CHAPTER 6
Software Quality Assurance

Quality

Quality is a complex and multifaceted concept that can be described from five different points of view:

- **Transcendental view:** something that immediately recognizes, but cannot explicitly define.
- **User view:** If a product meets an end user's specific goals, it exhibits quality.
- **Manufacturer's view:** Product conforms to the original specification.
- **Product view:** Quality can be tied to inherent characteristics (e.g., functions and features) of a product.
- **Value-based view:** How much a customer is willing to pay for a product?

Quality

- quality encompasses all of these views and more.
- **user satisfaction = compliant product + good quality + delivery within budget and schedule**

Software Quality

- *"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 points:

- **Effective software process:** establishes the infrastructure that supports any effort at building a high-quality software product.
- **Useful product:** delivers the content, functions, and features that the end user desires in a reliable, error-free way.
- **Measurable values:** For,

producer:

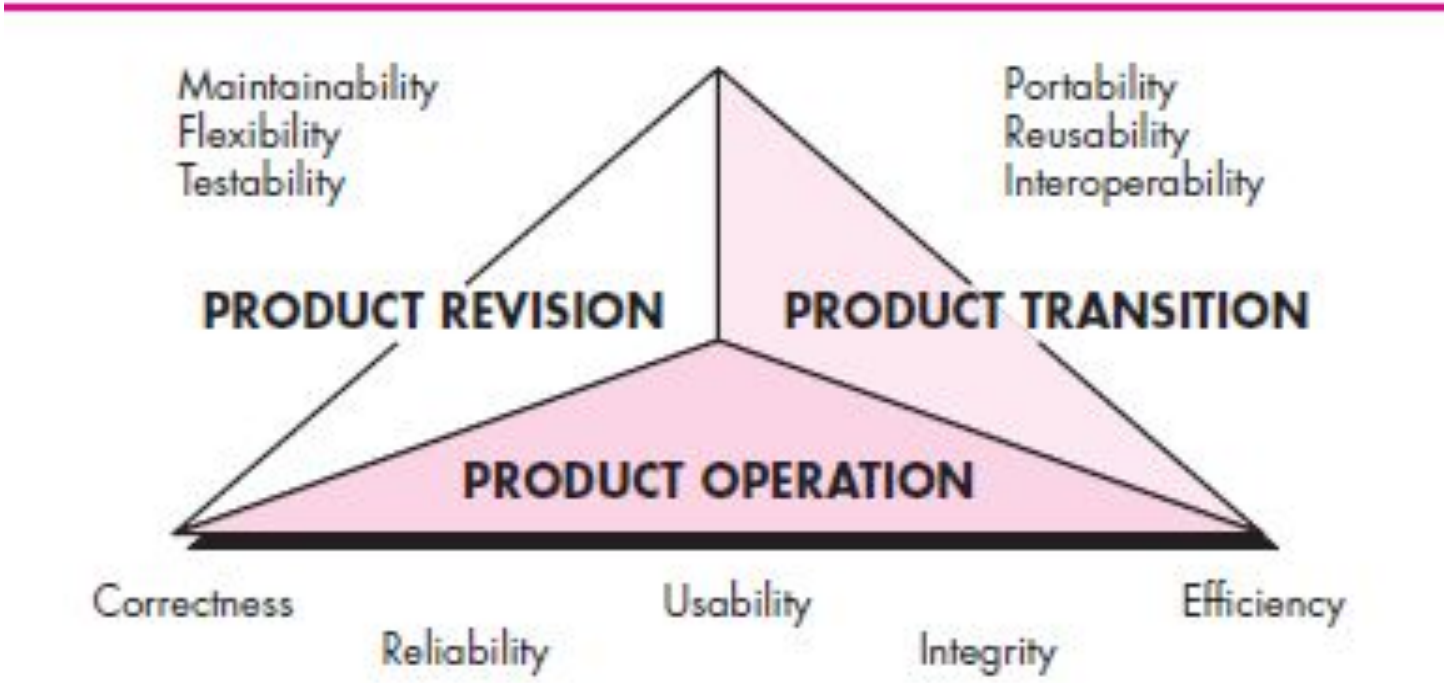
High-quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.

end user:

(1) greater software product revenue

(2) better profitability when an application supports a business process

McCall's software quality factors



- ***Correctness****:* The extent to which a program satisfies its specification and fulfills the **customer's mission objectives.**
- ***Reliability.*** The extent to which a program can be expected to perform its **intended function** with required precision
- ***Efficiency****.* The amount of computing resources and code required by a program to perform its function.
- ***Integrity****.* Extent to which access to software or data by unauthorized persons can be controlled.

