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.

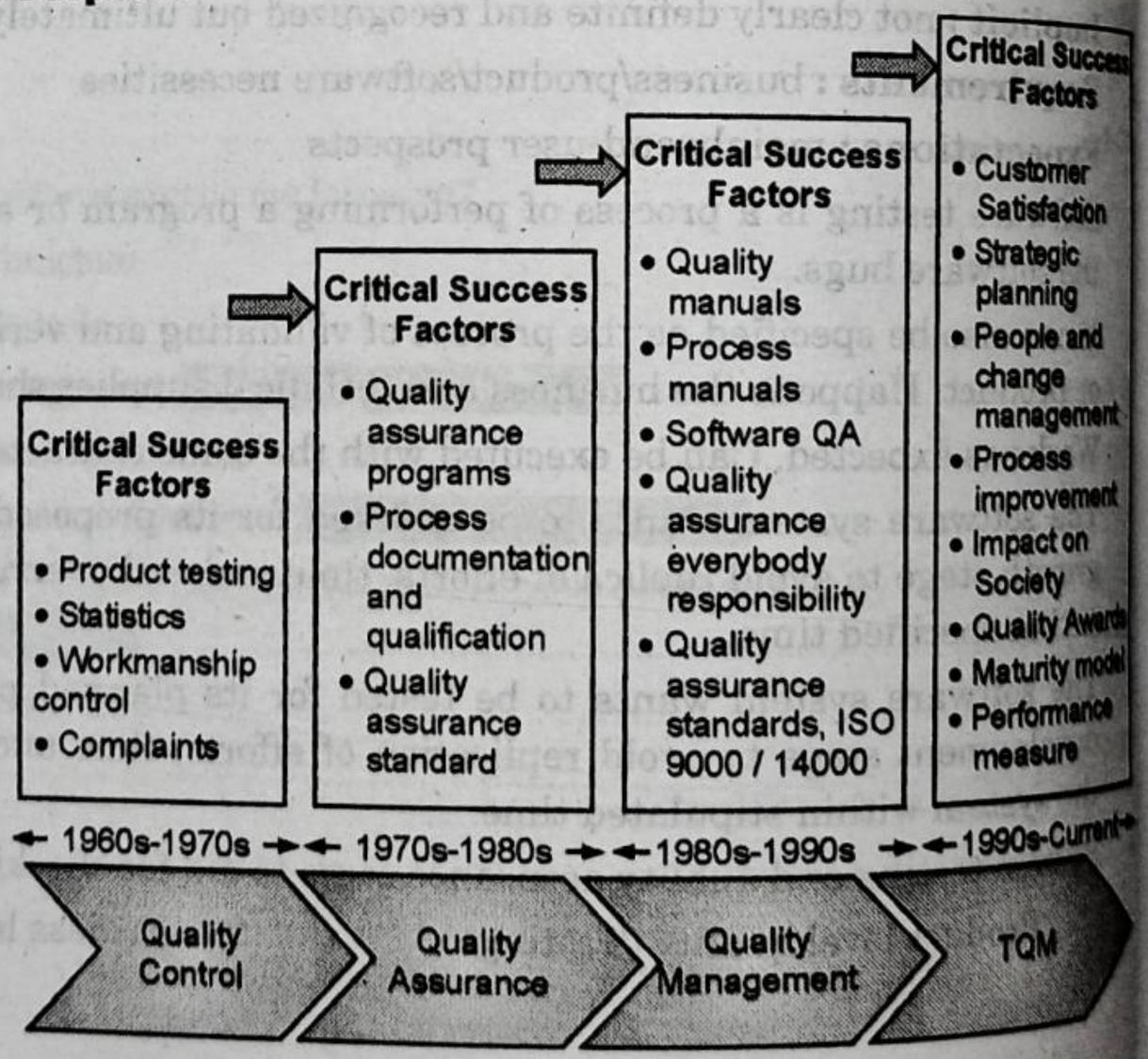
1.1.3 Why is Software Testing Necessary?

- 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

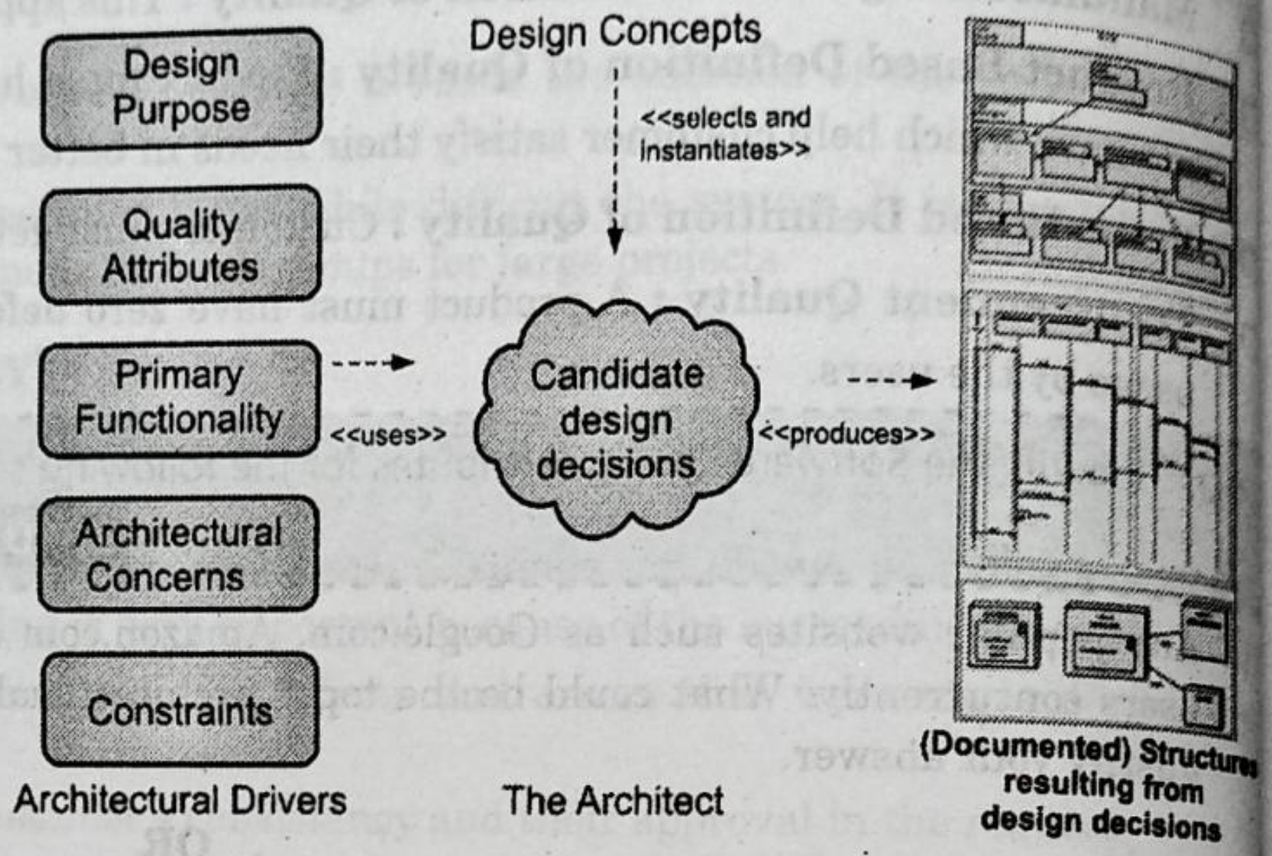
(iii) A software company is in a process of building social networking site which will have very large number of users in near future. Also company wish to add new features in this site and during addition of new features site should provide all the current features without any disturbance What top 2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

