- Execution schedule steps can be any of the following types :
  1. **Test suite execution record :** Records that specify the execution environments for each test suite and track the status of each test suite that is run
  2. **Test case execution record :** Records that specify the execution environments for each test case and track the status of each test case that is run
  3. **Automation :** Scripts that automatically run tasks such as installing or uninstalling a product build
- Execution schedules are reusable. You can schedule them to run repeatedly at specified times (daily or weekly, for example) or have they triggered by an event. You can also select the machines or test cells on which to run execution schedules.
- A test cell is a collection of lab resources that meet the requirements for a specific test environment.
- If an execution schedule is triggered by a build, all test case results and test suite results are associated with the corresponding build record.

2.4.3 Use Case Testing

GQ. Explain Use Case Testing with example.

- Use case testing is a technique that helps to identify test cases that cover the entire system, on a transaction by transaction basis, from start to finish. It is a description of a particular use of the system by a user. It is used widely in developing tests or systems for acceptable levels.
- In a use case, there will be a set of actions for the user to complete. The actions can be :
  - Withdraw funds,
  - Balance transfer, and/or
  - Balance enquiry,
  - Other actions relating to the software that is being developed.

Use Case Example

- When developing use cases, a test case table is usually developed. There will be a success scenario as well as the steps that the user should complete.
- Examples of the steps can be :
  - Insert card,
  - Enters a PIN,
  - Allows access to the account.
  - Validates card and asks for a PIN,
  - Validates a Pin, and
- Following this, there will also be a list of extensions within the table. It could happen, for example, that upon validating the card, the system determines that something is incorrect.
- The extensions can be listed as follows :

  2a) Card not valid (Display message and reject card),

  3a) Pin not valid (Display message and ask for re-try – twice), and

  4a) Pin invalid 3 times (eat card and exit).
- Many times, software testers and developers refer to users as 'actors'. Use cases are associated with the following :
  - Actors (human users, external hardware or other components or systems), and
  - Subjects (the component or system to which the use case is applied).
- Each use case specifies some behavior that a subject can perform in collaboration with one or more actors.
- A use case specifies a type of behavior that a subject can perform in collaboration with one or more actors, and it can be described by interactions and activities as well as preconditions, post conditions and natural language where appropriate.
- Need software testers who can do a comprehensive and thorough test on all your applications and software ? Then contact Inspired Testing today.

2.4.4 Scenario Testing

GQ. Explain Scenario Testing with example.

- **Scenario Testing** is a Software Testing Technique that uses scenarios i.e. speculative stories to help the tester work through a complicated problem or test system.
- The ideal scenario test is a reliable, complicated, convincing or motivating story the outcome of which is easy to assess.

- Usually these tests are different from test cases as the test cases are single steps whereas scenarios cover a number of steps.
- Scenario testing is performed to ensure that the end to end functioning of software and all the process flow of the software are working properly.
- In scenario testing, the testers assume themselves to be the end users and find the real world scenarios or use cases which can be carried out on the software by the end user.
- In scenario testing, the testers take help from clients, stakeholders and developers to create test scenarios.
- Scenario testing helps testers to know how the software will exactly work when end user will use it. As the scenario testing tests the business process flow of the software so it helps in figure out a lot of defects which cannot be found with the help of other testing.
- Scenario testing is carried out by creating test scenarios which copy the end users usage. A test scenario is a story which describes the usage of the software by an end user.

☞ Characteristics of Scenario Testing

A scenario test has five key characteristics :

1. Story
2. Motivating
3. Credible
4. Complex
5. Easy to evaluate

☞ Scenario Testing Process

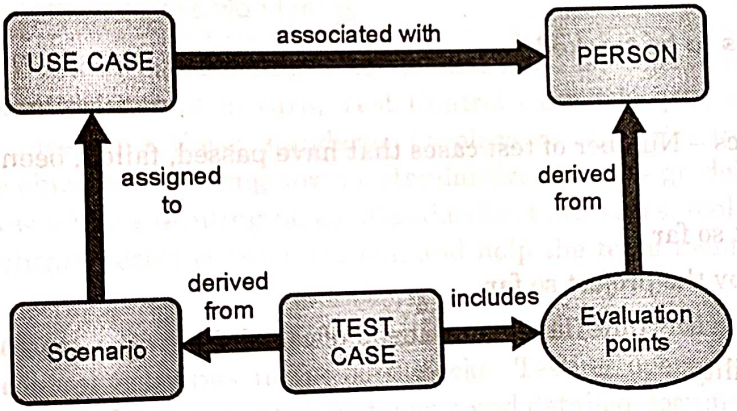


Fig. 2.4.1 : Scenario Testing

☞ Methods in Scenario Testing

There are two methods in scenario testing :

1. **System scenarios :** Scenario tests used in this method are only those sets of realistic, user activities that cover various components in the system.
2. **Use-case and role-based scenarios :** In use-case and role-based scenario method the focus is specifically on how the system is used by a user with different roles and environment.

☞ Risks of Scenario Testing

- Scenario testing is complex involving many features.
- Scenario testing is not designed for coverage of the program.
- Scenario testing is often heavily documented and used time and again.

2.4.5 Test Monitoring &amp

GQ. Write a short note on : Test Monitoring with example.

What is testing monitoring?

- Image result for Test Monitoring &amp
- Test Monitoring is the process of evaluating and providing feedback on the test proceedings that are currently in progress. It comprises techniques to ensure that specific targets are met at every stage of testing so that they meet predetermined benchmarks and objectives.
- Monitoring works by comparing the current status of testing-related tasks against a previously established plan and evaluating whether the process is yielding the expected results.
- Test Monitoring involves :
  - Offering feedback to the QA team and other stakeholders regarding the progress of sprint test cycles
  - Conveying test results achieved so far to all relevant parties
  - Identifying and tracking relevant test metrics
  - Planning and Estimation to determine the future course of action, based on the metrics being tracked

Relevant Test Metrics to be tracked

- Test Coverage
- Test Execution Metrics – Number of test cases that have passed, failed, been blocked, or are on hold
- Defect Metrics
- The cost of the project so far
- Resources consumed by the project so far
- Status of each task – on schedule, lagging or ahead of schedule
- Requirement Traceability

When to collect data for Test Monitoring?

- The frequency with which data should be collected for monitoring purposes depends on the nature of the project.
- For example, if the project is slated to be completed in one month, then it makes sense to collect monitoring data weekly. However, if the test suites are especially complicated and require close supervision, then data might have to be accrued twice a week or so.

GQ. How to evaluate progress through collected data ?

- It is best to establish the evaluation process in the monitoring plan laid out to teams at the beginning of the project so that they know exactly how their work will be judged.
- Two easy methods to define progress would be :
  1. Compare the progress in plan with the actual progress made by the team
  2. Look at the previously defined criteria to evaluate the project's progress. For example, if the effort to complete a task was 20% higher than it was meant to be (according to the plan), that is a marker of how the project is progressing.

GQ. What is Test Control.

- Test Control occurs based on the results of Test Monitoring. It refers to taking corrective action based on test monitoring reports to improve quality and efficiency.
- Some examples of test control activities would be :
  - Prioritize testing efforts in a different way
  - Reorganize test schedules and deadlines
  - Restructure the test environment
  - Reprioritize the test case and conditions
- Test Control is essentially modifying the testing process so that it becomes better suited for meeting the defined objectives. This may require adding extra resources, reducing the scope of release, or splitting the release into multiple releases, etc. Obviously, what specific test control activities will be implemented depends on a variety of factors – stakeholders' opinions, budget, project complexity, availability of testers, and the like.
- Test Control goes hand-in-hand with Test Monitoring. Obviously, once Monitoring identifies any bottlenecks that may prevent a test cycle from meeting its goals, Control activities will have to come into play to ensure otherwise.

☞ Best Practices in Test Monitoring and Test Control

GQ. Write a note on best practices in Test Monitoring.

1. **Establish standards:** Without establishing a set of benchmarks, objectives, and standards in the planning phase, Test Monitoring and in turn, Test Control cannot be performed. QA managers and other senior personnel must put these standards in place so that the testing team knows their individual and collective objectives. Setting down a standardized process guideline also gives the team a unified structure to follow while executing tasks. Standardized processes, tools, templates, and metrics make analysis easy, facilitate better communication, and help the team members navigate the project more efficiently.
2. **Prioritize Documentation :** Document everything – test results, measurement values, minutes of stakeholder meetings, control activities to fix bottlenecks. Testing operations, especially when they involve automation testing can be complicated. Extensive and detailed documentation is the best way to avoid miscommunications and keep all stakeholders on the same page.
3. **Be Proactive and Prepare for Change :** Every testing project carries with it a high possibility of disruptions showing up in the midst of it. QA managers and testers have to go into test suites with a proactive mindset that is ready to adapt and resolve issues detected during execution. These issues can vary from the budget, time, quality, infrastructure, scope, and availability of human resources. Whatever the problem, it is best that testers expect them to occur, and be prepared to deal with them.

☞ The Role of Real Devices in Test Monitoring and Test Control

- No matter the activity, all testing operations must be executed on real devices. Test Monitoring and Test Control can only be effective when tests are being run in real user conditions.
- Running them on emulators or simulators cannot provide 100% accurate results, and therefore QA managers won't be able to evaluate the testing process with precision. Any results yielding from Test Monitoring would be only partially correct, and therefore Test Control activities based on these results will not modify the test cycle for maximum productivity.

