1.1 Quality Concepts

- According to David Garvin of the Harvard Business School suggest that "quality is a complex and multifaceted concept" that can be described from five different point of view.
- The **transcendental view** argues(like persig) that quality is something that you immediately recognize but cannot explicitly defined.
- The **user view** sees quality in terms of and end users specific goals, if the product meets the goal those goals it exhibit quality.
- The **manufacturer view** defines quality in terms of the original specification of the product. If the product conforms to the spec, If exhibits quality.
- The **product view** suggests that quality can be tied to inherent characteristics(eg. Functions and features) of a product.
- Finally, the **value based view** measures quality based on how much a customer is willing to pay for a product.

SOFTWARE QUALITY

“In the most general sense, software quality can be defined as “An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.”

The definition serves to emphasize three important point:

1. An effective software process establishes the infrastructure that support any effort at building a high-quality software product.
2. A useful product delivers the content, functions and features that the end user desires, but as important, it delivers these actions in a reliable, error-free way.
3. By adding value for both the producer and user of a software product, high quality software provides benefits for the software organization and the end user community.

ISO 9126 QUALITY FACTOR

The ISO 9126 standard was developed in an attempt to identify the key quality attributes for computer software. The standard identifies six key quality attributes.

Functionality:

Degree to which software satisfies stated needed as indicated by sub attribute :suitablity, accuracy, interoperability compliance and security.

Reliability:

The amount of time that the software is a available for use as indicated by the following sub attributes maturity, fault tolerance. Recoverability

Usability:

The degree to which the software is easy to use as indicated by the sub-attributes: understandability, learnability, operability.

ISO 9126 QUALITY FACTOR

Efficiency:

The degree to which the software makes optimal use of system resources as indicated by the following attributes:time behavior, resource behavior.

Maintainability:

The ease with which repair may be made optimal use of system resources as indicated by attributes: time behavior. Resource behavior.

Portability:

The easer with which the software can be transposed from one environment to another as indicated by the attribute:adaptability, installability, conformance, replaceability.

9.2 Software Quality Assutance

•Software quality assurance (often called quality management) is an umbrella activity that is applied throughout the software process.

•Software quality assurance encompasses a broad range of concerns and activities that focus on the management of software quality.

•Software quality assurance (SQA) encompasses:

▪An SQA process,

▪Specific quality assurance and quality control tasks (including technical reviews and a multi-tiered testing strategy)

▪Effective software engineering practice (methods and tools)

▪Control of all software work products and the changes made to them

▪A procedure to ensure compliance with software development standards (when applicable)

▪Measurement and reporting mechanisms

9.2 SOFTWARE QUALITY ASSUTANCE

SQA includes following major activities:

1.Software Quality Planning–selection of appropriate procedures and standards from this framework and adaptation of these to specific software project

2.Software Quality Control–definition and enactment of processes that ensure that project quality procedures and standards are being followed by the software development team

3.Software Quality Metrics–collecting and analyzing quality data to predict and control quality of the software product being developed

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

1. Standards
2. Reviews and audits
3. Testing
4. Error/defect collection and analysis
5. Change management
6. Education
7. Vendor management
8. Security management
9. Safety
10. Risk management

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

1. Standards:

The job of SQA is to ensure that standards that have been adopted are followed and that all work products conform to them.

2. Reviews and audits:

Technical reviews are a quality control activity. Their intent is to uncover errors.

Audits are a type of review performed by SQA personnel with the intent of ensuring that quality guidelines are being followed for software engineering work.

3. Testing:

The job of SQA is to ensure that testing is properly planned and efficiently conducted so that it has the highest likelihood of achieving its primary goal.

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

4. Error/defect collection and analysis:

SQA collects and analyzes error and defect data to better understand how errors are introduced and what software engineering activities are best suited to eliminating them.

5. Change management:

SQA ensures that adequate change management practices have been instituted.

6. Education:

The SQA organization takes the lead in software process improvement and is a key proponent and sponsor of educational programs.

7. Vendor management:

The job of the SQA organization is to ensure that high-quality software results by suggesting specific quality practices that the vendor should follow (when possible), and incorporating quality mandates as part of any contract with an external vendor.

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

8. Security management:

SQA ensures that appropriate process and technology are used to achieve software security.

9. Safety:

SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.

