Unit-6
Quality Assurance & Management

Topics

- Quality Concepts and Software Quality Assurance
- Software Reviews (Formal Technical Reviews)
- Software Reliability
- The Quality Standards
  - ISO 9000
  - CMM
  - Six Sigma for SE
  - SQA Plan
  - Software Maintenance and Configuration Management

What is Quality

- Quality

Developed product meets it's specification

- Quality Management

Ensuring that required level of product quality is achieved

- Defining procedures and standards
- Applying procedures and standards to the product and process
- Checking that procedures are followed
- Collecting and analysing various quality data

Software Quality Assurance (SQA)

- Software quality assurance (also called quality management) is an umbrella activity that is applied throughout the software process
- It is planned and systematic pattern of activities necessary to provide high degree of confidence in the quality
- Software quality assurance (SQA) encompasses
  - An SQA process
  - Specific quality assurance and quality control tasks
  - Effective software engineering practice
  - Control of all software work products
  - A procedure to ensure compliance with software development standards
  - Measurement and reporting mechanisms

SQA Activities

- Prepare an SQA plan for a project
- Participate in the development of the project's software process description
- Review software engineering activities to verify compliance with the defined software process
- Audit designated software work products to verify compliance with those defined as part of the software process
- Ensure that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reporting to senior management

SQA Techniques

- Statistical quality assurance implies the following steps:
  1. Information about software defects is collected and categorized
  2. An attempt is made to trace each defect to its underlying cause
     - Ex., non-conformance to specifications, design error, violation of standards, poor communication with the customer
  3. Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the "vital few").
  4. Once the vital few causes have been identified, move to correct the problems that have caused the defects.
- Some of the defects are uncovered as software is being developed.
- Other are encountered after the software has been released.

SQA Techniques (Cont.)

Uncovered errors can be tracked to (one or more) of the following causes

- Incomplete or erroneous specifications (IES)
- Misinterpretation of customer communication (MCC)
- Intentional deviation from specifications (IDS)
- Violation of programming standards (VPS)
- Error in data representation (EDR)
- Inconsistent component interface (ICI)
- Error in design logic (EDL)
- Incomplete or erroneous testing (IET)
- Inaccurate or incomplete documentation (IID)
- Error in programming language translation of design (PLT)
- Ambiguous or inconsistent human/computer interface (HCI)
- Miscellaneous (MIS)

Software Reviews (Formal Technical Reviews)

- A formal technical review (FTR) is a software quality control activity performed by software engineers (and others)
- The objectives of an FTR are:
    1. To uncover errors in function, logic, or implementation; for any representation of the software
    2. To verify that the software under review meets its requirements
    3. To ensure that the software has been represented according to predefined standards
    4. To achieve software that is developed in a uniform manner
    5. To make projects more manageable
- During the FTR,
    - A reviewer (the recorder) actively records all issues that have been raised
- These are summarized at the end of the review meeting, and a reviewed issues list is produced
- In addition, a formal technical review summary report is completed

Review Guidelines

Guidelines for conducting formal technical reviews must be established in advance, distributed to all reviewers, agreed upon & then followed

- Review the product, not the producer
- Set an agenda and maintain it
- Limit debate and denial
  Speak problem areas, but don't attempt to solve every problem noted
- Take written notes
- Conduct meaningful training for all reviewers
- Limit the number of participants and insist upon advance preparation
- Develop a checklist for each product that is likely to be reviewed
- Allocate resources and schedule time for FTRs
- Review your early reviews