2.5 CONTROL- TEST METRICS –TEST CASE PRODUCTIVITY

GQ. Explain Software Testing Metrics.

☞ **Software Testing Metrics**

- **Software Testing Metrics** are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process.
- The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.
- A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute.
- The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.

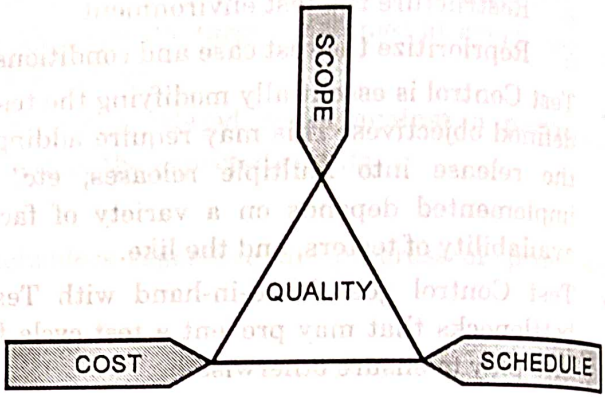


Fig. 2.5.1 : Test Matrices

- Software testing metrics – Improves the efficiency and effectiveness of a software testing process.
- Software testing metrics or software test measurement is the quantitative indication of extent, capacity, dimension, amount or size of some attribute of a process or product.
- **Example for software test measurement :** Total number of defects

GQ. Why Test Metrics are Important ?

"We cannot improve what we cannot measure" and Test Metrics helps us to do exactly the same.

- Take decision for next phase of activities
- Evidence of the claim or prediction
- Understand the type of improvement required
- Take decision or process or technology change

☞ **Types of Test Metrics** (Refer Fig. 2.5.2)

1. **Process Metrics :** It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle)
2. **Product Metrics :** It deals with the quality of the software product
3. **Project Metrics :** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

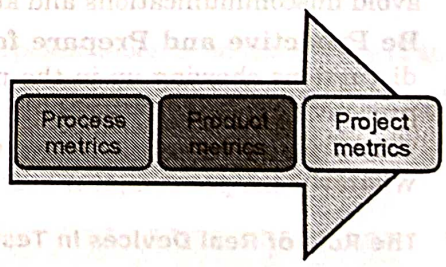


Fig. 2.5.2

☞ **How do you calculate test case productivity ?**

- Productivity (for test case preparation) = Actual number of test cases/ Actual effort expended in test case preparation.
- Productivity (for test case execution) = Actual number of test cases / actual effort expended in testing.

2.5.1 Test Case Coverage

GQ. What is Test Coverage?

- Test coverage is defined as a metric in Software Testing that measures the amount of testing performed by a set of test. It will include gathering information about which parts of a program are executed when running the test suite to determine which branches of conditional statements have been taken.
- In simple terms, it is a technique to ensure that your tests are testing your code or how much of your code you exercised by running the test.

GQ. What Test Coverage does ?

- Finding the area of a requirement not implemented by a set of test cases
- Helps to create additional test cases to increase coverage
- Identifying a quantitative measure of test coverage, which is an indirect method for quality check
- Identifying meaningless test cases that do not increase coverage

GQ. How Test Coverage can be accomplished?

- Test coverage can be done by exercising the static review techniques like peer reviews, inspections, and walkthrough
- By transforming the ad-hoc defects into executable test cases
- At code level or unit test level, test coverage can be achieved by availing the automated code coverage or unit test coverage tools
- Functional test coverage can be done with the help of proper test management tools.

Benefits of Test Coverage

- It can assure the quality of the test
- It can help identify what portions of the code were actually touched for the release or fix
- It can help to determine the paths in your application that were not tested
- Prevent Defect leakage
- Time, scope and cost can be kept under control
- Defect prevention at an early stage of the project lifecycle
- It can determine all the decision points and paths used in the application, which allows you to increase test coverage
- Gaps in requirements, test cases and defects at the unit level and code level can be found in an easy way

Formula to calculate Test Coverage

- To calculate test coverage, you need to follow the below-given steps :

Step 1 : The total lines of code in the piece of software quality you are testing

Step 2 : The number of lines of code all test cases currently execute

- Now, you need to find (X divided by Y) multiplied by 100. The result of this calculation is your test coverage %.
- For example:
- If the number of lines of code in a system component is 500 and the number of lines executed across all existing test cases is 50, then your test coverage is :

$$(50 / 500) * 100 = 10\ \%$$

☞ **Examples of Test Coverage**

Example 1 :

- For example, if "knife" is an Item that you want to test. Then you need to focus on checking if it cuts the vegetables or fruits accurately or not.
- However, there are other aspects to look for like the user should able to handle it comfortably.

Example 2 :

- For example, if you want to check the notepad application. Then checking it's essential features is a must thing.
- However, you need to cover other aspects as notepad application responds expectedly while using other applications, the user understands the use of the application, not crash when the user tries to do something unusual, etc.

☞ **Drawbacks**

- Most of the tasks in the test coverage manual as there are no tools to automate. Therefore, it takes lots of effort to analyze the requirements and create test cases.
- Test coverage allows you to count features and then measure against several tests. However, there is always space for judgment errors.

2.5.2 Defect Acceptance &amp

- Software Go-Live is always a big event for any software product. It is important to absolutely make sure that everything works and that we are releasing quality software to the users.
- A bad or premature or unstable or difficult to use product can cause a lot of losses financially and could also make the user lose trust in the brand itself.
- Often times, we hear that testing should be done until we meet the exit criteria. We also hear that defects have to be fixed to an acceptable level.

☞ **The following are important considerations**

- High severity and priority defects are usually the ones that would impact the day to day usage of the software. These types of defects are the ones that must be fixed before we go-live. No exceptions.
- Sometimes Functional defects are classified as Change Requests as they were not a part of the originally given requirements. Such CRs, that are a must for business to work after Go-live, are also a must to be implemented.
- Classification of defects and prioritization of functional defects are done by the UAT coordinators in collaboration with business users and Business analysts.
- Usually, the customer has an e**An example exit criteria for go-live :**
  - This is just an example. It can vary from project to project.
  - 100 % of Priority 1 defects are closed (Severity Critical and priority 1)
  - 90 % of priority 2 defects are closed (Severity high and priority 2) with a logical workaround being available for the rest of the 10% of the defects. And, a plan is available for closing the rest of the 10 % of the defects.
  - Production deployment and sanity checklist is ready.
  - Production support team has been formed and ready for closing tickets.
  - 70% of priority 3 defects are closed and a plan is in place for closing rest of the 30% of low defects.

A few points to note

- All severity and priority definitions are decided during the business meetings between the customer and vendor at the start of the program.
- After all the UAT defects are logged and all the other defects are being closed, UAT coordinators and Business sponsors meet to take stock of pending and open defects. If all defects that are required for Day-1 go-live are closed, the business sponsors see their readiness for go-live and get the software into production.

2.6 REJECTION

GQ. Why Developers Reject Defects ?

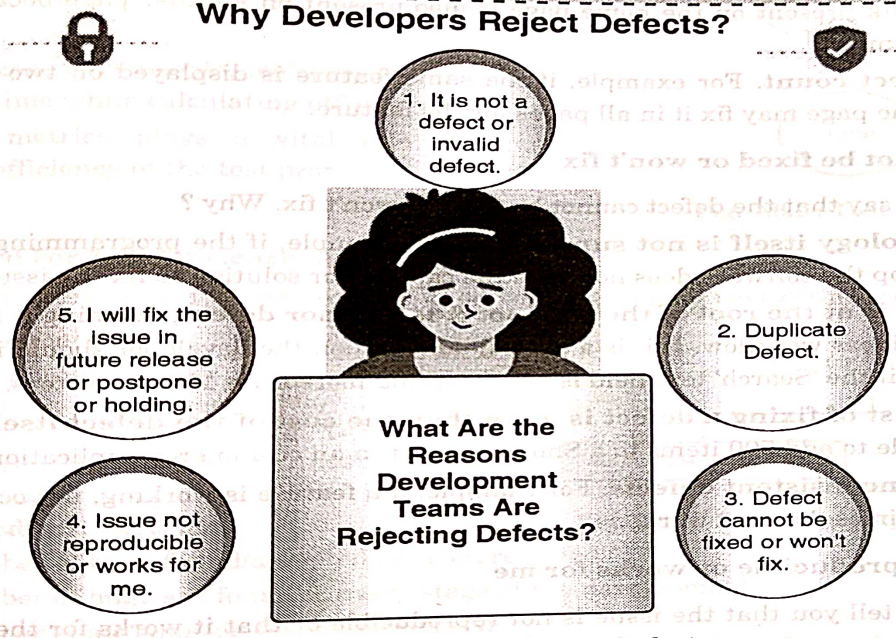


Fig. 2.6.1 : Developers Reject Defects

- As testers, defect tracking and reporting are key aspects within our role, as it is crucial to communicate such results back to our developers or higher management in order to ensure we release a stable and flawless product.
- However, we know that a lot of the times when we raise a defect, developers reject it for some reason.

1. It is not a defect or it's an invalid defect

Why does the Dev team say it is not a defect or that it is an invalid defect ? By seeing the defect report, the Developer might say it is not a defect due to the following reasons :

1. **Misunderstanding of the requirement**. For example, a developer understands a feature as a link, whereas a tester understands it as a button.
2. **When the build or software is wrongly installed**. For example, when there's a mismatch in the build steps.

