MODULE 6: QUALITY MANAGEMENT

-by
Asst Prof Rohini Sawant

SOFTWARE QUALITY

- Crosby defines quality as conformance to requirements. This means that the product should be developed according to the pre-specified requirements. If the requirements are misunderstood and the product is developed with incorrect requirements, then the product lacks quality.
- Juran & Gryna defines quality as fitness for use. It means whether the product is actually usable by the user or not. If the user is satisfied with the product, it is of good quality.
- This shows that quality is not a single idea; rather it is a multi-dimensional concept.
- On the basis of the multi-dimensional concept, quality can be defined as 'the degree to which a product or service possesses a desired combination of attributes'.
- We know that software is not similar to any physical product, so the concept of software quality also differs from physical products. Software quality is not easily definable. Software has many possible quality characteristics.
- In the narrowest sense, software quality revolves around defects. It is commonly recognized as lack of bugs in the software. This definition is the same as the classical definition of quality—'conformance to requirements'.

SOFTWARE QUALITY

- In the early days of computers, software developers mainly focused on product functionalities, and most of the end users were highly qualified professionals, such as mathematicians, scientists, and engineers.
- Development of personal computers and advances in computer networks, the World Wide Web, and graphical user interface made computer software highly accessible to all kinds of users.
- There has been increasing customer expectations in terms of better quality in software products, and developers are under tremendous pressure to deliver high-quality products at a lower cost.
- Even though competing products deliver the same functionalities, it is the lower cost products with better quality attributes that survive in the competitive market.
- Therefore, all stakeholders—users, customers, developers, testers, and managers—in a product must have a broad understanding of the overall concept of software quality

SOFTWARE QUALITY

The following five viewpoints help us in understanding different aspects of the quality concept:

- Transcendental View: In the transcendental view quality is something that can be recognized through experience but is not defined in some tractable form. Quality is viewed to be something ideal, which is too complex to lend itself to be precisely defined. However, a good-quality object stands out, and it is easily recognized. Because of the philosophical nature of the transcendental view, no effort is made to express it using concrete measures.
- User View: The user view concerns the extent to which a product meets user needs and expectations. Quality is not just viewed in terms of what a product can deliver, but it is also influenced by the service provisions in the sales contract. In this view, a user is concerned with whether or not a product is fit for use.

SOFTWARE QUALITY

- User View: This view is highly personalized in nature. Because of the personalized nature of the product view, a product is considered to be of good quality if it satisfies the needs of a large number of customers.
- Value-Based View: The value-based view represents a merger of two independent concepts: excellence and worth. Quality is a measure of excellence, and value is a measure of worth. The central idea in the value-based view is how much a customer is willing to pay for a certain level of quality. The reality is that quality is meaningless if a product does not make economic sense. Essentially, the value-based view represents a trade-off between cost and quality

SOFTWARE QUALITY

- Manufacturing View: The manufacturing view has its genesis in the manufacturing sectors, such as the automobile and electronics sectors. In this view, quality is seen as conforming to requirements. Any deviation from the stated requirements is seen as reducing the quality of the product. The concept of process plays a key role in the manufacturing view. However, there is no guarantee that conforming to process standards will lead to good product. However, product quality can be incrementally enhanced by continuously improving the process.
- Product View: The central hypothesis in the product view is this: If a product is manufactured with good internal properties, then it will have good external qualities. The product view is attractive because it gives rise to an opportunity to explore causal relationships between internal properties and external qualities of a product. In this view, the current quality level of a product indicates the presence or absence of measurable product properties. An example of the product view of software quality is that high degree of modularity, which is an internal property, makes a software testable and maintainable.

Software quality management

- Concerned with ensuring that the required level of quality is achieved in a software product.
- Involves defining appropriate quality standards and procedures and ensuring that these are followed.
- Should aim to develop a 'quality culture' where quality is seen as everyone's responsibility.

What is quality?

- Quality, simplistically, means that a product should meet its specification.
- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.