- ***Usability****:* Effort required to learn, operate, prepare input for, and interpret output of a program.
- ***Maintainability:*** Effort required to locate and fix an error in a program.

- ***Flexibility*** *:*Effort required to modify an operational program.
  Testability: Effort required to test a program to ensure that it performs its intended function.
- ***Portability:*** Effort required to transfer the program from one hardware and/or software system environment to another.

- ***Reusability:*** Extent to which a program [or parts of a program] can be reused in other applications.
- ***Interoperability****:* Effort required to couple one system to another.

Quality Factors

- 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:

ISO 9126

- **Functionality**
- **Reliability**
- **Usability**
- **Efficiency.**
- **Maintainability.**
- **Portability.**

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 sum 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.

Software Quality Dilemma

- 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.

Software quality attributes

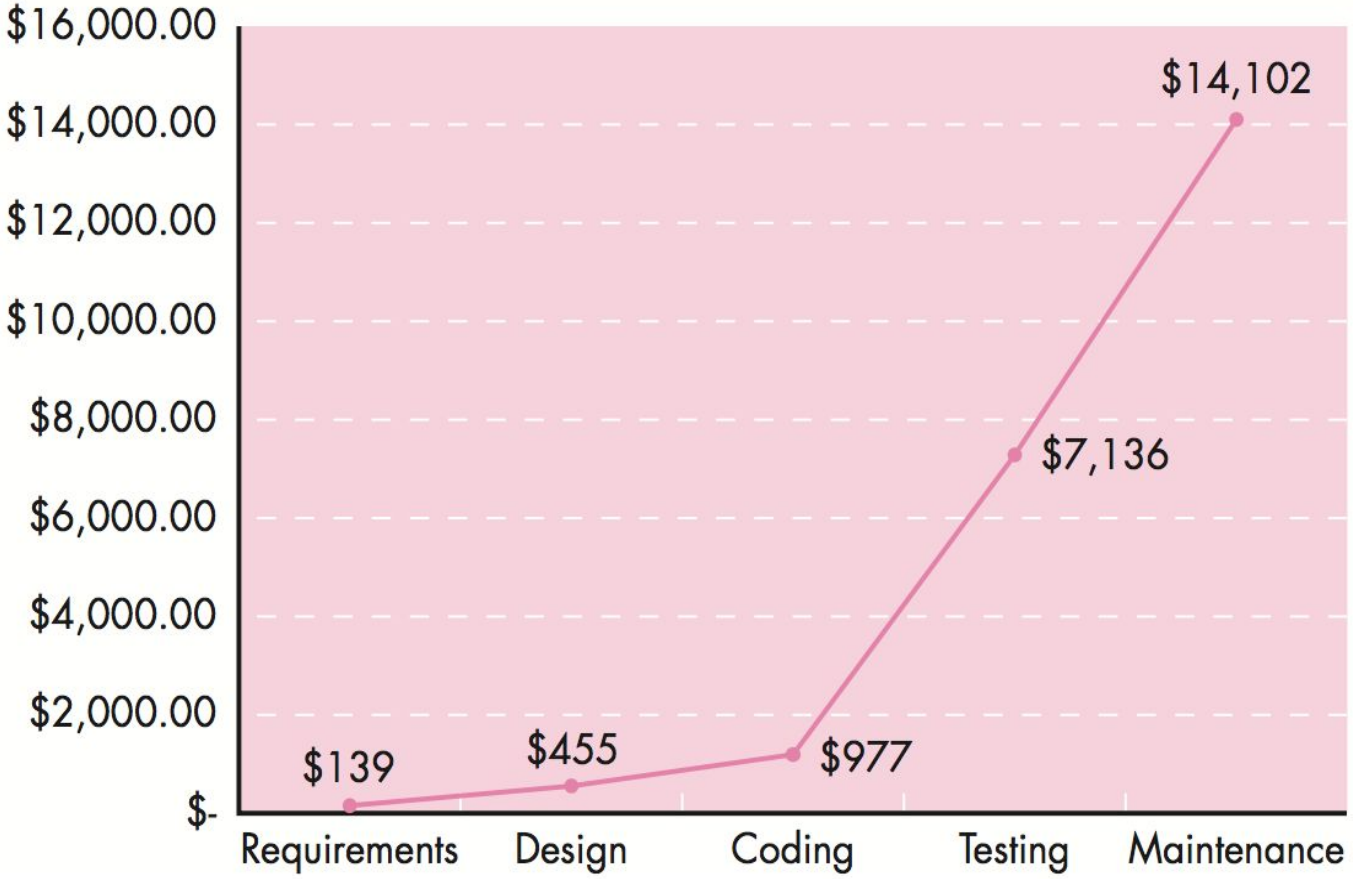
- Integrity : security , how system handles unwanted threats .
- Flexibility : Easy to change, how flexible to change
- Interoperability : how effectively communicating one system to another system.