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

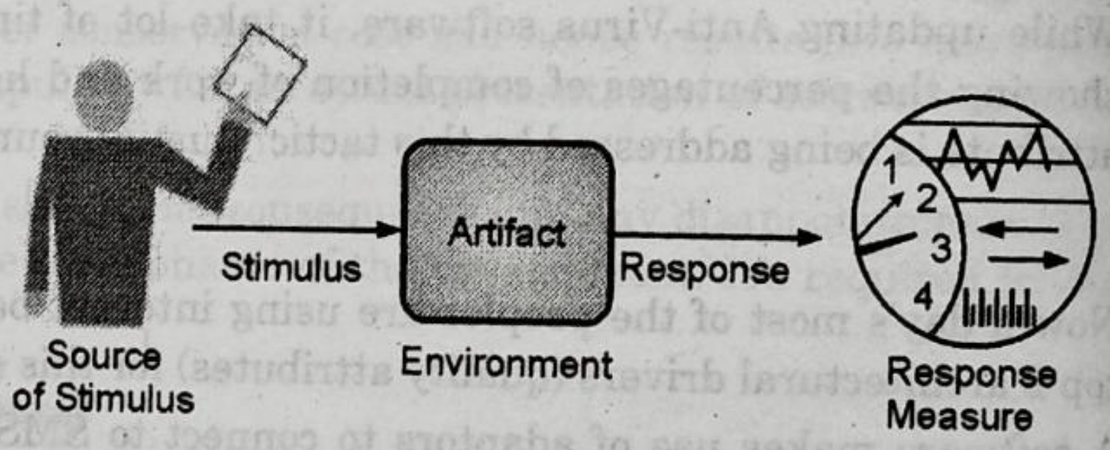
Already beginning design with ADD (or with any other enterprise method, for that matter), you essential to consider about what you are doing and why. While this statement may look extremely obvious, the devil is, as usual, in the details. We classify these what and why as architectural drivers questions.

- 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.
- 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.



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

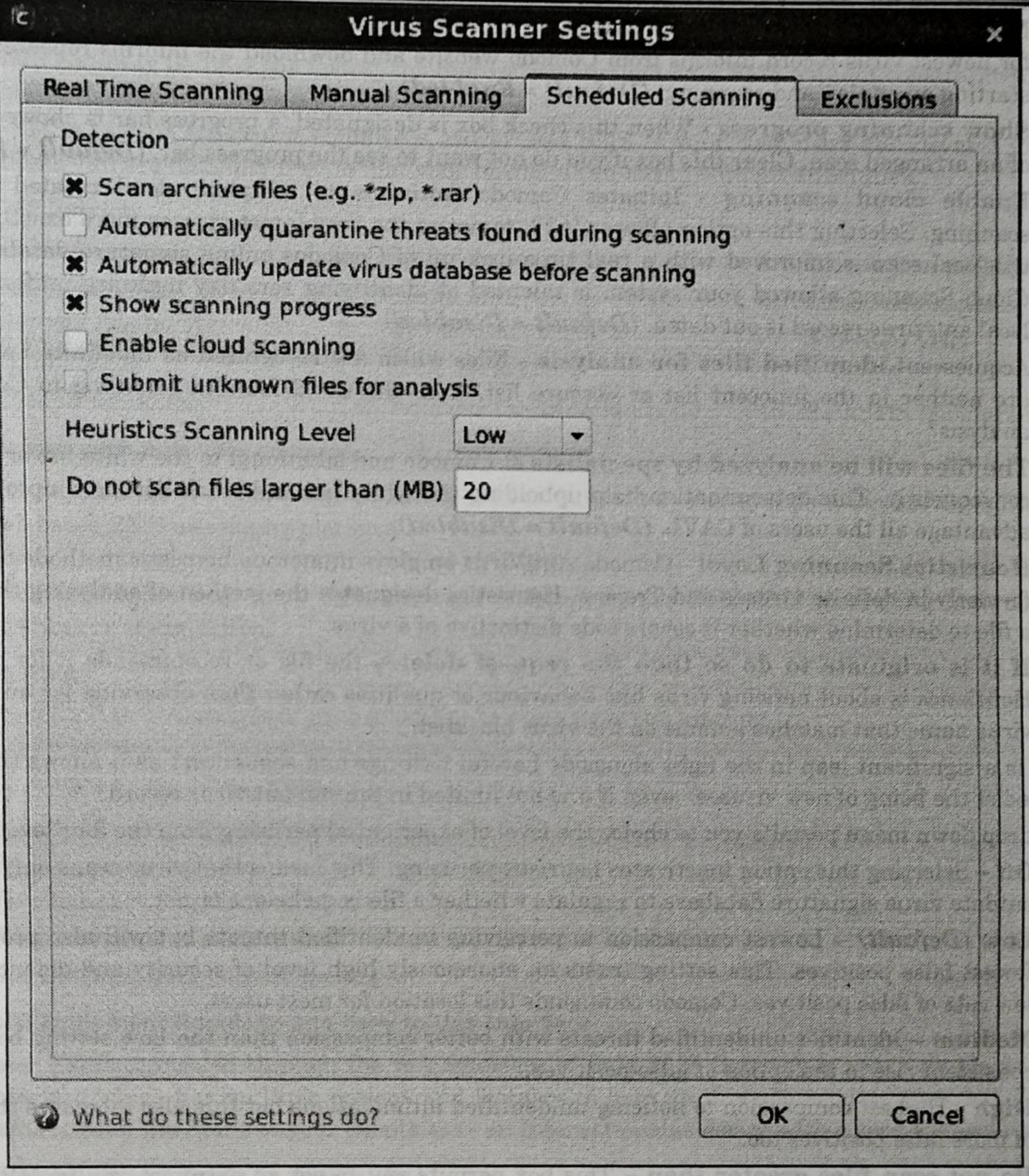


(1A3)**Fig. 1.1.3 : The six portions of a quality attribute situation**

- 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 ch for newest virus record informs from Comodo website and download the informs repeatedly bef starting an on-demand scanning. ***(Default = Enabled)***
- **Show scanning progress** - When this check box is designated, a progress bar is showed on st of an arranged scan. Clear this box if you do not want to see the progress bar. ***(Default = Enable***
- **Enable cloud scanning** - Initiates Comodo Antivirus to complete cloud created antivir scanning. Selecting this option allows CAVL to notice the very latest viruses more exactly becau the local scan is improved with a real time look up of Comodos online signature database. Wh Cloud Scanning allowed your system is talented of identifying zero day malware uniform if yo local antivirus record is out dated. ***(Default = Disabled)***
- Acquiescent **identified files for analysis** - Files which are recognized as indefinite i.e. the file are neither in the innocent list or obscure list, from the cloud founded scanning to Comodo f analysis?
- **The files will be analyzed by specialists** at Comodo and additional to the white list or black lis consequently. This determination help upholding the white list and black list more upto date an advantage all the users of CAVL. ***(Default = Disabled)***
- **Heuristics Scanning Level** - Comodo AntiVirus employs numerous heuristic methods to identif formerly in definite viruses and Trojans. Heuristics designates the method of analyzing the code 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 accurat 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 engin 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 th 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 th fewest false positives. This setting trusts an enormously high level of security and defence with 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 consistent rise in the option of false positives.
  - **High –** Highest compassion to noticing unidentified intimidations but this also increases the optio of more false positives too.
  - **Do not scan files greater than –** This box permits you to set an extreme size (in MB) for th separate files to be perused during on admittance scanning. Files greater than the size quantifie 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 mobil devices. Without applying technical changes and advertising methods you cannot imagine growing i this inexpensive world.
- Sending Bulk SMSs is, therefore, measured the most expediency and fast way to transport material t multiple users within a blink of eyes. Online Bulk SMS facilities, providing by Bulk SMS Gateway ar 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**