3. **Referring to the old requirement**. For example, when a feature is enhanced, the developer develops it with the new SRD (System Requirements Document), whereas the tester tests it using the old SRD.
4. **Adding extra features**. For example, the developer thinks it will be useful to add extra features, but the tester is using the requirement document that lacks the new features.

2. Duplicate defect

By reviewing the defect report, developers might say the defect is a duplicate defect due to the following reasons:

1. **Testing a common feature**, testers might find the same defect and thus it is duplicated. For example, a link present on the home page is also present on another page because the navigating page is the same.
2. **Reduce defect count**. For example, if the same feature is displayed on two or more pages and fixing it on one page may fix it in all pages for the feature.

3. Defect cannot be fixed or won't fix

Developers might say that the defect cannot be fixed or won't fix. Why ?

1. **If the technology itself is not supported**. For example, if the programming language which is used to develop the software does not have the capacity or solution to fix the issue.
2. **If the defect is at the root of the product & is a minor defect**, meaning it is not impacting the customer business workflow. If it is a critical defect, then the developer should indeed fix the issue. For example, if the 'Search' text field is not accepting more than 100 characters.
3. **When the cost of fixing a defect is more than the cost of the defect itself**. For example, if a user is not able to add 500 items to a Shopping Cart in an ecommerce application.
4. **Because of inconsistent defects**. For example, if a feature is working, but occasionally the same feature sometimes does not work.

4. Issue not reproducible or works for me

- Developers might tell you that the issue is not reproducible or that it works for them well. That might be due to a build mismatch and/or not enough test data. For example, if the platform/Browser/OS is not mentioned in the defect report, then the developer will try to reproduce it on another platform, and it might just work fine for him.
- We need to make sure to deliver proper defect reports with all the relevant information for reproducing the defect.

5. I will fix the issue in a future release or postpone or put on hold

There are also scenarios in which developers might say they will fix the issue in a future release, postpone it or put in on hold, due to the following reasons :

1. Finding a minor defect at the end of the release, and the developer might not have sufficient time. For example a spelling mistake on a link.
2. If a customer is planning to do a lot of requirement changes.

2.6.1 Test Efficiency

GQ. What Is Efficiency Testing ?

- Efficiency testing tests the number of test cases executed divided by the unit of time. The unit of time is generally in hrs. It tests the measure of code and the testing resources that are required by an application to perform a specific function.
- It evaluates how many resources were planned and how many were actually utilized for testing. It is all about getting the task done with minimal effort.
- Test efficiency considers people, tools, resources, processes, and time while calculating efficiency.
- Creating test metrics plays a vital role in measuring the efficiency of the test processes.

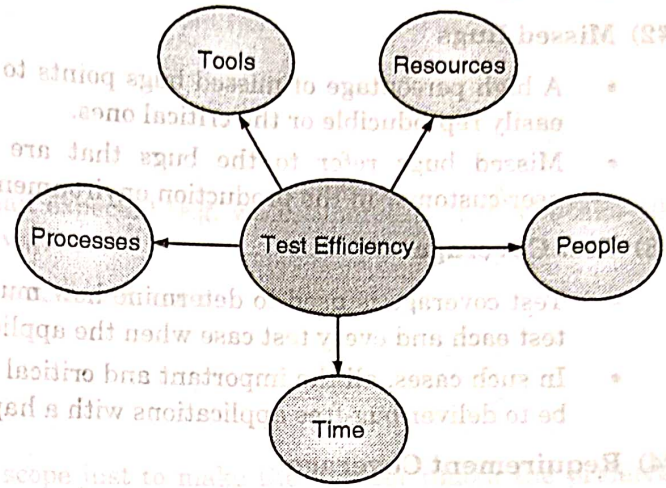


Fig. 2.6.2 : Test Efficiency

Techniques Used For Test Efficiency (Refer Fig. 2.6.3)

Both the techniques, given below, can be used for evaluating test efficiency :

#1) Metric Based Approach

Metric based approach helps to get an idea of enhancing the testing processes when it is not progressing as expected. The prepared Test Metrics must be analyzed properly, as it helps to estimate the efficiency of the testing process.

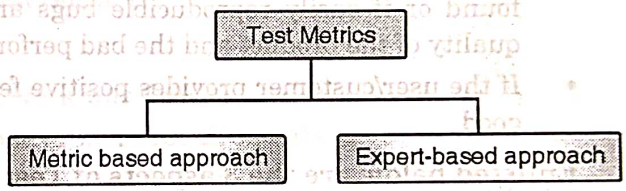


Fig. 2.6.3 : Both Techniques for test metrics

Commonly used test metrics:

- A total number of bugs found/accepted/rejected/resolved.
- A total number of bugs are found in every stage of the development.
- A total number of automation test cases written.

The mostly used metric is :

The total number of bugs found in different phases of testing :

- (Total number of bugs resolved)/ (Total number of bugs raised) *100
- There are several metrics but the best one can be created by experienced testers themselves based on knowledge and analysis.
- Certain Metrics like written automation test cases, and the number of bugs found are of not much use as the number of test cases can be higher. However, if the major cases are missing, then it is not useful. In the same way, the number of bugs raised can be higher but missing the major functionality bugs can be an issue.

Let's go through a few metrics that can be used in a project.

1. Rejected bugs
2. Missed bugs
3. Test coverage
4. Requirement coverage
5. User feedback

#1) Rejected Bugs

The percentage of rejected bugs gives an overview of how much the testing team is aware of the product that is under testing. If the percentage of rejected bugs is high, then it clearly exhibits a lack of knowledge and understanding of the project.

#2) Missed Bugs

- A high percentage of missed bugs points to the testing team's capability especially if the bugs are easily reproducible or the critical ones.
- Missed bugs refer to the bugs that are missed by the testing team and are found by the user/customer in the production environment.

#3) Test Coverage

- Test coverage is used to determine how much the application has been tested. It is not possible to test each and every test case when the application is complex or too large.
- In such cases, all the important and critical features should be tested properly and the focus should be to deliver bug-free applications with a happy path.

#4) Requirement Coverage

For efficiency testing, the requirement covered by the application, and the number of requirements tested and passed for a feature play an important role.

#5) User Feedbacks

- Testing efficiency can be calculated based on the feedback provided by the user. If critical bugs are found or if easily reproducible bugs are reported by the user, then it clearly points to the bad quality of the product and the bad performance of the testing team.
- If the user/customer provides positive feedback then the efficiency of the testing team is considered good.

Enlisted below are the 3 aspects of Test Efficiency:

1. Client requirements are being fulfilled by the system.
2. Software specifications to be achieved by the system.
3. Efforts were put in to develop a system.

Thus, Metric based approach is based on the calculations.

#2) Expert-Based Approaches

- The expert-based approach is based on the experience of the tester who tests the software along with the knowledge gained from his previous projects.
- Test effectiveness is measured by how well the system behaves as per the user's expectation. If the system is effective, the user easily achieves the goals set for testing.

2.6.2 Efforts and Schedule Variance

GQ. What is Effort Variance ?

Effort Variance

- Effort Variance = [(Actual Effort - Planned Effort) / Planned Effort] * 100
- It provides variance of Actual Effort vs. Planned Effort.
- As an example suppose the Quality Objective is that the **Effort Variance** should not exceed 5%.

Project X

- Planned Effort = 80
- Actual Effort = 95
- Effort Variance = (95 – 80)/80 × 100 = 18.75

Project Y

- Planned Effort = 110
- Actual Effort = 115
- Effort Variance = (115 – 110)/110 100 = 4.55
- The **effort variance** may come out to be more than expected (e.g. we estimated that it will take 100 hours but in actual it took 110 hours, + 10 % effort variance)

Some of the causes why this variance might have occurred are :

- Estimation parameters were wrong.
- Scope was not understood in totality.
- Inefficient process.
- Someone changed the estimate without changing scope just to make the number match the preferred schedule
- Added capability the customer did not request.
- The **effort variance** may come out to be less than expected (e.g. we estimated that it will take 100 hours but in actual it took 90 hours, -10% effort variance)

Some of the causes why this variance might have occurred are :

- Estimation parameters were wrong.
- There was an improvement in the process and brilliant work was done.
- We did not complete the task and probably missed on one or more requirements.
- Missed out on some steps.

GQ. What Is Schedule Variance ?

- Schedule Variance (SV) is a term for the difference between the earned value (EV) and the planned value (PV) of a project. It is used a measure of the variance analysis that forms an element the earned value management techniques. An alternative but less common classification of this technique is earned schedule management or analysis.
- The schedule variance indicates whether the performance – i.e. the authorized work performed – exceeds, falls below or is equal to the planned performance. Projects that apply the PMI methodology use SV typically in their "control schedule" process.
- There are two different types of schedule variances :
  1. Point-in-time or period-by-period schedule variance and
  2. Cumulative schedule variance.
- While both types share the same calculation approach, their meanings can differ significantly. Learn more about these differences in the following sections and the examples below.
- The corresponding indicator for the cost controlling in a project is called cost variance.

GQ. What is Point-in-time / Period-by-Period Schedule Variance ?

- Point-in-time or period-by-period schedule variance refers to the difference between earned value (as observed and measured in a period) and planned value with respect to a single period.