Scope of quality management

- Quality management is particularly important for large, complex systems. The quality documentation is a record of progress and supports continuity of development as the development team changes.
- For smaller systems, quality management needs less documentation and should focus on establishing a quality culture.

Quality management activities

- Quality assurance
  - Establish organisational procedures and standards for quality.
- Quality planning
  - Select applicable procedures and standards for a particular project and modify these as required.
- Quality control
  - Ensure that procedures and standards are followed by the software development team.
- Quality management should be separate from project management to ensure independence.

QUALITY MANAGEMENT

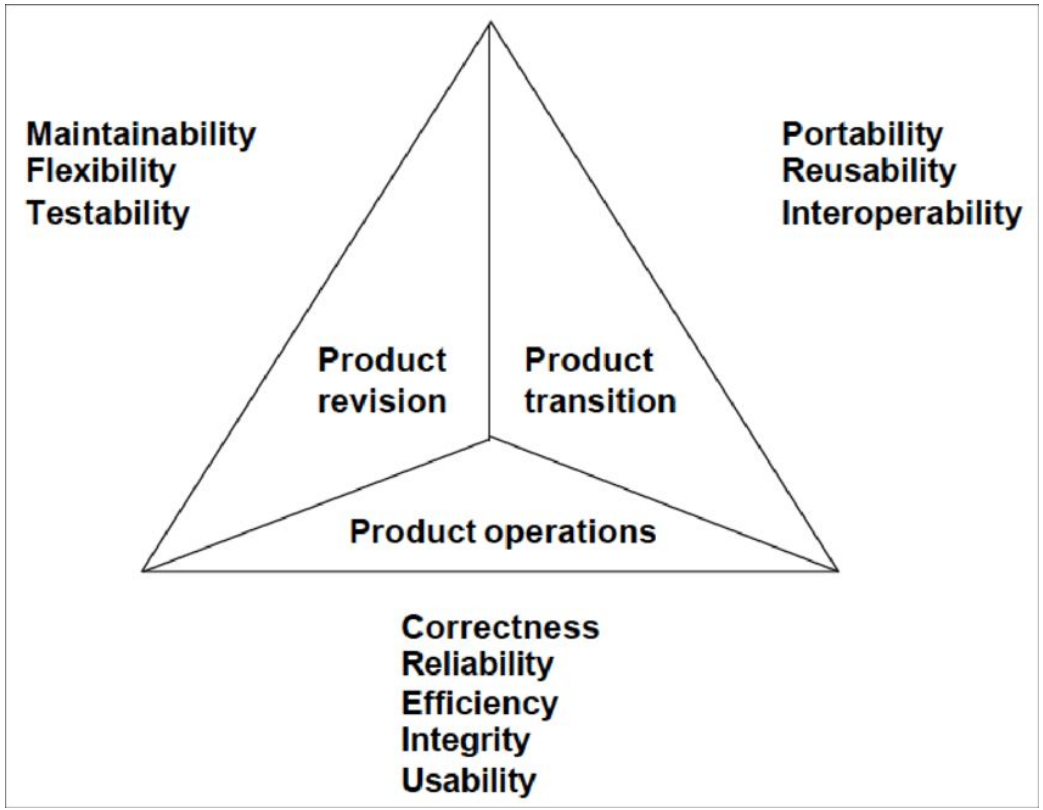
- We have discussed that quality assurance is a management activity performed by managers.
- At the managerial level, quality needs to be planned on a larger scale and all controlling and assurance activities are performed according to quality plans.
- The managerial root has given the term ‘quality management’ whose scope is larger than quality control and quality assurance. It is now an established way of managing the quality of a product.
- The task of quality management is to
- „ plan suitable quality control and quality assurance activities.
- „ define procedures and standards which should be used during software development and verify that these are being followed by everyone.
- „ properly execute and control activities.
- If quality at some point is not under control, then it is the responsibility of quality managers to manage the resources such that the required level of quality is achieved.