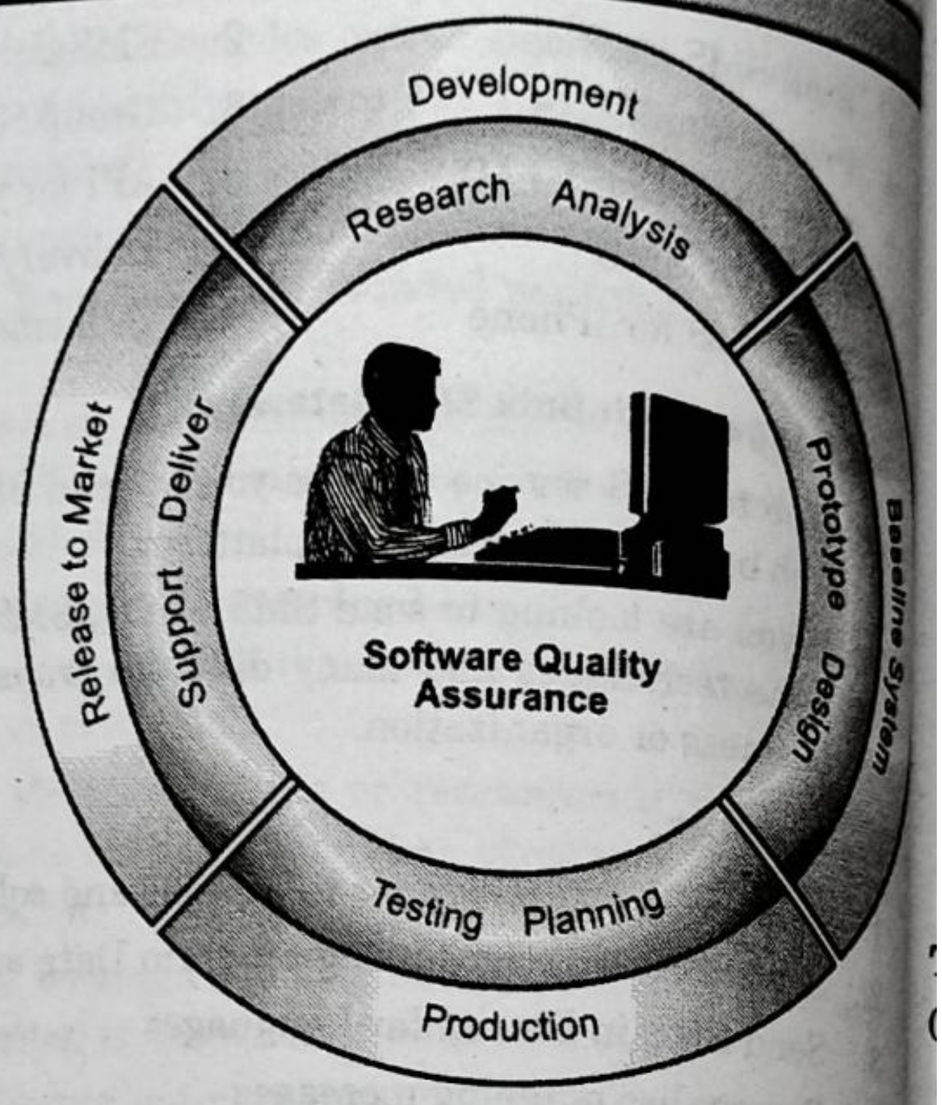
- Listing for your BulkSMS account is simple and easy and will only take a limited moments.
- To start transfer bulk SMS messages, Its Free Registration No Hidden controls, You can use bulk s with free credits.

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 wh 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 th organization.
- Suppliers may be customer for some organization and external customers accepts the output of th 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 syste administration as their customer.

☞ **External Customer**

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

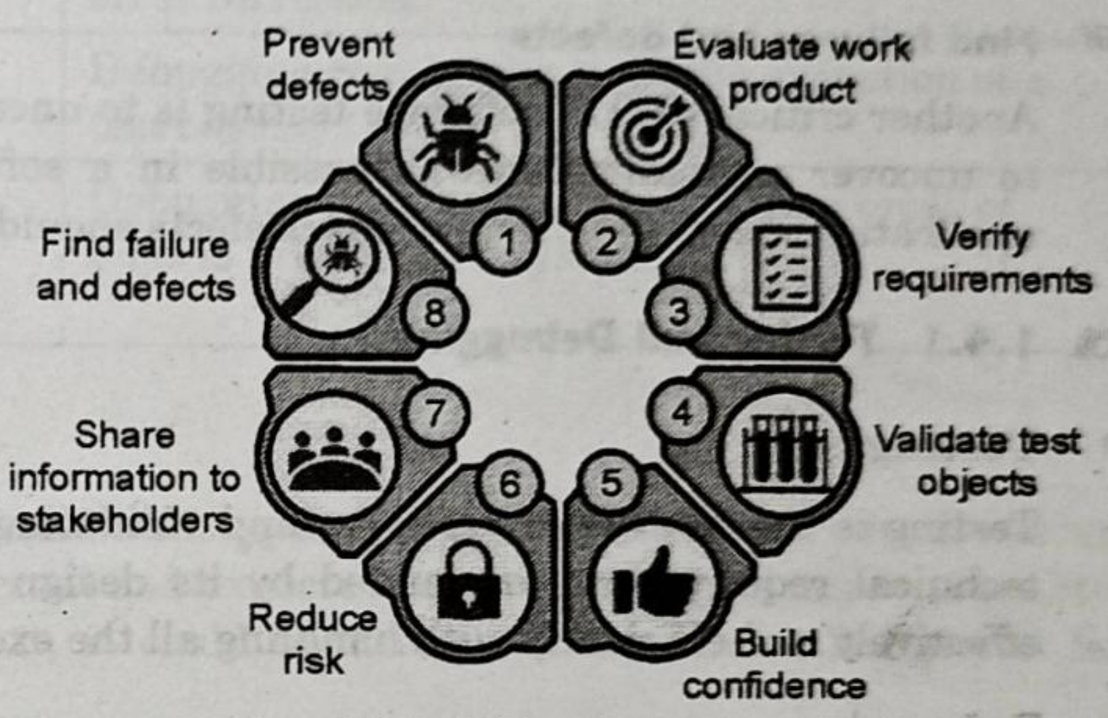


Fig. 1.4.1 : Objectives of testing

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 by the customers it is intended for.
- Those who specialize in software application testing are familiar with the needs of customers, 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 Resou department in a company that needs to track its employees and their performance would have entirely different software package from a hospital administrator trying to evaluate which hosp departments need more resources. That's why just like software developers, software testers 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 necess 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 negat user experience.
- Testing eliminates such errors in the performance while adding to the compatibility a adaptability of the software.
- Of course, one can design software that's exclusively limited for use on a desktop, but given that much of networking and work is now also conducted on smartphones, how useful would t software really be? Likewise, fewer customers will choose a software that only efficiently opera 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 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 glitch 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 long time to regain customer confidence, so it's much better and ultimately more convenient 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 experien glitchy software. These additional expenses can amount to a significant amount of damages, as only has the client been dissatisfied with the product for which they have paid, but a client's ti 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 test themselves "break" software.
- Testers do feed large amounts of data to software applications, but that's what is necessary software testing: if software cannot support large data-loads, what purpose does it achieve businesses or individual customers ?

5. To Accelerate Software Development

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

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 perfor by highly skilled and experienced testers who represent different end user profiles and use a wide ra of devices and operating systems.

- To save time whilst testing, manual testing can be supplemented with frequent automated te Automation is the most appropriate solution when performing white box testing, load tests and o performance testing.
- Any test that needs to be performed repeatedly should be automated. Automated tests are practical a 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 tes performed in-house with crowdtesting. With this approach, your product can be tested on a much lar 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 anot 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.

