Q. 9 Which step in the risk management process defines the probability and impact of each risk to determine the severity ?

(a) Risk Identification (b) Risk Analysis

(c) Risk Planning (d) Risk Tracking and control

Q. 10 Which of the following is true about risk identification?

(a) Risk events should be identified early and projects reviewed regularly to ensure that no significant risk events are missed.

(b) The risk identification process should be repeated on a regular basis as part of your key checkpoints.

(c) Be extremely detailed about possible risks, even if they do not at first seem realistic risks; this allows for coverage of all possibilities.

(d) Start a new risk register at the beginning of each project phase.

Q. 11 Which of the following is done to create a risk management plan?

(a) Accept that risk might not change any of the project elements.

(b) Identify risks, analyze risks, and define what, if anything, should be done about the risks.

(c) Assign the actions to project team members.

(d) Both B and C.

Q. 12 Which of the following is true about the importance of risk management ?

(a) It ensures that the project will be on budget.

(b) It helps reduce the impact of risk in a project.

(c) It eliminates the impact of all risk.

(d) It ensures that the project will be on schedule.

Review Questions

Q. 1 Write short notes on :

(a) Risk Management (b) Risk Planning

(c) Risk Monitoring (d) Risk Projection (e) Risk Refinement

Q. 2 Explain the seven principles of Risk Management.

Q. 3 Explain Risk Identification and the methods involved in it

Q. 4 Explain the different categories of Risks.

Q. 5 Explain the different types of risks?

Q. 6 What is Risk Management? Explain different stages involved in Risk Management.

Q. 7 Explain RMMM plan in detail.

□□□

UNIT III

CHAPTER 10

Software Quality Assurance

Syllabus

Elements of SQA, SQA Tasks, Goals, Metrics, Formal Approaches to SQA, Six Sigma, Software Reliability, The ISO 9000 Quality Standards, Capability Maturity Model.

10.1 Quality

Definition

"Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected from all professionally developed software."

10.1.1 Quality Perceptions

The software is developed by an engineer, it is required by the customer, and is used by various end users. Thus, each of them has a different perception about quality that decides whether the software is good or not.

- Engineer may judge :
  - User satisfaction,
  - Portability,
  - Maintainability,
  - Robustness
  - Efficiency
- Customer may judge : cost
- User may judge :
  - Reliability (i.e. No Defects that may either stop computing or may produce unexpected results),
  - Usability.

10.1.2 Importance of Software Quality

Q. Why is software quality important ?

- Aim of Quality Measure is : **Satisfying Customer**

- Why Quality is important can be inferred from the following reasons :
  - **Quality is a competitive issue :** Our product's quality should be better than our competitor's product.
  - **Quality is must for Survival :** Just taking advantage of your so called Fame and good marketing is not enough to survive in the market; Quality of your product decides your survival in the market.
  - **Quality gives entry into international Market :** Only quality products are given ISO marks and permission to enter in the global market.
  - **Quality is cost effective :** If the quality is maintained from the initial phases of SDLC, then the cost expenditure in correcting the risks or errors is very less as compared to the maintenance cost of the defects detected after the delivery of the product.
  - Quality helps retain customers & increase profits.
  - Quality is the hall-mark of world class business.

10.1.3 Quality Challenges

1. Quality of software should be given importance from the very initial Stage of Software Development, not just at the end of coding.

   Example : If a pharmacist is making a new medicine and if its sample is not tested in the chemical laboratory from the beginning of its development and if it is delivered to the stores *without verifying and validating it's quality*, then it may lead to the death of lots of people + the financial and reputational loss to the company.
2. Monitor not only individual segments (complete or complex or large system is divided into simple or individual systems) but also interaction between the segments.

   Example : Consider a subject (complete system) with different chapters (individual segments) – Each chapter individually has a meaning. But check out whether each chapter is related (interaction between segments) to the other or not. Only if individual chapters have a relative meaning with each other, only then they collectively make a subject. In an English Literature subject, you cannot insert a maths chapter.
3. The Quality criterion differs from phase to phase of Software Development.

   Example : Precise Req., Detailed SRS, Cost and Time Estimation, Risk Analysis are the quality criteria's at Requirement Analysis phase; Coupling & Cohesion in Design phase; Code Efficiency & Reusability in Coding phase.
4. Quality measures used for small systems may not be appropriate for larger ones.

   Example : Indica car is a very good and quality car but can be used to accommodate 5 people not to accommodate 10 people.

10.1.4 Causes of Errors In a Software Product

Q. What are the various causes behind the errors in a software product ?

- Incomplete or erroneous specification
- Misinterpretation of customer communication
- Intentional deviation from specification
- Violation of programming standards
- Error in data representation

- Inconsistent module interface
- Error in design logic
- Incomplete or erroneous testing
- Inaccurate or incomplete documentation
- Error in programming language translation of design
- Ambiguous or inconsistent human-computer interface
- Miscellaneous.

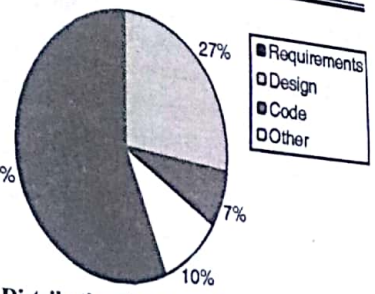

Fig. 10.1.1 : Distribution of Defects/Errors in the SDLC

The pie chart (Fig. 10.1.1) defines the defects that are caused at different phases of Software Development Life Cycle (SDLC) :

10.2 Quality Control and Quality Assurance

Q. State the difference between quality control and quality assurance.

- Quality Assurance is the process of defining how software quality can be achieved and how the development organization knows that the software has the required level of quality.
- *Quality control* and *Quality assurance* are the essential activities for any organization in the process of developing a product.

Table 10.2.1 : Software Quality Control v/s Software Quality Assurance

Sr. No.	Software Quality Control	Software Quality Assurance
1.	It is a *corrective approach* – corrects the faults when they occur.	It is an *preventive approach* – prevents the faults from occurring by providing rules and methods.
2.	*Finds defects* and then corrects it.	*Prevents defects* from occurring.
3.	It is a task conducted on the *product.*	It is a task conducted in the *process.*
4.	Gives confidence to *producer.*	Gives confidence to *customer.*
5.	Process of comparing the product quality with applicable standards and taking appropriate action when non-conformance is detected.	It is a set of planned and systematic set of activities that provides adequate confidence about establishing the requirements properly and assures that the products or services conform to specified requirements.
6.	Only tester is responsible for QC.	Entire team is responsible for QA.
7.	Detects and reports and corrects the defects.	Prevents the introduction of issues or defects.
8.	Examples: Walkthrough, inspections, testing conducted on the documents or on software product.	Examples : Defining process, automated engineering and testing tools i.e. CASE (engineering) and CAST (testing) tools, training, quality audits are used in development process.

10.2.1 Changing View of Quality

- The first quality assurance and control function was introduced in Bell labs in 1916 and spread rapidly throughout the manufacturing company across the world.
- Before twentieth century, quality control was not given that importance. It was considered as the sole responsibility of developers. The Table 10.2.2 describes the changing view of quality:

Table 10.2.2 : Changing view of Quality

Sr. No.	PAST	PRESENT
1.	Quality was considered as the responsibility of developers	Quality is considered as the responsibility of everyone's responsibility
2.	Defects are intentionally hidden from customers and management so hat developers are not burdened with extra work of correcting these errors.	Defects are highlighted and brought to surface so as to conduct corrective measures and achieve user satisfaction
3.	Quality problems led to blame each other and giving faulty justifications and excuses. This only led to quarrels among the team members and an unhealthy team work.	Quality problems lead to co-operative solutions so as to prevent the failures.
4.	Not much work was welcomed. Say, for example, correctness measures accompanied with minimum documentation.	Documentation is considered as essential to note down the lessons learnt so that mistakes are not repeated.
5.	Developers misunderstood that increased quality measures will also increase the project costs.	Developers know that improved quality increases the business profit.
6.	Quality was internally focused and understood that software has quality if it is just easy to maintain, reusable and flexible.	Quality is customer focused and it is described in terms of integrity, reliability, usability and accuracy.
7.	Quality task is conducted at project execution.	Quality task is conducted as project initiates.