- Variances of other periods, such as excesses or shortfalls, are not considered.
- **Example :** If you are calculating the schedule variance for the 2nd month, for instance, you would not take the SV of the 1st and 3rd month into account.
- This is the key difference to the cumulative schedule variance. Read on for more details.

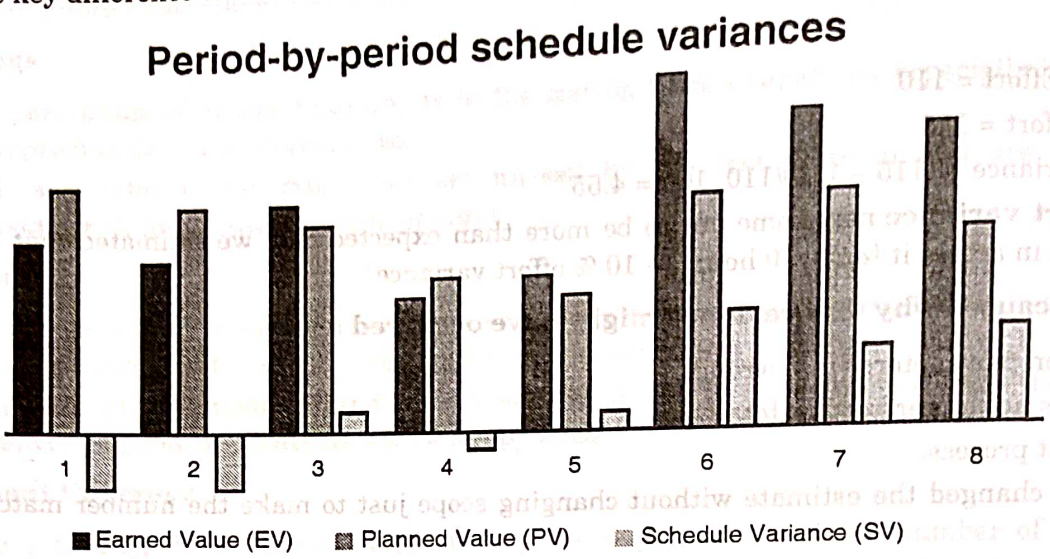


Fig. 2.6.4 : Schedule Variances

GQ. What Is Cumulative Schedule Variance ?

- The cumulative SV refers to the difference between earned and planned value over several – mostly consecutive – periods. It is both the sum of the point-in-time schedule variances of all periods in scope or the difference of the sum of EVs and the sum of PVs for these periods.
- **Example:** If you intend to calculate the cumulative schedule variance for the periods 1 to 3, for instance, you will sum up the period-by-period SVs of the periods 1, 2 and 3. Alternatively, you can determine the sum of EV(per.1) + EV(per.2) + EV(per.3) and deduct the sum of PV(per.1) + PV(per.2) + PV(per.3) which results in the cumulative schedule variance.

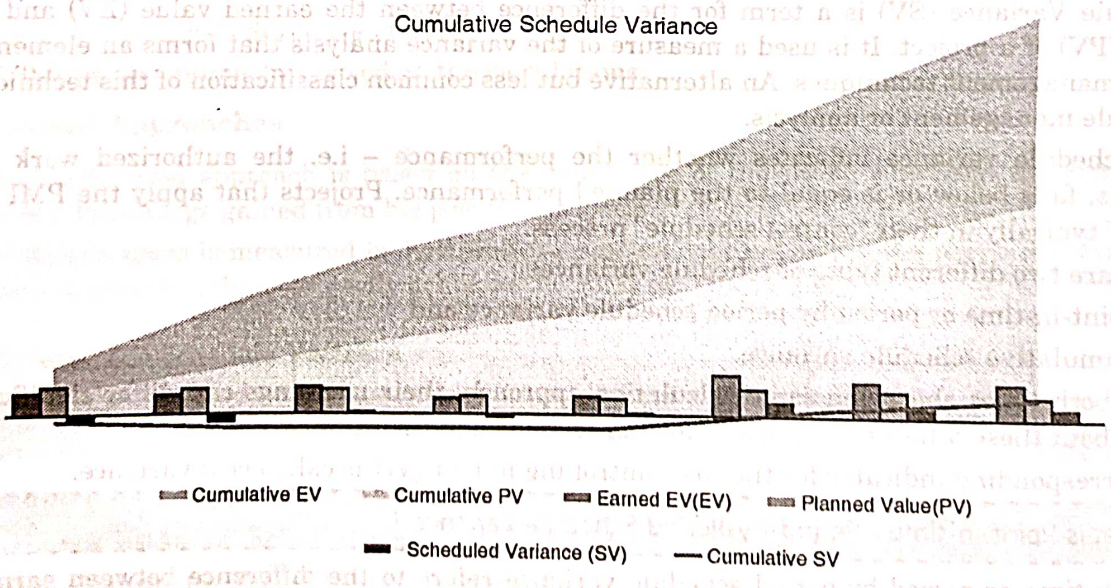


Fig. 2.6.5 : Cumulative Schedule Variance

GQ. How Is Schedule Variance Calculated ?

- You can use the following formula to calculate the schedule variance (SV) of one or several periods:

  SV = EV – PV,

  where EV = Earned value; PV = Planned value.
- Earned value is determined in the earned value analysis. It indicates how much of the authorized work (measured in allocated budget) has been completed within a single or a time frame of several periods.
- Planned value is the part of the budget that is allocated to the amount of work that should have been completed in a period or several periods. It is derived from the project baseline (or project plan).
- Both parameters must be denominated in the same unit – typically a currency unit (like $) or man-days – and refer to the same period(s).
- You have probably noted that both the EV and the PV definitions imply two options: they can either refer to a single period or multiple periods. While the basic calculation is basically identical for either case, the basis of the parameters would be different. This is further explained in the following sections.

GQ. How Is the Period-by-Period Schedule Variance Calculated ?

- The calculation of the period-by-period or point-in-time schedule variance follows the previously introduced basic formula:

  SV (period) = EV (period) – PV (period).
- Earned value (EV) and planned value (PV) refer to a single period in this case.

GQ. How Is the Cumulative Schedule Variance Calculated ?

- For the cumulative schedule variance, the basic formula is used with input values for multiple periods :

  SV (cumulative) = EV (cumulative) – PV (cumulative)

  or SV (cumulative) = Sum of SV (all periods),

  where SV(all periods) refers to all point-in-time SVs of the periods in scope.
- It is common in projects that the cumulative schedule variance is measured for the first to the most recent period. However, it can also be calculated for a different time frame, e.g. the 2nd to 5th period of a project, if this information is required.

GQ. What is the Meaning of the Calculated Schedule Variance Values ?

- Similar to other variance indicators in project management, the schedule variance comes with three potential value ranges that have their own respective meaning :
  - A negative schedule variance (SV < 0) indicates that the project is behind the schedule, as earned value does not meet the planned value.
  - A positive schedule variance (SV > 0) indicates that the earned value exceeds the planned value in the reference period(s), i.e. the project is ahead of the schedule.
  - If the schedule variance is 0 this indicates that that the schedule baseline is met, i.e. the earned value is equal to the planned value.
- Note that the values of the different types of SV may vary within the same project, depending on the reference period(s) that you have selected.
- For instance, the SV of a single period can be negative while the cumulative schedule variance can as well be 0 or positive at the same time.

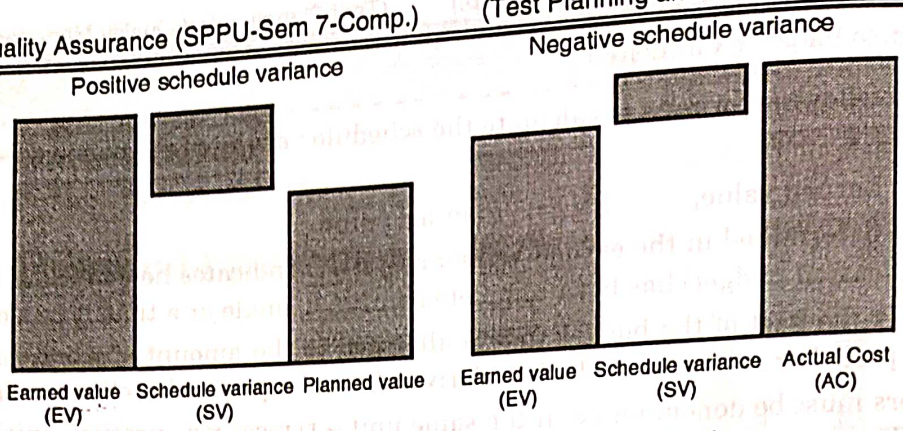


Fig. 2.6.6 : Positive and Negative schedule Variance

☞ Examples of Schedule Variance Calculation and Analysis

- In the following 2 examples, we will guide you through the use and calculation of the schedule variance in certain project situations.
- You will find a corresponding example with identical EV and PV amounts in our article on the schedule performance index.

☞ Example 1 : A Simple Calculation of Cumulative and Point in Time Schedule Variances

In this example, the PMO has observed the following numbers as part of their ongoing earned value analysis.

	Month 1	Month 2	Cumulative (Month 2)
Planned Value (PV)	50	150	200
Earned Value (EV)	60	130	190

☞ Calculation of Schedule Variances

- To determine the variance, the planned value is subtracted from the earned value for each period.
- The same approach is applied to the cumulative numbers. The calculation and results are illustrated in this table.

