Software Quality Assurance

Module 2

Software Quality

Objectives

- Introduce software quality
- Provide stakeholder-centric visions of quality and defect
- Describe some quality models

Module 2 - Content Outline (Agenda)

➔ Software quality
  - Definitions of quality
  - A pragmatic definition of defect
  - Dimensions of quality
  - Measuring quality
- McCall’s quality model
- ISO 9126 Quality Characteristics
- ISO 9000:2000 Software Quality Standard

Discussion Exercise 2.1: Define *Quality*

- Form pairs
- Define *quality*
  - Write the definition down
  - Is this your company’s definition?
  - Is this your partner’s company’s definition?
  - Does your partner agree with his/her corporate definition of quality?

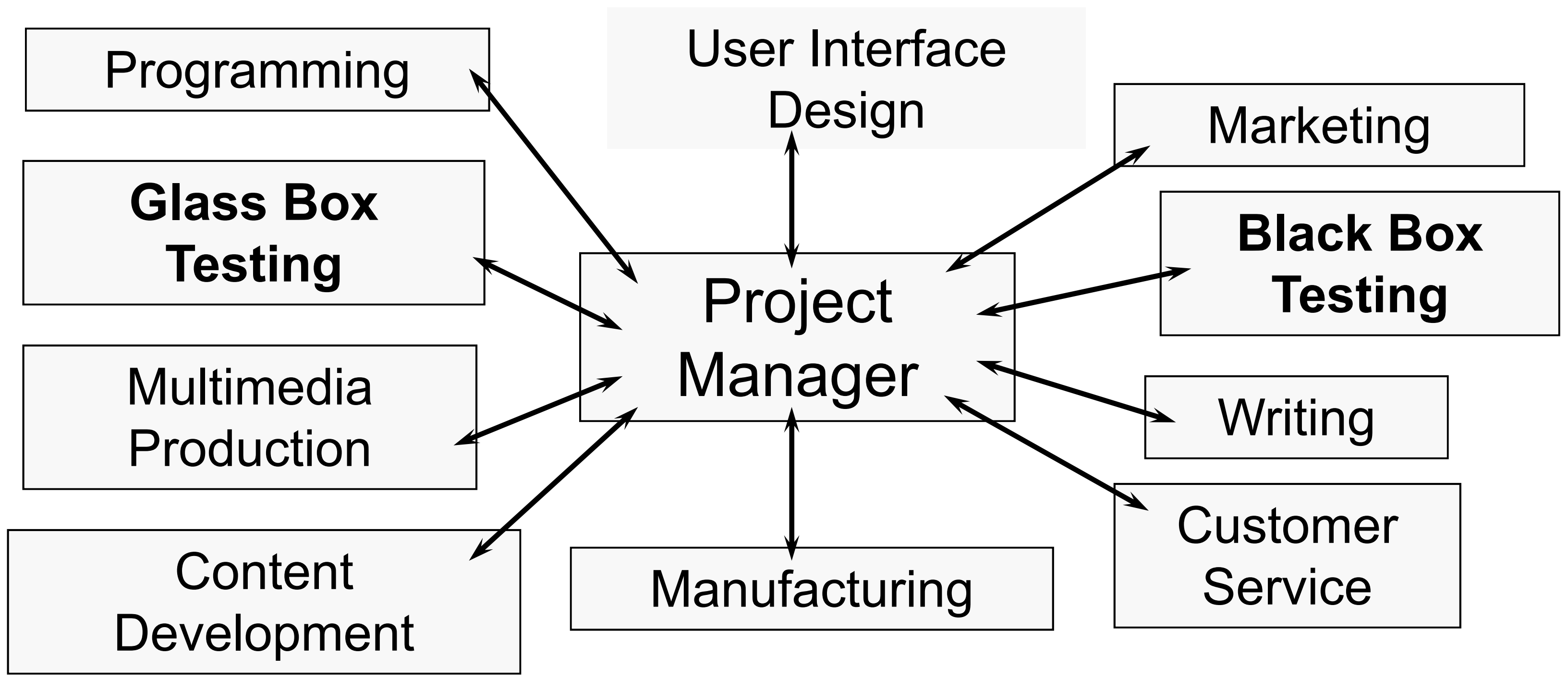
How Some Experts Have Defined Quality

- Define quality
  - Fitness for use (Dr. Joseph M. Juran)
  - The totality of features and characteristics of a product that bear on its ability to satisfy a given need (American Society for Quality)
  - Conformance with requirements (Philip Crosby)
  - The total composite product and service characteristics of marketing, engineering, manufacturing and maintenance through which the product and service in use will meet expectations of the customer (Armand V. Feigenbaum)
- Note absence of “conforms to specifications.”