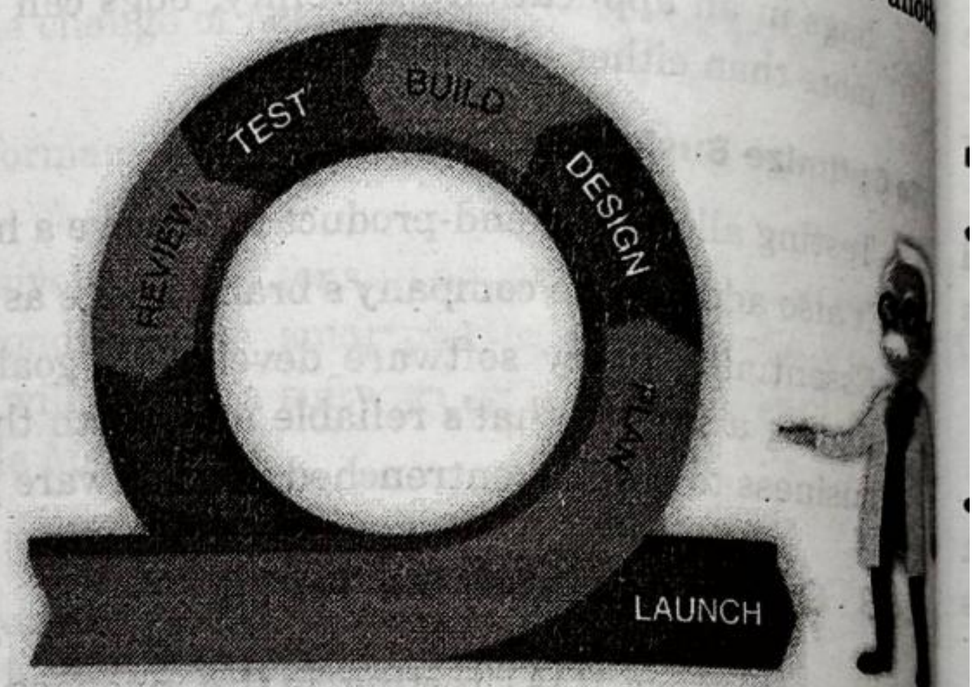


Fig. 1.4.3

- This is a collaborative approach that requires designers, developers, the QA team, and sometimes e users to communicate or work together. In order to facilitate collaboration, you can use a sin repository for the app code.
- Your teams will go through a short design or development cycle, followed by a targeted quality cont and testing phase for the new feature that was just added. Additional regression testing, secur testing, and stress testing can be undertaken as needed. The outcome of this phase will determine wh happens during the next design or development cycle.
- Leveraging automation will keep things moving once you adopt this approach. Test automation spe up the targeted testing phases and helps you to move onto the next development cycle in a matte hours or days. You will need to schedule manual tests after some key design or development cycle 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 sh testing phases. It's not enough to simply undergo functional testing - you need to incorporate feedb 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. Involv 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 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.

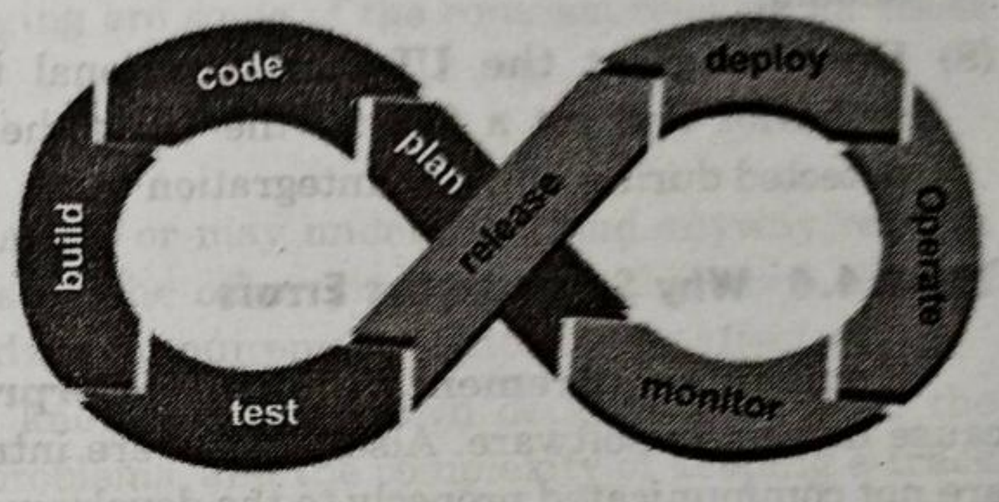


Fig. 1.4.4 : Continuous integration

- 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 lo testing to user acceptance testing (UAT) - so make sure you understand the differences and how to them.

(3) **Use regression tests :** testing a main feature once isn't enough. New additions to the code reposit 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. W 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 this data to determine where bugs are likely to occur. This data will help you to develop new tests th address problem areas.

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

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

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

1.4.4 Why Software has Errors

Unclear requirements and misinterpretation of requirements are the two major factors th cause defects in software. Also, defects are introduced in the development stage if the exact requiremen 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 mi is "Why does software have bugs?" and "How will these bugs occur?". This tutorial aims at answering the 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 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 humn 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 correcti 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 stakeholder development, and testing teams. Unclear requirements and misinterpretation of requirements a 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 re is ***Software Bugs***. In many organizations, management provides no incentive for programmer 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 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 project and the poorly documented code. Many times it takes a longer time to make even sli changes in poorly documented code, as there is a huge learning curve before making any c change.

(8) Software Development Tools

- Visual tools, class libraries, compilers, scripting tools, etc. often introduce their own bugs or a poorly documented, resulting in added bugs.
- Continuously changing software tools are used by software programmers. Keeping pace with t 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 automatio team's record/write any test script but forget to update it over a period, then that test could becom obsolete.
- If the automation test is not validating the results properly, then it won't be able to catch th 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 appl 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 Developmen 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 applicatio 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.

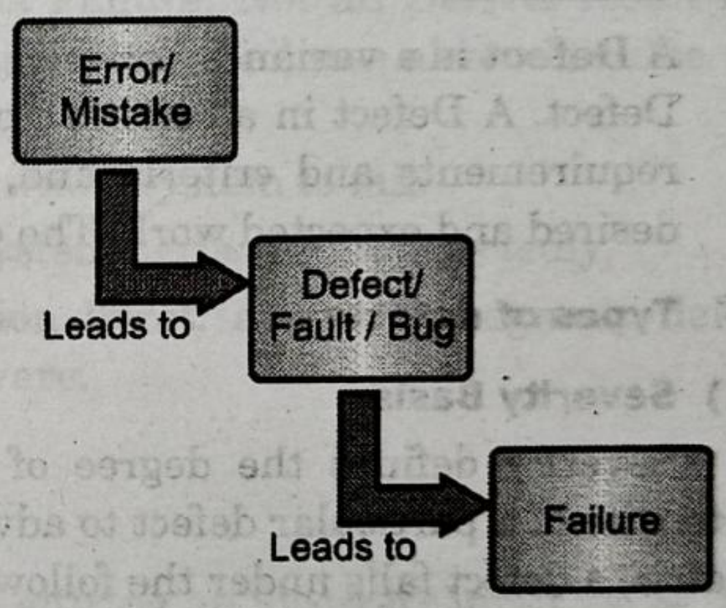


Fig. 1.4.5 : Error , Defect , and Failure

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 system. Such as missing or incorrect functionality of the system, no backup function or reverse funct available, etc.

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

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

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

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

(6) **Testing errors :** It implies the errors that occurred when implementing and executing the test proce 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 compatibility with the device.

➡ Defect

A **Defect** is a variance between expected and actual results. An Error that the tester finds is known Defect. A Defect in a software product reflects its inability or inefficiency to comply with the specifi requirements and criteria and, subsequently, prevent the software application from performing t 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 intensity of a particular defect to adversely impact a software product or its operation. Based on the severit metric, a defect falls under the following categories:

(1) **Critical :** Defects that are "**critical**" require immediate attention and treatment. A critical defe 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 Majo Defects. Although, these defects do not result in the complete failure of a system but may bring severa primary functions of the software to rest.