Speed
Quality
Features

Good Enough Software

- Good enough software delivers high-quality functions and features that users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.
- The software vendor hopes that the vast majority of end users will overlook the bugs because they are so happy with other application functionality.

Cost of Quality



Cost of Quality

- The industry average cost to correct a defect during code generation is approxi- mately $977 per error.
- The industry average cost to correct the same error if it is discovered during system testing is $7,136 per error

Cost of Quality

- The cost of quality can be divided into costs associated with **prevention, appraisal, and failure.**

  Prevention costs include:

(1) the cost of management activities required to plan and coordinate all quality control and quality assurance activities,

(2) the cost of added technical activities to develop complete requirements and design models,

(3) test planning costs, and

(4) the cost of all training associated with these activities.

Cost of Quality

Appraisal costs: include activities to gain insight into product condition the "first time through" each process. E.g

- ✔ *Cost of conducting technical reviews*
- ✔ *Cost of data collection and metrics evaluation*
- ✔ *Cost of testing and debugging*

Failure*: costs* are those that would disappear if no errors appeared before or after shipping a product to customers

- Failure costs may be subdivided into internal failure costs and external failure costs
- *Internal failure costs* are incurred when you detect an error in a product prior to shipment

Cost of Quality

- *External failure costs* are associated with defects found after the product has been shipped to the customer.

Quality Control

- Quality control encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals.
- Models are reviewed to ensure that they are complete and consistent.
- Code may be inspected in order to uncover and correct errors before testing commences.

- A series of testing steps is applied to uncover errors in processing logic, data manipulation, and interface communication.
- A combination of measurement and feedback allows a software team to tune the process when any of these work products fail to meet quality goal

Formal Technical Review

- 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; and

(5) to make projects more manageable.

- In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation.
- The FTR also serves to promote backup and continuity because a number of people become familiar with parts of the software that they may not have otherwise seen

Formal Technical Review

- The FTR is actually a class of reviews that includes *walkthroughs* and *inspections.*
- Each FTR is conducted as a meeting and will be successful only if it is properly planned, controlled, and attended

The Review Meeting

- Between three and five people (typically) should be involved in the review.
- Advance preparation should occur but should require no more than two hours of work for each person.
- The duration of the review meeting should be less than two hours.

Formal Technical Review

- **The Players of Review Meeting**

Producer—the individual who has developed the work product
  - Informs the project leader that the work product is complete and that a review is required.
- **Review leader**—evaluates the product for readiness, generates copies of product materials, and distributes them to two or three reviewers for advance preparation.
- **Reviewer(s)**—expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work.
- **Recorder**— a reviewer who records (in writing) all important issues raised during the review.

Formal Technical Review

Review Summary Report

- What was reviewed?
- Who reviewed it?
- What were the findings and conclusions?

FTR Guidelines

1. *Review the product, not the producer*
2. *Set an agenda and maintain it.*
3. *Limit debate and rebuttal*
4. *Enunciate problem areas, but don't attempt to solve every problem noted*

FTR Guidelines

5.Take written notes.

6.Limit the number of participants and insist upon advance preparation

7.Develop a checklist for each product that is likely to be reviewed

8.Allocate resources and schedule time for FTRs

9.Conduct meaningful training for all reviewers

*10.**Review your early reviews***

Elements of software Quality Assurance

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

- **Standards. T**he IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents

- **Reviews and audits.** Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors

- **Testing.** Software testing is a quality control function that has one primary goal to find errors.

Elements of Software Quality Assurance

- **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
- **Change Management**: If it is not properly managed, change can lead to con-fusion, and confusion almost always leads to poor quality.
- **Education:** The SQA organization takes the lead in software process improvement and is a key proponent and sponsor of educational programs.
- **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

- **Security Management:** SQA must ensures that appropriate process and technology are used to achieve software security.
- **Safety:** SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
- **Risk Management:** SQA organization must ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

SQA task, Goals and Metric

- **Prepares an SQA plan for a project :**
- **Participates in the development of the project's software process description.**

SQA task, Goals and Metric

- **Reviews software engineering activities to verify compliance with the defined software process.**
- **Audits designated software work products to verify compliance with those defined as part of the software process**

SQA task Goals and Matric

- **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.**

Statistical Software quality assurance

- Statistical quality assurance reflects a growing trend throughout industry to become more **quantitative** about quality
- For software, statistical quality assurance implies the following steps:

Statistical Software quality assurance

1. Information about software errors and defects is collected and categorized.
2. An attempt is made to trace each error and defect to its underlying cause (e.g., nonconformance 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 errors and defects.