Quality As Satisfiers and Dissatisfiers

- Joseph Juran distinguishes between Customer *Satisfiers* and *Dissatisfiers* as key dimensions of quality:
  - Customer Satisfiers
    - the right features
    - adequate instruction
  - Customer Dissatisfiers
    - unreliable
    - hard to use
    - too slow
    - incompatible with the customer’s equipment

Quality Involves Many Stakeholders



- In a project team meeting, each person in the room has a different vision of what a “quality” product would be. Fixing defects is just one issue.

Exercise 2.2: Quality Has Many Stakeholders (1/2)

- A programming group agrees on variable naming conventions and follows them. Then they hire a new senior programmer, who won’t follow the conventions. Programming the way he did in the 1970’s, he gives variables names like Mabel and Al. Some of the other programmers are unhappy about this and so they want to enter a bug report into the bug tracking system, saying that “Mabel is not a proper variable name.”

- **Question**: Does this report belong in the change request system? Why or why not?

Exercise 2.2: Quality Has Many Stakeholders (2/2)

- A company is developing a product that they will release in English first. The company expects to create other language versions by changing strings and other resource variables from the English version. They plan to localize versions without recompiling or relinking the code. However, the English code has a hyphenation error that doesn’t affect English words but will mishandle some German words.
- The localization manager wants this filed as a bug against the English base code. Fixing the problem in the German version would require a code change and recompilation. The American project manager says that it is not a bug because the English version works OK. Who is right? Should the change request go with the English product, the German product, or both?

A Working Definition of Quality

Quality is value to some person.

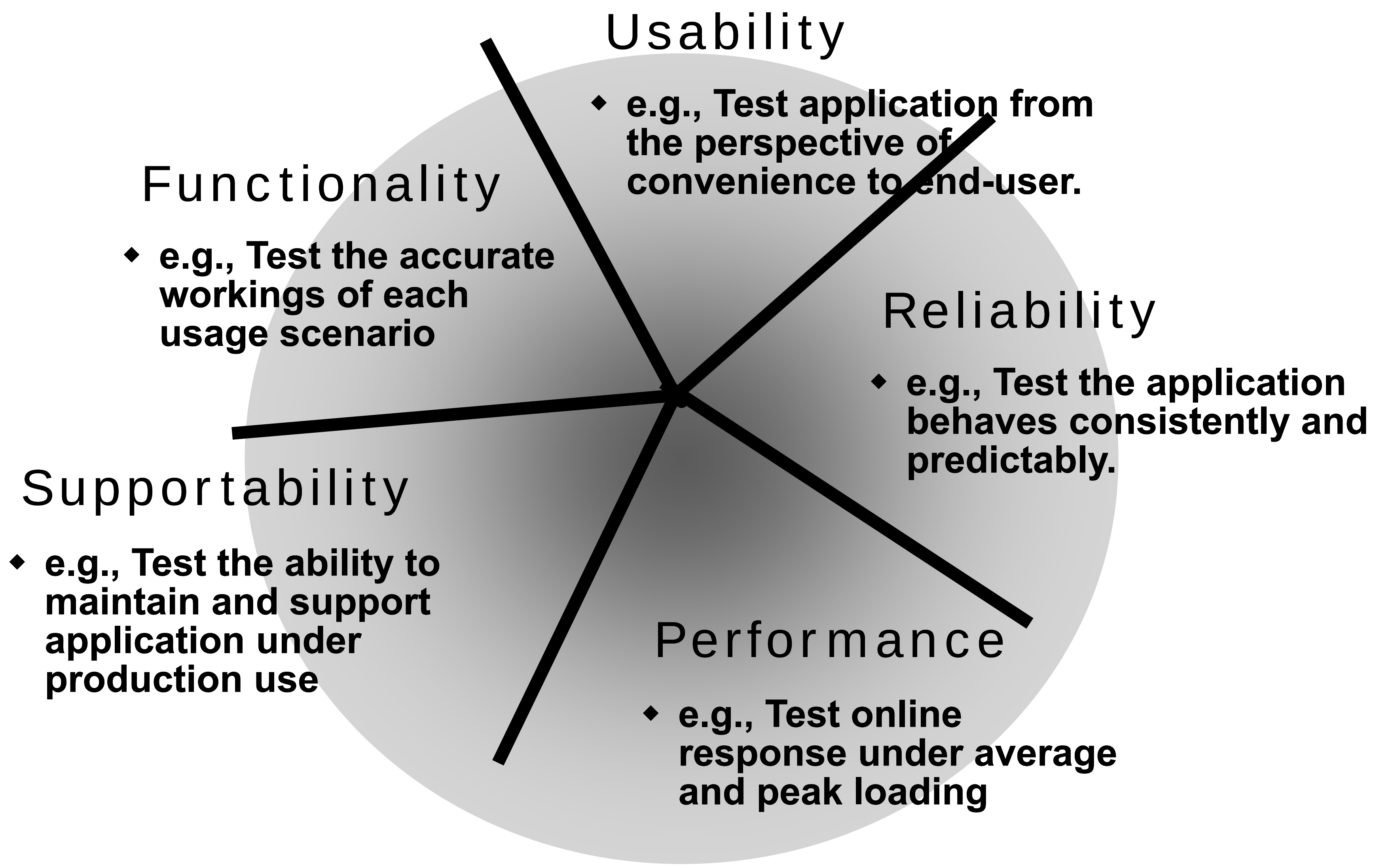
---- Gerald M. Weinberg

Change Requests and Quality

- A “defect” – in the eyes of a project stakeholder– can include anything about the program that causes the program to have lower value.