10. Risk management:

Although the analysis and mitigation of risk is the concern of software engineers, the SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

SQA TASKS

Prepares an SQA plan for a project:

The plan identifies evaluations to be performed, audits and reviews to be conducted, standards that are applicable to the project, procedure for error reporting and tracking, work products that are produced by the SQA group.

Participates in the development of the projects software process description

The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards(eg. ISO 9001) and other parts of the software project plan

Review software engineering activities to verify compliance with the defined software process:

The SQA group identifies, documents and tracks deviations from the process and verifies that corrections have been made.

SQA TASKS

- Audits designated software work products to verify compliance with those defined as part of the software process:
- The SQA group reviews selected work products; identifies, documents and tracs deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager.
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure:
- Deviations may be encountered in the project plan, process description, applicable standards or software engineering work products.
- Records any noncompliance and reports to senior management:
- Noncompliance items are tracked until they are resolved.

FORMAL APPROACHES TO SQA

- Need of more formal approaches for SQA
- Generally, they are
- Proof of coorectness
- Statistical SQA
- Clean room software engineering

FORMAL APPROACHES TO SQA

Proof of Correctness:

Basically, focuses correctness of code.

FORMAL APPROACHES TO SQA

STATISTICAL SQA

Approach focusing industry to become more quantitative about quality.

•Statistical SQA implies the following steps:

▪**Collection** of information about **software defects** is done.

▪**Software defects** are **categorized**.

▪**Cause** of each **defects** are **traced**.

▪Using **Pareto principle**, vital **causes** of **errors** are **identified**.

▪Move to **correct** these **causes** are taken.

FORMAL APPROACHES TO SQA

Clean Room Software Engineering

- Intended to produce software with a certifiable level of reliability
- Focus is on defect prevention, rather than defect removal.
- The name “cleanroom” was chosen to evoke the cleanrooms used in the electronics industry to prevent the introduction of defects during the fabrication of semiconductors.

Central principles

- Software development based on formal methods
- Incremental implementation under statistical quality control
- Statistically sound testing.

FORMAL APPROACHES TO SQA

- **STATISTICAL QUALITY ASSURACE –SIX SIGMA**

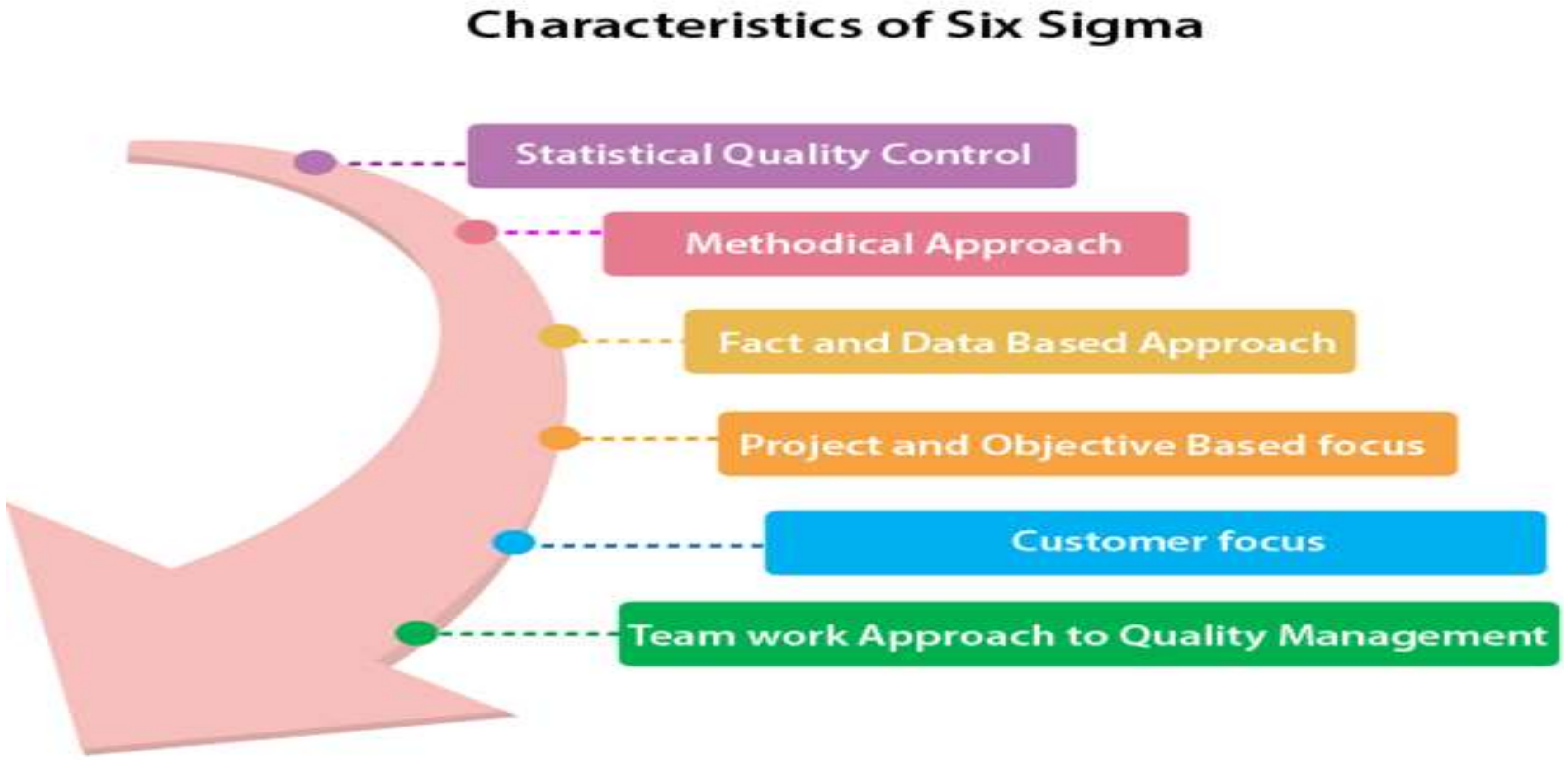
•Six Sigma is the most widely used strategy for statistical quality assurance in industry today.