10.2.2 PDCA Cycle

Q. Explain PDCA cycle in brief.

The most popular tool used to determine QA is the Shewhart Cycle, developed by Dr. W. Edwards Deming. This cycle consists of four steps: *Plan*, *Do*, *Check*, and *Act* i.e. PDCA.

1. **Plan :** Establish objectives and processes required to develop the proposed software product.
2. **Do :** Implement the planned process.
3. **Check :** Examine and evaluate the implemented process by testing the results against the predeter-mined objectives.
4. **Act :** Apply necessary actions for improvement if the result requires any changes.

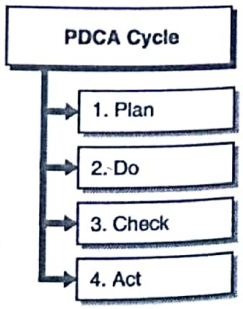

Fig. C10.1 : PDCA Cycle

10.3 SQA : Software Quality Assurance

- SQA serves as a customer's in-house representative that assists the software engineering team in achieving high-quality.
- SQA is considered as an *umbrella activity* that is applied throughout the SDLC process.
- SQA is a conformance to *explicitly stated software requirements* (functional requirements), *explicitly documented development standards, and implicit characteristics* (such as ease of use, maintainability, reliability, etc.) expected from the software.

10.3.1 Need of SQA

- SQA is a process that defines how software quality can be achieved and how the development organization knows that the software has the required level of quality.
- SQA group serves as the customer's in-house representative i.e. these people look at the software from customer's point of view.
- SQA monitors the software engineering processes and methods to ensure quality.
- SQA is a process of verifying or confirming that whether the products and services meet the customer expectations or not.
- SQA is a *process-driven approach* with specific steps to attain development goals. This process of QA considers design, development, production, and service.
- SQA group checks whether;
  - The software adequately meets the quality factors.
  - Software development has been conducted according to pre-established standards.
  - The technical disciplines (software engineers and testers) have properly performed their roles as part of the SQA activity.

Syllabus Topic : SQA Tasks

10.3.2 SQA Tasks

SQA is looked after by two different groups :

1. **Software engineers** who develop the software and have lot of technical knowledge.
   Software engineers address the quality by :
   - Applying effective technical methods and measures
   - Conducting formal technical reviews
   - Performing well-planned software testing
2. **Software quality assurance persons** who address the quality by :
   - Planning the quality assurance
   - Record keeping
   - Error reporting so as to improve the development process and prevent bugs from ever occurring.
   - Monitoring the development activities
   - Analysis and

The Software Engineering Institute (SEI) encompasses a set of SQA activities to evaluate the development process of the software product.

SQA activities

1. Prepare SQA plan for the project.

- The SQA plan is developed during project planning and after the requirement gathering.
- This SQA plan is then reviewed by all the stakeholders. Stakeholders are the persons who directly or indirectly interact with system such as customers, end-users, developers, testers, analysts, reviewers and etc.
- The SQA plan identifies :
  - Evaluations to be performed
  - Audits and reviews to be performed
  - Quality standards that are applicable to the project
  - Procedures for error reporting and tracking
- Documents to be produced by the SQA group
- Amount of feedback provided to the software project team

2. Participate in the development of the project's software process description.
   - The Software engineers select some process to develop the software and to help them, the SQA team reviews the process description to check:
     - The compliance of the development process with organizational policy
     - Internal software standards
     - Externally imposed standards
     - Other parts of the software project plan
3. Reviews the software engineering activities to verify it's compliance with the defined software process.
   The SQA team identifies, then documents and tracks the deviations from the process and verifies that the required corrections have been made.
4. Audits the designated software work products to verify their compliance with those defined as part of the software process.

SQA group audits

- The software work product
- Identifies, documents and tracks the deviations
- Verifies that the needed corrections have been made and
- Periodically reports the reviewed results of the software work product to the project manager.

5. Ensures that deviations in software work and work products are documented and handled according to a document procedure.
   These deviations may be encountered in the project plan, process description, applicable standard or technical work products.
6. Records any non-compliance and reports to senior management.
   The SQA team keeps track of the non compliance things until they are resolved.

Syllabus Topic : Goals

10.3.3 SQA Goals

Q. State the goals of SQA.

Goals of SQA activities in terms of Software Development

1. Assures that the software development will conform to the user requirements.
2. Assures that the software development will conform to planned schedules and budget.

3. Initiating and managing the activities to achieve greater efficiency of software development and SQA activities.

Goals of SQA activities in terms of Software Maintenance

1. Assures that the software maintenance activities will comply with the functional requirements.
2. Assures that the software maintenance activities will conform to planned schedules and budget.
3. Initiating and managing the activities to achieve greater efficiency of software maintenance and SQA activities

Syllabus Topic : Elements of SQA

10.3.4 Elements of SQA

- *Standards* and *Procedures* are the building blocks of SQA task; since; these provide the framework from which the software evolves.
- *Standards* are the pre-established criteria to which the *software products* are compared and *Procedures* are the established criteria to which the *software development* and *control processes* are compared.
- Therefore, standards and procedures both of them establish the prescribed methods for developing the software.
- Thus, proper documentation of standards and procedures is necessary to guide the SQA group in monitoring the process and evaluating the product.

Types of standards

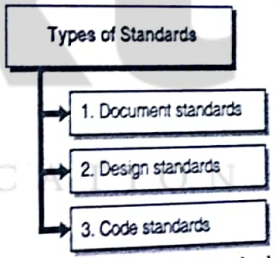

Fig. C10.2 : Types of Standards

1. **Document standards :** specifies the form and content for planning, controlling. The documentation provides consistency throughout the project. The NASA Data Item Descriptions (DIDs) provide the documentation standards.
2. **Design standards :** specifies the form and content of the design product. They provide rules and methods for translating the software requirements into the software design and then representing it in the design documentation.
3. **Code standards :** specifies the programming language in which the code is to be written and also define the restrictions on use of language features. It defines the legal language structure, style conventions, rules for data structure and interfaces and internal code documentation.
   - Procedures are explicitly stated criteria or steps that must be followed in carrying out a process.
   - All processes must have documented procedures.
   - Few processes in which procedures are required are configuration management, reporting the nonconformities and taking corresponding corrective actions, testing and also to conduct formal inspections.
   - Thus, in short, the SQA activities assure that both product and process comply with the established standards and procedures described in the QA portion of the management plan.

As we have already stated the ***standards*** are used for ***product evaluation*** and ***procedures*** are used for ***process monitoring***, we will see these both things in little detail.

Product evaluation

- It is an SQA activity that assures that quality standards regarding the product are being followed.
- The first products monitored by SQA group should be the project's standards and procedures.
- SQA assures that clear and achievable standards exist and thus these standards are achieved by the software product.
- Product evaluation assures that the software product reflects the requirements of the applicable standards as identified in the management plan.