	Month 1	Month 2	Cumulative (Month 2)
Planned Value (PV)	50	150	200
Earned Value (EV)	60	130	190
Schedule Variance (SV) per period	60 – 50 = 10	130 – 150 = -20	
Schedule Variance, cumulative			190 – 200 = -10 or: 10 + (-20) = 10

☞ Interpretation of the Calculated SV

- The project is slightly behind schedule which is indicated by the negative cumulative schedule variance of -10.
- Considering the period-by-period numbers, a positive schedule variance is observed in the first period (earned value exceeds the planned value). However, the project missed its schedule baseline in month 2 where the planned value has not been met at all.
- In practice, such a breakdown can be a good starting point for a further root cause analysis if the project fails to meet the schedule baseline.

Example 2 : Case Study of a Project in a Turnaround Situation

- This example illustrates how the numbers for a turnaround situation can look like.
- The following table shows the numbers for months 1 to 3 of a project.

	Month 1	Month 2	Month 3	Cumulative (Month 3)
Planned Value (PV)	100	130	200	430
Earned Value (EV)	60	120	220	400

Calculating the Cumulative and Period-by-Period SV

- For the cumulative schedule variance, the numbers of the last column are used and inserted into the previously introduced formula :

$$\text{cumulative SV} = 400 - 430 = -30$$

- This negative cumulative SV indicates that the project is behind schedule after 3 months.
- On a period-by-period basis, the schedule variance is as follows :

	Month 1	Month 2	Month 3	Cumulative (Month 3)
Schedule Variance (SV)	– 40	– 10	20	– 30

Interpretation of the Schedule Variances

- The schedule variances starter with a negative SV of -40 in the first month which was then managed down to -10 in month 2. The situation improved eventually with a positive SV of + 20 in the third month.
- While the project still fails to meet the overall schedule baseline, the breakdown by months shows a positive trend: the initially negative schedule variance improved incrementally over time.
- If this development is sustainable and positive, or at least 0, variances can be retained, the project will have a realistic chance to be back on track within a few months. In practice this may also depend on other factors though.

2.6.3 Test Efforts Biasing Factors

GQ. Write a note on Test Efforts Biasing Factors.

- When you create test plans and estimate the testing effort and schedule, you must keep these factors in mind otherwise your plans and estimates will mislead you at the beginning of the project and betray you at the middle or end.
- The test strategies or approaches you pick will have a major influence on the testing effort. In this section, let's look at factors related to the product, the process and the results of testing.
- In Product factors the presence of sufficient project documentation is important so that the testers can figure out what the system is, how it is supposed to work and what correct behaviour looks like. This will help us do our job more efficiently.

The factors which affect the test effort are :

- While good project documentation is a positive factor, it's also true that having to produce detailed documentation, such as meticulously specified test cases, results in delays.
- During test execution, having to maintain such detailed documentation requires lots of effort, as does working with fragile test data that must be maintained or restored frequently during testing.
- Increasing the size of the product leads to increases in the size of the project and the project team.

- Increases in the project and project team increases the difficulty of predicting and managing them. This leads to the disproportionate rate of collapse of large projects.
- The life cycle itself is an influential process factor, as the V-model tends to be more fragile in the face of late change while incremental models tend to have high regression testing costs.
- Process maturity, including test process maturity, is another factor, especially the implication that mature processes involve carefully managing change in the middle and end of the project, which reduces test execution cost.
- Time pressure is another factor to be considered. Pressure should not be an excuse to take unwarranted risks. However, it is a reason to make careful, considered decisions and to plan and re-plan intelligently throughout the process.
- People execute the process, and people factors are as important or more important than any other.
- Important people factors include the skills of the individuals and the team as a whole, and the alignment of those skills with the project's needs. It is true that there are many troubling things about a project but an excellent team can often make good things happen on the project and in testing.
- Since a project team is a team, solid relationships, reliable execution of agreed-upon commitments and responsibilities and a determination to work together towards a common goal are important. This is especially important for testing, where so much of what we test, use, and produce either comes from, relies upon or goes to people outside the testing group. Because of the importance of trusting relationships and the lengthy learning curves involved in software and system engineering, the stability of the project team is an important people factor, too.
- The test results themselves are important in the total amount of test effort during test execution. The delivery of good-quality software at the start of test execution and quick, solid defect fixes during test execution prevents delays in the test execution process.
- A defect, once identified, should not have to go through multiple cycles of fix/retest/re-open, at least not if the initial estimate is going to be held to.

2.6.4 Test Report &amp

- Test reporting is essential for making sure your web or mobile app is achieving an acceptable level of quality.
- Done right, test reporting and analysis can add true value to your development lifecycle by providing the right feedback at the right time.
- In this blog, I'll share :
  - The basics of test reporting.
  - Test reporting challenges that Agile teams encounter.
  - Components of an effective test report.

GQ. What Is a Test Report ?

- A test report is an organized summary of testing objectives, activities, and results. It is created and used to help stakeholders (product manager, analysts, testing team, and developers) understand product quality and decide whether a product, feature, or a defect resolution is on track for release.
- Beyond product quality, a test report also provides insight into the quality of your testing and test automation activities. Organizations typically have four high-level questions about their test automation.
  - What's wrong with my automation scripts ?
  - What's wrong with my backend ?
  - What's wrong with my lab ?

- What's wrong with my executions ?

- Finally, test reporting should help you understand the achieved value of testing. For example, are you testing anything unnecessarily ? Are your tests stable ? Were you able to uncover issues early in the process ?
- A good test reporting process gives insight and answers to all these important questions. You can not only improve quality of an app, but you can accelerate your releases.

Challenges in Test Reporting & Analysis

Agile, DevOps, CI/CD - these hallmarks of modern development have changed the requirements for a "good" test report. Below, a few issues that can get between you and a timely, accurate test report.

Galloping Release Cadences

- Traditionally, a test report was compiled and summarized (using spreadsheets!) as one of the final stages of a waterfall development process.
- Releases were few and far between, so there was time to compile results, create a report, and make decisions.
- The fast release cadences made standard by Agile and DevOps movements have dramatically changed this.
- Testing needs to happen quickly. Decisions about quality need to be made not in the timeframe of months, but within weeks, days, even hours. If that feedback isn't available in time, the release is either stalled or shipped with questionable quality.

Noisy, High-Volume Data

- Today's testing teams generate mountains of data from tests. Mountains created, in large part, by both test automation (more testing) and device proliferation (more devices, browsers, and versions).
- The more data, the better, right ? Yes and no.
- Yes, if it's actionable. No, if it's not. Many organizations suffer from too much testing data. In that case, it is difficult to make sense of what is valuable and what is just noise.
- Noise is created from flaky test cases, environment instability, and other issues that cause false negatives for which we don't understand the root cause. In today's reality, digital enterprises must go through each failure that is being marked in the report.
- Reporting, then, is burdened by high volumes of irrelevant information.

Divided Data

- Another issue, particularly for larger organizations, is due to the number and variety of teams, tools, and frameworks.
- Simply :
  - There is a lot of testing data.
  - It is coming from many different people and teams (SDETs, developers, business testers, API testers).
  - It is arriving via different frameworks and formats (Selenium, Appium, etc.)
- Without a uniform way to capture and sort this data across the organization, good test reporting becomes dangerously difficult.

What Should a Test Report Contain ?

- What goes in a test report ? That depends on the mix of stakeholders using it as well as the sophistication of the team.

- Regardless, its contents should supply fast, actionable feedback. Everything should be described (or displayed in a test automation tool) as simply as possible - but not too simply. It needs the right granularity in the right areas to be useful.
- Remember, the test report is used to analyze quality and make decisions. If it is too simplistic, important nuances can be lost and result in poor decisions. If it is too granular, you and the team will have difficulty getting a sense of the overall quality picture.

☞ **Basic Test Report Summary**

- A very basic test report for a small application or organizations should include, at a minimum, the following :
  1. **Executive Overview :** Summary of key findings.
  2. **Test Objective :** Information about test type and purpose.
  3. **Test Summary :** Defining passed, failed, and blocked test cases.
  4. **Defects :** described with priority and status.
- For a larger organization, or for an organization implementing more sophisticated testing, the minimum will not be enough.
- Each of the test reports must include sufficient artefacts like logs, network traffic (HAR Files), screenshots, video recording, and other relevant data to help the reviewer make data-driven decisions.
- Test history - including defects found by the test, problematic platform or feature in the product - can provide immense value to the test reporting reviewers around next steps, test impact analysis, and test coping for the next cycle.

☞ **Test Reporting for Continuous Testing**

- When you're releasing quickly, often, and with the help of test automation - as most modern organizations do smarter testing and analysis is a necessity.
- To start, you need to time testing activities so that reporting and analysis are delivered at the most relevant time in your development pipeline.
- In the example below, you'll see unit, smoke, and regression testing timed to align with when they are relevant to the team. For example, conduct unit testing too late (or get the feedback too late) and you risk delaying a release. Sync regression tests on a nightly basis, so the team can get feedback and take action the next day.
- Good test reporting is delivered to the right teams at the right time.
- Aside from that, you will want a test reporting dashboard that is perfected for the pipeline. This would include the following :
- **Executive Overview :** Highlighting real-time trends for testing in the Continuous Integration pipeline.

2.7 CONFIGURATIONS MANAGEMENT

GQ. What is Software Configuration Management ?

- In Software Engineering, **Software Configuration Management (SCM)** is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.
- The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

GQ. Why do we need Configuration management ?