Differentiation of SQA & SQC

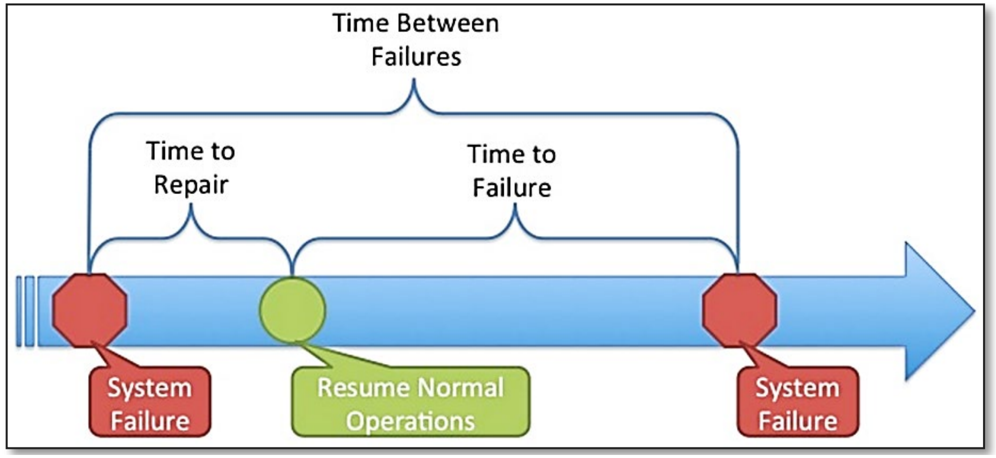
Criteria	SQA	SQC
Definition	SQA is a set of activities for ensuring quality in software engineering processes (that ultimately result in quality of software products). The activities establish and evaluate the processes that produce products	SQC is a set of activities for ensuring quality in software products. The activities focus on identifying defects in the actual products produced
Focus	Process focused	Product focused
Orientation	Prevention oriented	Detection oriented
Breadth	Organization wide	Product/project specific
Scope	Relates to all products that will ever be created by a process	Relates to specific product
Activities	Process Definition and Implementation, Audits, Training	Reviews, Testing

Software Reliability

Software reliability is defined in statistical terms as

The probability of failure-free operation of a computer program in a specified environment for a specified time

A simple measure of reliability is meantime-between-failure (MTBF):



MTBF = MTTF + MTTR

MTTF = mean-time-to-failure, MTTR = mean-time-to-repair

Measures of Reliability

- Many researchers argue that MTBF is a far more useful measure than other quality-related software metrics
- An end user is concerned with failures, not with the total defect count
- Because each defect contained within a program does not have the same failure rate, the total defect count provides little indication of the reliability of a system
- An alternative measure of reliability is failures-in-time (FIT)
  - A statistical measure of how many failures a component will have over one billion hours of operation

Software Safety

- Software safety is a software quality assurance activity
  - that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail
- If hazards can be identified early in the software process,
  - software design features can be specified that will either eliminate or control potential hazards
- A modelling and analysis process is conducted as part of software safety
- Initially, hazards are identified and categorized by criticality and risk
- Although software reliability and software safety are closely related to one another, it is important to understand the subtle difference between them
  - Software reliability uses statistical analysis to determine the likelihood that a software failure will occur
  - However, the occurrence of a failure does not necessarily result in a hazard or accident
  - Software safety examines the ways in which failures can lead to an accident

The quality standards ISO 9001

ISO 9000 and 9001
Six Sigma
CMM

- In order to bring quality in product and service, many organizations are adopting Quality Assurance System
- ISO standards are issued by the International Organization for Standardization (ISO) in Switzerland
- Proper documentation is an important part of an ISO 9001 Quality Management System.
- ISO 9001 is the quality assurance standard that applies to software engineering
- It includes, requirements that must be present for an effective quality assurance system
- ISO 9001 standard is applicable to all engineering discipline

ISO 9001 Cont.

The requirements described by ISO 9001:2000 address topics such as

- Management Responsibility
- Quality System
- Design Control
- Document
- Product Identification
- Traceability
- Inspection
- Testing
- Control of Quality Records
- Internal Quality
- Training
- Servicing
- Contract Review
- Process Control
- Data Control
- Preventive Action
- Audits

In order for a software organization to become registered to ISO 9001:2000

1. It must establish policies and procedures to address each of the requirements just noted
2. Able to demonstrate that these policies and procedures are being followed

Six Sigma

Several Software Packages available to assist in measuring yield, defects per million opportunities, etc.

- Six sigma is "A generic quantitative approach to improvement that applies to any process."
- Six Sigma is a disciplined, data-driven approach and methodology for eliminating defects in any process - from manufacturing to transactional and from product to service
- To achieve six sigma, a process must not produce more than 3.4 defects per million opportunities
  - 4 Sigma → 6210 defects per million opportunities
  - 5 Sigma → 230 defects per million opportunities
- Six sigma have two methodologies.
  - DMAIC (Define, Measure, Analyze, Improve, Control)
  - DMADV (Define, Measure, Analyze, Design, Verify)

DMAIC - Six Sigma

Define: Define the problem or process to improve upon related to the customer and goals

Measure: How can you measure this process in a systematic way?

Analyze: Analyze the process or problem and identify the way in which it can be improved. What are the root causes of problems within the process?

Improve: Once you know the causes of the problems, present solutions for them and implement them.