•It was introduced by Engineer Sir Bill Smith while working at Motorola in 1986.

•The term Six Sigma is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high-quality standard.

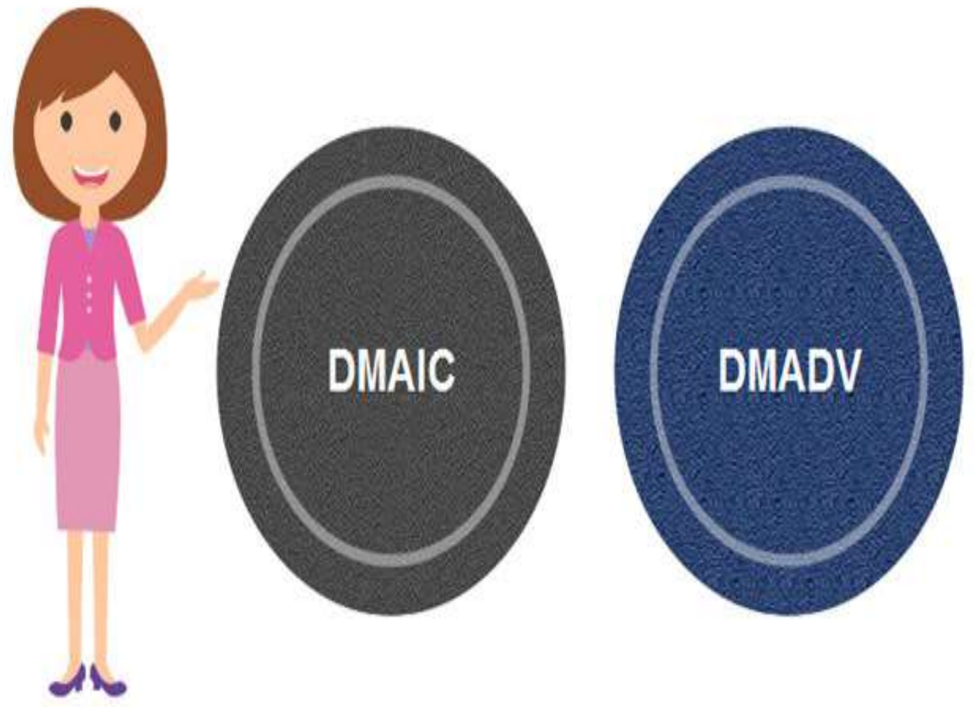
FORMAL APPROACHES TO SQA

CHARATERISTICS OF SIX SIGMA


Characteristics of Six Sigma
Statistical Quality Control
Methodical Approach
Fact and Data Based Approach
Project and Objective Based focus
Customer focus
Team work Approach to Quality Management

