Chapter 6

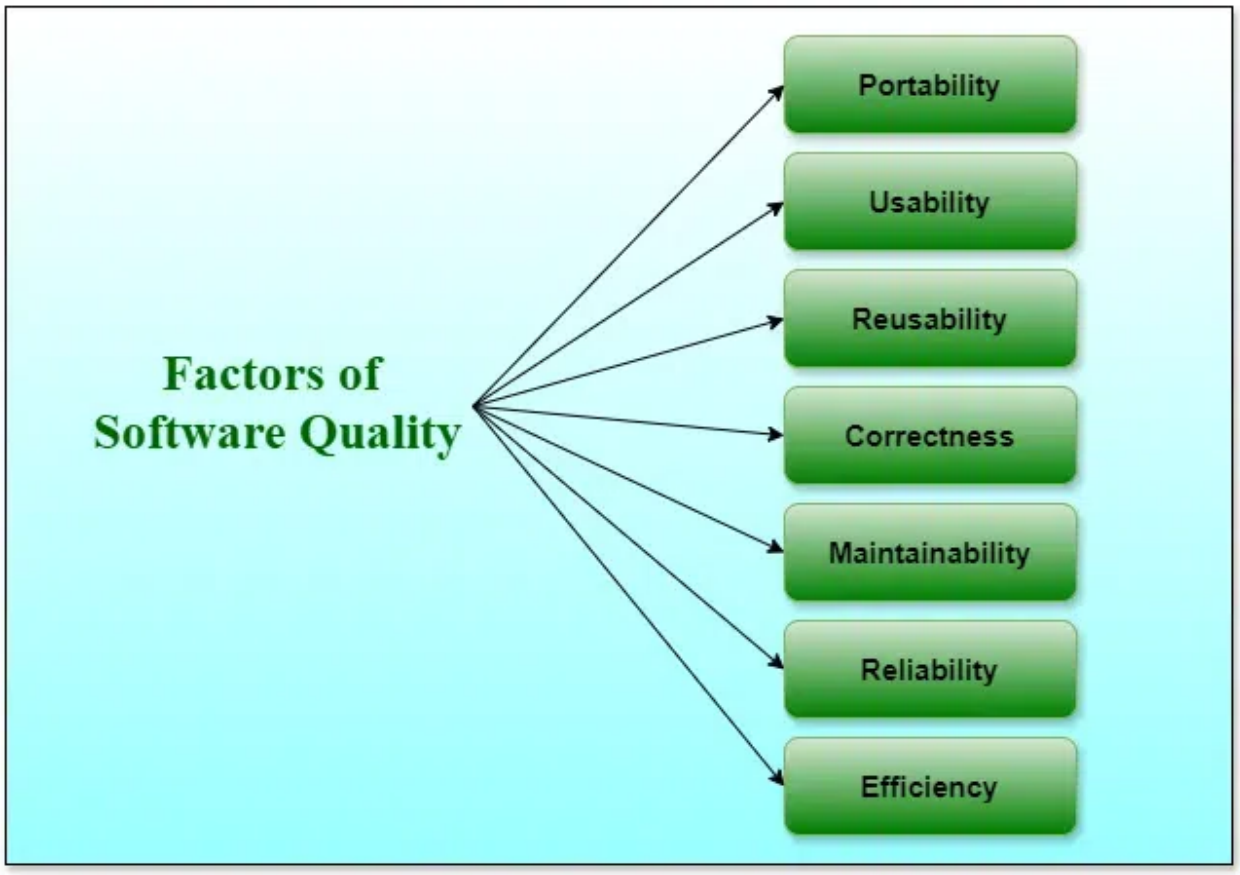
SOFTWARE QUALITY ASSURANCE PROCESS

6.1. Introduction To Software Quality:

Software Quality shows how good and reliable a product is. To convey an associate degree example, think about functionally correct software.