Process Monitoring

- It is an SQA activity that assures that appropriate steps are undertaken to carry out the software development process.
- SQA monitors the process by comparing the actual steps carried out with those in the documented procedures.
- The quality assurance portion of the management plan specifies the methods to be used by the SQA process monitoring activity.

SQA Audit Activity

- This is also a building block of SQA because it is a fundamental SQA technique that looks at the process and product in depth and compares it with the established standards and procedures.
- Audits conduct the review of management, technical and assurance process to provide an appropriate quality and status if the software product.
- Audits assure that proper control procedures are being followed that include maintenance of documentation and accurate status reports of the process conducted.

Syllabus Topic : Metrics

10.4 Metrics for Software Quality

The following are the software quality factors and the needed metrics and measurements for each :

Fig. C10.3 : Software Quality Factors

1. Correctness

- A program must operate correctly else it provides little value to its users. Correctness is measured as the degree to which the software performs its required functionality.
- The most common measure for correctness is number of defects per KLOC where a defect is lack of conformance to requirements.

2. Maintainability

- Maintainability is the ease with which :
- An application can be corrected if an error is encountered

- A defect can be fixed
- A application is adapted in a changed environment
- A application is enhanced
- A simple time oriented metric is Mean Time to Change (MTTC) i.e. the time it takes to analyze the changed request, design an appropriate modification, implement the modification, test it and deliver it to the client. Fig. 10.4.1 which gives the scenario of online detection and offline repair. The measures MTBF (Mean Time Between Failure), MTTD (Mean Time To Detect) and MTTR (Mean Time To Repair) are the average times to failure, to detection and to repair.

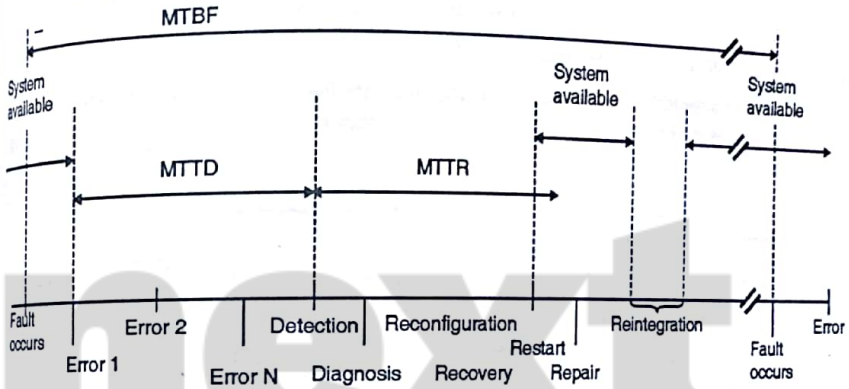

Fig. 10.4.1 : MTBF, MTTD and MTTR measures of maintainability

3. Integrity

- This attribute measures a system's ability to withstand attacks to its security. Attacks can be made on all three components of software: programs, data and documents.
- We cam measure integrity through following two additional attribute :
  1. Threat: It is the probability that an attack of a specific type will occur within a given time.
  2. Security: It is the probability that the attack of a specific type will be repelled. The integrity of a system van be then defined as :

  $$\text{Integrity} = \sum (1-(\text{threat} * (1-\text{security})))$$

4. Usability

- If a program is not easy to use, it is often meant to failure. Even if the functions that it performs are valuable.
- Usability is an attempt to quantify ease-of-use and can be measured in terms of characteristics.

Defect Removal Efficiency

- It is a quality metric that provides benefit at both the project and process level.
- DRE is a measure of the filtering ability of quality assurance and control activities as they are applied throughout all process framework activities.
- DRE is defined as: DRE = E/(E+D) where E is the number of errors found before delivery of the software to the end user and D is the number of defects found after delivery. The ideal value for

DRE is 1 i.e. no defects are found in the software. Generally, D is greater than 0 but DRE can be 1 when, as E increases the overall value of DRE begins to approach 1.

Other nine Classes of Measures

A. Product Measures

1. Errors, faults, failures
2. Mean-time-to-failure
3. Realibility growth and projection
4. Remaining products faults
5. Completeness and consistency
6. Complexity

B. Process Measures

1. Management control
2. Coverage
3. Risk, benefit, cost evaluation

C. Defect Amplification and Removal

- A defect amplification model can be used to illustrate the generation and detection of errors during the preliminary design, detail design and coding steps of the software engineering process.

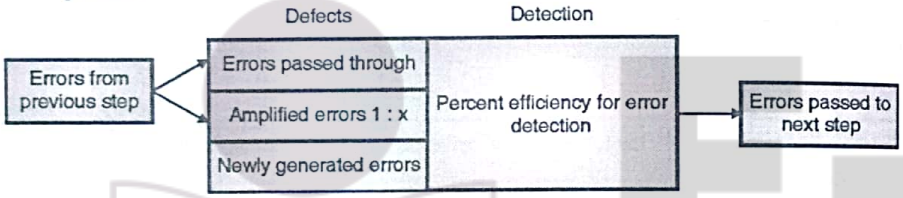

Fig. 10.4.2 : Defect Amplification model

- In case if the technical reviews fail to discover the newly generated defects, these may get amplified. The Fig. 10.4.2 represents the percent of efficiency in detecting errors.

Establishing a Software Metric Program

Following are the steps to conduct a goal-driven software metric program :

Step 1 : Identify your business goals
Step 2 : Identify what you want to know or learn
Step 3 : Identify your sub goals
Step 4 : Identify the entities and attributes related to your sub goals.
Step 5 : Formalize your measurement goals
Step 6 : Identify the quantifiable questions and the related indicators that you will use to help you achieve your measurement goals.
Step 7 : Identify the data elements that you will collect to construct the indicators that help answer your questions.
Step 8 : Define the measures to be used, and make these definitions operational.
Step 9 : Identify the actions that you will take to implement the measures.
Step 10 : Prepare a plan for implementing the measures.

10.5 Quality Factors

McCall's Quality Factors : Jim McCall produced the McCall's Quality Model in 1977 for the US Air Force to bridge the gap between users and developers. He tried to map the user view with the developer's priority.

McCall has categorized the software quality factors in to 3 categories:

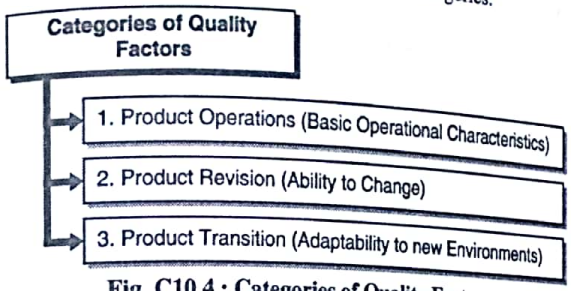

Fig. C10.4 : Categories of Quality Factors

1. **Product Operations (Basic Operational Characteristics) :** The quality criteria specified in this category talk about how easy it should be to handle and how efficiently it should use the resources and give bug free processing and expected outputs.
2. **Product Revision (Ability to Change) :** The quality criteria specified in this category say that the software should be easy enough to test, maintain and make any changes if required.
3. **Product Transition (Adaptability to new Environments) :** The quality criteria specified in this category says that it must be easy to transit (carry or load or install) the software on any platforms, should be able to share its code with the other languages on the platform and try to write the code i.e. reusable in future in any other software development.

