UNIVERSITY OF RAJSHAHI
Rajshahi, BANGLADESH.

Course Code:

ICE-3151

Course Title :

Information System Analysis and **Software Engineering**

Chapter:

Software Quality Management

Software Engineering

ICE-3151

Section-B

Unit

Software Quality Management

Software Quality Management: Concept of Quality, Quality Factors, Achieving Software Quality, Elements of Software Quality Assurance, SQA Tasks, Goals, and Metrics, Formal Approaches to SQA. Statistical Software Quality Assurance, Software Reliability, The ISO 9000 Quality Standards, The SQA Plan.

What is Quality?

- Quality is something 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.

What is 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.

Quality of Design?

- Quality of design refers to the characteristics that designers specify for a product.
- The grade of materials, tolerances, and performance specifications all contribute to the quality of design.
- As higher-grade materials are used, tighter tolerances and greater levels of performance are specified.
- The design quality of a product increases if the product is manufactured according to specifications.

Quality of Design in Software Development

- In software development,
  - ***quality of design*** encompasses the degree to which the design meets the functions and features specified in the requirements model.
  - ***Quality of conformance*** focuses on the degree to which the implementation follows the design and the resulting system meets its requirements and performance goals.
- But the quality of design and quality of conformance the only issues that software engineers must consider?

User satisfaction = compliant product +
good quality +
delivery within budget and schedule

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.
  - An effective software process establishes the infrastructure that supports any effort at building a high-quality software 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.
  - 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.

Garvin's Eight Quality Dimensions

- Although Garvin's eight dimensions of quality were not developed specifically for software, they can be applied when software quality is considered:
  - ✓ ***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?

Garvin's Eight Quality Dimensions (Conti….)

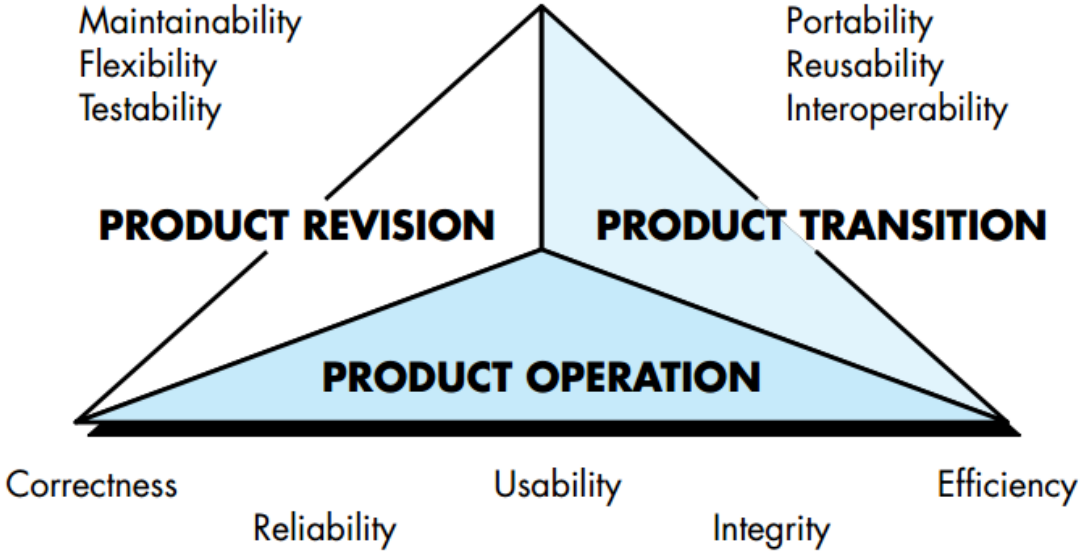
- ✓ ***Conformance:*** Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?
- ✓ ***Durability:*** 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?

Garvin's Eight Quality Dimensions (Cont…)

- ✓ ***Aesthetics:*** There's no question that each of us has a different and very subjective vision of what is aesthetic. And yet, 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 are evident nonetheless. Aesthetic software has these characteristics.
- ✓ ***Perception:*** In some situations, you have a set of prejudices that will influence your perception of quality. For example, if you are introduced to a software product that was built by a vendor who has produced poor quality in the past, your guard will be raised and your perception of the current software product quality might be influenced negatively. Similarly, if a vendor has an excellent reputation, you may perceive quality, even when it does not really exist.