Control: Utilize Statistical Process Control to continuously measure your results and ensure you are improving

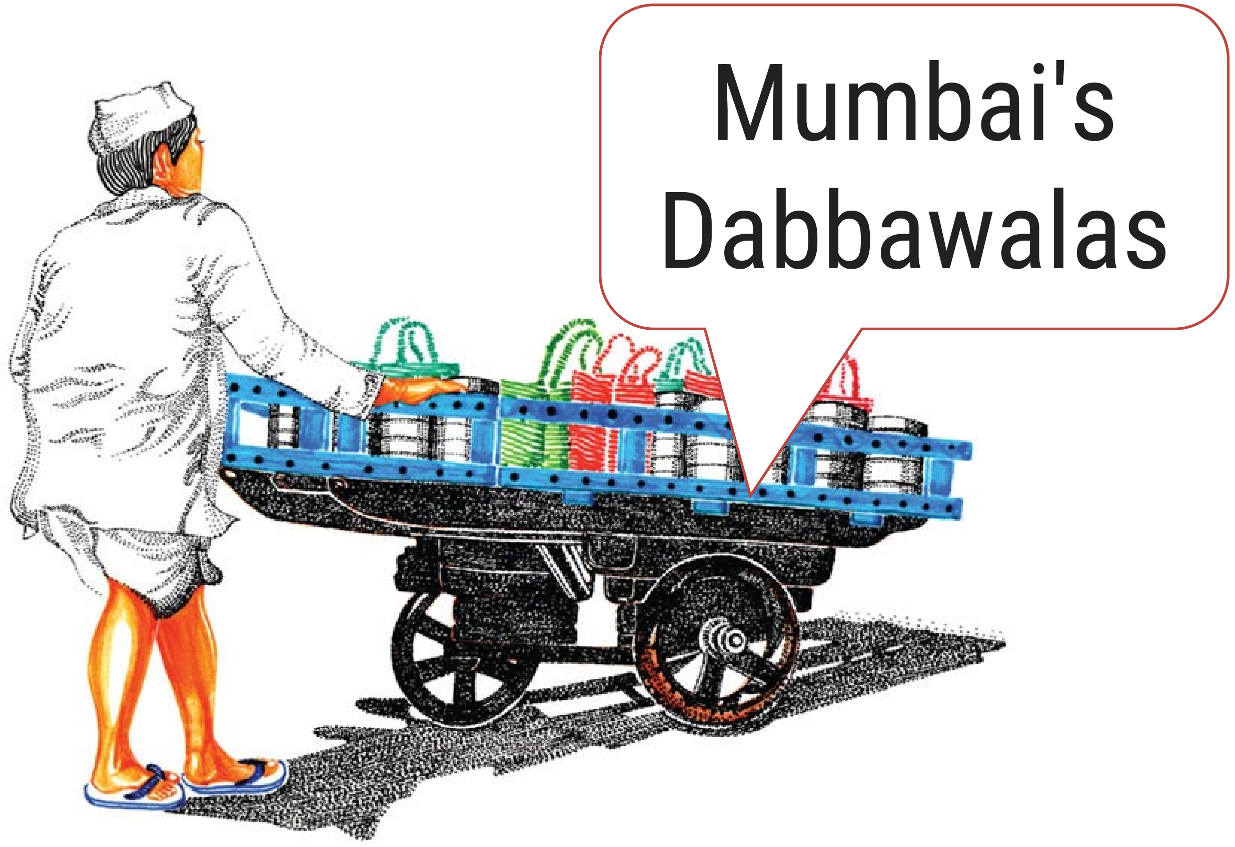
DMADV - Six Sigma

Define, Measure and analyze are similar to above method

Design: Avoid root causes of defects and meet the customer requirements

Verify: To verify the process, compare the process with the standard plan and find differences

Example of Six Sigma Company



For over 100 years they have delivered food to every part of the city, earning them a Six Sigma rating (a Forbes rating of 99.9 % which means one error in 6 per million transactions).

CMM (Capability Maturity Model)

- To determine an organization’s current state of process maturity, the SEI (Software Engineering Institute) uses an assessment that results in a five point grading scheme
- The grading scheme determines compliance with a capability maturity model (CMM) that defines key activities required at different levels of process maturity

The SEI approach establishes five process maturity levels that are defined in the following manner

Level 1: Initial

- The software process is characterized as ad hoc and occasionally
- Few processes are defined and success depends on individual effort

Level 2: Repeatable

- Basic project management processes are established to track cost, schedule, and functionality.
- The necessary process discipline is in place to repeat earlier successes on Project

Level 3: Defined

- The software process for both management and engineering activities is documented, standardized and integrated
- This level includes all characteristics defined for level 2