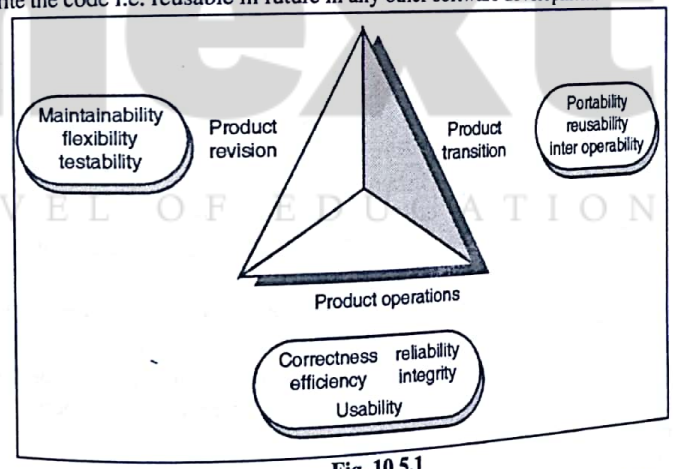

Fig. 10.5.1

The attributes of a quality/good software as shown in Fig. 10.5.1, categories are as below :

1. **Correctness :** Extent to which the functionality matches the specification
   It means the required functionality and correct results. Customer satisfaction depends on the degree to which customer requirements and expectations have been met. It should provide all the functionalities desired by the customer.
2. **Reliability :** Extent to which the system fails
   It means the *Bug free execution* i.e. assured performance, fully verified and validated processes, secured and safe to work when break down or power crisis. It should be reliable in all conditions. Reliability is measured in terms of the mean time between failures (average time between the

consecutive failures in a given period of life of a system), mean time to repair the equipment and mean time to recover (average time taken to return a system to the point where it failed and continue with its operation. Software reliability can be viewed from three different dimensions that specify the overall system reliability.

- *Hardware reliability* : It is the probability of hardware component failure and what time it takes to repair that component.
- *Software reliability* : It is the probability of the software failure i.e. how frequently software produces an unexpected and incorrect output. Software failures are different from hardware failures.
- *Operator reliability* : It is the probability of how frequently the operator of a system will make an error.

3. **Efficiency :** System resource (including CPU, disk, memory, network) usage.

It means the efficient use of resources. Software is said to be efficient if it uses all its resources i.e. memory, processor and storage in an effective manner. The software design and architecture should be such that it gives you the response in the least processing time, using the resources in the best possible way.

4. **Integrity :** Protection from unauthorized access

Integrity is related with the extent of access to software by unauthorized persons can be controlled.

5. **Usability :** Ease of use.
   - It means that the software must be user friendly. The software should have a good documentation and user manual which may include the installation and the process of using the software. This makes easy for the new to learn and operate just by studying the manual.
   - It acts as a bridge between the user requirements and the developed system. Usability counts in the terms of effectiveness, efficiency and satisfaction.
6. **Maintainability :** Ability to find and fix a defect.
   - For all the changes desired by the customer or the user, the software engineer has to respond fast. And this is possible only if the software design and its architecture are so chosen that changes can be carried out in the shortest time, without affecting the overall integrity of the software.
   - The change could be to correct the mistakes, expand its scope or adapt to new technology.
7. **Flexibility :** Ability to make changes required as dictated by the business.

The software should be developed so that if user demands any changes in the system at coding or testing phase i.e. in the middle of the SDLC, then it should be easy to insert these changes in the existing modules.

8. **Testability :** Ability to validate the software requirements.

It should need less effort to test a program so that it performs its intended function. As the complexity of the program increases, the efforts to test the software also increase. Testability can be defined in terms of the following attributes that try to uncover maximum number of defects with minimum efforts.

(i) **Operability :** The better the software works, the more efficiently it can be tested

(ii) **Observe-ability :** It is based on the fact that 'what you see is what you get' (WYSIWYG)

(iii) **Controllability :** The better way to control the software quality is to automate and optimize the tests.

(iv) **Decomposability :** isolating the problems and performing smarter re-testing.

(v) **Simplicity :** less there is to test, the more quickly it can be tested.

(vi) **Stability :** Fewer the changes, fewer the disruptions to testing.

(vii) **Understandability :** The more information we have about the system, the smarter we can test

9. **Portability :** Ability to transfer the software from one environment to another.

It is about developing the software to run on one operating system or hardware configuration while being conscious of how it might be shaped with minimum effort to run on other operating systems and hardware configurations as well.

10. **Reusability :** Ease of using existing software components in a different context.

It gives the concept of 'Write once and Use many times!' For example, writing functions or sub procedures to receive variable parameters. The calling code passes the values to the parameters and the called procedure processes them as needed. The advantage of this approach is that – once the code is written and fully tested, it can be used lot number of times anywhere.

11. **Interoperability :** Ease to which software components work together.

The software development should be so that it can interact with other products. For example, Word processor can incorporate charts from Ms-Excel or data from databases. It deals with the interface between software products over a communication network.

12. **Cost-effective :**

The development of the software within the cost and budget depends on efficient design and a high level of project management effort.

Syllabus Topic : Formal Approaches to SQA

10.6 Formal Approaches to SQA

Three formal approaches to SQA are :

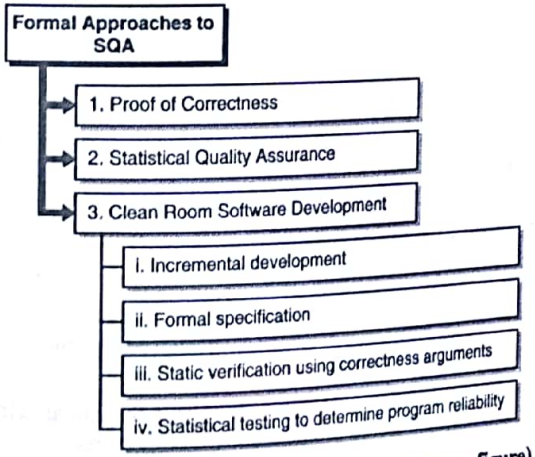

Fig. C10.5 : Formal Approaches to SQA (Change figure)

10.6.1 Proof of Correctness

The aim of this approach is to prove a program correct.

- It is a mathematical proof that when executed by computer program yields correct results.
- Hypothesis : it is a condition that the program must satisfy before the program is executed. This condition is also called as 'pre-condition'.
- Thesis : it is a condition that the program must satisfy after the execution of the program. This condition is also called as 'post-condition'.

10.6.2 Statistical Quality Assurance

- Identify software defects
- Each defect is tracked back to its cause
- correct the causes behind the defects.

10.6.3 Clean Room Software Development

- The name 'Clean Room Software Development' is derived from the 'Cleanroom' process of semiconductor fabrication.
- The philosophy behind this technique is *defect avoidance* rather than defect removal.
- Clean room software development is an engineering process to develop a high-quality software with certified reliability.
- As part of IBM's Federal Defence System, Harlan Mills, Dyer and Linger developed the CSE methodology in the early 1980s.
- Their followed the motto 'prevention is better than care' and thus, aimed to prevent software errors before they happen rather than correcting them after they occur.
- The four main components of clean room approach : to CSE are :

i. **Incremental development :** In clean-room process, each increment is developed separately and tested in a simulated environment to check the quality of the subsystem. The results of previous increments can be used to improve the quality of next increments which makes the process more successful. Incremental, prototyping and spiral SDLC models are useful here to manage the risks.

Advantages

- Allows early and continuous quality assessment throughout the development process.
- Provides increased user feedback at each and every step of development.
- Repairs if any process related problems occur.
- Allows the users to change their requirements

ii. **Formal specification :** The Box Structure Method is used for specification and design of the CSE process.

iii. **Static verification using correctness arguments :** Verification of program correctness is done by the team review based on correctness questions. Mathematical verification techniques are used to verify the program correctness.

iv. **Statistical testing to determine program reliability :** Statistical principles are used to test the software and certify its reliability. The sequence of clean room tasks for each increment is described in Fig. 10.6.1

Task 1 : Requirement gathering

A more detailed and descriptive user requirements are gathered and sustomer-level specification is developed.