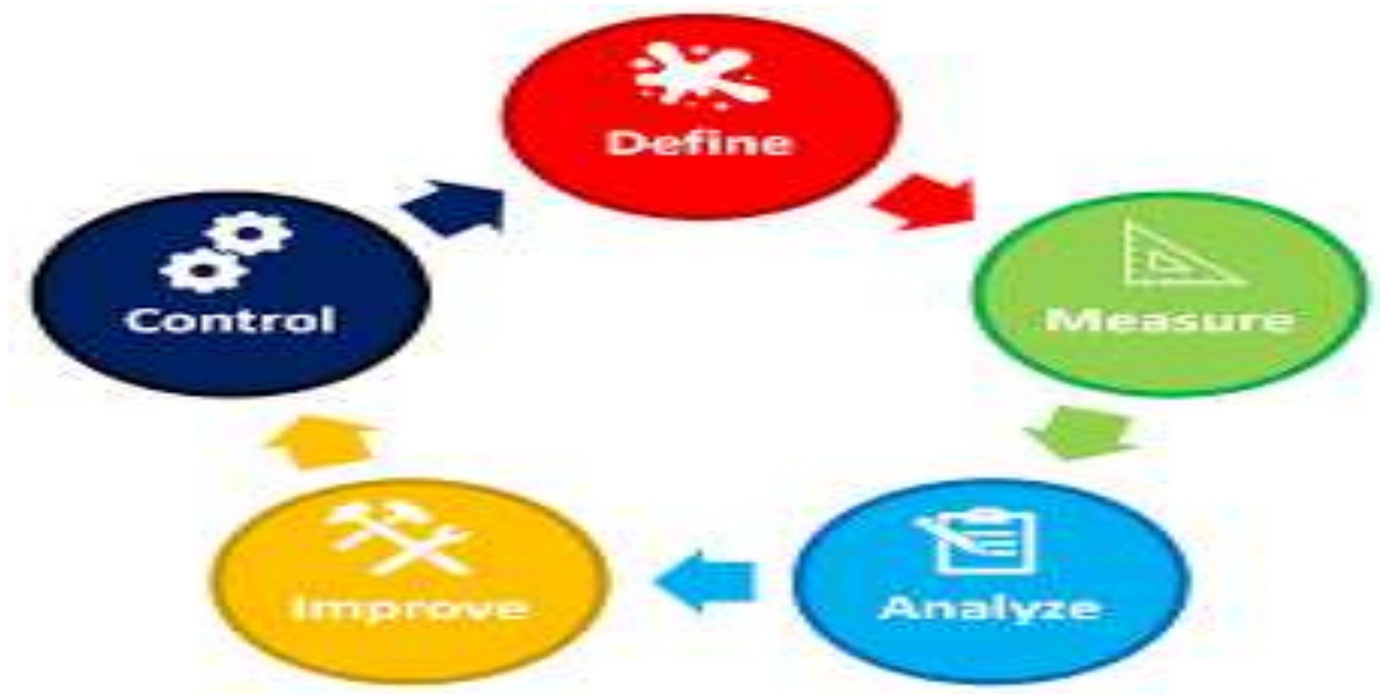
FORMAL APPROACHES TO SQA

Two six sigma methodologies;



FORMAL APPROACHES TO SQA

DMAIC PHASES:



FORMAL APPROACHES TO SQA

DMAIC PHASES:

- If an existing software process is in place, but improvement is required.
- •Process
- ▪**Define**: customer requirements and deliverables and project goals via well-defined methods of customer communication.
- ▪**Measure**: the existing process and its output to determine current quality performance (collect defect metrics).
- ▪**Analyze**: defect metrics and determine the vital few causes.
- ▪**Improve**: the process by eliminating the root causes of defects.
- ▪**Control**: the process to ensure that future work does not reintroduce the causes of defects.

