A.D. PATEL INSTITUTE OF TECHNOLOGY

(A Constituent College of CVM University)

DEPARTMENT OF INFORMATION TECHNOLOGY

Unit 7: Quality Assurance and Management

Subject: Software Engineering
Subject code: 102045602

Prepared By: Prof. Neha Chauhan

Contents

Quality Concepts

- **Today, software quality remains an issue, but who is to blame?**
  - Customers blame developers, arguing that sloppy practices lead to low-quality software.
  - Developers blame customers (and other stakeholders), arguing that irrational delivery dates and a continuing stream of changes force them to deliver software before it has been fully validated.

Software Quality

- 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.

Who does it? & Why is it important?

- Everyone- Software Engineers, Managers, all stakeholders involved in the software process is responsible for quality.
- If a software team stresses quality in all software engineering activities, it reduces the amount of rework that it must do. That results in lower costs, and more importantly, improved time to market.

Steps

- To achieve high quality software , four activities must occur.

1. Proven software engineering process & practice
2. Solid project management
3. Comprehensive quality control
4. Quality assurance infrastructure

Work product

- Software that meets its customer’s needs, performs accurately and reliably, and provides value to all who use it.
- Track quality by examining the results of all quality control activities, and measure quality by examining errors before delivery and defects released to the field.

- The transcendental view argues (like Persig) that quality is something that you immediately recognize, but cannot explicitly define.
- The user view sees quality in terms of an end-user’s specific goals. If a product meets those goals, it exhibits quality.
- The manufacturer’s view defines quality in terms of the original specification of the product. If the product conforms to the spec, it exhibits quality.

- The product view suggests that quality can be tied to inherent characteristics (e.g., 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. In reality, quality encompasses all of these views and more.

- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.
  - User satisfaction = compliant product + good quality + delivery within budget and schedule

Effective Software Process

- An **effective software process** establishes the infrastructure that supports any effort at building a high quality software product.
- The **management aspects** of process create the checks and balances that help avoid project chaos—a key contributor to poor quality.
- **SW engineering practices** allow the developer to analyze the problem and design a solid solution—both critical to building high quality software.
- Finally, **umbrella activities** such as change management and technical reviews have as much to do with quality as any other part of software engineering practice.

Useful Product

- A useful product delivers the content, functions, and features that the end-user desires.
- But as important, it delivers these assets in a reliable, error free way.
- A useful product always satisfies those requirements that have been explicitly stated by stakeholders.
- In addition, it satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

Adding Value

- 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.
- The software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- The user community gains added value because the application provides a useful capability in a way that expedites some business process.

- The end result is:
  - (1) greater software product revenue,
  - (2) better profitability when an application supports a business process, and/or
  - (3) improved availability of information that is crucial for the business.

Quality Dimensions

- David Garvin [Gar87]:
    - **Performance Quality.** Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?
    - **Feature quality.** Does the software provide features that surprise and delight first-time end-users?
    - **Reliability.** Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?

- **Conformance.** Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?
- **Durability.** Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?

- **Serviceability.** Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?
- **Aesthetics.** Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious “presence” that are hard to quantify but evident nonetheless.
- **Perception.** In some situations, you have a set of prejudices that will influence your perception of quality.

The Software Quality Dilemma

- If you produce a software system that has terrible quality, you lose because no one will want to buy it.
- If on the other hand you spend infinite time, extremely large effort, and huge sums of money to build the absolutely perfect piece of software, then it's going to take so long to complete and it will be so expensive to produce that you'll be out of business anyway.
- Either you missed the market window, or you simply exhausted all your resources.
- So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away, such as during evaluation, but also not the object of so much perfectionism and so much work that it would take too long or cost too much to complete. [Ven03]

Cost of Quality

- Prevention costs include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- Internal failure costs include
  - rework
  - repair
  - failure mode analysis
- External failure costs are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work

Achieving Software Quality

- Critical success factors:
  - **Software Engineering Methods**
  - **Project Management Techniques**
  - **Quality Control**
  - **Quality Assurance**

Software Quality Assurance

- Software quality assurance (often 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 of a product.
- 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 and the changes made to them.
  - A procedure to ensure compliance with software development standards.
  - Measurement and reporting mechanisms.