- It’s appropriate to report any aspect of the software that, in your opinion (or in the opinion of a stakeholder whose interests you advocate) causes the program to have lower value.

Dimensions of Quality: FURPS



Usability

- e.g., Test application from the perspective of convenience to end-user.

Functionality

- e.g., Test the accurate workings of each usage scenario

Reliability

- e.g., Test the application behaves consistently and predictably.

Supportability

- e.g., Test the ability to maintain and support application under production use

Performance

- e.g., Test online response under average and peak loading

The traditional view of FURPS

1. Functional testing

verifies that a system executes the required use-case scenarios as intended.

Functional tests may include the testing of features, usage scenarios and security.

The traditional view of FURPS

2. Usability testing

evaluates the application from the user’s perspective. Usability tests focus on human factors, aesthetics, consistency in the user interface, online and context-sensitive help, wizards and agents, user documentation, and training materials.

The traditional view of FURPS

3. **Reliability testing**

verifies that the application performs reliably and is not prone to failures during execution (crashes, hangs, memory leaks). Effective reliability testing requires specialized tools. Reliability tests include integrity, structure, stress, contention and volume tests.

The traditional view of FURPS

4. **Performance testing**

checks that the target system works functionally and reliably under production load. Performance tests include benchmark tests, load tests, and performance profile tests.

The traditional view of FURPS

5. **Supportability testing**

verifies that the application can be deployed as intended. Supportability tests include installation and configuration tests.

A Broader Definition of Dimensions of Quality

- Accessibility
- Capability
- Compatibility
- Concurrency
- Conformance to standards
- Efficiency
- Installability and uninstallability
- Localizability
- Maintainability
- Performance
- Portability
- Reliability
- Scalability
- Security
- Supportability
- Testability
- Usability

Collectively, these are often called *Qualities of Service*, *Nonfunctional Requirements*, *Attributes*, or simply *the -ilities*

Measuring quality

- Measurement allows us to establish baselines for qualities. Developers must know the minimum level of quality they must deliver for a product to be acceptable.
- Organizations make continuous improvements in their process models—and an improvement has a cost associated with it. Organizations need to know how much improvement in quality is achieved at a certain cost incurred due to process improvement.
- The present level of quality of a product needs to be evaluated so the need for improvements can be investigated.

Measurement of User’s View

- The user’s view encompasses a number of quality factors, such as functionality, reliability, and usability.
- It is easy to measure how much of the functionalities a software product delivers by designing at least one test case for each functionality.
- The ratio of the number of passed test cases to the total number of test cases designed to verify the functionalities is a measure of the functionalities delivered by the product.

Measurement of User’s View

- Among the qualities that reflect the user’s view, the concept of *reliability* has drawn the most attention of researchers.
- In the ISO 9126 quality model, *usability* has been broken down into three sub-characteristics, namely, *learnability*, *understandability*, and *operability*.
- *The quality concept is broken down into component parts until each can be stated in terms of directly measurable attributes*.

Measurement of Manufacturer’s View

- Manufacturers are interested in obtaining measures of the following two different quantities:
  - **Defect Count:** How many defects have been detected?
  - **Rework Cost:** How much does it cost to fix the known defects?

Defect count

- Defect count represents the number of all the defects that have been detected so far.
- If a product is in operation, this count includes the defects detected during development and operation.
- A defect count reflects the quality of work produced.
- Merely counting the defects is of not much use unless something can be done to improve the development process to reduce the defect count in subsequent projects.

Defect count

- One can analyze the defects as follows:
  - For each defect identify the development phase in which it was introduced and the phase in which it was discovered.
  - Categorize the defects based on modules.
  - To compare defects across modules and products in a meaningful way, normalize the defect count by product size.
  - Separate the defects found during operation from the ones found during development.

Rework Cost

- After defects are detected, the developers make an effort to fix them.
- Ultimately, it costs some money to fix.
- The rework cost includes all the additional cost associated with defect-related activities, such as fixing documents.
- Rework is an additional cost that is incurred due to work being done in a less than perfect manner the first time it was done.

Rework Cost

- It is obvious that organizations strive to reduce the total cost of software development, including the rework cost.
- The rework cost can be split into two parts as follows:
  - **Development Rework Cost:** This is the rework cost incurred *before* a product is released to the customers.
  - **Operation Rework Cost:** This is the rework cost incurred *when* a product is in operation.