Factors of Software Quality

The modern read of high-quality associates with software many quality factors like the following:



1. **Portability:** A software is claimed to be transportable, if it may be simply created to figure in several package environments, in several machines, with alternative code merchandise, etc.
2. **Usability:** A software has smart usability if completely different classes of users (i.e. knowledgeable and novice users) will simply invoke the functions of the merchandise.

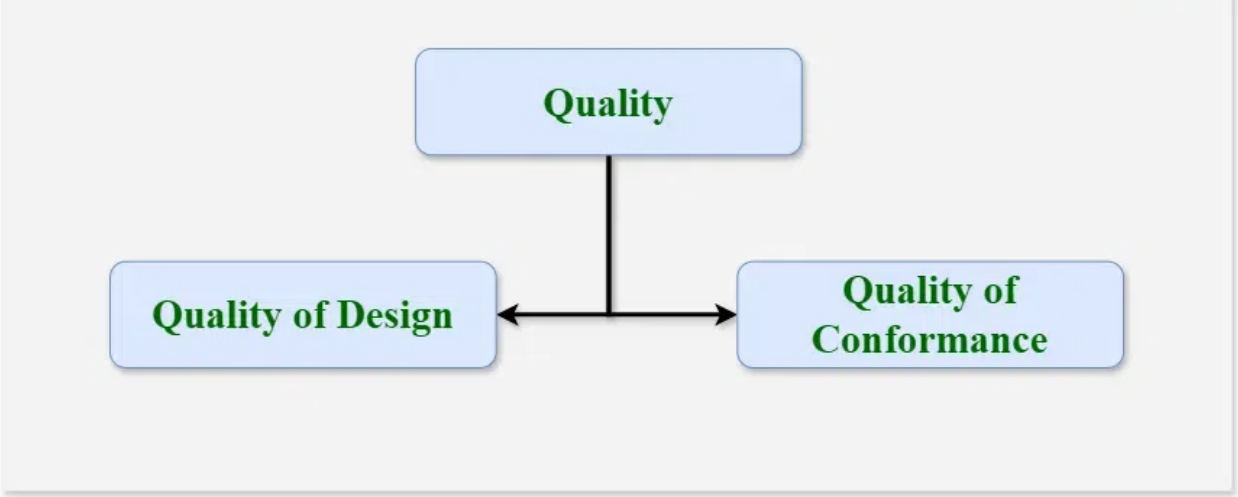
3. **Reusability:** A software has smart reusability if completely different modules of the merchandise will simply be reused to develop new merchandise.
4. **Correctness:** Software is correct if completely different needs as laid out in the SRS document are properly enforced.
5. **Maintainability:** A software is reparable, if errors may be simply corrected as and once they show up, new functions may be simply added to the merchandise, and therefore the functionalities of the merchandise may be simply changed, etc
6. **Reliability:** Software is more reliable if it has fewer failures. Since software engineers do not deliberately plan for their software to fail, reliability depends on the number and type of mistakes they make. Designers can improve reliability by ensuring the software is easy to implement and change, by testing it thoroughly, and also by ensuring that if failures occur, the system can handle them or can recover easily.
7. **Efficiency.** The more efficient software is, the less it uses of CPU-time, memory, disk space,network bandwidth, and other resources. This is important to customers in order to reduce their costs of running the software, although with today's powerful computers, CPU time, memory and disk usage are less of a concern than in years gone by.

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities that ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process that works parallel to Software Development. It focuses on improving the process of development of software so that problems can be prevented before they become major issues. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

What is quality?

Quality in a product or service can be defined by several measurable characteristics. Each of these characteristics plays a crucial role in determining the overall quality.