Importance of SQA

- Quality control and assurance are essential activities for any business that produces products to be used by others.
- Prior to the twentieth century, quality control was the sole responsibility of the craftsperson who built a product.
- As time passed and mass production techniques became commonplace, quality control became an activity performed by people other than the ones who built the product.
- Software quality is one of the pivotal aspects of a software development company.
- Software quality assurance starts from the beginning of a project, right from the analysis phase.

- SQA checks the adherence to software product standards, processes, and procedures.
- SQA includes the systematic process of assuring that standards and procedures are established and are followed throughout the software development life cycle and test cycle as well.
- The compliance of the built with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, project management etc.

- The major reason of involving software quality assurance in the process of software product development is to make sure that the final product built is as per the requirement specification and comply with the standards

Elements of SQA

- Standards
- Reviews and Audits
- Testing
- Error/defect collection and analysis
- Change management
- Education
- Vendor management
- Security management
- Safety
- Risk management

Role of SQA

- **Prepares an SQA plan for a project.**
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - 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
- **Participates in the development of the project's software process description.**
  - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

- **Reviews software engineering activities to verify compliance with the defined software process.**
  - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- **Audits designated software work products to verify compliance with those defined as part of the software process.**
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - 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.**
- **Records any noncompliance and reports to senior management.**
  - Noncompliance items are tracked until they are resolved.

SQA GOALS

- **Requirements quality.** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- **Design quality.** Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality.** Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness.** A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

Statistical QA

•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 (e.g., 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 per- cent 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.

- A software engineering organization collects information on defects for a period of one year.
- Some of the defects are uncovered as software is being developed.
- Others are encountered after the software has been released to its end-users. Although hundreds of different errors are uncovered, all 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.

•***Review Reporting and Record Keeping***

- 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 review 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, and then followed.
  - A review that is un-controlled can often be worse than no review at all.
  - 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.
- 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
- Conduct meaningful training for all reviewers.
- Review your early reviews.

- ***Sample-Driven Reviews***
  - In an ideal setting, every software engineering work product would undergo a formal technical review.
  - In the real word of software projects, resources are limited and time is short.
  - As a consequence, reviews are often skipped, even though their value as a quality control mechanism is recognized.

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”.

Measures of Reliability

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

- **MTBF = MTTF + MTTR**
  - Where the acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to- repair, respectively.
  - 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 modeling 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 result in conditions that can lead to an accident.

Quality Standards

- ***ISO 9001***
  - In order to bring quality in product & 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.

- The requirements delineated by ISO 9001:2000 address topics such as
  - Management responsibility
  - quality system
  - contract review
  - design control
  - document
  - data control
  - product identification
  - Traceability
  - process control
  - inspection
  - Testing
  - preventive action
  - control of quality records
  - internal quality
  - Audits
  - Training
  - Servicing
  - Statistical techniques.

- In order for a software organization to become registered to ISO 9001:2000, it must establish policies and procedures to address each of the requirements just noted (and others) and then be able to demonstrate that these policies and procedures are being followed.

Six Sigma

- 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 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.
- 5 Sigma -> 230 defects per million
- 4 Sigma -> 6210 defects per million
- Six sigma have two methodologies

(1) **DMAIC (Define, Measure, Analyze, Improve, Control)**

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

(2)DMADV: (Define, Measure, Analyze, Design, Verify)

- – 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.

CMM (Capability Maturity Model)

- To determine an organization’s current state of process maturity, the SEI 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 provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels that are defined in the following manner:

- **Level 1: Initial.** The software process is characterized as adhoc and occasionally even chaotic. 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 projects with similar applications.

- **Level 3: Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2.
- **Level 4: Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. 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.

SQA Plan

- 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.

- 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.