But there is more to quality management. The major role of QM is to develop a *quality culture* in the organization. Quality culture means that every member of the team is aware and conscious about the quality and is working towards a high-quality end-product. It is the responsibility of quality managers to encourage the team members to take responsibility for their work, not only to function correctly but also for improving the quality (in the sense of defects). Though QM specifies the quality specifications and standards, yet there are some aspects which cannot be standardized, e.g. elegance, readability, etc. are intangible aspects of software quality. Quality managers also encourage the team members to keep an eye on these intagible aspects of quality. Thus, every team member works in unison to achieve the common goal of high-quality software.

QM involves verifying and improving the software process. The quality of intermediate results at each stage and the final end-product is measured, reviewed, and analysed. If the target does not seem to be achievable with the current process, then problems and difficulties are reported to senior management, and finally the current process can be improved based on this feedback.

QM is different from project management (PM), in the sense that QM is not associated with a particular project, rather it is an organization-wide activity for many projects. On the other hand, PM is specific to a particular project and manages the cost schedule, resources, and quality of that project only.

Mccall's Quality Factors and Criteria



McCall's Quality Factors and Criteria

- The 11 quality factors defined in Table 17.1 have been grouped into three broad categories (See Table 17.2.)
  - Product operation
  - Product revision
  - Product transition

Quality Categories	Quality Factors	Broad Objectives
Product Operation	Correctness	Does it do what the customer wants?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Does it quickly solve the intended problem?
	Integrity	Is it secure?
	Usability	Can I run it?
Product Revision	Maintainability	Can it be fixed?
	Testability	Can it be tested?
	Flexibility	Can it be changed?
Product Transition	Portability	Can it be used on another machine?
	Reusability	Can parts of it be reused?
	Interoperability	Can it interface with another system?

Table 17.2: Categorization of McCall's quality factors [10].

McCall's Quality Factors and Criteria

Quality Factors	Definitions
Correctness	The extent to which a program satisfies its specifications and fulfills the user'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 a function.
Integrity	The extent to which access to software or data by unauthorized persons can be controlled.
Usability	The effort required to learn, operate, prepare input, and interpret output of a program.
Maintainability	The effort required to locate and fix a defect in an operational program.
Testability	The effort required to test a program to ensure that it performs its intended functions.
Flexibility	The effort required to modify an operational program.
Portability	The effort required to transfer a program from one hardware and/ or software environment to another.
Reusability	The extent to which parts of a software system can be reused in other applications.
Interoperability	The effort required to couple one system with another.

Table 17.1: McCall's quality factors [10].