(3) **Minor :** These defects produce less impact and have no significant influence on a software product. Th results of these defects are visible in the operation of the product. However, it does not prevent user 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, w 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 chance that the user will find it. Depending on the likelihood or the possibility of a defect in a software product i 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**

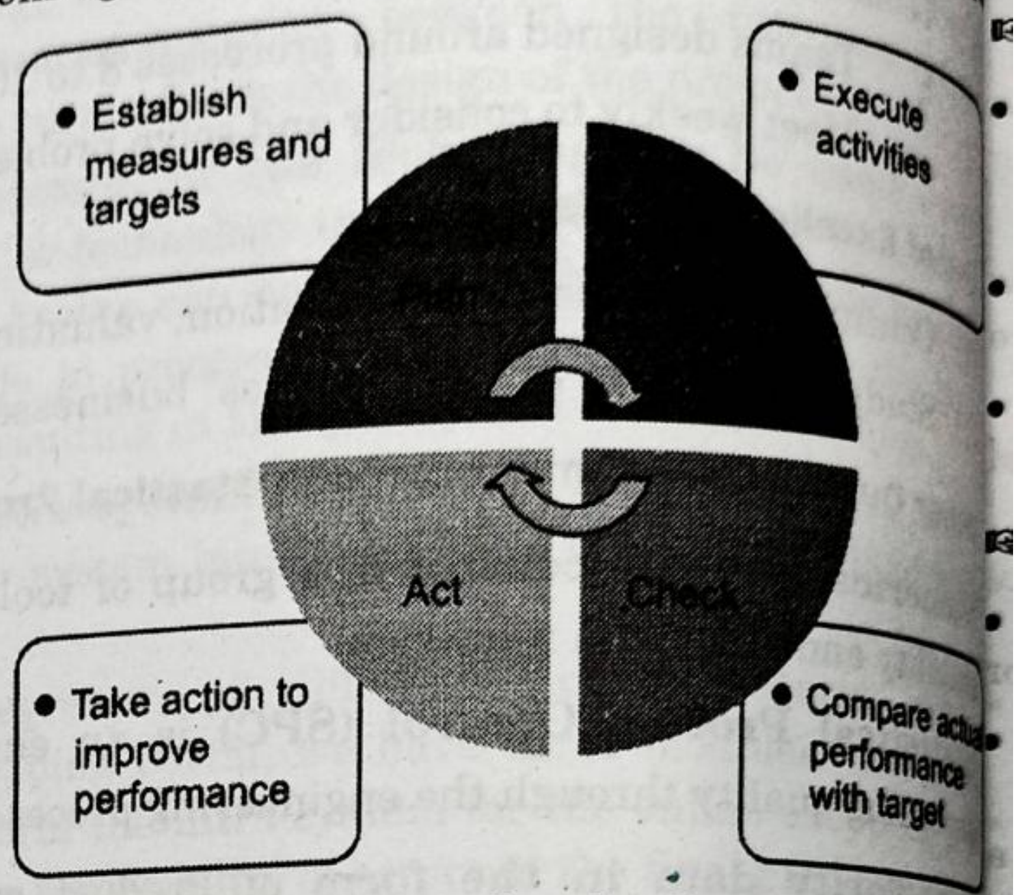
- Subset of items created to use for examination Quality Management through cultural change defines quality improvements as a cultural change driven by management. It involves
- Identifying areas in which quality can be improved depending upon process capability measurements and organizational priorities.
- 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 (PD framework. **SPPU - Nov./Dec.19 (End S**

- Continuous improvement is an ongoing effort to improve products, services or processes. These eff 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 p **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) cy**

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 t domains. Some examples of some domains showing customer expectations in terms of quality of vari 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

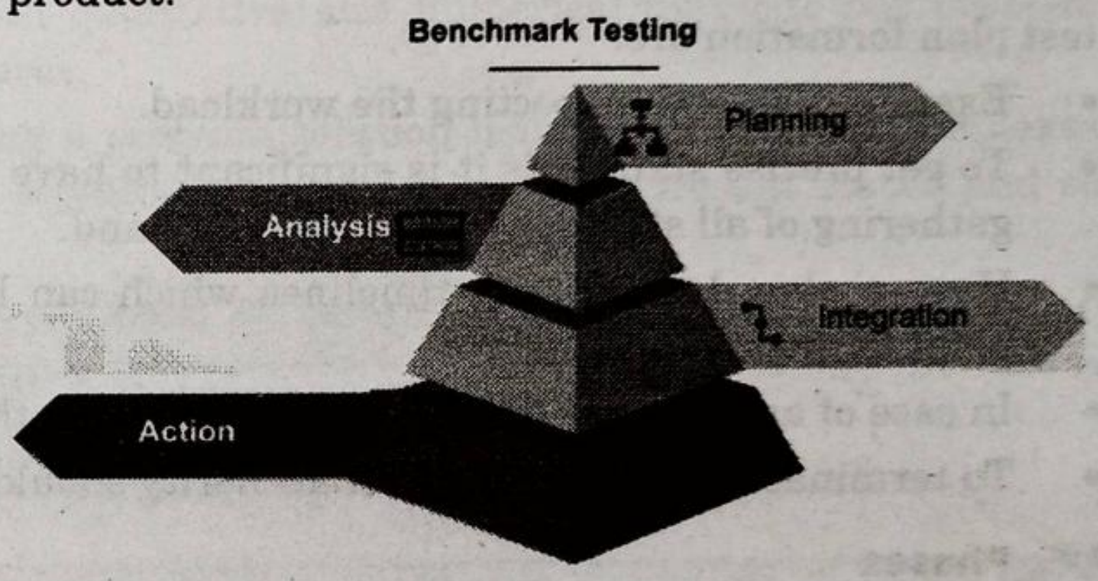
- It is important concept in Quality Function Deployment (QFD). It is concept of qualitative /quantitative metrics or measurable variables which can used to access product quality on several scales against benchmarks..
- Benchmarking include price of product paid by customer, time required to acquired 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.

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.



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

- 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 formation occupied care of.
- The test bags should be well distinct and separated as essentials as per their dissimilar functionali
- While the difficult is being approved upon it should be chequered for its constancy and control ev 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 circumstances.
- The software and hardware components must always be in line with the requirements or specifica of the production environment as the benchmark should be set for the production. The challe 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 advanc can be whichever an organization standard which helps in outcome the quantity competences under po quantified circumstances.

The second variety is the request benchmark which assistances in discovery the quantity capabiliti 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 step 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 th 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 term 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 help delivering standard software in the least.

▶ 2. Analysis Phase

- The analysis phase assistances in receiving a quality product and helps in classifying the root c of any subjects which were handled previous.
- By responsibility this you can naturally classify some alterations which are wanted and set a 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 pe 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.