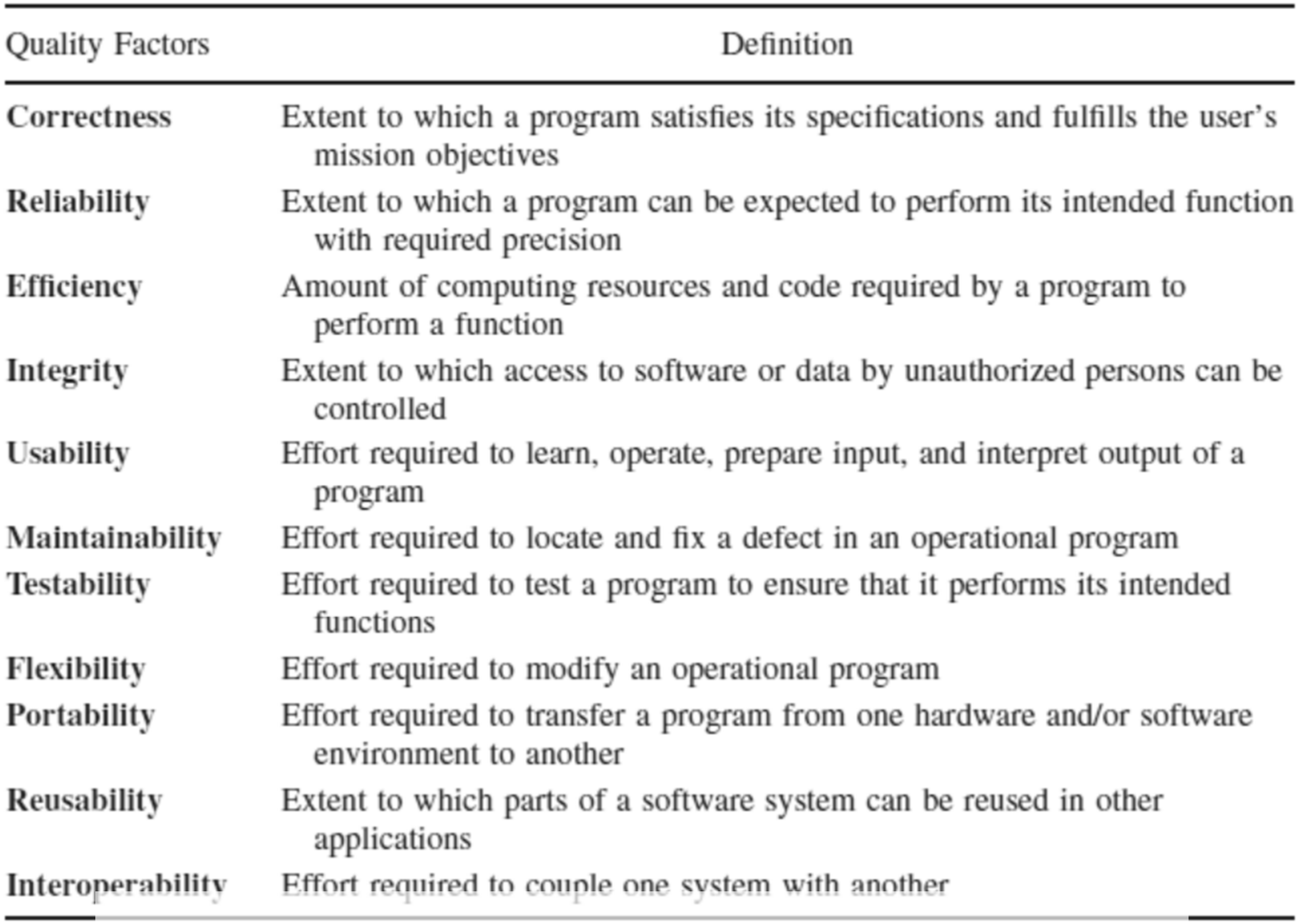
Module 2 - Content Outline (Agenda)

- ◆ Software quality
- ➔McCall’s quality model
- ◆ ISO 9126 Quality Characteristics
- ◆ ISO 9000:2000 Software Quality Standard

Quality Factors

- A *quality factor* represents a behavioral characteristic of a system.
- Quality factors are external attributes of a software system.
- Customers, software developers, and quality assurance engineers are interested in different quality factors to a different extent.