FORMAL APPROACHES TO SQA

DMADV PHASES

- If an organization is developing a software process (rather than improving an existing software).
- •Process
- ▪**Define**: Team will define the project's goals and deliverables.
- ▪**Measure**: The team measures the project's factors that are critical to its deliverables.
- ▪**Analyze**: Working in conjunction with the measure phase is the analyze phase, when the team analyzes the process options that will best meet the customer's required deliverables.
- ▪**Design**: The team will document the detailed process that meets the customer's deliverables.
- ▪**Verify**: The team will verify that the customer's needs are met through the use of the newly designed process.

The SQA Plan

Comprises of the procedures, techniques, and tools that are employed to make sure that a product or service aligns with the requirements defined in the SRS(software requirement specification)

•The SQA Plan provides a road map for instituting software quality assurance.

•Developed by the SQA group (or by the software team if an SQA group does not exist), the plan serves as a template for SQA activities that are instituted for each software project.

•Identifies the SQA responsibilities of a team

•Lists the areas that need to be reviewed and audited

THE SQA PLAN

A standard for SQA plans has been published by the IEEE. The standard recommends a structure that identifies:

1.The purpose and scope of the plan

2.A description of all software engineering work products (e.g., models, documents, source code) that fall within the purview of SQA

3.All applicable standards and practices that are applied during the software process

4.SQA actions and tasks including reviews and audits) and their placement throughout the software process

5.The tools and methods that support SQA actions and tasks

6.Software configuration management procedures

7.Methods for assembling, safeguarding, and maintaining all SQA-related records

8.Organizational roles and responsibilities relative to product quality

SOFTWARE REVIEW

A systematic inspection of a software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC).

Purpose

serves to uncover errors in analysis, design, coding, and testing.

Why

- To improve the productivity of the development team.
- To make the testing process time and cost effective.
- To make the final software with fewer defects.
- To eliminate the inadequacies.

SOFTWARE REVIEW

Types of Reviews:

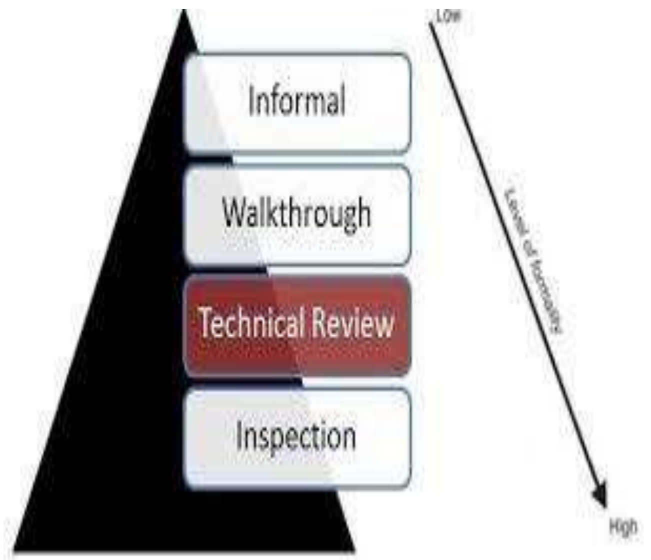
1.**Informal reviews**

▪informal meeting and informal desk checking

2.Formal reviews

▪Walkthrough, technical reviews, inspection

TYPES OF REVIEWS?



Software Review

Formal Technical Reviews:

Objectives of FTR:

•To uncover errors in function, logic, or implementation

•To verify the software under review meets its requirements

•To ensure that the software has been represented according to predefined standards

•To develop software in a uniform manner

•To make projects more manageable

Purposes of FTR:

•Serves as a training ground for junior engineers

•Promote backup and continuity

SOFTWARE REVIEW

Review meeting’s constraints

- 3-5 people involved in a review
- Advanced preparation (no more than 2 hours for each person)
- The duration of the review meeting should be less than 2 hours
- Focus on a specific part of a software product

People involved in a review meeting

- Producer, review leader, 2 or 3 reviewers (one of them is recorder)

SOFTWARE REVIEW

Before The FTR:

- **The preparation of a review meeting**
- •A meeting agenda and schedule (by review leader)
- •Review material and distribution (by the producer)
- •Review in advance (by reviewers)

SOFTWARE REVIEW

During the FTR:

- Moderator, usually QA leader, is responsible for running the meeting and orchestrating how the issues are to be presented.
- •Reviewers raise issues based on their advance preparation.
- •Get the big problem issues on the table early. Don't get lost in minutely small clerical details.
- •Identify problems, don't solve them.
- •Focus on material, not the producer
- •Elicit questions, don't answer them.
- •Don't explain how your product works. An FTR is not a product demo.
- •Watch the clock and keep momentum to get through all the issues in the alotted time.
- •Recorder actively records all issues raised.