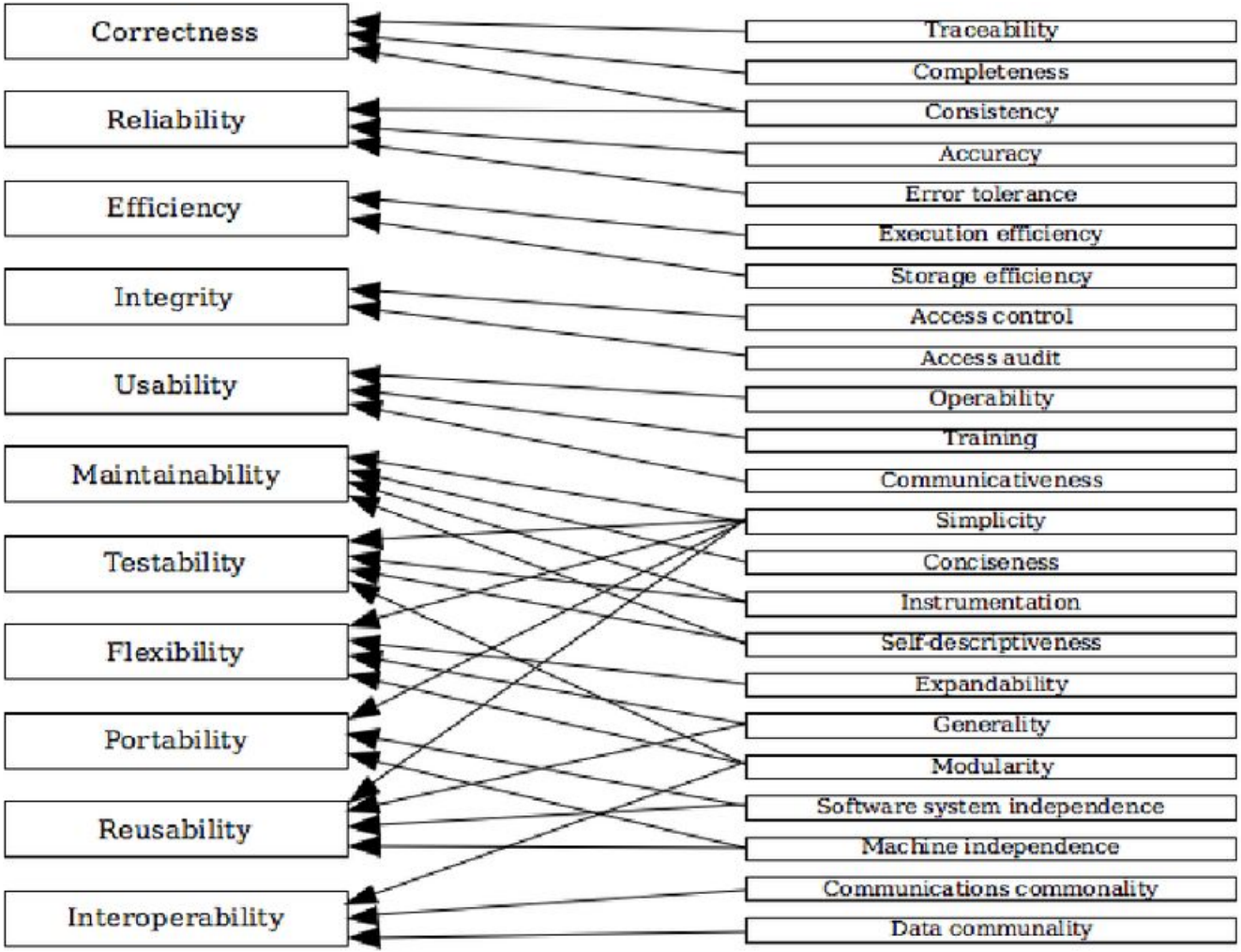

Correctness
Reliability
Efficiency
Integrity
Usability
Maintainability
Testability
Flexibility
Portability
Reusability
Interoperability
Traceability
Completeness
Consistency
Accuracy
Error tolerance
Execution efficiency
Storage efficiency
Access control
Access audit
Operability
Training
Communicativeness
Simplicity
Conciseness
Instrumentation
Self-descriptiveness
Expandability
Generality
Modularity
Software system independence
Machine independence
Communications commonality
Data communality

TABLE 17.3 McCall's Quality Criteria

Quality Criteria	Definition
Access audit	Ease with which software and data can be checked for compliance with standards or other requirements
Access control	Provisions for control and protection of the software and data
Accuracy	Precision of computations and output
Communication commonality	Degree to which standard protocols and interfaces are used
Completeness	Degree to which a full implementation of the required functionalities has been achieved
Communicativeness	Ease with which inputs and outputs can be assimilated
Conciseness	Compactness of the source code, in terms of lines of code
Consistency	Use of uniform design and implementation techniques and notation throughout a project
Data commonality	Use of standard data representations
Error tolerance	Degree to which continuity of operation is ensured under adverse conditions
Execution efficiency	Run time efficiency of the software
Expandability	Degree to which storage requirements or software functions can be expanded
Generality	Breadth of the potential application of software components