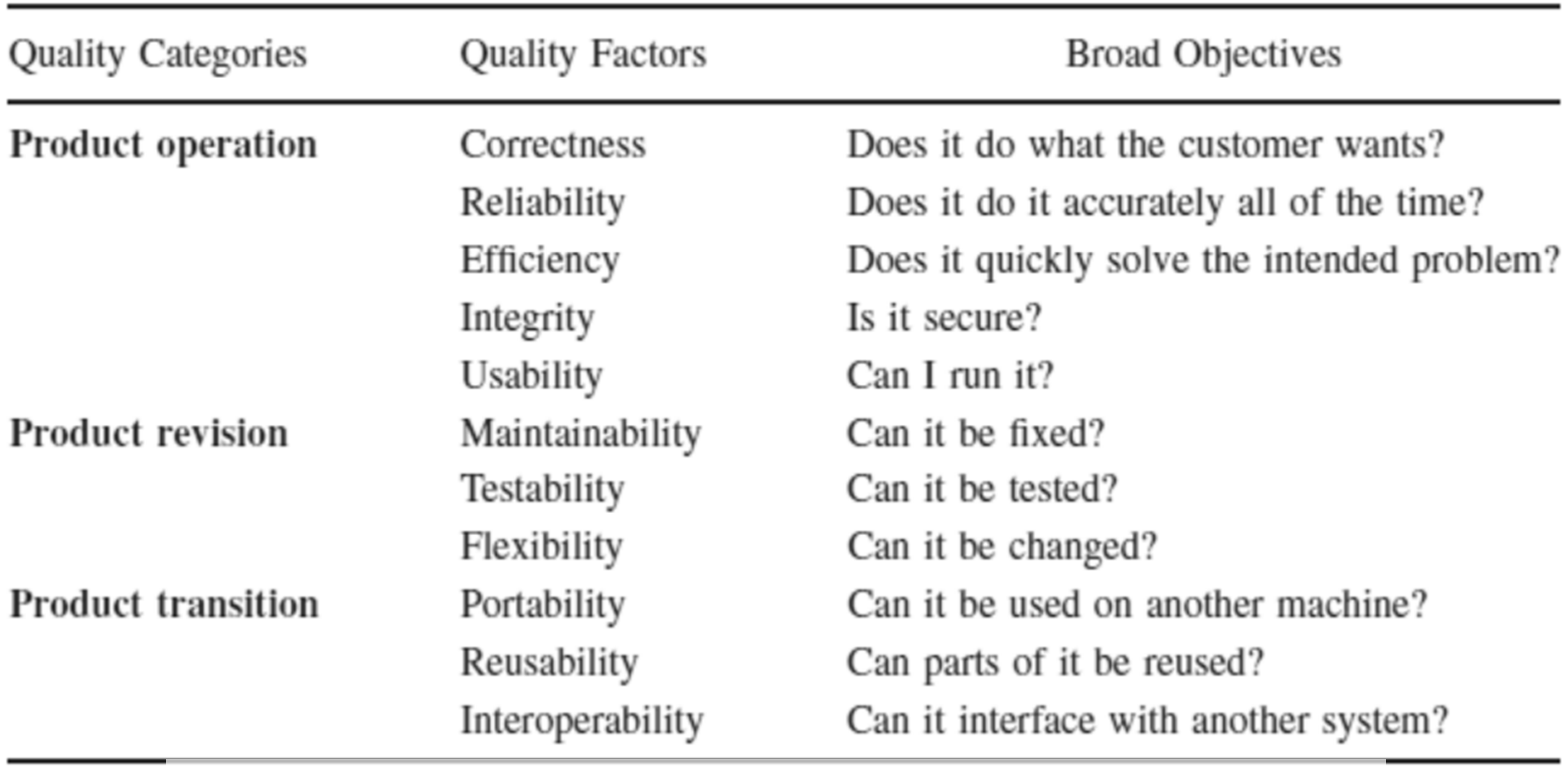
Quality Factors



Quality Factors	Definition
Correctness	Extent to which a program satisfies its specifications and fulfills the user's mission objectives
Reliability	Extent to which a program can be expected to perform its intended function with required precision
Efficiency	Amount of computing resources and code required by a program to perform a function
Integrity	Extent to which access to software or data by unauthorized persons can be controlled
Usability	Effort required to learn, operate, prepare input, and interpret output of a program
Maintainability	Effort required to locate and fix a defect in an operational program
Testability	Effort required to test a program to ensure that it performs its intended functions
Flexibility	Effort required to modify an operational program
Portability	Effort required to transfer a program from one hardware and/or software environment to another
Reusability	Extent to which parts of a software system can be reused in other applications
Interoperability	Effort required to couple one system with another