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

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

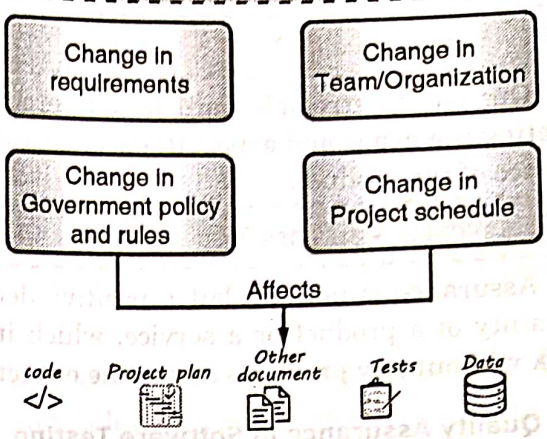


Fig. 2.7.1 : Configuration Management

☞ Tasks in SCM process

- Configuration Identification
- Change Control
- Configuration Audits and Reviews
- Baselines
- Configuration Status Accounting

☞ Configuration Identification

Configuration identification is a method of determining the scope of the software system. With the help of this step, you can manage or control something even if you don't know what it is. It is a description that contains the CSCI type (Computer Software Configuration Item), a project identifier and version information.

☞ Activities during this process

- Identification of configuration Items like source code modules, test case, and requirements specification.
- Identification of each CSCI in the SCM repository, by using an object-oriented approach
- The process starts with basic objects which are grouped into aggregate objects. Details of what, why, when and by whom changes in the test are made
- Every object has its own features that identify its name that is explicit to all other objects
- List of resources required such as the document, the file, tools, etc.

Example :

- Instead of naming a File login.php its should be named login_v1.2.php where v1.2 stands for the version number of the file
- Instead of naming folder "Code" it should be named "Code_D" where D represents code should be backed up daily.

2.7.1 Quality Assurance Process

GQ. What is Quality ?

Quality is extremely hard to define, and it is simply stated : "Fit for use or purpose." It is all about meeting the needs and expectations of customers with respect to functionality, design, reliability, durability, & price of the product.

GQ. What is Assurance ?

Assurance is nothing but a positive declaration on a product or service, which gives confidence. It is certainty of a product or a service, which it will work well. It provides a guarantee that the product will work without any problems as per the expectations or requirements.

Quality Assurance in Software Testing

- **Quality Assurance in Software Testing** is defined as a procedure to ensure the quality of software products or services provided to the customers by an organization.
- Quality assurance focuses on improving the software development process and making it efficient and effective as per the quality standards defined for software products. Quality Assurance is popularly known as QA Testing.
  - How to do Quality Assurance: Complete Process
  - What is Quality Control ?
  - Difference between Quality Control and Quality Assurance ?
  - Differences between SQA and Software Testing
  - Best practices for Quality Assurance
  - Quality Assurance Functions
  - Quality Assurance Certifications
  - CMMI level
  - Test Maturity Model (TMM)

How to do Quality Assurance : Complete Process

- Quality Assurance methodology has a defined cycle called PDCA cycle or Deming cycle. The phases of this cycle are :

  1. Plan 2. Do 3. Check 4. Act

- These above steps are repeated to ensure that processes followed in the organization are evaluated and improved on a periodic basis. Let's look into the above QA Process steps in detail :

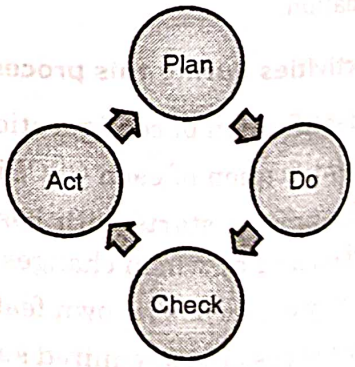


Fig. 2.7.2 : Quality Assurance Process

1. **Plan :** Organization should plan and establish the process related objectives and determine the processes that are required to deliver a high-Quality end product.
2. **Do :** Development and testing of Processes and also "do" changes in the processes
3. **Check :** Monitoring of processes, modify the processes, and check whether it meets the predetermined objectives
4. **Act :** A Quality Assurance tester should implement actions that are necessary to achieve An organization must use Quality Assurance to ensure that the product is designed and implemented with correct procedures. This helps reduce problems and errors, in the final product.

GQ. What is Quality Control ?

Improvements in the processes

- Quality control popularly abbreviated as QC. It is a Software Engineering process used to ensure quality in a product or a service. It does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome.
- The main aim of Quality control is to check whether the products meet the specifications and requirements of the customer. If an issue or problem is identified, it needs to be fixed before delivery to the customer.
- QC also evaluates people on their quality level skill sets and imparts training and certifications. This evaluation is required for the service based organization and helps provide "perfect" service to the customers.

GQ. What are Difference between Quality Control and Quality Assurance?

- Sometimes, QC is confused with the QA. Quality control is to examine the product or service and check for the result.
- Quality Assurance in Software Engineering is to examine the processes and make changes to the processes which led to the end-product.

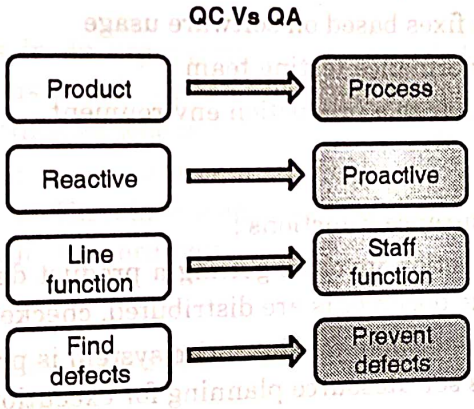


Fig. 2.7.3 : Quality Control Vs Quality Assurance

Quality Control Activities	Quality Assurance Activities
Walkthrough	Quality Audit
Testing	Defining Process
Inspection	Tool Identification and selection
Checkpoint review	Training of Quality Standards and Processes

- **The above activities are concerned with Quality Assurance and Control mechanisms for any product and not essentially software.** With respect to software
  - QA becomes SQA (Software Quality Assurance)
  - QC becomes Software Testing.

☞ Differences between SQA and Software Testing

Following table explains on differences between SQA and Software Testing :

SQA	Software Testing
Software Quality Assurance is about engineering process that ensures quality	Software Testing is to test a product for problems before the product goes live
Involves activities related to the implementation of processes, procedures, and standards. Example – Audits Training	Involves actives concerning verification of product Example – Review Testing
Process focused	Product focused
Preventive technique	Corrective technique
Proactive measure	Reactive measure
The scope of SQA applied to all products that will be created by the organization	The scope of Software Testing applies to a particular product being tested.

☞ Best practices for Quality Assurance

- Create a Robust Testing Environment
- Select release criteria carefully
- Apply automated testing to high-risk areas to save money. It helps to fasten the entire process.
- Allocate Time Appropriately for each process
- It is important to prioritize bugs fixes based on software usage
- Form dedicated security and performance testing team
- Simulate customer accounts similar to a production environment

☞ Quality Assurance Functions

There are 5 primary Quality Assurance Functions :

1. **Technology transfer :** This function involves getting a product design document as well as trial and error data and its evaluation. The documents are distributed, checked and approved
2. **Validation :** Here validation master plan for the entire system is prepared. Approval of test criteria for validating product and process is set. Resource planning for execution of a validation plan is done.
3. **Documentation :** This function controls the distribution and archiving of documents. Any change in a document is made by adopting the proper change control procedure. Approval of all types of documents.
4. **Assuring Quality of products**
5. **Quality improvement plans**

☞ Quality Assurance Certifications

- There are several certifications available in the industry to ensure that Organizations follow Standards Quality Processes.
- Customers make this as qualifying criteria while selecting a software vendor.

☞ ISO 9000

- This standard was first established in 1987, and it is related to Quality Management Systems. This helps the organization ensure quality to their customers and other stakeholders.
- An organization who wishes to be certified as ISO 9000 is audited based on their functions, products, services and their processes.

- The main objective is to review and verify whether the organization is following the process as expected and check whether existing processes need improvement.
- This certification helps –
  - Increase the profit of the organization
  - Improves Domestic and International trade
  - Reduces waste and increase the productivity of the employees
  - Provide Excellent customer satisfaction

2.7.2 Documentation Risk &amp

GQ. Write a Note on Documentation Risk.

- Risk Management documentation is what helps companies understand what risks can potentially disrupt or benefit business operations. These living documents are instrumental in identifying, assessing, mitigating, monitoring, and reporting threats. With effective risk documents in place, businesses can take a proactive approach on how to address and respond to unexpected incidents when they occur.
- Here are some key documents that should be a part of your risk management process. This can help prepare for unforeseen events that may impact your business.

Valuable Risk Management Documentations

Risk Register

- An inventory of identified risks in an organization, with detailed steps and explanations on how to mitigate them. Also referred to as a risk log, this document helps businesses record and monitor the probability of each risk. It also indicates who is responsible for managing each risk.
- In addition, auditors can gather useful data from a risk register to determine if a risk is unmanageable, and the cost to alleviate said risks. Therefore, companies must frequently update a risk register to stay current on emerging risks and adjust accordingly.

Risk Management Plan

- This is a document that outlines how a business will identify, analyze, and respond to risks that may be a threat to business operations.
- Staff across all departments use risk management documentation to communicate and react appropriately during unforeseen circumstances, thus minimizing their impact.