Task 2 : Box structure specification

It makes use of box structures to describe the functional specification that confirms to the operational analysis principles. The box structures isolate and separate the creative definition of behaviour, data and procedures at each level of refinement.

Task 3 : Formal design

Clean-room uses box structure specification for specification and design of the CSE process. The box structure makes the formal design in three levels of abstractions :

1. *behavioral view* : represented by black box that describes the interaction of the whole system with the application environment.
2. *finite state machine view* : represented by state box
3. *procedural view* : represented by clear box that describes the procedures in the system.

Task 4 : Correctness verification

A series of rigorous correctness verification activities are conducted on the design and then on the code. Verification begins by verifying the highest level box structure i.e. the black box verification and then moves towards the design details i.e. state box and then towards the code i.e. clear box verification. This technique finds the errors faster and reduces the development and testing costs. This technique replaces the unit testing because the errors are found before testing.

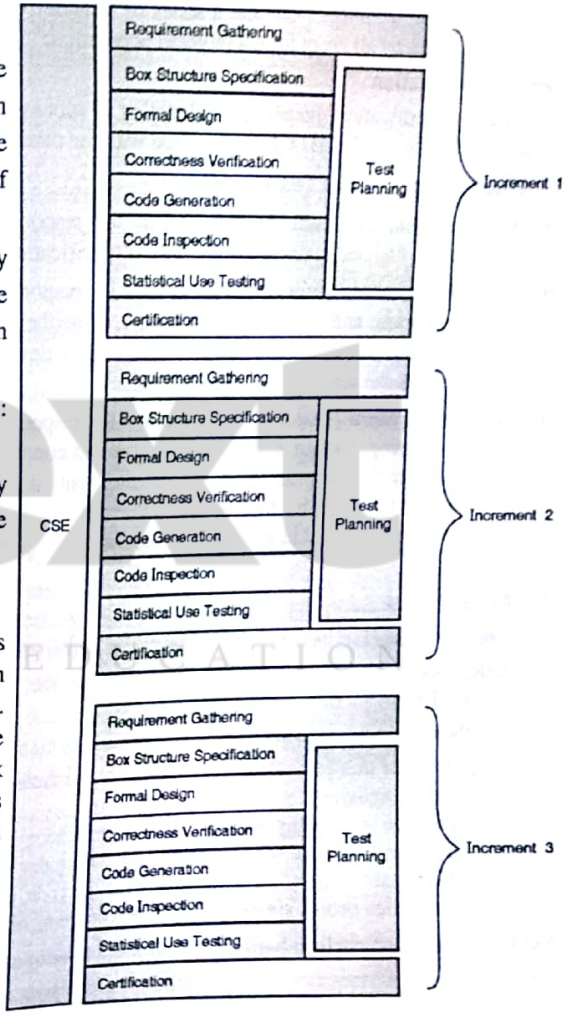

Fig. 10.6.1 : Clean-room Software Engineering (CSE) process

Task 5 : Code generation

The box structure specification represented in a specialized language is translated into appropriate programming language.

Task 6 : Code inspection

Standard Walkthroughs and inspections are conducted to ensure semantic and syntactic conformance of the code.

Task 7 : Test planning

A suite of test cases that exercise a probability distribution of usage is planned and designed. Test suits are developed in parallel with specification, verification and code generation.

Task 8 : Statistical use testing

Statistical use techniques execute a series of tests derived from a statistical sample of all possible program execution by all users from a targeted population.

Task 9 : Certification

Once the verification, inspection and testing is successfully completed, that particular increment (module) is certified as READY for integration with the other increments (or modules or components).

Cleanroom process Team

1. **Specification team :** This group is responsible for developing and maintaining the system specification.
2. **Development team :** This group is responsible for developing and verifying the software. It neither executes the software nor does the compilation. It only develops and inspects the software.
3. **Certification team :** This group is responsible for developing a set of statistical test cases and conducts these tests on the software after the development. It also uses reliability growth models to determine the software reliability and if it is acceptable, it certifies the increment.

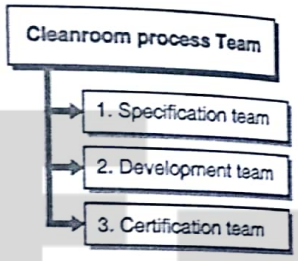

Fig. C10.6 : Cleanroom process Team

Advantages

- The results of CSE have been very impressive in delivered systems.
- Independent incremental assessment shows that the process is no more expensive than other approaches.
- But then, the process is not widely used because of the following reasons:
  - A belief that cleanroom methodology is too theoretical, too mathematical and too radical for use in reality.
  - Replace unit testing with correctness verification and statistical quality control that represents a major difference with the techniques used today in software development.
  - No constraints are set on how to write the code, code review is done mentally and verbally, no written proofs are required and no compiling of code is done.

Table 10.6.1 : Comparison between Traditional development models and Cleanroom development

	Typical Development	Cleanroom Development
1.	Specification is usually incomplete for external behavior.	Specification is complete providing the precise and complete description for external behavior
2.	From specification, code is informal, debugging is done to verify.	Box Structures are used to refine and verify the specification and designs.
3.	Failures are common and accepted	Failures are not at all accepted.
4.	It attempt to perform code coverage test but poor reliability prediction.	It is usage model based and predicts the field reliability

Syllabus Topic : Six Sigma

10.7 Six Sigma

- *Sigma* is the Greek letter that represents standard deviation in statistics. *Six Sigma* generated from 'sigma' has the same idea behind it i.e. it relies heavily on statistical techniques to reduce defects and measure quality.
- Six sigma stands for six standard deviations that strives for perfection – allows only 3.4 defects per million opportunities for each product or service transaction.
- Six-sigma was first started by Motorola in 1980's in its manufacturing division where millions of parts are made using the same process repeatedly.
- Slowly, Six Sigma was being used in other non manufacturing processes. Today, Six Sigma is applied to many fields such as Services, Medical and Insurance Procedures, Call Center, etc.
- Six-Sigma methodology involves the techniques and tools needed to improve the capability of the development process and thus, reduce the defects in it. Therefore, Six-Sigma is the method of constantly reviewing and re-tuning the process.
- Six Sigma experts (Green Belts and Black Belts) can either evaluate the existing business process or determine the ways to improve the existing process or design a brand new business process using *DFSS* (*Design For Six Sigma*) principles. Usually, it is easier to design a new process than refining an existing process.

Six-Sigma methodology defines **three core steps** abbreviated as **DMA :**

1. Define the customer requirements, deliverables and project goals via well defined methods of customer communications.
2. Measure the performance of the existing process and its output to determine the current quality performance.
3. Analyze the defect metrics and determine the vital few causes.

If Six Sigma methodology is used in ***improving the existing software process***, it suggests two additional steps with the above three core steps that can in all be abbreviated as DMAIC

1. Improve the process by eliminating the root causes of defects.
2. Control the process to ensure that future work doesn't re-introduce the causes of further defects.

If Six Sigma methodology is used to ***create a brand new software process*** from ground up, it suggests two other additional steps to the three core steps that can in all be abbreviated as

DMADV

1. Design the process :
   - To avoid the root causes of defects
   - To meet the customer requirements
2. Verify that the process model will in fact avoid the defects and meet customer requirements.