Software Quality Assurance (SQA) encompasse s

SQA process Specific quality assurance and quality control tasks (including technical reviews and a multitiered 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

Elements of Software Quality Assurance (SQA)

1. **Standards:** The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. 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 performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel (people employed in an organization) with the intent of ensuring that quality guidelines are being followed for software engineering work.
3. **Testing:** Software testing is a quality control function that has one primary goal—to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted for primary goal of software.
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:** Every software organization wants to improve its software engineering practices. A key contributor to improvement is education of software engineers, their managers, and other stakeholders. The SQA organization takes the lead in software process improvement which is key proponent and sponsor of educational programs.
7. **Security management:** SQA ensures that appropriate process and technology are used to achieve software security.
8. **Safety:** SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.

The Software Quality Assurance (SQA) focuses on the following

- **Software's portability:** Software's **portability** refers to its ability to be easily transferred or adapted to different environments or platforms without needing significant modifications. This ensures that the software can run efficiently across various systems, enhancing its accessibility and flexibility.
- **software's usability: Usability** of software refers to how easy and intuitive it is for users to interact with and navigate through the application. A high level of usability ensures that

users can effectively accomplish their tasks with minimal confusion or frustration, leading to a positive user experience.

- **software's reusability: Reusability** in software development involves designing components or modules that can be reused in multiple parts of the software or in different projects. This promotes efficiency and reduces development time by eliminating the need to reinvent the wheel for similar functionalities, enhancing productivity and maintainability.
- **software's correctness: Correctness** of software refers to its ability to produce the desired results under specific conditions or inputs. Correct software behaves as expected without errors or unexpected behaviors, meeting the requirements and specifications defined for its functionality.
- **software's maintainability: Maintainability** of software refers to how easily it can be modified, updated, or extended over time. Well-maintained software is structured and documented in a way that allows developers to make changes efficiently without introducing errors or compromising its stability.
- **software's error control: Error control** in software involves implementing mechanisms to detect, handle, and recover from errors or unexpected situations gracefully. Effective error control ensures that the software remains robust and reliable, minimizing disruptions to users and providing a smoother experience overall.

Software Quality Assurance (SQA) Include

1. A quality management approach.
2. Formal technical reviews.
3. Multi testing strategy.
4. Effective software engineering technology.
5. Measurement and reporting mechanism.

Major Software Quality Assurance (SQA) Activities

1. **SQA Management Plan:** Make a plan for how you will carry out the SQA throughout the project. Think about which set of software engineering activities are the best for project. check level of SQA team skills.

2. **Set The Check Points:** SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.
3. **Measure Change Impact:** The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to check the compatibility of this fix with whole project.
4. **Multi testing Strategy:** Do not depend on a single testing approach. When you have a lot of testing approaches available use them.
5. **Manage Good Relations:** In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of SQA team with programmers team will impact directly and badly on project. Don't play politics.
6. **Maintaining records and reports:** Comprehensively document and share all QA records, including test cases, defects, changes, and cycles, for stakeholder awareness and future reference.
7. **Reviews software engineering activities:** The SQA group identifies and documents the processes. The group also verifies the correctness of software product.
8. **Formalize deviation handling:** Track and document software deviations meticulously. Follow established procedures for handling variances.

Benefits of Software Quality Assurance (SQA)

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. Improving the process of creating software.
5. Improves the quality of the software.

Disadvantage of Software Quality Assurance (SQA)

There are a number of disadvantages of quality assurance.

- **Cost:** Some of them include adding more resources, which cause the more budget its not, Addition of more resources For betterment of the product.
- **Time Consuming:** Testing and Deployment of the project taking more time which cause delay in the project.

- **Overhead** : SQA processes can introduce administrative overhead, requiring documentation, reporting, and tracking of quality metrics. This additional administrative burden can sometimes outweigh the benefits, especially for smaller projects.
- **Resource Intensive** : SQA requires skilled personnel with expertise in testing methodologies, tools, and quality assurance practices. Acquiring and retaining such talent can be challenging and expensive.
- **Resistance to Change** : Some team members may resist the implementation of SQA processes, viewing them as bureaucratic or unnecessary. This resistance can hinder the adoption and effectiveness of quality assurance practices within an organization.
- **Not Foolproof** : Despite thorough testing and quality assurance efforts, software can still contain defects or vulnerabilities. SQA cannot guarantee the elimination of all bugs or issues in software products.
- **Complexity** : SQA processes can be complex, especially in large-scale projects with multiple stakeholders, dependencies, and integration points. Managing the complexity of quality assurance activities requires careful planning and coordination.

Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

Software Quality management should be independent of project management to ensure independence of cost and schedule adherences. It directly affects the process quality and indirectly affects the product quality.

Activities of Software Quality Management:

- **Quality Assurance -** QA aims at developing Organizational procedures and standards for quality at Organizational level.
- **Quality Planning -** Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.
- **Quality Control -** Ensure that best practices and standards are followed by the software development team to produce quality products.

Quality Planning in the realm of software development is a systematic process that defines the standards, procedures, and methods to be used in the creation of software products. This process is not a standalone activity but an integral part of the broader quality management framework. It encompasses various components, each playing a pivotal role in shaping the final quality of the software.

These components include:

1. **Quality Goals:** Setting clear, measurable objectives for the software's performance, functionality, and reliability.
2. **Standards and Procedures:** Establishing industry-standard practices and custom methodologies tailored to the project's needs.
3. **Resource Allocation:** Determining the necessary resources, including tools, technologies, and team expertise, to achieve the desired quality.
4. **Risk Management:** Identifying potential risks and developing strategies to mitigate them effectively.
5. **Continuous Improvement**: Implementing a feedback loop to learn from each project and continually enhance the quality planning process.

Why Is Software Quality Planning Important?

The significance of Software Quality Planning in software projects is multifaceted. At its heart, quality planning ensures that the end product not only functions as intended but also delivers a user experience that meets, if not exceeds, customer expectations. This aspect of planning is crucial in a market where customer satisfaction is paramount to success.

Real-world examples abound where effective quality planning has led to successful software projects. For instance, consider a major e-commerce platform that implemented rigorous quality planning. The result was not only a robust and user-friendly platform but also one that could handle massive transaction volumes with minimal downtime. This success is a testament to the effectiveness of well-executed quality planning.

What are the Objectives of Software Quality Planning?

Software Quality Planning (SQP) is an essential facet of software development, underpinning the success of the final product. The objectives of SQP are multifarious, each contributing to the overall efficacy and quality of the software. Here are the key objectives:

1. **Defining Quality Standards:** SQP aims to establish specific quality standards that the software must meet. These standards often align with industry benchmarks and customer expectations, ensuring that the software is not just functional but also competitive in the market.
2. **Identifying Key Quality Metrics:** A critical objective of SQP is to determine the metrics by which software quality will be measured. These metrics can include performance indicators, defect frequency, user satisfaction scores, and compliance with regulatory requirements.
3. **Resource Allocation for Quality Assurance:** Allocating the right mix of resources – including skilled personnel, tools, and time – is vital to ensure that the quality objectives are met. This strategic allocation is a crucial planning aspect to ensure efficient and effective quality assurance activities.
4. **Risk Assessment and Mitigation:** SQP involves identifying potential risks that could affect the quality of the software and developing strategies to mitigate these risks. This proactive approach helps in avoiding costly and time-consuming issues later in the development process.
5. **Ensuring Compliance with Regulatory Standards:** For many software products, especially in sectors like healthcare and finance, compliance with regulatory standards is non-negotiable. SQP plays a pivotal role in ensuring that the software adheres to all relevant laws and regulations.
6. **Continuous Improvement:** A fundamental objective of SQP is to establish a framework for continuous improvement. This involves learning from each project, gathering feedback, and making iterative improvements to the quality planning process.

Software Quality Planning Techniques

The techniques used in Software Quality Planning are as varied as the types of software projects they serve. However, some key methodologies are widely recognized for their efficacy. These include:

1. **Quality Function Deployment (QFD):** QFD is a customer-driven planning process that helps in translating customer requirements into specific engineering targets, ensuring the final product meets customer expectations.