Contingency Plan and Mitigation Plans

The goal of both documents is to outline policies, processes, and procedures for unexpected events – however, each is implemented at different times. For a mitigation plan, a business should enforce this before a risk takes place. This plan will help detect potential risks and determine the appropriate action, so you can control and minimize damage. If specific events occur the contingency plan dictates what steps to take after that fact.

Risk Reports

- Documents that communicate the progress or status of your risk management plan. A risk report addresses the most pressing risks and how well they are being managed.
- Furthermore, the report should include details on any emerging risks, information on inadequate policies and procedures, and data that will be crucial to future decision-making.

Risk Response Strategies

- A course of action on how to handle each risk. Also known as risk treatment, this document will detail different types of strategies ranging from avoidance, transference, mitigation, or acceptance.
- Each strategy involves a different approach how to deal with any potential risk. Furthermore, risk response planning may include details on the duties and responsibilities of staff, resources needed to mitigate risks, the time frame on when to implement the process, and the regular review of each strategy.

Risk Assessment Matrix

A diagram that evaluates the likelihood and impact of potential risks. This tool can help businesses visualize certain risks and prioritize them into categories of probability and severity. This level of detail makes it easier and faster for businesses to calculate risks.

2.8 ISSUES IN SOFTWARE QUALITY

GQ. What are the Issues in Software Quality ?

That said, software QA teams need to address several challenges before they can deliver results. Here are some of the main challenges, they should consider.

1. Facilitation of Quality

- To accept a system for testing software, it's extremely important to understand its nature. This is why product owners must join hands QA and testing professionals so they can test the product according to the business model.
- QA teams need to give their input in the initial discussions because that's necessary in facilitating quality.

2. QA Culture

- The QA culture plays a key role in streamlining QA processes. Companies need to change their testing culture and explore innovative ideas. Without experimenting with unique techniques and different QA testing tools, it's difficult for organizations to test software products efficiently and quickly.
- Developing a comprehensive QA is necessary to meet market demands and deliver quality.

3. Early Testing

- Similar to development engineers, QA and testing professionals have a crucial role in the Software Development Lifecycle. If you begin testing in the early stages of the development cycle, you can identify several quality issues throughout the whole cycle more effectively.
- It's easier to give valuable early feedbacks and offer solutions on features, the scope of the project, sustainable development, and architecture. Each one of these elements cuts high additional expenses that are a result of late detection of errors. At the same time, it reduces unnecessary delays in time-to-market, giving teams a chance to compete more effectively.
- However, as your software quality assurance needs to test code frequently, it's hard to document everything accurately. You need a fully committed and dedicated testing team that can create self-sufficient stories for testing. Although implementing testing early is not easy, the benefits of early testing outweigh the several challenges it poses.

4. **The Merge of Development with Testing**

- To build an effective development team, organizations need to mix the development teams with software quality assurance teams in terms of knowledge sharing. That being said, QA and testing professionals should be encouraged to have some knowledge about the development process of a product while the developers should be encouraged to take active parts in enhancing test automation infrastructure and should possess knowledge about different QA testing tools.
- Knowledge sharing is the key in this situation. Organizations must therefore make the development team work together with the software assurance team. This will result in a developed and effective quality development team.
- Developers must learn about the testing process. On the other hand, testers must also have sufficient knowledge about software development. Hence, this will ensure us that the software we are using is definitely of good quality.

5. **Time**

- When it comes to the time it takes to test, device and browser coverage, developers taking up QA efforts, and the small number of hours in a day all add up. If a team needs to test for all of the newest web and mobile devices and browsers, the team is either continually stuck with the cost of updating their device farm, or they're relying on simulation software to provide accurate representations of the customer experience.
- Not only is the former expensive, it's also time consuming. On the other hand, the latter is more scalable but also a less accurate depiction of customer experience.
- Additionally, QA can seriously become a bottleneck for engineering teams when developers are stuck combing through a product for bugs as opposed to focusing on the next build. Developers should being looking ahead at what's next, not stuck critiquing their own creations.
- Lastly, even with a fully scaled QA process or in-house team, there are only so many hours in the day and only so many hours that can occupy the time of dedicated QA resources. Maybe you need testing overnight, over the weekend, or over the holidays ? These duties are simply difficult to impart upon an in-house team.

6. **People**

- Moreover, one could argue for a connection between the comprehensiveness of testing and the amount of personnel available to provide this testing.
- Scalability is a critical factor in software testing, as the number of testers per test cycle can impact both the time it takes to receive results as well as the quality and overall completeness of findings.
- Even if a team does have a robust personnel count dedicated to QA, how often are tests running ?
- Eventually, any team will hit capacity, and there will be a backup of scheduled tests. One way of providing against this is with an unlimited testing model, something that is only possible with crowd testing in particular.

7. **Automation**

- Many teams don't have automation in place yet and assume it to be the answer to everything. They may also lack the time or resources to get it running efficiently. Many teams don't have the time to manage automation and subsequent documentation when they're trying to move quickly.
- Every time a minor change is made, a given test case might be affected. This often necessitates that someone be hands-on-deck for all subsequent changes. Even issues as simple as broken links (sometimes due to changes in content or functionality on the website), can be hard to keep up with without a large in-house QA team keeping up-to-date on all, even minor, product changes.

- If conversations with our prospective customers have taught us anything, it's that if we want to see real improvement in the QA process, we need to affect the common challenges facing most teams.
- We need decrease the time required to test effectively, minimize the amount of internal employees needed to test completely, and properly support the development of automation processes.
- No QA process is perfect, but changes in QA technology - crowd testing for one - are beginning to address the above challenges, allowing teams to scale and move much more quickly, with fewer restraints on personnel, finances, and time.
- Technology is developing fast. The field of software quality assurance has grown rapidly. This growth is making developers face a lot of challenges.
- Despite of these challenges, developers were not fazed. On the contrary, they made developers more determined to do good with their job. What challenges did SQA face ?

8. Test Coverage

- Requirements change as fast as technology develops. There is always the possibility that important testing functions will be missed.
- Because of this, test requirements must be traced thoroughly. Maintaining traceability of requirements is most important.

9. Quality Assurance Culture

- Quality assurance companies need to explore. In fact, the testing and quality assurance culture must experience change. This change will be resulting in performing SQA activities properly.
- Finding unique techniques is a way of dealing with this challenge. Exploring on innovative ideas will help. Product testing will be then be done faster and efficiently. Being able to meet the demand for high quality products in the market will be ensured.

10. Build Verification

Frequent builds may increase the possibility of code breaking existing features. Automated testing should therefore be used then. Therefore it is not advisable to use manual testing.

11. Facilitation Of Quality

- It is very important for a quality assurance team to understand that a system needs to be verified.
- The nature of the business must be considered hence. The Quality assurance team must coordinate and work together with product owners and business experts.
- The quality assurance team must be involved in discussing about important concerns. This will consequently make them facilitate product quality.

12. Collaboration

- Another challenge that the quality assurance team may encounter is collaboration.
- Involving the team from the early stages of development which, as a result, lead to proper collaboration. Hence making software developed and supported effectively.

13. Skill Gaps

Sometimes, your team members have gaps in their skills and the type of expertise the project demands. As a QA project manager, you have to identify what skills your testing resources lack, so you can create an appropriate training plan for them. For instance, if some of your team members have to improve their testing skills, you should help them close that skill gap.

14. Training and Assessment

- In human resource planning, you should consider how existing members are trained and developed to gain the required skills and develop expertise.
- QA project managers need to create a training plan and apply it promptly to fill in the talent gap.

15. Evaluation

- Putting your testing resources is not sufficient on its own. Instead, it's important for QA project managers to monitor and evaluate these programs frequently, so they can ensure they are effective.
- For instance, you can work on your developer's training expertise by assessing his progress and assigning him different tasks to see if he has learned from the training initiatives. If the developer is finds testing difficult, the manager should either replace him or consider an alternative training method.

Effective Test Team is the Key Point of the Success

- QA teams build the testing infrastructure, related to automated test design and integration.
- Since complex software products rely on several testing procedures, QA project managers help create large, multi-dimensional project testing teams to maintain the best software quality.
- As DevOps principles slowly become the norm, development environments too will become more agile in terms of roles and responsibilities.
- Because integrated DevOps environments don't have the space to focus on standards and procedures separately, QA might become a regular feature of the development process with time.
- Here are a few ways Quality Assurance, influences application development nowadays :
  - As QA ensures the implementation of standard requirements and procedures, the planned course of action by the QA team guarantees product quality and success.
  - QA helps the entire software team in terms of reporting paperwork, documentation, and associated data.
  - QA ensures that the all essential procedures are being followed throughout the entire development cycle.
  - QA team performs activities such as testing, process monitoring, audits, as well as product evaluation to improve productivity.
  - QA gives a transparent outlook of collaboration and management for DevOps teams.
- Finally, an efficient testing strategy should comprise of different types of QA testing that include manual, automated and exploratory testing. All of these contribute to effectively tighten the release cycles and lower risks.
- Three types of QA testing have to be included in a robust testing strategy. These are unit tests to validate trivial components, integration tests to validate the compatibility of the components and functional tests to validate end-user scenarios.

2.8.1 Quality Management Importance

GQ. What is Quality Management ?