The Six-Sigma methodology leads to defect reduction and improvement in profits, product quality and customer satisfaction.

The three key elements of Six-Sigma are :

1. Customer Satisfaction.
2. Defining Processes and defining Metrics and Measures for Processes.

3. Team Building and Involving Employees - The company involves all employees and provides opportunities and incentives for employees to focus their talents and ability to satisfy customers.

Syllabus Topic : Software Reliability

10.8 Software Reliability

Reliability means the probability of failure-free operation of software in a specified hardware environment for a specified time.

It is the probability of the software failure i.e. how frequently software produces an unexpected and incorrect output. Software failures are different from hardware failures.

One of distinct characteristics of reliability is that it is objective, measurable, and can be estimated else many of the software quality attributes are subjective. These measurable criteria are called as *software metrics.*

☞ **Need**

With increasing number of software and expanding scope, software is in boom and simultaneously causes lot of failures. And the causes for these failures are the poorly designed GUIs and program coding.

10.8.1 Reliability Measures

- Reliability metrics are the units used to measure the system reliability
- **Software reliability** can be measured by counting the number of operational failures and relating these to the demands made on the system at the time of failure.
  - Probability of Failure on Demand (POFOD) = 0.001
- Where, for one in every 1000 requests the service fails per time unit of operation.
  - Rate of Fault Occurrence (ROCOF) = 0.02
- Where, two failures for each 100 operational time units of operation
- ***Software Reliability*** can be measured in terms of Mean-Time-Between-Failures (MTBF) where;

  Reliability = MTBF = MTTF + MTTR ;

  Where; MTTF is mean-time-to-failure and MTTR is mean-time-to-repair respectively.
- Measuring MTBF is more useful than measuring number of defects/KLOC because each defect only provides very little indication of the reliability of a system.
- ***Software availability*** is also a quality attribute which is described as the probability that a program is operating according to requirements at a given point in time. It is measured as;

  Availability = [MTTF / (MTTF + MTTR)] * 100%

☞ **Reliability Measurement Process**

The process of measuring reliability is illustrated in Fig. 10.8.1

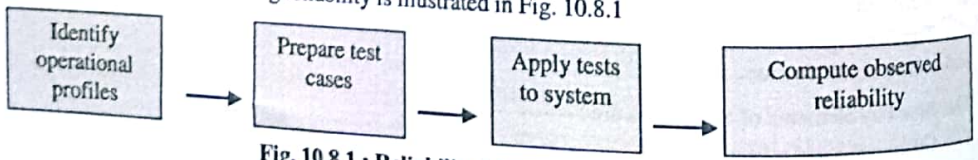

Fig. 10.8.1 : Reliability Measurement Process

Step 1 : Study the existing systems of the same type to identify the operational profile.

Step 2 : Then design a set of test cases that reflect the identified operational profile. These test cases can be generated from studying the existing system as stated in the above step using the test data generator tool which is a Computer Aided Software Testing (CAST) tool.

Step 3 : Apply the tests to the system and count the number and type of failures that occur.

Step 4 : After statistically computing the number of failures, you can compute the software reliability and then find the appropriate reliability metric value. This, this process is also called as statistical testing which aims to measure the software reliability.

10.8.2 Reliability Models

Software reliability models are useful in describing the characteristics of how and why software fails also quantifies the software reliability.

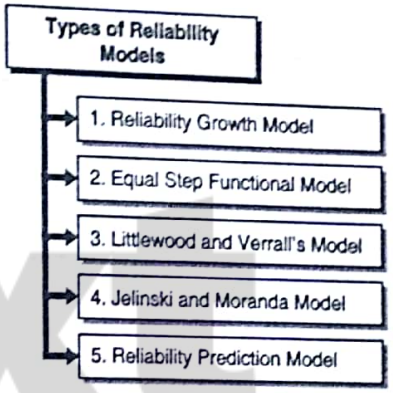

Fig. C10.7 : Types of Reliability Models

➔ **1. Reliability Growth Model**

- This model describes how the system reliability changes over time.
- During the testing process, as failures are discovered; the faults causing these failures are fixed so as to improve the software reliability i.e. changing of reliability from poor to better.

➔ **2. Equal Step Functional Model**

- This model illustrates the concept of reliability growth that describes that the reliability goes on increasing each time a fault is discovered and repaired and then a new version of software is created.
- This model assumes that the software repairs are always correctly implemented so as to reduce the number of software faults and associated failures in each new version of the system. All this description is signified in Fig.10.8.2

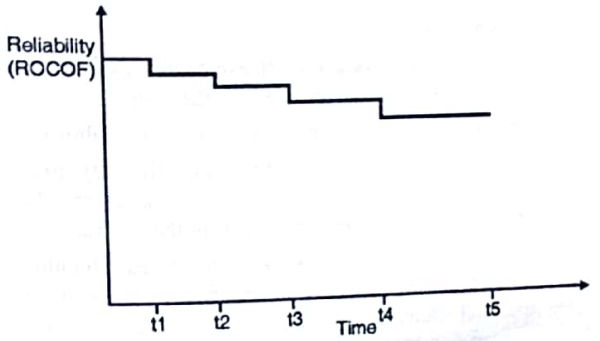

Fig. 10.8.2 : Equal Step Function Model representing Reliability Growth

- In practice, debugging doesn't always fix the software faults because changing a program sometimes introduces new faults into it. The probability of occurrence of these faults may be higher than the occurrence of the faults that have been repaired.
- Repairing the most common faults rather than repairing the occasionally occurring faults improves the reliability growth comparatively.

3. Littlewood and Verrall's Model (Random Step Function Model)

- This model considers the problem by introducing a random element in the reliability growth improvement effected by a software repair. Therefore, each repair does not result in an equal reliability improvement but varies on the basis of *random* perturbation.
- This model allows a negative growth when a software repair produces further errors thus decreasing the reliability. The reason behind this is that the frequently occurring faults are likely to be discovered early in the testing process.

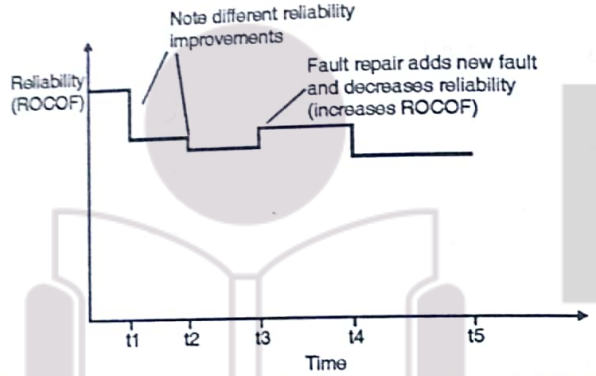

Fig. 10.8.3 : Random Step Function Model representing Reliability Growth

4. Jelinski and Moranda Model

- This model states that each time an error is repaired the reliability does not increase by a constant amount.
- Rather the reliability growth is by assuming that fixing of an error is proportional to the number of errors present in the system at that time.

5. Reliability Prediction Model

- Testing is very expensive but important to achieve quality software. However, there is a need to stop testing as soon as possible and not over-test the system.
- Testing must be stopped when the required level of system reliability is achieved.
- This required level of reliability can be checked using reliability prediction model and if this model predicts that the required level of reliability will never be achieved, then in this case, the manager must decide to rewrite the components of the software.
- The reliability can be predicted by comparing the measured reliability value with the known reliability model. That means, reliability is predicted by matching the achieved reliability value to some historical reliability data. You can extrapolate the model to the required level of reliability and observe at what point is the required level of reliability is achieved.