TABLE 17.3 McCall's Quality Criteria

Quality Criteria	Definition
Hardware independence	Degree to which the software is dependent on the underlying hardware
Instrumentation	Degree to which the software provides for measurement of its use or identification of errors
Modularity	Provision of highly independent modules
Operability	Ease of operation of the software
Self-documentation	Provision of in-line documentation that explains implementation of components
Simplicity	Ease with which the software can be understood.
Software system independence	Degree to which the software is independent of its software environment—nonstandard language constructs, operating system, libraries, database management system, etc.
Software efficiency	Run time storage requirements of the software
Traceability	Ability to link software components to requirements
Training	Ease with which new users can use the system

SIX SIGMA

Six sigma is a quality model originally developed for manufacturing processes. It was developed by Motorola. Six sigma derives its meaning from the field of statistics. Sigma is the standard deviation for a statistical population. Six sigma means, if one has six standard deviations between the mean of a process and the nearest specification limit, there will be practically no items that fail to meet the specifications. Therefore, the goal of this model is to have a process quality of that level. Eventually, six sigma evolved and was applied to other non-manufacturing processes. Today, you can apply six sigma to many fields like services, medical, and insurance procedures, call centers including software.

Six sigma is a way to achieve strategic business results emphasizing on lower costs with less number of defects. Six sigma processes will produce less than 3.4 defects per million opportunities. To achieve this target, it uses a methodology known as DMAIC with the following steps:

- **D**efine opportunities
- **M**easure performance
- **A**nalyse opportunity
- **I**mprove performance
- **C**ontrol performance

DMAIC (Define—Measure—Analyze—Improve—Control)

DMAIC is the more well-known and most-used LSS project method. DMAIC focuses on improving an existing process by incorporating the following phases:

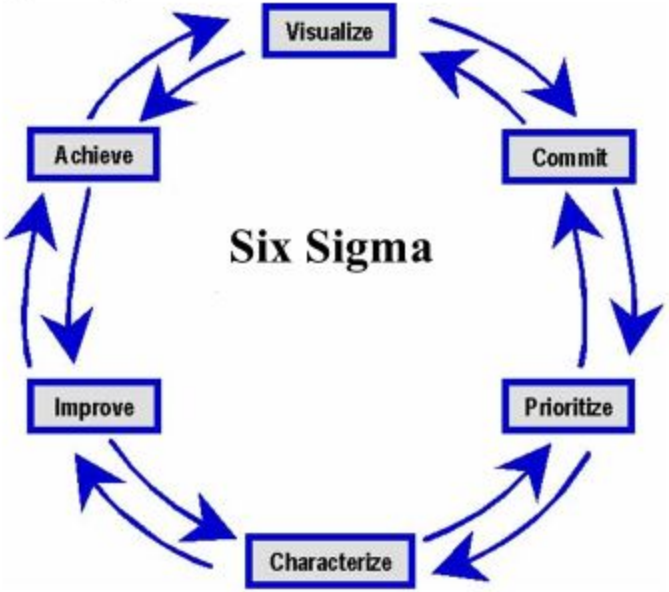
Phase	Description
Define	Define the problem, output to be improved, customers, and process associated with the problem.
Measure	Collect data from the process to establish a baseline for the improvements.
Analyze	Analyze the data to find the root causes of defects.
Improve	Develop, test, and implement solutions to improve the process.
Control	Implement process controls to sustain the improvements.

DMADV (Define—Measure—Analyze—Design—Verify)

DMADV is focused on the process of designing a new product, service or process, incorporating the following phases:

Phase	Description
Define	Define the process and design goals.
Measure	Measure and identify critical-to-quality characteristics of the product, service or process. This includes risk and production capabilities.
Analyze	Analyze the data to find the best design.
Design	Design and test the product, service or process.
Verify	Ensure that the design output meets the design input requirements (verification) and that the designed product performs satisfactorily under real or simulated conditions of intended use (validation).