Six Sigma for Software Engineering

- *Six Sigma* is the most widely used strategy for statistical quality assurance in industry today
- Originally popularized by Motorola in the 1980s, the Six Sigma strategy "is a rigorous and disciplined methodology that uses data and statistical analysis.

Six Sigma for Software Engineering

The Six Sigma method-ology defines three core steps:

- ***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
- *Analyze* defect metrics and determine the vital few causes.

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"
- To illustrate, program *X* is estimated to have a reliability of 0.999 over eight elapsed processing hours.
- In other words, if program *X* were to be executed 1000 times and require a total of eight hours of elapsed processing time (execution time), it is likely to operate correctly (without failure) 999 times.

Measure of Reliability and Availability

- If we consider a computer-based system, a simple measure of reliability is ***mean-time-between-failure*** **(MTBF):**
- MTBF = MTTF + MTTR
  where the acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to Repair*.

Measure of Reliability and Availability

- In addition to a reliability measure, you should also develop a measure of availability.
- *Software availability* is the probability that a program is operating according to requirements at a given point in time and is defined as
- Availability= (MTTF)/(MTTF+ MTTR) *100
- The MTBF reliability measure is equally sensitive to MTTF and MTTR.
- The avail- ability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.

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.

THE SQA PLAN

- The *SQA Plan* provides a **road map** for establishing 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 established for each software project.

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. Description of all software engineering work products (e.g., models, documents, source code)

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

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

THE SQA PLAN

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

7. Software configuration management procedures

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

9. Organizational roles and responsibilities relative to product quality.

International Standard Organization(ISO)

- The ISO 9000 standards specifies the guidelines for maintaining a quality system.
- ISO 9000 is a series of three standards: ISO 9001, ISO 9002, and ISO 9003
- The ISO 9000 series of standards is based on the premise that if a proper process is followed for **production**, the good quality products are bound to **automatically** follow.

ISO

- The types of software industries to which the different ISO standards apply are as follows:
- ISO 9001: The standard applies to the organization engaged in the design, development, production, and serving of the goods.
- This is the standard that is applicable to most software development organization.

ISO

- ISO 9002 : This standard applies to the organizations that do not design the product but are only involved in the production.
- Example: steel or car manufacturing industries who buy the product plant design and involve in only manufacturing

ISO

- ISO 9003: Applied for those organizations which involved only in installation and testing of the products

Why to get ISO certification

- Confidence of customers in an organization is enhanced
- ISO 9000 requires a well documented software production process which **contributes to higher quality and repeatable developed software** .
- Makes the development process focused, efficient and cost effective
- Points out the weak points of an organizations and recommends remedial actions.
- Sets the basic framework for development of an optimal process.

How to get ISO Certifications

- An organizations intending to obtain the ISO certifications applies to an ISO 9000 registrar for registration.
- The ISO 9000 registration process consists of the following stages :
- Application Stage:
- Pre-assessment
- Document review and adequacy audit
- Compliance Audit
- Registration
- Continued Surveillance

Capability Maturity Model(CMMM)

- The intent of the maturity model is to provide an overall indication of the **“process maturity” exhibited by a software organization**.
- That is, an indication of the **quality of the software process**, **the degree to which practitioner’s understand and apply the process**, and **the general state of software engineering practice**

- Capability Maturity Model suggests five levels of maturity

  Level 1, Initial

- Processes are disorganized, ad hoc and even chaotic.
- Success likely depends on individual efforts and is not considered to be repeatable

Level 2, Repeatable

- Requisite processes are established, defined and documented.
- Basic project management processes are established to track cost, schedule, and functionality.
- requisite processes are established, defined and documented Planning and managing new products is based on experience with similar projects

- **Level 3, Defined**
- An organization develops its own standard software development process.
- These defined processes enable greater attention to documentation, standardization and integration.
- All projects use an approved, tailored version of the organization’s standard software process for developing software.

- **Level 4, Managed**
- At the managed level, an organization monitors and controls its own processes through data collection and analysis.
- Meaningful variations in process performance can be distinguished from random noise, and trends in process and product qualities can be predicted.

- **Level 5, Optimized**
- Processes are constantly improved through processes are constantly improved through monitoring feedback from processes and introducing innovative processes and functionality.
- The organization has quantitative feedback systems in place to identify process weaknesses and strengthen them pro-actively.
- Project teams analyze defects to determine their causes; software processes are evaluated and updated to prevent known types of defects from recurring.

5 levels of the Capability Maturity Model

LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4	LEVEL 5
Initial	**Repeatable**	**Defined**	**Managed**	**Optimizing**
Software development processes are disorganized.	Processes are defined and documented.	Processes are standardized.	Processes are monitored and controlled.	Processes are continuously improved.