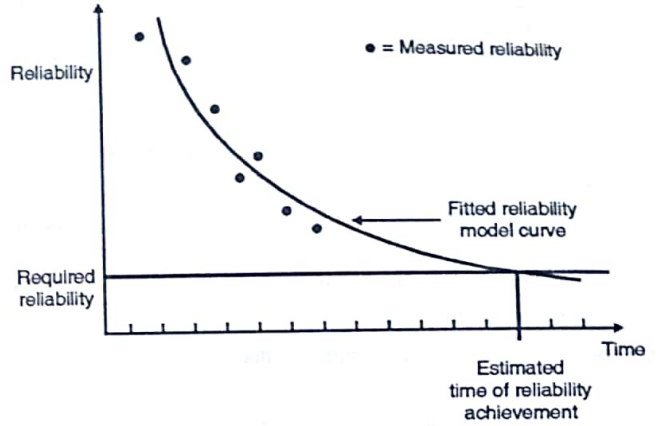

Fig. 10.8.4 : Reliability Prediction Model

Predicting the system reliability has two main benefits

1. **Planning of testing :** Using the testing schedule, we can predict the completion time of testing. And if this prediction says that the completion time is after the planned delivery date, then you have to use additional resources to conduct testing and debugging so as to accelerate the rate of reliability growth.
2. **Customer negotiations :** Sometimes the reliability model shows that reliability growth is very slow and testing is also giving relatively very less benefit, or if the model predicts that the required reliability level will never be reached, then in these cases, it would be worth to renegotiate the reliability requirements with the customer of the system.

10.9 SQA Planning and Standards

- SQA planning is the process of developing a quality plan that lists the desired software qualities and describes how these are to be verified.
- SQA plan involves those organizational standards that are appropriate to a particular product and development process.

Quality plan includes

1. **Proposed product introduction :** It provides detail description of the product, its intended market and quality expectations from the product.
2. **Product plans :** It includes the critical release dates and responsibilities for the product along with plans for distribution and product servicing.
3. **Process description :** It describes the development and service processes that will be used for product development and management.
4. **Quality tools :** It describes the identification and justification of critical product quality attributes.
5. **Risk and risk management :** It describes the most probable risks that may affect the product quality and the actions to address these risks.

The overview of the quality plan is as described above but it differs in describing the details depending on the size and type of the system that is being developed. But the plan should be designed as short as possible because if it is too long then people will not read it and the purpose of preparing the quality plan is ultimately lost.

☞ **The SQA plan**

- The SQA plan provides a road map for monitoring the software quality assurance.
- The SQA plan is developed by the SQA group.
- The SQA plan serves as a template for conducting the SQA activities that are assigned for each software project.
- The IEEE has published a standard format for SQA plan. This plan has the following structure.

1. **Purpose and Scope of the plan**
2. **References** i.e. which existing system or documents are studied to get the knowledge about the new system.
3. **Management** describes the roles and responsibilities of SQA in the organizational structure, tasks of SQA group, their roles and responsibilities related to product quality.
4. **Documentation** describes all software engineering work products produced as part of the software development process. It includes project documents, design models, technical documents and user manuals.
5. All applicable **Standards, Practices, and Conventions** list the standards/practices applied during the software development process.
6. **Reviews and Audits** provide a brief overview of the review and audit approach to be conducted during the project development.
7. **Tests** describe the test plan and procedures and the record keeping requirements.
8. **Problem reporting and corrective actions** define the appropriate procedures required for reporting, tracking and ultimately resolving the errors.
9. **Tools and Methods** that describe the assembling, safeguarding and maintaining of all SQA related records.

10.9.1 SQA Standards

SQA defines a set of standards that should be applied to the software development process or software product.

QA establishes two types of standards :

1. **Product standards :** These set of standards are applied to the software product that is being developed. They include the document standards such as the structure of requirements, standards commitment header from an object class definition and coding standards that define how a programming language should be used.
2. **Process standards :** These set of standards are applied to the software process that are followed while developing the software. They include the definitions of specifications, design and validation processes and a description of the documents that should be written in the course of these processes.

The product and process standards are inter-linked with each other – product standards are applied to the output of the software process and process standards include the software process activities that ensure that product standards are followed.

☞ **Need of SQA standards**

1. Provide the knowledge about the best appropriate practices followed in the company which is often achieved after a great deal of trial and error work. All these best practices are built up as a standard which guides the company from avoiding repeating the past mistakes.
2. Standards provide a framework for enforcing quality assurance activities as they inculcate bet practices ensuring that the standards have been properly followed.
3. They help in continuity where work carried out by one person can be handed over and continued by other person.
4. Standards ensure that all the engineers within an organization adapt the same practices. Therefore, the effort needed to learn new practices is reduced.

☞ **Setting steps of SQA standards**

Quality managers who set the standards of the organization have to be well-equipped with skills and resources and should follow the below steps to build the standards;

1. Involve the software engineers in the selection of the product and process standards. Engineers should understand the importance of standards, why they have been designed and must be committed to these standards. The standards document should not just state the standards but they should also mention why particular standardization decisions have been made and what is its use.
2. Review and modify the standards periodically to cope with the changing technologies. Once the standards are developed, they need to be displayed in the organization's standards hand book and must be periodically updated to reflect the changing requirements and technologies.
3. Provide the software tools to support the standards whenever possible. Using the manual development process makes it difficult to comply with the standards but the use of tools provides comparatively more compliance and also reduces the extra efforts needed in that process.

10.9.2 ISO Quality Standards

- There are various international set of standards set out by International Standards Organization (ISO) that can be applied in the development of a quality management system across all industries.
- These ISO standards are concerned with the quality process in organizations that design, develop and maintain the products.
- One of such ISO standards is ISO 9126 that is specially aimed to software development and sets out some key quality attributes to assure the quality of the product being developed.

Syllabus Topic : The ISO 9000 Quality Standards

10.9.2(A) ISO 9000 Quality Standard

ISO 9000 is a set of QA standards that define a basic set of good practices to guide the company in delivering the user satisfying product.

☞ **Benefits of ISO 9000**

1. It focuses on the development process and not the product. It is concerned about the way an organization goes about its work and not the results of the work. And this is because having a quality development process will ultimately help in achieving quality product.
2. It only illustrates the process requirements and not how they are to be achieved. For example, the ISO 9000 standards says that a software team should plan and perform product design reviews but it doesn't say how that requirements should be accomplished.

The sections of ISO 9000 that deal with software are ISO 9001 and ISO 9000-3.

- The ISO 9001 is for business that designs, develops, installs, and services the products.
- The ISO 9000-3 is for business that develops, supplies, installs, and maintains the computer software.

10.9.2(B) ISO 9001

- It describes various aspects of the quality process and lays out the organizational standards and procedures that company should define.
- These should be *documented* in an organizational quality manual which should include the descriptions of the defined processes that have been followed during product development.

Documentation standards in ISO 9001

Documentation standards are important because documents are the only tangible way of representing the software and the software process. There are three major types of documentation standards :

1. **Document process standards :** These standards define the process that should be followed for document production.
2. **Document standards :** These standards govern the structure and presentation of documents.
3. **Document interchange standards :** These ensure that all electronic copies of documents are compatible.

The Fig. 10.9.1 depicts the model of a documentation process where - drafting, checking, revising and redrafting are iterative processes. It should continue until the document of acceptable quality is produced. The acceptable quality level depends on the document type and the potential readers of the document.