- Quality management is the way to manage all activities and actions that must be completed to keep a certain degree of distinguish, including developing and executing a quality policy, as well as generating and implementing quality planning and assurance, quality control, and quality improvement.
- The staff working in the Quality management field refers to it as Quality Management System (QMS). The deployment of a QMS often begins at the executive level, with the definition of corporate goals and objectives.

- These aims and goals are translated into policies, methods, and standard operating procedures (SOPs) that are implemented throughout the business.

GQ. What is the Importance of Quality Management ?

- Quality management is critical to the growth and performance of any firm. It is also a valuable resource in the struggle for client connections since it strives to provide a better customer service experience. Quality must be maintained at all levels for your business to prosper.
- Companies may put in place systems to guarantee that their goods satisfy the highest quality requirements and work properly. The aim is to increase customer happiness while driving corporate growth.

Let us provide you with 6 reasons why it is important to implement a quality system in your corporate :

1. Coherent quality and production of the products

- The importance of quality management resides in its potential to assist businesses in improving the dependability, durability, and performance of their goods. These elements help a company stand out from its competition.
- Better goods result in happy consumers and increased income. Quality management systems, such as ISO 9001, establish clear communication structures, roles, and duties across all divisions, in addition to product quality. As a result, staff morale rises, performance improves, and efficiency rises.

2. Achieve customer satisfaction

- Consumers are more demanding than ever in today's competitive economy. Because to technological advancements, they may pick from thousands of brands and have access to millions of outlets. If you want your company to stand out, you must meet or surpass their expectations.
- Quality management may assist you in converting prospects into loyal clients by constantly refining your goods, absorbing modifications, and eliminating flaws. It also provides businesses with the knowledge they want to build items and services that customers desire.
- In the long term, this increases your market share and offers your company a competitive advantage.

3. Higher Productivity Levels

- Employee productivity increases when the company recognizes and implements the Importance of Quality Management in all its business activities.
- They are aware and realize that they are working on something unique and of high quality, and that because of the impediments and bottlenecks which are immediately ironed out resulting in enhancing their production levels.

4. Reduced Risk Possibilities

- Risk management isn't only about selecting proper company insurance and investing in cutting-edge data security tools. There are several concerns to consider after your items leave the facility.
- Recalls, for example, can cause considerable long-term financial losses as well as negatively impact the customer experience.
- They may also have a negative impact on your brand and reputation. You, as a business owner, are accountable for the expenses of product recalls.
- In the worst-case situation, you may be forced to deal with lawsuits and perhaps declare bankruptcy. As a result, businesses cannot afford to neglect or dismiss the significance of quality management.

5. **Less Human Errors**

- When a company adheres to the Importance of Quality Management, it also adheres to a set of standards and principles that have been developed for each of its business processes. And everyone in the business, from the top management to the management trainees, must follow the same rules.
- Consequently, there are fewer human mistakes, which increases productivity and job efficiency. Furthermore, with fewer human mistakes, there is a far lower probability of harm.

6. **Stand out in the competition**

- Small firms exert so many efforts to compete with their larger counterparts. It is critical to provide great products and services. Quality management systems give information and rules to ensure that things are done correctly.
- Furthermore, they assist your company in achieving maximum cost efficiency and resource usage.
- In the long term, these actions build your company's brand, propelling you beyond your competition and leading an advanced market position since they enhance your goods and business processes.

5 Key Elements of Quality management

For the management system to reach its full potential, the corporate must work to implement some elements.

1. **Leadership is key to conquer all difficulties**

- Instead of a boss, there must be a leader that is, a devoted person in charge of each firm, and strives to achieve the specified objectives, a person who leads the whole team and operate as a unit, rather than focusing just on directing like a boss would.
- Leadership is combined with effective communication to other members of the organization; once we know where we're going, the next step to achieving participation from other members is to inspire and invite them to feel ownership of the Management System and to be able to put all your skills at their disposal.

2. **For a clear vision of the future, Planning is the solution**

- To ensure that a proper quality management system is in place, the many personnel aspects of the business must collaborate in the same direction. As a result, every employee in the business will feel included and encouraged to accomplish the standards.
- It is not enough to be clear about where we want to go and set goals; we must also spend time identifying the way. We sometimes envisage very long-term plans, which may be difficult and even demotivating.

3. **Continuous Improvement**

- Only the best will survive in the market, which will always be competitive. As a result, to remain competitive, all businesses must adapt and change. Continuous adaptation and improvement are vital to a company's success.
- Compliance requirements, risk-based thinking, quality planning processes, innovation, safety design, and QMS evaluation are all effective ways to accomplish this.

4. **Teamwork**

- Teamwork is also an important aspect of TQM for corporate success. The usage of teams will provide the firm with faster and better problem-solving solutions. Teams can also deliver more long-term enhancements to procedures and operations.

- People feel more comfortable bringing up difficulties in teams, where they may obtain aid from other workers to develop and implement a solution.

5. Documents and records

- Document management is primarily about information management, and the two areas overlap extensively. Documents such as standard operating procedures (SOPs) ensure that each process is consistent.
- Documents should be maintained up to date, accurate, and safe on a centralized platform. Choosing the best document management software should be a top concern for every growing business.
- This sounds like a great help for corporate, but do you think it might contribute to the occurrence of some flaws ? Let's look at the advantages and disadvantages of the Implementation of a Quality Management System.

☞ **Quality Management Advantages**

- It can aid in the prevention of the sale of defective products and services.
- It has no effect on output since workers continue to work while inspectors check.
- As with any quality system, the company may profit from a better reputation for quality, which may lead to increased sales
- Quality Management Disadvantages
- It does not avoid resource waste when items are defective.
- The process of checking products or services is costly, such as the salary given to inspectors and the expense of testing goods in a laboratory.
- It does not encourage all employees to be quality conscious.

Adopting the right Quality Management system requires careful study and resources, but it is all worth the hustle once you've seen your corporate competes ferociously. To learn more about the courses related to Quality Management here

2.8.2 Quality Best practices

- Having a QMS with established practices helps improve customer satisfaction and loyalty while meeting regulatory requirements.
- A strong reputation for quality is an important differentiator in today's market. Better products equals happier customers and higher revenue.
- Globalization and a highly distributed supply chain have an impact on the quality of the products and services. With dispersed teams and remote workers, organizations need to design and deliver high-quality products on time while meeting both customer and regulatory requirements.
- As a result, an organization needs to have next-generation quality management processes and tools that support the entire organization and not just a small set of key personnel. However, to have a successful quality management system, quality management principles must be established and reviewed consistently.

GQ. What are the top 10 Best Practices for Ultimate Quality Management ?

- The concept of quality management has four different components :
  1. Determination of quality policy
  2. Quality planning and assurance
  3. Quality Control
  4. Quality Improvement

Some of the best practices for implementing quality management in an organization are listed below.

1. Document quality measures and metrics
2. Involve stakeholders to identify and define quality standards
3. Get feedback on quality metrics and proposed measures from within the organization, customers, and other stakeholders
4. Implement a proactive approach for addressing quality issues instead of a reactive approach
5. Conduct continuous quality management process throughout the lifecycle
6. Track quality metrics and measures to understand the trend, improve or modify quality standards, and remain compliant
7. Periodically review quality standards and metrics
8. Establish thresholds that can be used to define CAPA
9. Analyze quality impact across the industry, including product, finance, customer service, and suppliers management
10. Constantly look to improve quality

☞ **#Exemplar/Case Studies**

1. **Online Recommendation System Graph Databases Enable Personalized Recommendations**

- Using graph analytics for recommendations is the first step towards faster and more personalized recommendations. It's worth remembering that the standard collaborative filtering algorithm is a 3-hop graph query :
- (**People** → who bought the **product** ← which **you** just bought) → also bought these other **products.**
- But all graph databases are not created equal. TigerGraph easily handles multiple hops, while many others struggle with more than two hops. TigerGraph's deep link analysis lets vendors customize and extend their analytics, enabling more hops to collect product features, customer demographics, and the context of the current situation, resulting in more accurate and more personalized recommendations.

2. **Quality Engineering services for Medical Devices Company Case Study**

Case studies of innovative medical device companies from India : barriers and enablers to development

☞ **Abstract**

Background

- Over 75% of the medical devices used in India are imported. Often, they are costly and maladapted to low-resource settings.
- We have prepared case studies of six firms in Bangalore that could contribute to solving this problem. They have developed (or are developing) innovative health care products and therefore are pioneers in the Indian health care sector, better known for its reverse engineering skills.
- We have sought to understand what enablers and barriers they encountered.

Methods

- Information for the case studies was collected through semi-structured interviews. Initially, over 40 stakeholders of the diagnostics sector in India were interviewed to understand the sector. However the focus here is on the six featured companies.
- Further information was obtained from company material and other published resources.

Results

- In all cases, product innovation has been enabled by close interaction with local medical practitioners, links to global science and technology and global regulatory requirements.
- The major challenges were the lack of guidance on product specifications from the national regulatory agency, paucity of institutionalized health care payers and lack of transparency and formalized Health Technology Assessment in coverage decision-making.
- The absence of national evidence-based guidelines and of compulsory continuous education for medical practitioners was key obstacles in accessing the poorly regulated and fragmented private market.

Conclusions

- Innovative Indian companies would benefit from a strengthened capacity and interdisciplinary work culture of the national device regulatory body, institutionalized health care payers and medical councils and associations.
- Continuous medical education and national medical guidelines for medical practitioners would facilitate market access for innovative products.

...Chapter Ends