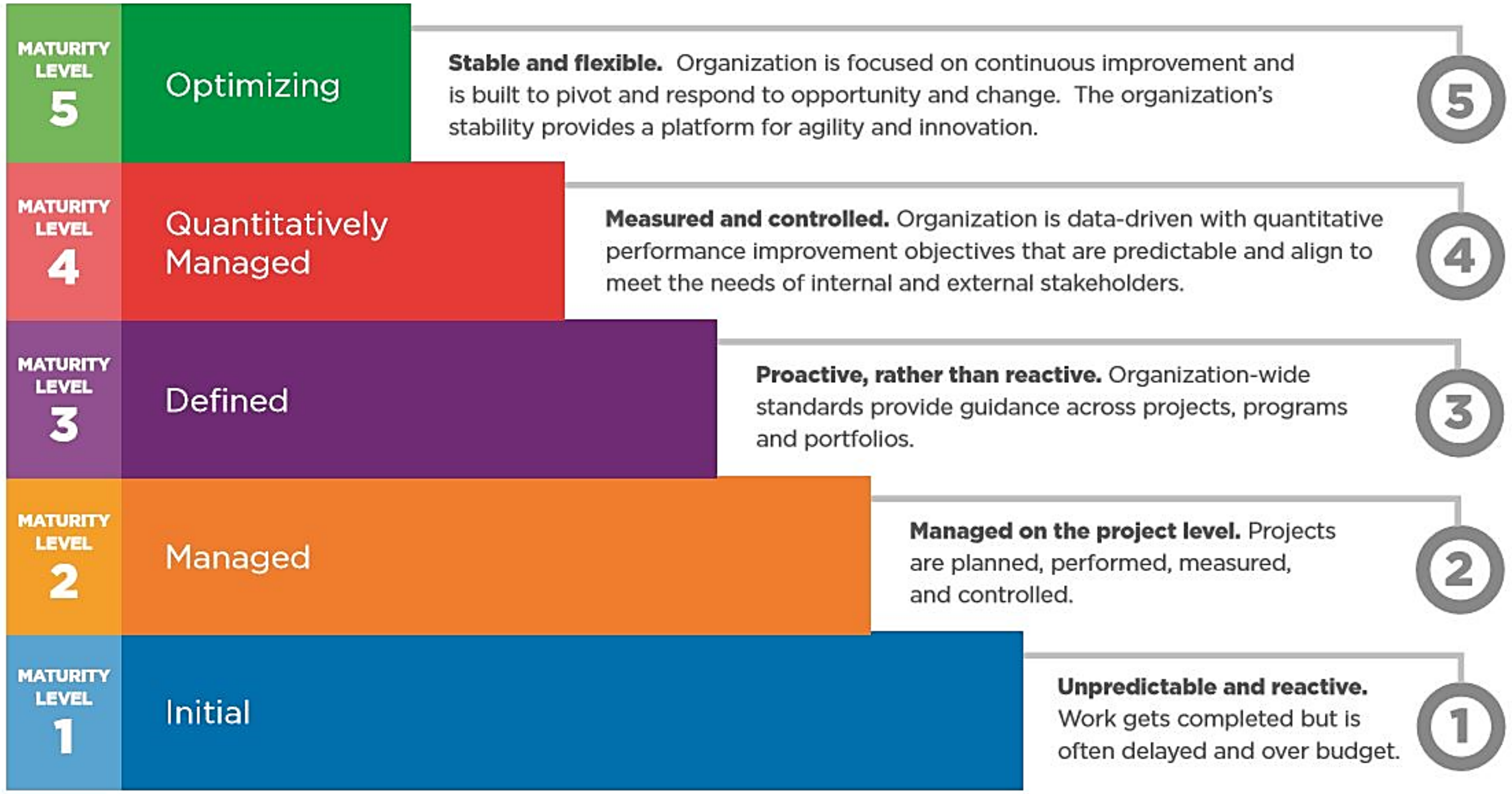
CMM (Capability Maturity Model) Cont.

Level 4: Managed

- Detailed measures of the software process and product quality are collected
- This level includes all characteristics defined for level 3

Level 5: Optimizing

- Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies
- This level includes all characteristics defined for level 4



Stable and flexible. Organization is focused on continuous improvement and is built to pivot and respond to opportunity and change. The organization's stability provides a platform for agility and innovation.

Measured and controlled. Organization is data-driven with quantitative performance improvement objectives that are predictable and align to meet the needs of internal and external stakeholders.