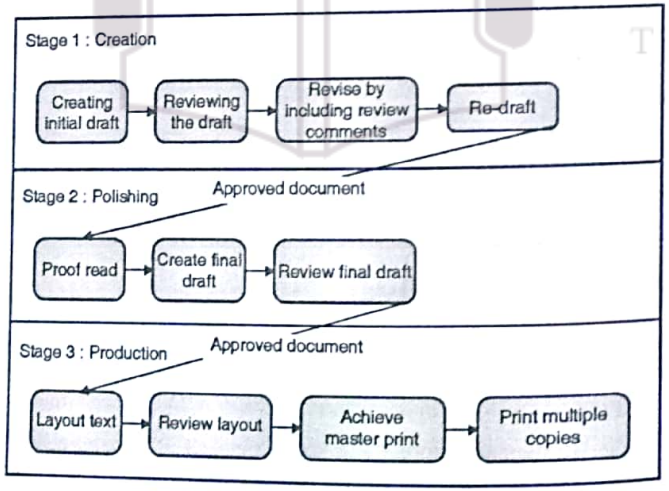

Fig. 10.9.1 : Documentation Procedure

Few Documentation Structures that may be developed are;

- **Document identification standards :** Thousands of documents are developed in large projects where each needs to be identified uniquely. This identifier is usually defined y the configuration manager.
- **Document structure standards :** These define the conventions used for page numbering, page header and footer information and section and sub section numbering.
- **Document presentation standards :** These include the definition on fonts, and styles used in the documentation, the logos, the company name and the colour opted to highlight the document structure.
- **Document update standards :** It reflects the changes in the system i.e. it provides a consistent way of indicting the document changes.

Criteria set by ISO 9000-3 to create a better software development process :

- Develop detailed quality plan and procedures to control configuration management, verification and validation, bug fixing and corrective actions.
- Prepare and receive approval for a software development plan that includes a definition of the project, a list of the project's objectives, a project schedule, a product specification, a description of how the project is organized, a discussion of risks and assumptions and strategies for controlling it.
- Communicate the specification to customer in the language that makes them understand and helps them in validating during testing process.
- Plan, develop, document and perform software design review procedures.
- Develop and document software test plans
- Develop methods to test whether the software meets the user requirements.
- Perform software validation and acceptance tests.
- Maintain the test cases and test results.
- Prove that the product is ready before it is released.
- Use statistical techniques to analyze the software development process and to evaluate product quality.

10.9.2(C) ISO 9126 Quality Standard

The ISO 9126 standard identifies 6 key quality attributes.

- **Functionality** : Degree to which software satisfies stated needs.
- **Reliability** : The degree to which software is up and running in unfavourable conditions.
- **Usability** : The degree to which software is easy to use.
- **Efficiency** : The degree to which software makes optimum use of resources.
- **Maintainability** : The ease with which the software can be modified.
- **Portability** : The ease with which software works on different environments.

Few other Quality Standards are :

- TQM (Total Quality Management)
- SEI CMM (Software Engineering Institute set of Capability Maturity Models)

ISO 15504

Syllabus Topic : Capability Maturity Model

10.10 CMMI Process Improvement Framework

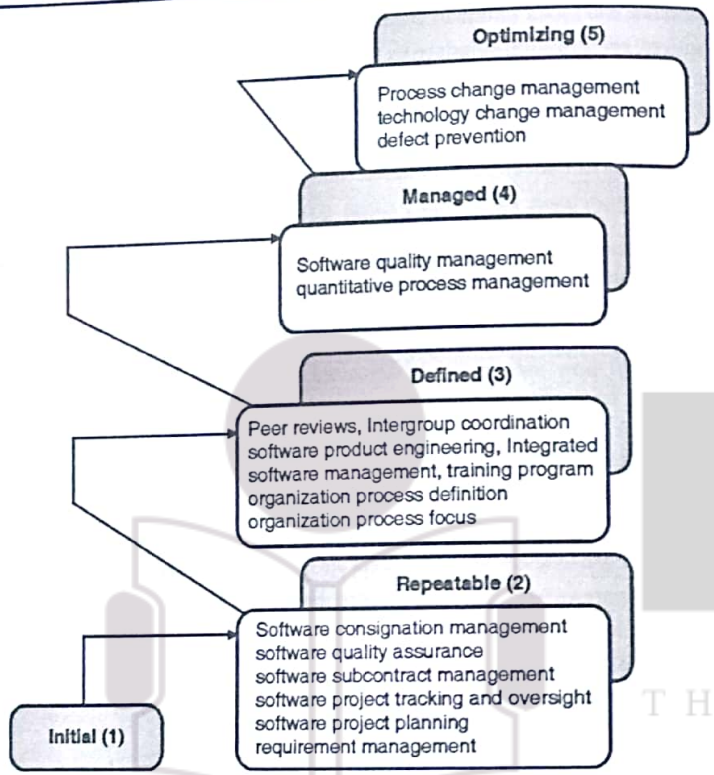

Fig. 10.10.1 : CMMI and Key Process Areas

- *Capability Maturity Model (CMM) is a maturity model applied within the context of Software Project Improvement framework.*
- *CMM is a specific* approach taken for quality assurance.
- The CMMI (Capability Maturity Model Integration) is used to implement Quality Assurance in an organization.
- The CMMI describes an evolutionary improvement path from an adhoc, immature process to a mature, disciplined process which describe the key elements of an effective software process.
- The CMMI includes key practices for planning, engineering, and managing the software development and maintenance which helps in improving the ability of organizations to meet goals for the cost, schedule, functionality, and product quality.
- The CMMI helps in judging the maturity of an organization's software development process and compares it to the state of practice of the industry.
- The CMMI categorizes five levels of process maturity :

  Level 1-Initial : The software process is characterized as ad hoc and chaotic. The success generally depends upon individual efforts.

Level 2-Repeatable : Basic project management processes are established to keep track of cost, schedule, and functionality. Then, the same process is used repeatedly for all the similar type of projects.

Level 3-Defined : The software development process is documented, and integrated into an approved standard software development process for the organization. All projects use this approved version of the organization's standard software process for developing and maintaining the software.

Level 4-Managed : Detailed measures for software process and product quality are collected, understood and implemented.

Level 5-Optimizing : Continuous process improvement is done by continuous feedback from the process and inculcating innovative ideas and technologies.

At this level, the entire organization focuses on continuous Quantitative feedback from previous projects which is used to improve the project management.

The software process at this level can be characterized as continuously improving the process performance of their projects.

Improvement is done both by incremental enhancements in the existing process and by innovations using new technologies and methods.

These levels are decomposed into several **key process areas.**

- **Process Change Management :** To identify the causes of defects and prevent them from re-occurring.
- **Technology Change Management :** To identify beneficial new technologies and incorporate them in an orderly manner.
- **Defect Prevention :** To continuously improve the process inorder to improve the quality productivity and thus decrease development schedule and cost.

Drawback of CMM

- The *CMM* does not describe how to create an effective software development process.
- The qualities it measures are practically difficult to implement in an organization.

Review Questions

Q. 1 Why is software quality important ?
Q. 2 What are the various causes behind the errors in a software product ?
Q. 3 State the difference between quality control and quality assurance.
Q. 4 Explain PDCA cycle in brief.
Q. 5 State the goals of SQA.
Q. 6 State and explain the quality metrics and measurements.
Q. 7 List down the McCalls quality factors.
Q. 8 What are the three formal approaches to SQA ? Explain cleanroom process in detail.
Q. 9 Explain six-sigma in detail.
Q. 10 Explain the software reliability measures in brief.
Q. 11 Write in short about various ISO quality standards.
Q. 12 Explain CMMI framework in detail.