Categorization of McCall’s Quality Factors

- The 11 quality factors have been grouped into three broad categories：



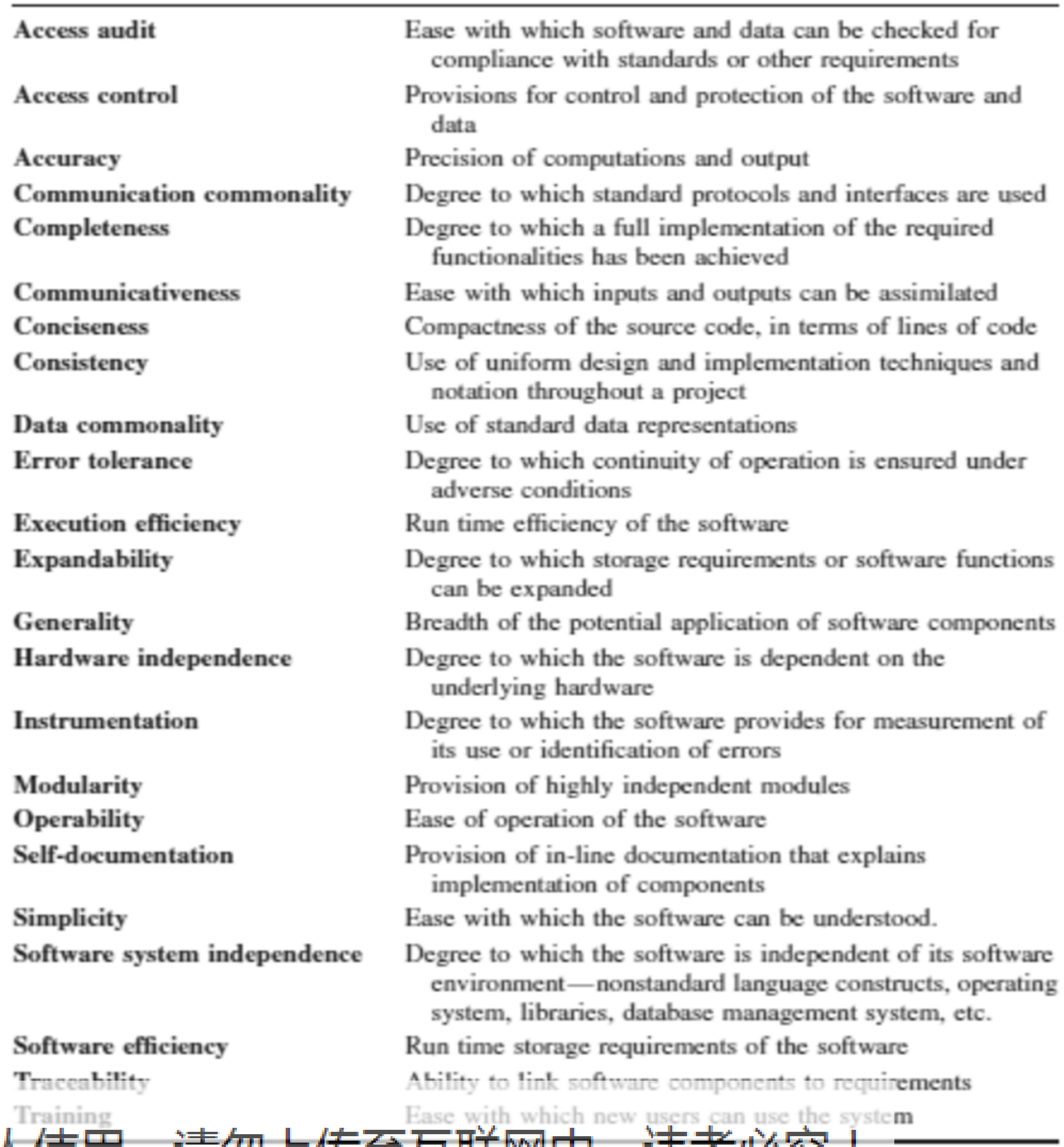
Quality Categories	Quality Factors	Broad Objectives
Product operation	Correctness	Does it do what the customer wants?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Does it quickly solve the intended problem?
	Integrity	Is it secure?
	Usability	Can I run it?
Product revision	Maintainability	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?

Quality Criteria

- A *quality criterion* is an attribute of a quality factor that is related to software development.
  - For example, modularity is an attribute of the architecture of a software system. A highly modular software allows designers to put cohesive components in one module, thereby increasing the maintainability of the system.
- Some quality criteria relate to products and some to personnel.
  - For example, modularity is a product-related quality criterion, whereas training concerns development and software quality assurance personnel.