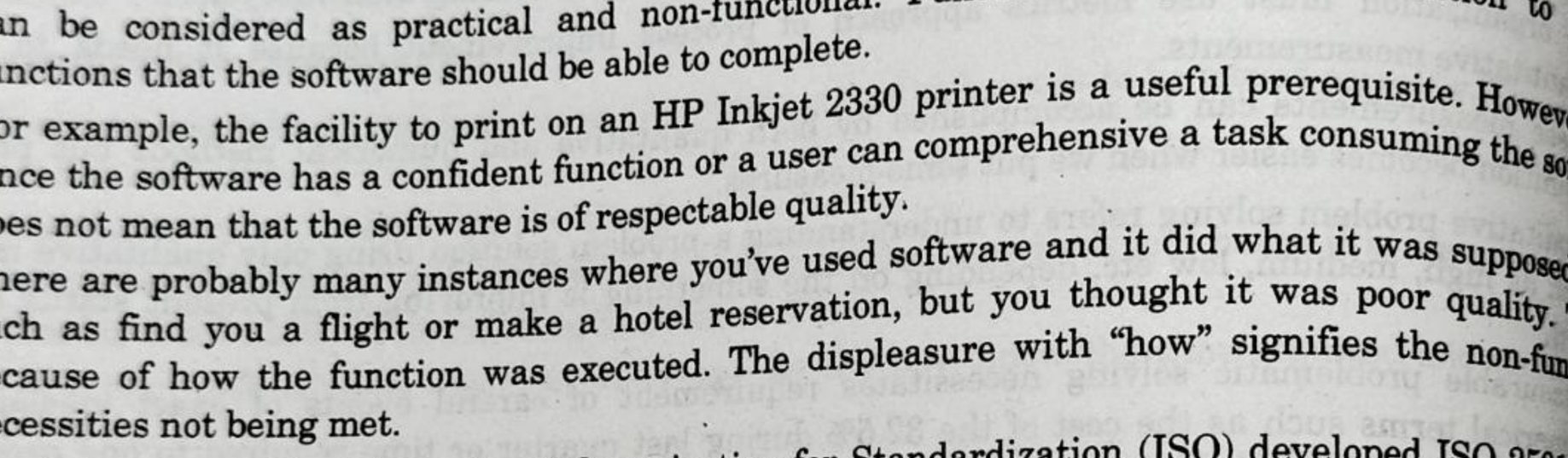
1.9 SOFTWARE QUALITY

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 requirements. But what are requirements ? Necessities, also called user levels in today's agile ter can be considered as practical and non-functional. Functional requirements mention to prec 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, j since the software has a confident function or a user can comprehensive a task consuming the softwa 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 such as find you a flight or make a hotel reservation, but you thought it was poor quality. This because of how the function was executed. The displeasure with "how" signifies the non-functio necessities not being met.
- For this purpose the International Organization for Standardization (ISO) developed ISO 25010 a model for specifying non-functional necessities. The classical shown below exemplifies the classificat 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 fi necessitates requiring and important. Only then can they be contented, and sufficient them can be ea more problematic than substantial functional necessities. So, what does this mean for you? Let's exam 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 produce it. Quality affects productivity.
- Productivity is tool of quantity that defines the competence of the association in relations of the ratio 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 so 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

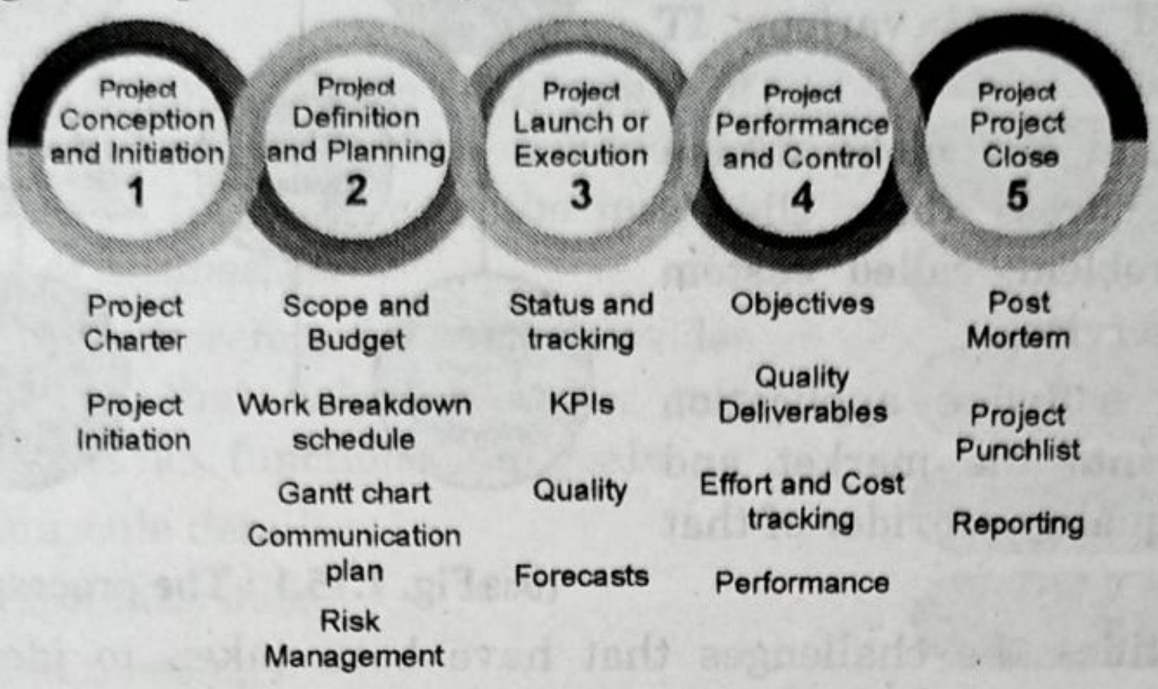
Quality culture is set of group standards that guide how developments are made to average work practices and resultant outputs. An organization's values can support entities at all levels make impro and more accountable choices involving issues of quality.

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 valu

(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.



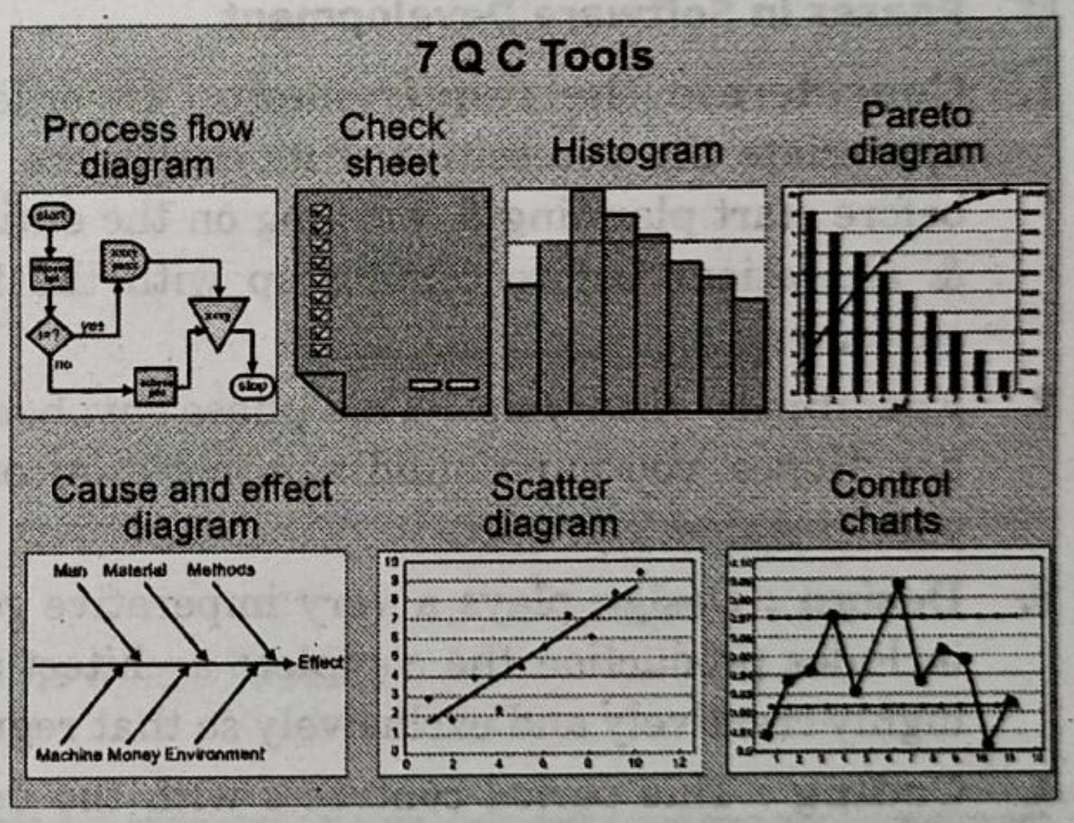
(1A11)**Fig. 1.14.1 : Steps in Project Management**

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.