Software Review

At the end of FTR

- Meeting decisions
  - Accept the work product w/o further modification
  - Reject the work product due to errors
  - Accept the work under conditions (such as change and review)

Software Review

After FTR:

•Write up the issues list.

- •Examine the issues raised, determine which to take action on.
- •Assign each issue to someone as an action item and revise work product as needed.
- •**Review issues list serves two purposes**
- ▪To identify problem areas in the project
- ▪To serve as an action item checklist (a follow-up procedure is needed)
- •**Review summary report**
- ▪What was reviewed?
- ▪Who reviewed it?
- ▪What were the findings and conclusions?
- •Have reviewers sign off on summary report, indicating proper corrective action has been taken for each issue.
- •Include signed report in deliverable.

RELIABILITY

- Reliability is a broad concept:
- It is applied whenever we expect something to behave in a certain way.
- Reliability is one of the metrics that are used to measure quality.
- It is a user-oriented quality factor relating to system operation:
- Intuitively if the users of a system rarely experience failure the system is considered to be more reliable than one that fails more often.
- A system without faults is considered to be highly reliable:
- Constructiong a correct system is a difficult task
- Even an incorrect system may be considered to be reliable if the frequency of failure is acceptable

Key concepts in discussing reliability:

- Fault
- Failure
- Time
- Three kinds of time intervals:MTTR,MTTF,MTBF

RELIABILITY

Failure:

A failure is said to occur if the observable outcome of a program execution is different from the expected outcome.

Fault:

A "fault" is the term used to describe the identified or determined cause of failure.

Example: A failure may be cause by a defective block of code.

Time:

Time is a key concept in the formation of reliability. If the time gap between two successive failure is short we say we say that the system is less reliable.

RELIABILITY

Definations of Software Reliability:

Software reliability is defined as the probability of failure free operation of a software system for a specified time in a specified environment.

Key elements of the above definition:

- Probability of failure free operation
- Length of time of failure free operation
- A given execution enviroment

Example: The probability that a PC in a store is up and running for eight hours without crash is 0.99.

SOFTWARE RELIABILITY METRICS

Reliability metrics are used to quantitatively express the reliability of a software product. Some reliability metrics are:

1. MTTF(Mean Time To Failure):

It is the average time between observed system failure.

MTTF=$\frac{total\ hours\ of\ operation}{total\ number\ of\ observed\ failure}$

2. MTTR(Mean Time To Repair):

It measures the average time it takes to track the errors causing the failure and then to fix them

MTTR=$\frac{total\ hours\ of\ Maintenance}{total\ number\ of\ repairs}$

3. MTBF(Mean time between failure)

The average time duration between inherent failures of a repairable system component.

MTBF=$\frac{total\ hours\ of\ Operation}{total\ number\ of\ failures}$

3. Availability:

Software availability is the probability that a program is operating according to requirements at a given point of time.

It is given by:

Availability=MTBF/(MTBF+MTTR)

Hence reliability can be calculated as

Reliability=$e^{-\lambda t}$

CLASSIFICATION OF SOFTWARE METRICS

Product metrics:

It describes the measurement of the product.

Generally includes the measurement of –

- Size
- Complexity
- Designs features
- Performance
- Quality level
- Reliability
- functionality

CLASSIFICATION OF SOFTWARE METRICS

Process metrics:

Used to improve the development process and maintenance activities of software.

Generally includes the measurement of –

- Effort required
- time to produce the product
- Number of defects found
- Tools and technology
- Quality and efficiency

Classification of Software Metrics

Project Metrics:

It describes project characteristics and execution

Generally measures-

- Number of software development
- Cost
- Schedule
- Productivity
- Quality
- Assess status of ongoing project

A Framework for Software Metrics

- Software metrics is a measure of some property of a piece of software ot its specifications.
- In framework for software metrics, Measures, Measurement, Metrics and indicators are often used interchangeably.