McCall's Quality Factors

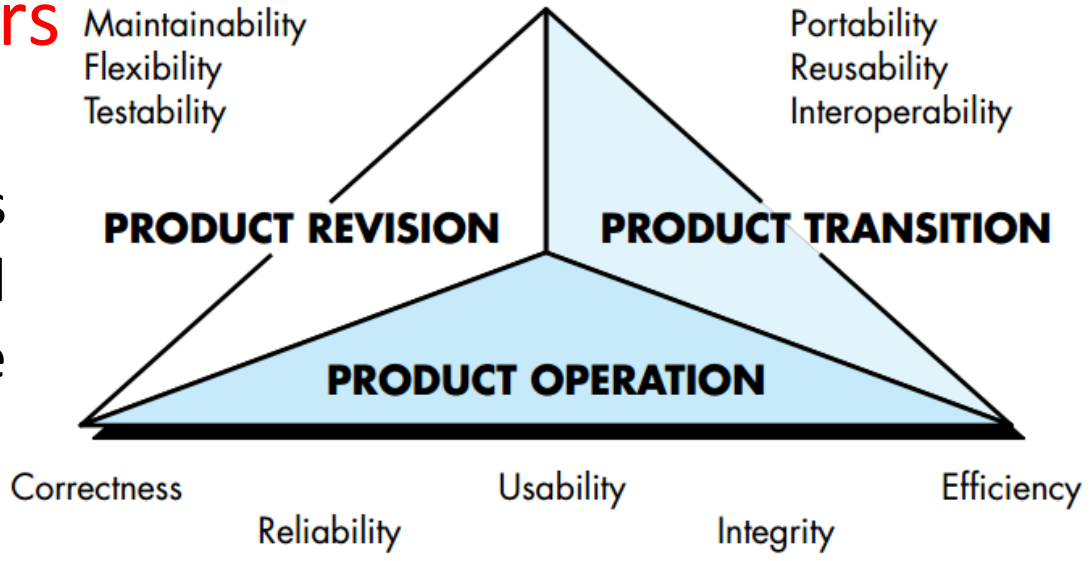
- McCall propose a useful categorization of factors that affect software quality. These software quality factors, shown in Figure bellow.



- These software quality factor focus on three important aspects of a software product:
    - its operational characteristics,
    - its ability to undergo change, and
    - its adaptability to new environments.

McCall's Quality Factors

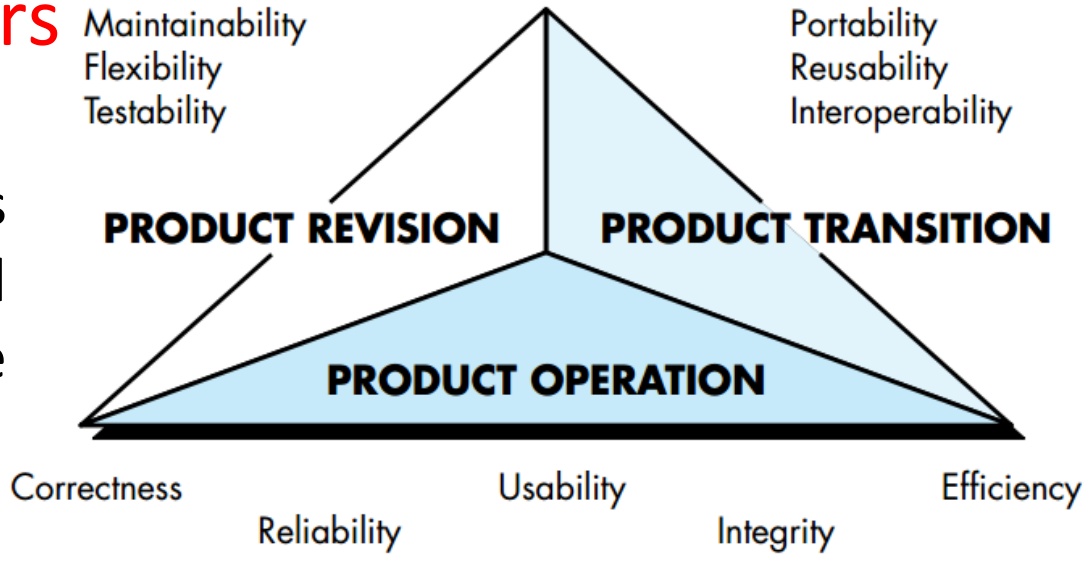
Referring to the factors noted in Figure, McCall and his colleagues provide the following descriptions:



- ***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.