Proactive, rather than reactive. Organization-wide standards provide guidance across projects, programs and portfolios.

Managed on the project level. Projects are planned, performed, measured, and controlled.

Unpredictable and reactive. Work gets completed but is often delayed and over budget.

SQA Plan

- The SQA Plan provides a road map for establishing software quality assurance
- The plan serves as a template for SQA activities that are instituted for each software project
- The standard recommends a structure that identifies:
    - The purpose and scope of the plan
    - A description of all software engineering work products (e.g., models, documents, source code).
    - All applicable standards that are applied during the software process
    - SQA actions and their placement throughout the software process
    - The tools and methods that support SQA actions and tasks
    - Software configuration management procedures
    - Methods for assembling, safeguarding and maintaining all SQA-related records
    - Organizational roles and responsibilities relative to product quality

Software Maintenance and Configuration Management.

- Software maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. This can include fixing bugs, adding new features, improving performance, or updating the software to work with new hardware or software systems.
- The goal of software maintenance is to keep the software system working correctly, efficiently, and securely, and to ensure that it continues to meet the needs of the users.

There are several key aspects of software maintenance, including:

➢ Bug fixing: The process of finding and fixing errors and problems in the software.

➢ Enhancements: The process of adding new features or improving existing features to meet the evolving needs of the users.

Software Maintenance and Configuration Management.

- Performance optimization: The process of improving the speed, efficiency, and reliability of the software.
- Porting and migration: The process of adapting the software to run on new hardware or software platforms.
- Re-engineering: The process of improving the design and architecture of the software to make it more maintainable and scalable.
- Documentation: The process of creating, updating, and maintaining the documentation for the software, including user manuals, technical specifications, and design documents.

Software maintenance is a critical part of the software development life cycle and is necessary to ensure that the software continues to meet the needs of the users over time. It is also important to consider the cost and effort required for software maintenance when planning and developing a software system.

Software Maintenance and Configuration Management.

Types of Software Maintenance:

1. Corrective Maintenance

- Corrective maintenance aims to correct any remaining errors regardless of where they may cause specifications, design, coding, testing, and documentation, etc.

2. Adaptive Maintenance

- It contains modifying the software to match changes in the ever-changing environment.

3. Preventive Maintenance

- It is the process by which we prevent our system from being obsolete. It involves the concept of reengineering & reverse engineering in which an old system with old technology is re-engineered using new technology. This maintenance prevents the system from dying out.

4. Perfective Maintenance

- It defines improving processing efficiency or performance or restricting the software to enhance changeability. This may contain enhancement of existing system functionality, improvement in computational efficiency, etc.

Software Maintenance and Configuration Management.

Software Configuration Management

- When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively.
- Several individuals (programs) works together to achieve these common goals. This individual produces several work product (SC Items) e.g., Intermediate version of modules or test data used during debugging, parts of the final product.
- A configuration of the product refers not only to the product's constituent but also to a particular version of the component.
- Therefore, SCM is the discipline which
- Identify change
- Monitor and control change
- Ensure the proper implementation of change made to the item.
- Auditing and reporting on the change made.

Software Maintenance and Configuration Management.

The key objectives of SCM are to:

- Control the evolution of software systems: SCM helps to ensure that changes to a software system are properly planned, tested, and integrated into the final product.
- Enable collaboration and coordination: SCM helps teams to collaborate and coordinate their work, ensuring that changes are properly integrated and that everyone is working from the same version of the software system.
- Provide version control: SCM provides version control for software systems, enabling teams to manage and track different versions of the system and to revert to earlier versions if necessary.
- Facilitate replication and distribution: SCM helps to ensure that software systems can be easily replicated and distributed to other environments, such as test, production, and customer sites.
- SCM is a critical component of software development, and effective SCM practices can help to improve the quality and reliability of software systems, as well as increase efficiency and reduce the risk of errors.

Software Maintenance and Configuration Management.

The main advantages of SCM are:

- Improved productivity and efficiency by reducing the time and effort required to manage software changes.
- Reduced risk of errors and defects by ensuring that all changes are properly tested and validated.
- Increased collaboration and communication among team members by providing a central repository for software artifacts.
- Improved quality and stability of software systems by ensuring that all changes are properly controlled and managed.

Software Maintenance and Configuration Management.

The main disadvantages of SCM are:

- ➢Increased complexity and overhead, particularly in large software systems.
- ➢Difficulty in managing dependencies and ensuring that all changes are properly integrated.
- ➢Potential for conflicts and delays, particularly in large development teams with multiple contributors.

Thank You…