Measures:

Provides a quantitative indication of the extent, amount. Dimension, capacity or size of some attributes of a product or process.

Measurement:

The act of determining a measure.

Metrics:

A quantitative measure of the degree to which a system, component or process processes a given attribute.

Indicator:

A metric or combination of metrics gives valuable information about the software process, project, or product, providing insights into its performance and quality.

A FRAMEWORK FOR SOFTWARE METRICS

Activities of a Measurement Process:

- Formation: This step involves choosing the right measures and metrics that represent the software's important aspects and characteristics.
- Collection: Here, data is gathered using a specific method to calculate the chosen metrics.
- Analysis: The collected data is processed using math to compute the metrics.
- Interpretation: The metrics are carefully studied to understand the software's quality and performance.
- Feedback: Based on the interpretation, suggestions and recommendations are given to the software development team for improvements.

METRICS FOR ANALYSIS AND DESIGN

The metrics for analysis are:

Functionality delivered:

It provides an indirect measure of the functionality that is packaged within the software.

System size:

It measures the overall size of the system defined in terms of information available as part of the analysis model.

Specification quality:

It provides as indication of the specific and completeness of a requirement specification.

The metrics for the design are:

Architectural metrics:

Provide an indication of the quality of the architectural design.

METRICS FOR ANALYSIS AND DESIGN

Component-level metrics:

It measures the complexity of software components and other characteristics that hava a bearing on quality.

Interface design metrics:

It focuses primarily on usability.

Specialized object oriented design metrics:

It measures characterisctics of classes and their communication and collaboration characteristics.

ISO STANDARD

- ISO stands for International Standard Organization.
- It is an independent , non-government international organization for developing standards.
- These standards are developed for ensuring quality, safety and efficiency of products, services and systems.

ISO 9000 Series/ ISO 9000 Model:

- It is a series of standards developed by ISO
- These standards have been developed for assuring quality(QA) for manufacturing and service industries.
- Initially ISO 9001,9002 and 9003 have been launched and later ISO 9004 also came into existence.

ISO STANDARD

ISO 9000(1987):

- It is defined as a set of international standards on quality management and quality assurance.
- Applicable to any size of organization.
- Be usable by all sectors.

ISO 9001(1987):

- Current version is released in sep 2015
- This standard is provided to the organization which are involved in creating new products.
- If focuses quality assurance in Design, development and production
- This standard is best suited for software development organization.

ISO STANDARD

ISO 9002:

This standard is applicable to those companies which do not design product but involved in manufacturing

ISO 9003:

Latest version ISO 9003:1994

Applicable to the organizations that are ony involved in the installation and testing of the products.

ISO 9004:(ISO 9004:2018)

- This standard gives the guidelines for enhancing an organizations ability to achieve sustained success.
- It provides a self assessment tool.

Software Certification

- Software certification is a process where an independent authority evaluates and approves a software system to ensure that it meets certain standards of reliability, safety, and performance.
- valuation is done using established criteria and testing methods to confirm that the software functions as intended and is free from significant issues or vulnerabilities.
- Once the software passes the certification, it is considered trustworthy and can be used with confidence by users and organizations.
- This certification provides assurance that the software has been thoroughly checked and meets specific quality standards, giving users peace of mind in using it for various applications and tasks.

ISO 9000 Certification Process

Application:

Once company has decided for certification, it applies to the registrar for the registration.

Pre-assessment:

During this stages register makes a rough assessment of the organization

Document Review and Adequecy of Audit:

During this stage, the registrar review the document and suggest an improvement.

Compliance Audit:

Registrar checks whether the organization has compiled the suggestion made by it during the review or not.

Registrations:

Registrar awards certification after completion of all phases.