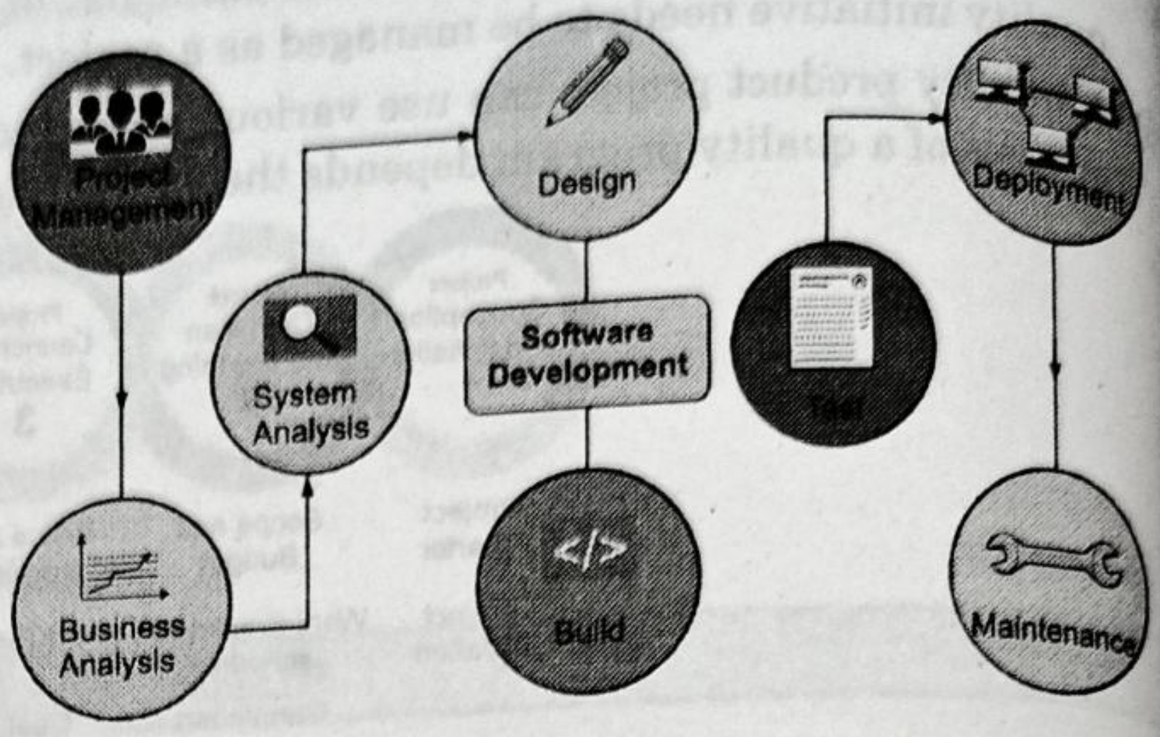
(1A12)**Fig. 1.14.2 : 7 QC Tools**

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

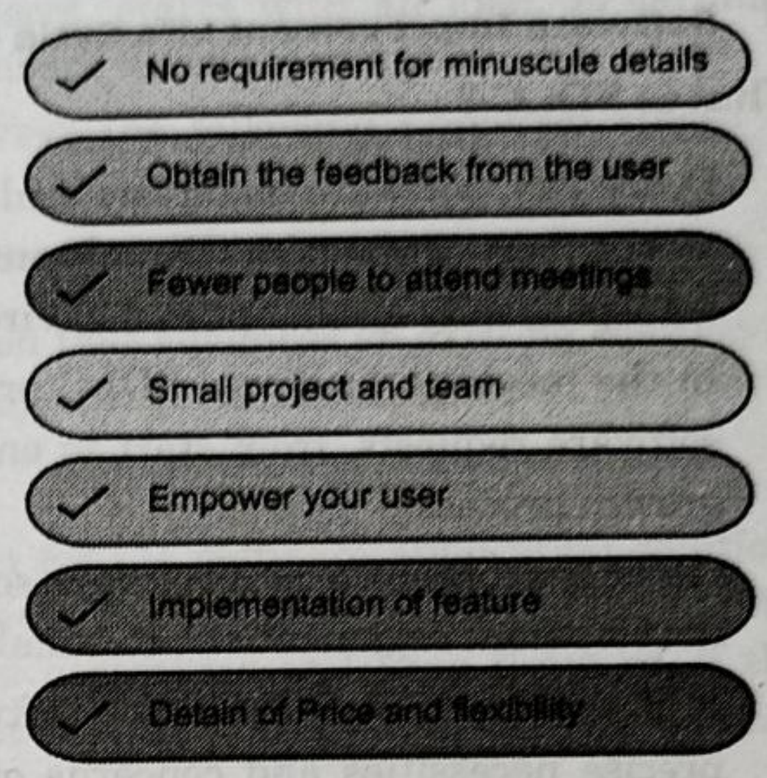
When it emanates to software development services, there are several approaches and models included in it. Here, we are conversing few major periods of software development methodologies:-

☞ 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 on 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.

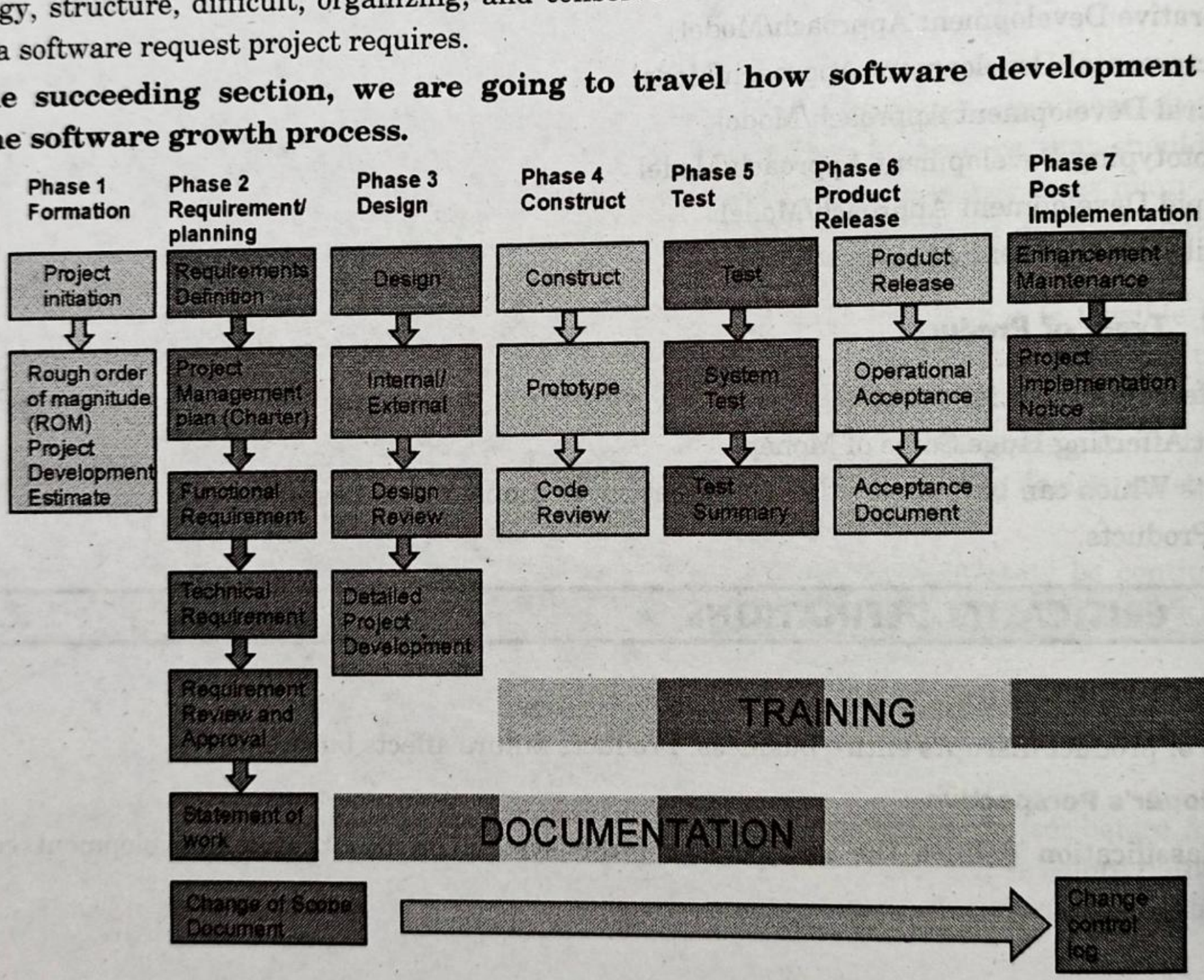
1.16.1 Problemstic Areas of SDLC

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 lifecycle 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