2. **Failure Modes and Effects Analysis (FMEA):** This technique involves identifying potential failure modes within a system and the effects of these failures. It's a proactive approach to prevent issues before they occur.
3. **Six Sigma:** Six Sigma is a data-driven approach aimed at improving quality by identifying and eliminating defects and variability in processes.
4. **Total Quality Management (TQM):** TQM is a holistic approach that focuses on continuous improvement in all aspects of an organization, with the goal of enhancing quality at every stage.
5. **Agile Methodology:** Although primarily a development methodology, Agile has significant implications for quality planning. Its iterative nature allows for continuous feedback and improvement in quality.

Process and Product Standards

- Standards are essential in software engineering to ensure consistency, quality, and reliability in both the development process and the final product.
- They provide a framework for best practices and help teams work more efficiently.

Process Standards

- Standards that define the procedures, methods, and practices to be followed during the software development lifecycle (SDLC).

Purpose:

Ensure consistency across projects.

Improve quality and reduce errors.

Facilitate communication and collaboration among team members.

Examples:

ISO/IEC 12207: Standard for software lifecycle processes.

CMMI (Capability Maturity Model Integration): Framework for process improvement.

Agile Manifesto: Guidelines for Agile development practices.

Key Areas Covered:

Requirements gathering and analysis.

Design and development.

Testing and quality assurance.

Deployment and maintenance.

Benefits:

Reduces risks and improves predictability.

Enhances customer satisfaction.

Promotes continuous improvement.

Product Standards

- Standards that define the characteristics and quality of the software product.

Purpose:

Ensure the product meets specified requirements.

Improve usability, reliability, and maintainability.

Examples:

ISO/IEC 25010: Standard for software product quality.

IEEE 829: Standard for software test documentation.

MISRA C: Coding standards for safety-critical systems.

Key Areas Covered:

Code quality and style.

Documentation standards.

User interface design.

Performance and security requirements.

Benefits:

Ensures consistency in the product.

Facilitates easier maintenance and updates.

Enhances user experience.

6.5. Capability Assessment and Process Improvement

- Capability assessment and process improvement are critical components of software engineering that focus on evaluating and enhancing the effectiveness of software development processes.
- These practices help organizations deliver high-quality software, meet customer expectations, and achieve business goals.

1. Capability Assessment

Capability assessment involves evaluating the maturity and effectiveness of an organization's software development processes. It helps identify strengths, weaknesses, and areas for improvement.

1.1 Purpose of Capability Assessment

To evaluate the current state of software processes.

To identify gaps and areas for improvement.

To benchmark against industry standards and best practices.

To ensure alignment with organizational goals.

Key Models for Capability Assessment

CMMI (Capability Maturity Model Integration):

- A framework for process improvement that integrates best practices from software engineering, systems engineering, and product development.

Levels of Maturity:

Level 1: Initial - Processes are ad hoc and unpredictable.

Level 2: Managed - Processes are planned and documented.

Level 3: Defined - Processes are standardized across the organization.

Level 4: Quantitatively Managed - Processes are measured and controlled.

Level 5: Optimizing - Continuous process improvement is practiced.

Benefits:

- Improves process predictability and quality.
- Reduces risks and costs.

ISO/IEC 15504 (SPICE - Software Process Improvement and Capability Determination):

- A framework for assessing and improving software processes.
- Focuses on process capability levels (0 to 5) and process attributes (e.g., performance, management, and innovation).

Benefits:

Provides a standardized approach to process assessment.

Supports international best practices.

TMMi (Test Maturity Model Integration):

A framework for assessing and improving testing processes.

Focuses on maturity levels similar to CMMI.

Benefits:

Enhances the effectiveness and efficiency of testing processes.

Steps in Capability Assessment

Define Objectives:

Identify the purpose and scope of the assessment.