McCall's Quality Factors

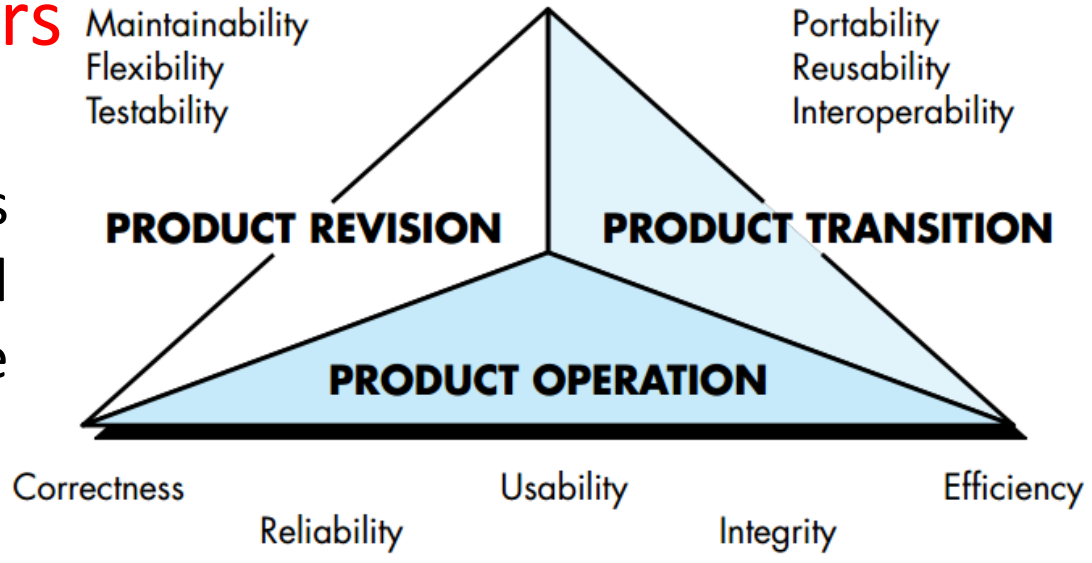
Referring to the factors noted in Figure, McCall and his colleagues provide the following descriptions:



- ***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.

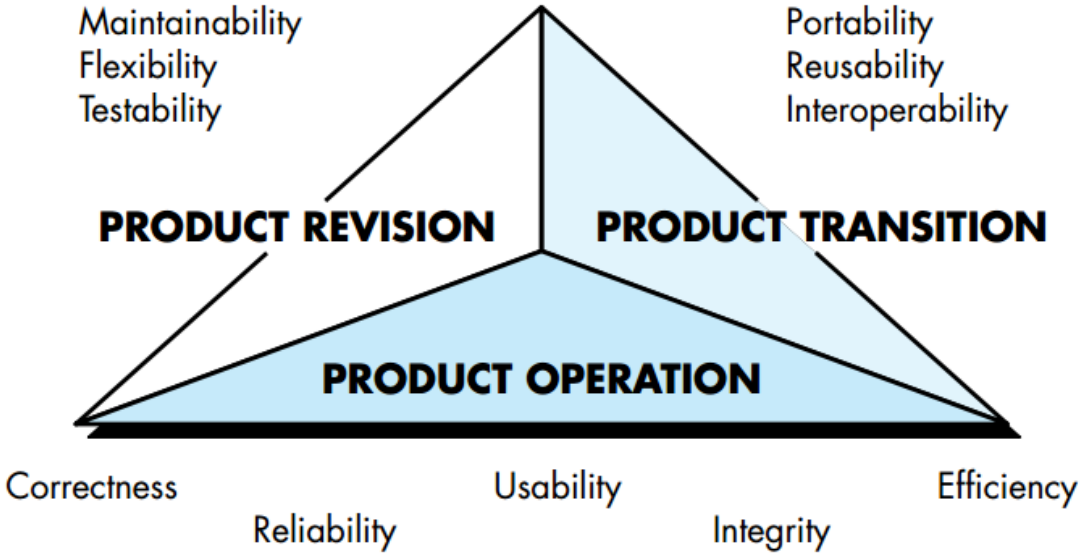
McCall's Quality Factors

Referring to the factors noted in Figure, McCall and his colleagues provide the following descriptions:



- ***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—related to the packaging and scope of the functions that the program performs.
- ***Interoperability:*** Effort required to couple one system to another

McCall's Quality Factors



- It is difficult, and in some cases impossible, to develop direct measures of these quality factors. In fact, many of the metrics defined by McCall and colleagues can be measured only indirectly.
- ✓ However, assessing the quality of an application using these factors will provide you with a solid indication of software quality