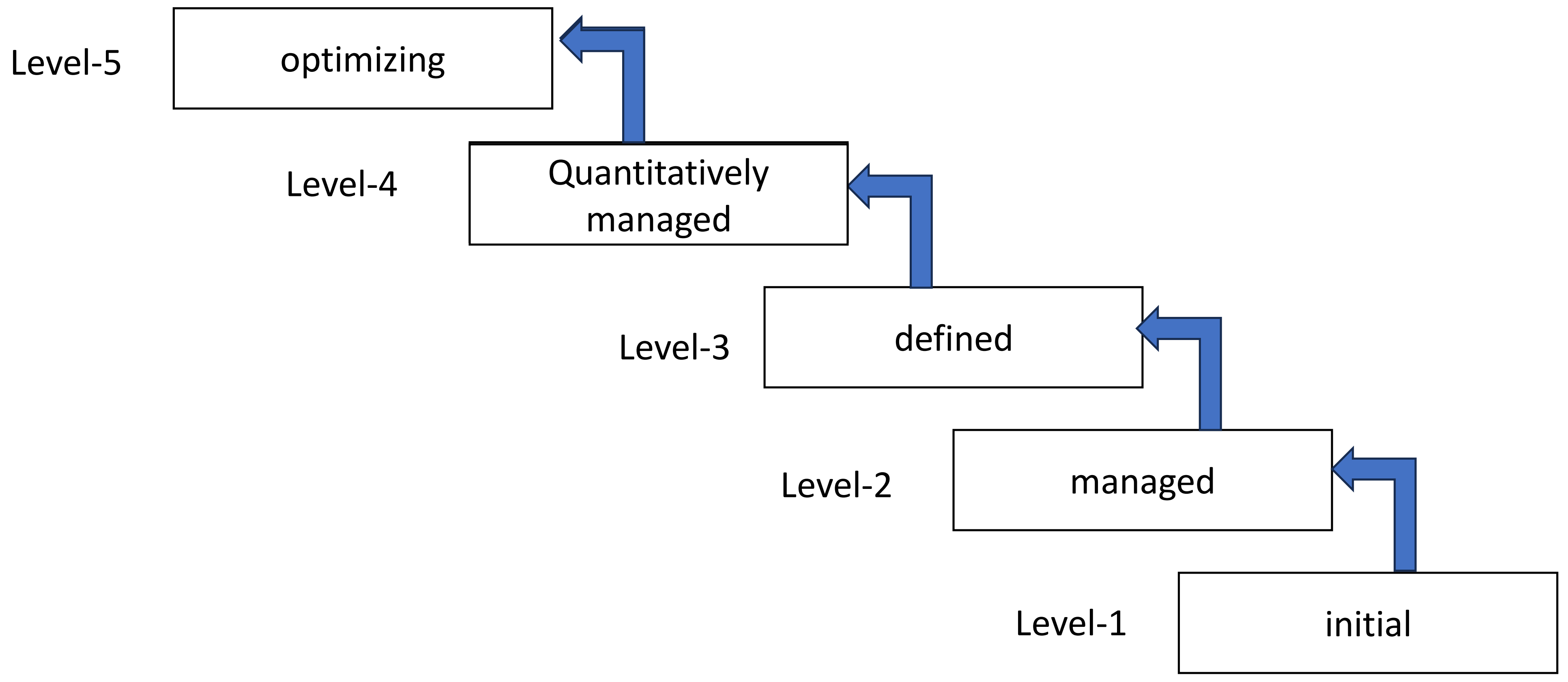
Continued Inspections:

The registrar continued to monitor the organization time to time.

Capability Maturity Model(CMM)

- CMM was developed and is promoted by the Software Engineering Institute SEI a research and development center.
- It is not a software process model. Capability Maturity Model is used as a benchmark to measure the maturity of an organizations software process.
- The model describes a five –level evolutionary path of increasingly organized and systematically more mature processes.

CAPABILITY MATURITY MODEL(CMM)



Capability Maturity Model(CMM)

Level one:Initial

- No engineering management, everything done on adhoc basis.
- Software process is unpredictable with respect to time and cost
- It depends on current staff, as staff change so does the process.

Level two:Repeatable

- Planning and managing of new projects are based on the experience with the similar projects.
- Realistic plans based on the performance based on the previous projects.

Level three: Defined(process standardization)

- Process of developing and maintaining software across the organization is documented including engineering and management.
- Training programs are implemented to ensure that the staff have skills and knowledge required.
- Risk management.

Level four: Managed(quantitative measurement)

- Organization set quantitative goals for both product and process.
- Here process is predictable both with res(pect to time and cost

Level five:Optimized(continuous process improvement)

- Here organization analysis defects to determine their causes and goals is to preventing the occurrence of defects.
- Here company continuously improve the process performance of their projects