Quality Criteri

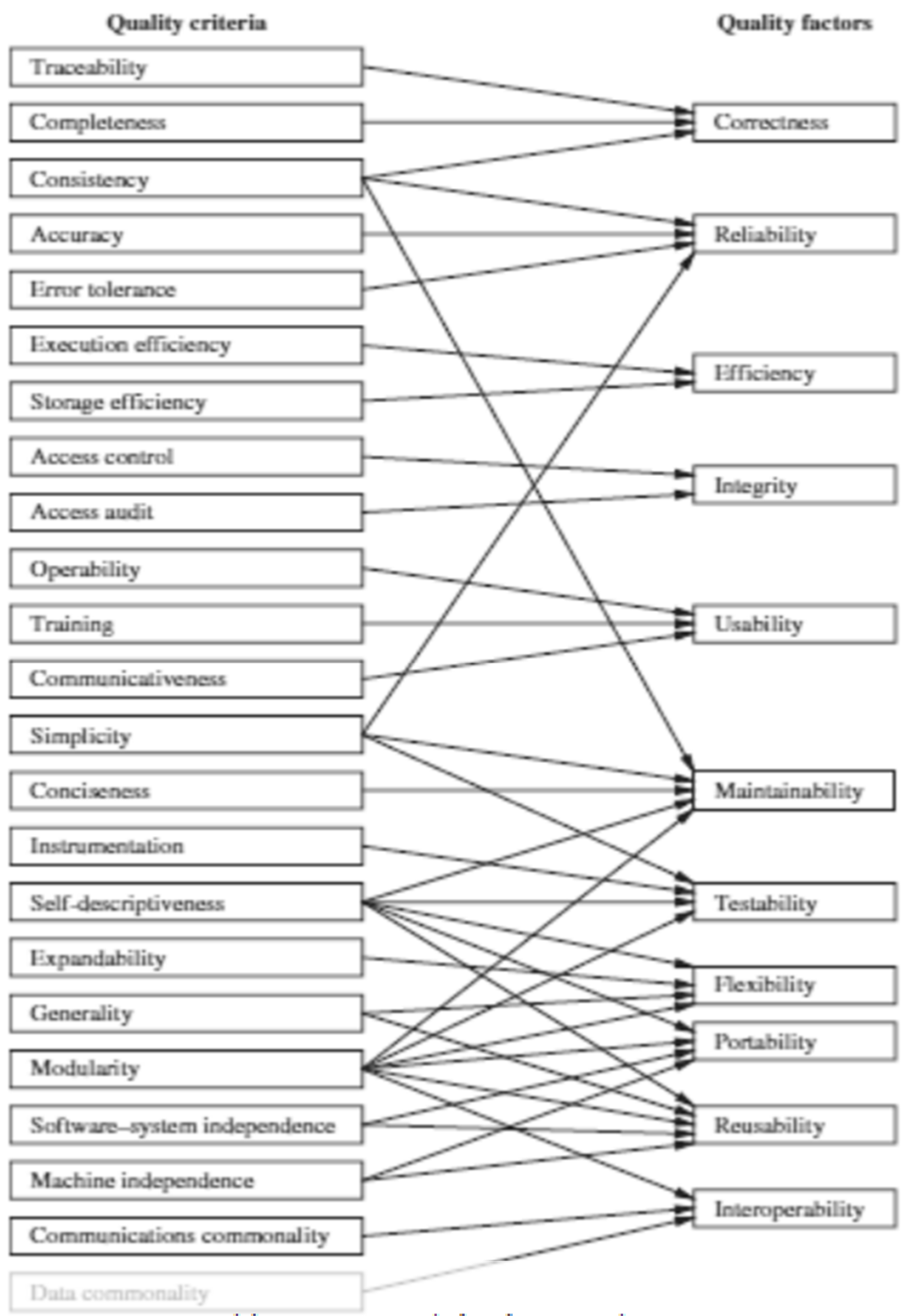
- 23 quality criteria：



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
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

Relationship between Qu

- The relationship between quality factors and quality criteria is shown in this figure.
- An arrow from a quality criterion to a quality factor means that the quality criterion has a positive impact on the quality factor.



Relationship between Quality Factors and Criteria

- Though it is desirable to improve all the quality factors, doing so may not be possible. This is because, in general, quality factors are not completely independent.
- Thus, we note two characteristics of the relationship as follows:
  - If an effort is made to improve one quality factor, another quality factor may be degraded. For example, if an effort is made to make a software product testable, the efficiency of the software is likely to go down.
  - Some quality factors positively impact others. For example, an effort to enhance the correctness of a system will increase its reliability.

Quality Metrics

- The high-level quality factors cannot be measured directly.
    - For example, we cannot directly measure the testability of a software system. Neither can testability be expressed in “yes” or “no” terms. Instead, the degree of testability can be assessed by associating with testability a few quality metrics, namely, *simplicity*, *instrumentation*, *self-descriptiveness*, and *modularity*.
- A *quality metric* is a measure that captures some aspect of a quality criterion.
- One or more quality metrics should be associated with each criterion.

Quality Metrics

- The metrics can be derived as follows:
  - Formulate a set of relevant questions concerning the quality criteria and seek a “yes” or “no” answer for each question.
  - Divide the number of “yes” answers by the number of questions to obtain a value in the range of 0 to 1. The resulting number represents the intended quality metric.
- The above way of computing the value of a metric is highly subjective.
- It is difficult to combine different metrics to get a measure of a higher level quality factor.