ISO 9126 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:**
  - ✓ ***Functionality:*** The degree to which the software satisfies stated needs as indicated by the following sub-attributes: ***suitability, accuracy, interoperability, compliance, and security.***
  - ✓ ***Reliability:*** The amount of time that the software is available for use as indicated by the following sub-attributes: ***maturity, fault tolerance, recoverability.***
  - ✓ ***Usability:*** The degree to which the software is easy to use as indicated by the following sub-attributes: ***understandability, learnability, operability.***

ISO 9126 Quality Factors

- ✓ **Efficiency.** The degree to which the software makes optimal use of system resources as indicated by the following subattributes: ***time behavior, resource behavior***.
- ✓ **Maintainability:** The ease with which repair may be made to the software as indicated by the following subattributes: ***analyzability, changeability, stability, testability***
- ✓ **Portability:** The ease with which the software can be transposed from one environment to another as indicated by the following subattributes: ***adaptability, installability, conformance, replaceability*** .

Achieving Software Quality

- Software quality doesn't just appear.
- It is the result of good project management and solid software engineering practice.
- Management and practice are applied within the context of four broad activities that help a software team achieve high software quality:
  - Software engineering methods,
  - Project management techniques,
  - Quality control actions, and
  - Software quality assurance.

Achieving Software Quality (Conti…)

Software engineering methods

- If you expect to build high-quality software, you must understand the problem to be solved.
- You must also be capable of creating a design that conforms to the problem while at the same time exhibiting characteristics that lead to software that exhibits the quality dimensions and factors.

Project management techniques

- Software quality is often influenced as much by management decisions as it is by technology decisions.
- Even the best software engineering practices can be subverted by poor business decisions and questionable project management actions.

Achieving Software Quality (Conti…)

Quality control actions

- 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.

Achieving Software Quality (Conti…)

Software Quality Assurances

- Quality assurance establishes the infrastructure that supports solid ***software engineering methods, rational project management, and quality control actions.***

Concern & Activities of Software Quality Assurance (SQA)

Software quality assurance encompasses a broad range of concerns and activities that can be summarized in the following manner:

- ***Standards:*** The IEEE, ISO, and other standards organizations have produced a broad array of software engineering standards and related documents. Standards may be adopted voluntarily by a software engineering organization.
- ***Reviews and Audits:*** Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors.

Concern & Activities of SQA (Cont…)

- ***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 so that it has the highest likelihood of achieving its primary goal.
- ***Error/defect collection and analysis:*** The only way to improve is to measure how you're doing. 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.

Concern & Activities of SQA

- ***Change Management:*** Change is one of the most disruptive aspects of any software project. If it is not properly managed, change can lead to confusion, and confusion almost always leads to poor quality. SQA ensures that adequate change management.
- ***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.

Concern & Activities of SQA

- ***Vendor management:*** Three categories of software are acquired from external software vendors—***shrink-wrapped packages*** (e.g., Microsoft Office), ***a tailored shell*** that provides a basic skeletal structure that is custom tailored to the needs of a purchaser, and ***contracted software*** that is custom designed and constructed from specifications provided by the customer organization.
- ***Security management:*** With the increase in cyber crime and new government regulations regarding privacy, every software organization should institute policies that protect data at all levels, establish firewall protection for WebApps, and ensure that software has not been tampered with internally.

Concern & Activities of SQA

- ***Safety:*** Because software is almost always a pivotal component of human-rated systems (e.g., automotive or aircraft applications), the impact of hidden defects can be catastrophic. SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.
- ***Risk management:*** Although the analysis and mitigation of risk is the concern of software engineers, the SQA organization ensures that risk management activities are properly conducted and that risk-related contingency plans have been established.

SQA Tasks

- Software quality assurance is composed of a variety of tasks associated with **TWO** different constituencies:
  - The software engineers who do technical work, and
  - an SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting.
    - ✓ Software engineers address quality (and perform quality control activities) by applying solid technical methods and measures, conducting technical reviews, and performing well-planned software testing.

SQA Tasks

- **Prepares an SQA plan for a project:** The plan is developed as part of project planning and is reviewed by all stakeholders.
- **Participates in the development of the project's software process description:** The software team selects a process for the work to be performed. 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:** The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

SQA Tasks

- **Audits designated software work products to verify compliance with those defined as part of the software process:** The SQA group reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made; and 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:** Deviations may be encountered in the project plan, process description, applicable standards, or software engineering work products.
- **Records any noncompliance and reports to senior management:** Noncompliance items are tracked until they are resolved.