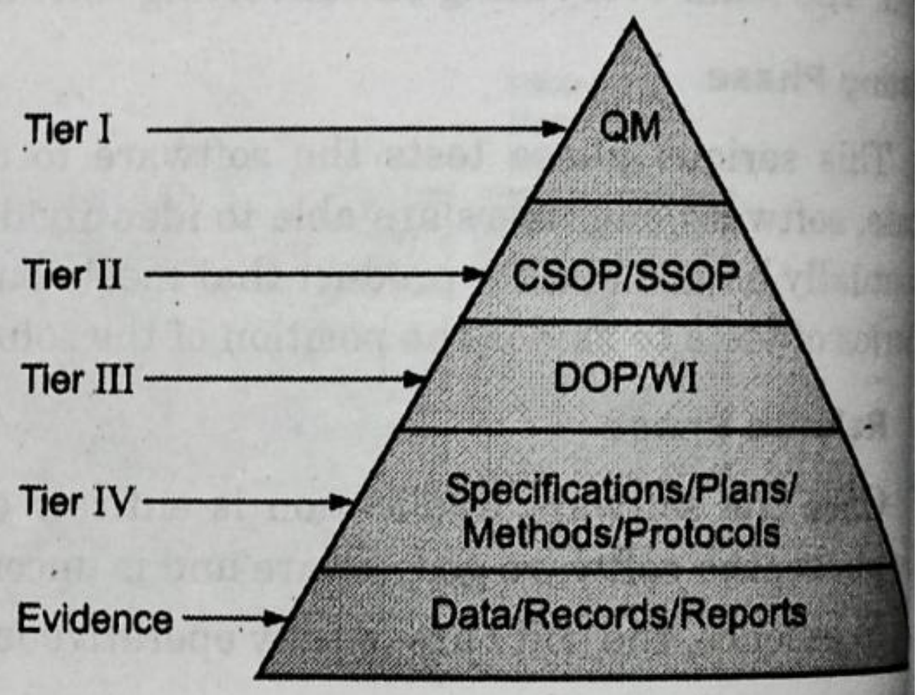
UQ. With respect Quality Management Systems explain the following?
(i) Quality Management Systems Structure
(ii) Pillars of Quality Management System
(iii) Important aspects of quality management
SPPU - Aug. 18(In Se

- Quality management involves management of all inputs and processing defined so that the output fr 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.

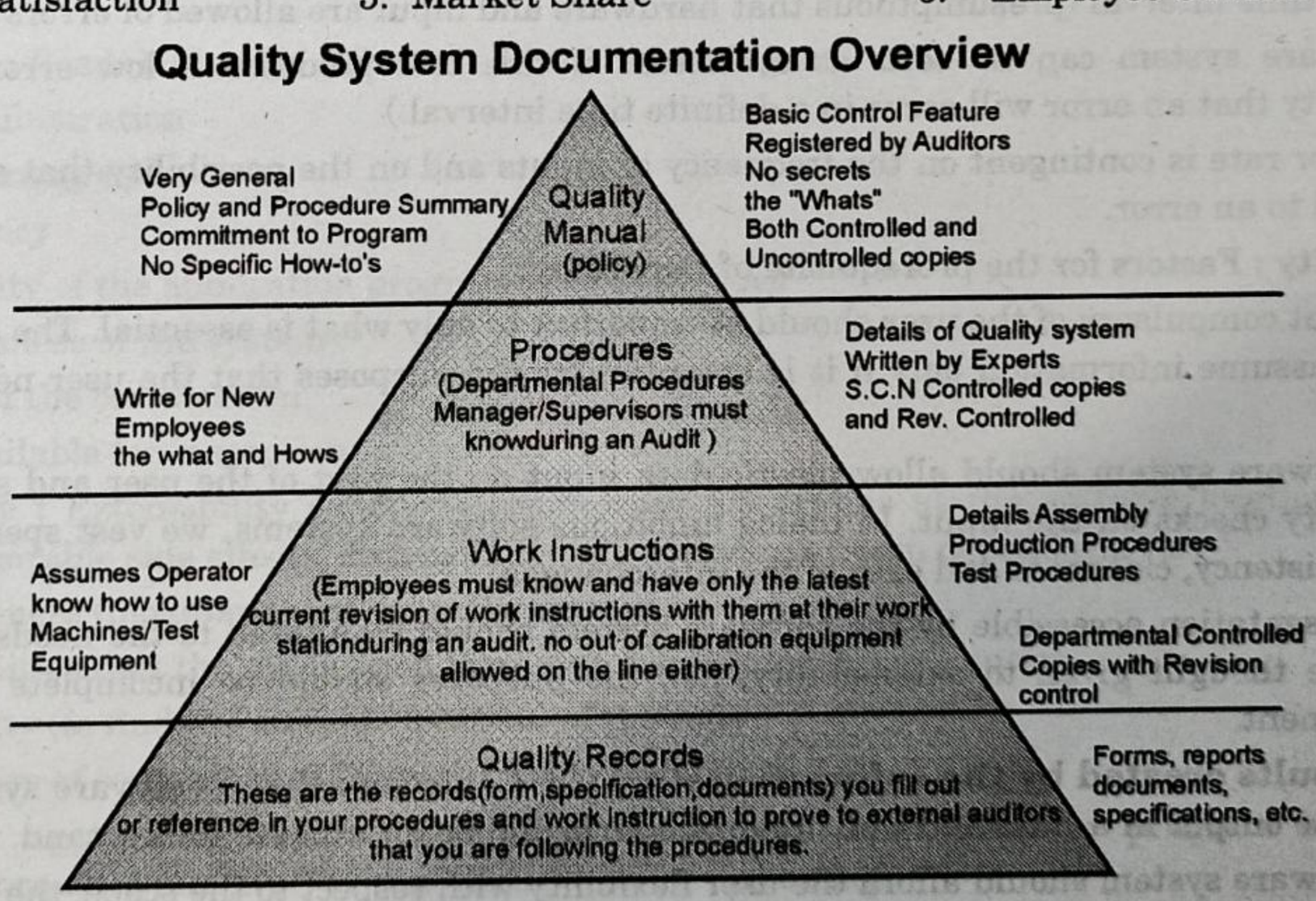
3. Define Your Quality Objectives

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 cont
(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, Robustne 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 compos 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 consisten of the software system.
   - **Reliability** of a software system is definite as the prospect that this system achieves a functi (resolute by the provisions) for a definite number of input trials under detailed input situations in 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., t 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 inp 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 syste should assume information only if it is essential for the purposes that the user needs to transm out.
   - The software system should allow plastic data input on the part of the user and should carry o credibility checks on the input. In dialog ambitious software systems, we vest specific standing the consistency, clearness and ease of the interchanges.
   - The presentation accessible by the software system should be altered to the needs of the operat with the thought given to extensibility; i.e., the purposes should be incomplete to these in t requirement.
   - **The results created by the software system :** The outcomes that a software system distribut 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 deta and the form of presentation of the results. Error messages must be provided in a form that 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 possibl 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 tha 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 progra mechanisms.
    - A software system can be said to be transferable if the effort required for porting it show meaningfully less than the effort essential for a new implementation.

...Chapter End