Quality Improvement – The Wheel of 6Sigma



Quality Improvement – Six Sigma Process

- **Visualize** – Understand how it works now and imagine how it will work in the future
- **Commit** – Obtain commitment to change from the stakeholders
- **Prioritize** – Define priorities for incremental improvements
- **Characterize** – Define existing process and define the time progression for incremental improvements
- **Improve** – Design and implement identified improvements
- **Achieve** – Realize the results of the change

This methodology improves any existing business process by constantly reviewing and improving the process.

Six sigma is not just all about statistics. It can be applied to software problems which affects its quality. Six sigma can be applied to analyse the customer requirements and define business goals correspondingly. Its approach to customer requirements, if applied to software development projects, is fundamentally different than those typically practiced in software deployment efforts. It does not start by asking the customers about the requirements first. But it begins by analysing what we need to learn. There are six sigma tools which help in identifying the prioritization of functionalities to be delivered.

ISO 9000:2000

- There are ongoing efforts at the international level for standardizing different aspects of computer communications and software development.
- The ISO has developed a series of standards, collectively known as the ISO 9000.
- The ISO has developed a series of standards, collectively known as the ISO 9000.
- The ISO 9000 standards are reviewed and updated from time to time, once every 5–8 years.
- The ISO 9000:2000 standard is based on the following eight principles:
- **Principle 1. Customer Focus:** Success of an organization is highly dependent on satisfying the customers. An organization must understand its customers and their needs on a continued basis. Understanding the customers helps in understanding and meeting their requirements. It is not enough to just meet customer requirements. Rather, organizations must make an effort to exceed customer expectations. By understanding the customers, one can have a better understanding of their real needs and their unstated expectations

ISO 9000:2000

- **Principle 2. Leadership:** Leaders set the direction their organization should take, and they must effectively communicate this to all the people involved in the process. All the people in an organization must Without a good understanding of the organizational direction, employees will find it difficult to know where they are heading. Leaders must set challenging but realistic goals and objectives. Employee contribution should be recognized by the leaders. Leaders create a positive environment and provide support for the employees to collectively realize the organizational goal.
- **Principle 3. Involvement of People:** In general, organizations rely on people. People are informed of the organizational direction, and they are involved at all levels of decision making. People are given an opportunity to develop their strength and use their abilities.

ISO 9000:2000

- **Principle 4. Process Approach:** A process is a sequence of activities that transform inputs to outputs. Organizations can prepare a plan in the form of allocating resources and scheduling the activities by making the process defined, repeatable, and measurable. Consequently, the organization becomes efficient and effective. Continuous improvement in processes leads to improvement in efficiency and effectiveness
- **Principle 5. System Approach to Management:** At any time, people are involved in one or more processes. A process is affected by the outcome of some other processes, and, in turn, it affects some other processes in the organization. It is important to understand the overall goal of the organization and the individual subgoals associated with each process. For an organization as a whole to succeed in terms of effectiveness and efficiency, the interactions among processes must be identified and analyzed

ISO 9000:2000

- **Principle 6. Continual Improvement:** Continual improvement means that the processes involved in developing, say, software products are reviewed on a periodic basis to identify where and how further improvements in the processes can be affected. . Continual process improvements result in lower cost of production and maintenance. Moreover, continual improvements lead to less differences between the expected behavior and actual behavior of the system.
- **Principle 7. Factual Approach to Decision Making:** Decisions may be made based on facts, experience, and intuition. Facts can be gathered by using a sound measurement process. There is a need for methods to validate the measured data and make the data available to those who need it. The measured data should be accurate and reliable. A quantitative measurement program helps organizations know how much improvement has been achieved due to a process improvement.

ISO 9000:2000

- **Principle 8. Mutually Beneficial Supplier Relationships:** Organizations rarely make all the components they use in their products. It is a common practice for organizations to procure components and subsystems from third parties. An organization must carefully choose the suppliers and make them aware of the organization's needs and expectations. The performance of the products procured from outside should be evaluated, and the need to improve their products and processes should be communicated to the suppliers. A mutually beneficial, cooperative relationship should be maintained with the suppliers.