SQA Goals

- **Requirements quality:** SQA must ensure that the software team has properly reviewed the requirements model to achieve a high level of quality.
- **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:** SQA should isolate those attributes that allow a reasonable analysis of the quality of code.
- **Quality control effectiveness:** A software team should apply limited resources in a way that has the highest likelihood of achieving a high-quality result.

Software Quality Goals, attributes, and metrics

Goal	Attribute	Metric
Requirement quality	Ambiguity	Number of ambiguous modifiers (e.g., many, large, human-friendly)
	Completeness	Number of TBA, TBD
	Understandability	Number of sections/subsections
	Volatility	Number of changes per requirement
		Time (by activity) when change is requested
	Traceability	Number of requirements not traceable to design/code
	Model clarity	Number of UML models
		Number of descriptive pages per model
		Number of UML errors

Software Quality Goals, attributes, and metrics

Goal	Attribute	Metric
Design quality	Architectural integrity	Existence of architectural model
	Component completeness	Number of components that trace to architectural model
		Complexity of procedural design
	Interface complexity	Average number of pick to get to a typical function or content
		Layout appropriateness
	Patterns	Number of patterns used
Code quality	Complexity	Cyclomatic complexity
	Maintainability	Design factors (Chapter 8)
	Understandability	Percent internal comments
		Variable naming conventions
	Reusability	Percent reused components
	Documentation	Readability index

Software Quality Goals, attributes, and metrics

Goal	Attribute	Metric
QC effectiveness	Resource allocation	Staff hour percentage per activity
	Completion rate	Actual vs. budgeted completion time
	Review effectiveness	See review metrics (Chapter 14)
	Testing effectiveness	Number of errors found and criticality
		Effort required to correct an error
		Origin of error

Formal Approaches to SQA

- Over the past three decades, a small, but vocal, segment of the software engineering community has argued that a more formal approach to software quality assurance is required.
- It can be argued that a computer program is a mathematical object.
- A rigorous syntax and semantics can be defined for every programming language, and a rigorous approach to the specification of software requirements is available.
- If the requirements model (specification) and the programming language can be represented in a rigorous manner, it should be possible to apply mathematic proof of correctness to demonstrate that a program conforms exactly to its specifications

Statistical SQA

- Statistical quality assurance reflects a growing trend throughout the industry to become more quantitative about quality. For software, statistical quality assurance implies the following steps:
  - Information about software errors and defects is collected and categorized.
  - An attempt is made to trace each error and defect to its underlying cause.
  - 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).
  - Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

A Generic Example: Statistical SQA

- To illustrate the use of statistical methods for software engineering work, assume that a software engineering organization collects information on errors and defects for a period of one year.
- Some of the errors are uncovered as software is being developed.
- Other defects are encountered after the software has been released to its end users.
- Although hundreds of different problems 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 Reliability

- There is no doubt that the reliability of a computer program is an important element of its overall quality.
- If a program repeatedly and frequently fails to perform, it matters little whether other software quality factors are acceptable.
- 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”.
- Software reliability, unlike many other quality factors, can be measured directly and estimated using historical and developmental data.

Measures of Software Reliability

- 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, respectively

- Many researchers argue that MTBF is a far more useful measure than other quality-related software metrics

Software Availability

- Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as

$$\text{Availability} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})} \times 100\%$$

- The MTBF reliability measure is equally sensitive to MTTF and MTTR. The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.

The ISO 9000 Quality Standards

- A quality assurance system may be defined as the organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.
- Quality assurance systems are created to help organizations ensure their products and services satisfy customer expectations by meeting their specifications.
- These systems cover a wide variety of activities encompassing a product's entire life cycle including planning, controlling, measuring, testing and reporting, and improving quality levels throughout the development and manufacturing process.
- ISO 9000 describes quality assurance elements in generic terms that can be applied to any business regardless of the products or services offered.

The ISO 9000 Quality Standards

- To become registered to one of the quality assurance system models contained in ISO 9000, a company's quality system and operations are scrutinized by thirdparty auditors for compliance to the standard and for effective operation.
- Upon successful registration, a company is issued a certificate from a registration body represented by the auditors.

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.
- A standard for SQA plans has been published by the IEEE.

SQA Plan

- The standard for SQA 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) that fall within the purview of SQA,
  - all applicable standards and practices that are applied during the software process,
  - SQA actions and tasks (including reviews and audits) 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, and
  - organizational roles and responsibilities relative to product quality.