Select a Model:

Choose an appropriate framework (e.g., CMMI, SPICE).

Gather Data:

Collect information about current processes through interviews, surveys, and document reviews.

Evaluate Processes:

Compare current processes against the selected model.

Identify Gaps:

Highlight areas where processes fall short of desired levels.

Report Findings:

Document assessment results and recommendations.

Plan Improvements:

Develop a roadmap for addressing identified gaps.

2. Process Improvement

- Process improvement focuses on enhancing the efficiency, effectiveness, and quality of software development processes. It is a continuous effort to align processes with organizational goals and industry standards.

2.1 Importance of Process Improvement

- Improves product quality and customer satisfaction.
- Reduces costs and development time.
- Enhances team productivity and morale.
- Ensures compliance with industry standards.

Key Approaches to Process Improvement

PDCA (Plan-Do-Check-Act):

A cyclical approach for continuous improvement.

Steps:

Plan: Identify problems and plan improvements.

Do: Implement changes on a small scale.

Check: Evaluate the results.

Act: Standardize successful changes or revise the plan.

Six Sigma:

A data-driven approach to eliminate defects and reduce variability.

Focuses on DMAIC (Define, Measure, Analyze, Improve, Control).

Benefits:

- Improves process efficiency and quality.
- Reduces waste and costs.
- Agile Process Improvement:
- Focuses on iterative and incremental improvements.
- Encourages collaboration, feedback, and adaptability.

Steps in Process Improvement

- **Identify Areas for Improvement:**
- Use capability assessment results or feedback from stakeholders.

- **Set Goals:**
- Define clear, measurable objectives for improvement.

Develop an Action Plan:

- Outline steps, resources, and timelines for implementing changes.

Implement Changes:

- Execute the action plan and monitor progress.

Evaluate Results:

- Measure the impact of changes and compare them to goals.

Standardize Improvements:

- Incorporate successful changes into standard processes.

Repeat:

- Continuously identify new areas for improvement.

6.6 Reviews and Inspections

- Reviews and inspections are critical activities in software engineering to identify defects, improve quality, and ensure compliance with standards.
- They involve examining software artifacts (e.g., requirements, design, code) to detect issues early in the development process.

Reviews

- A formal or informal evaluation process where software artifacts are examined by a team to identify defects and improvements.

Types of Reviews:

Informal Reviews:

Casual discussions or walkthroughs.

No formal documentation or process.

Formal Reviews:

Structured and documented.

Follow a defined process (e.g., Fagan Inspection).

Steps in a Formal Review:

Planning: Define objectives, scope, and participants.

Preparation: Distribute documents and prepare for the review.

Meeting: Discuss and identify defects.

Rework: Address identified issues.

Follow-up: Verify that issues have been resolved.

Benefits:

Early detection of defects.

Improved communication and collaboration.

Ensures adherence to standards.

6.6.2 Inspections

- A rigorous and structured review process focused on identifying defects in software artifacts.

Key Characteristics:

Led by a trained moderator.

Involves a checklist of common defects.

Focuses on specific artifacts (e.g., requirements, design, code).

Steps in an Inspection:

Planning: Define scope, objectives, and participants.

Overview: Provide context and background.

Preparation: Reviewers examine the artifact independently.

Inspection Meeting: Discuss and log defects.

Rework: Fix identified issues.

Follow-up: Verify fixes and ensure quality.

Benefits:

High defect detection rate.

Improves product quality and reduces rework.

Ensures compliance with standards.

6.6.3 **Differences Between Reviews and Inspections**

Aspect	Reviews	Inspections
Formality	Can be informal or formal.	Always formal and structured.
Focus	General evaluation and feedback.	Defect detection and compliance.
Process	Flexible and adaptable.	Rigorous and follows a strict process.
Participants	Can include anyone involved.	Trained reviewers and a moderator.
Documentation	May or may not be documented.	Fully documented and tracked.