Module 2 - Content Outline (Agenda)

- ◆ Software quality
- ◆ McCall’s quality model
- ➔ISO 9126 Quality Characteristics
- ◆ ISO 9000:2000 Software Quality Standard

Categories of quality characteristics

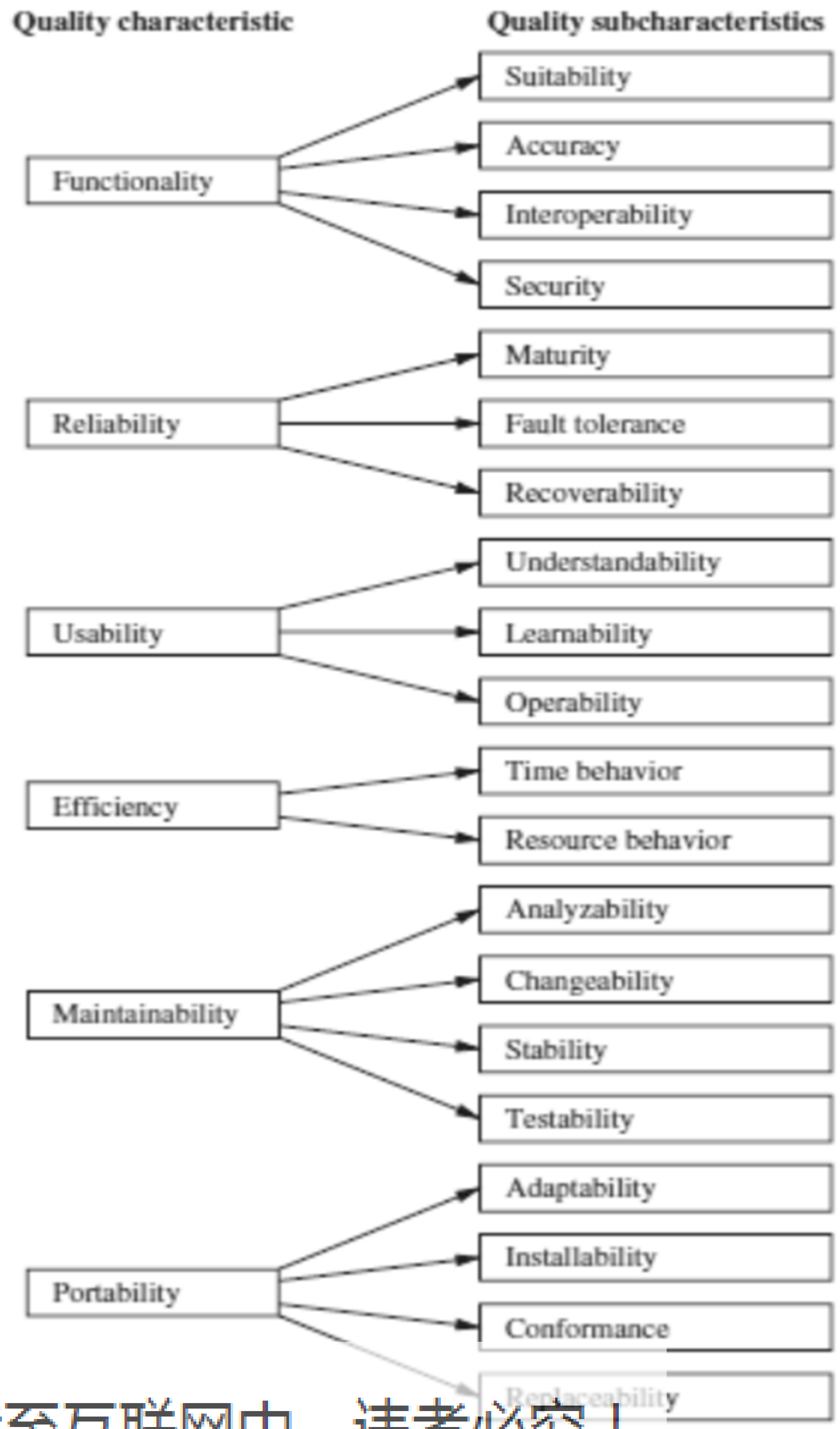
- *Functionality*: A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
- *Reliability*: A set of attributes that bear on the capability of software to maintain its performance level under stated conditions for a stated period of time.
- *Usability*: A set of attributes that bear on the effort needed for use and on the individual assessment of such use by a stated or implied set of users.

Categories of quality characteristics

- *Efficiency*: A set of attributes that bear on the relationship between the software’s performance and the amount of resource used under stated conditions.
- *Maintainability*: A set of attributes that bear on the effort needed to make specified modifications (which may include corrections, improvements, or adaptations of software to environmental changes and changes in the requirements and functional specifications).
- *Portability*: A set of attributes that bear on the ability of software to be transferred from one environment to another (this includes the organizational, hardware or, software environment).

An sample quality model

- decomposes the quality characteristics into more concrete subcharacteristics



ISO 9126 Quality Characteristics

- Organizations must define their own quality characteristics and subcharacteristics after a fuller understanding of their needs.
- In other words, organizations must identify the level of the different quality characteristics they need to satisfy within their context of software development.
- Reaching an ideally best quality level from the present one is a gradual process.
- Therefore, it is important to understand the need for moving on to the next achievable step toward the highest level—the ideally best level.

McCall’s quality model VS ISO 9126 model

- Since the two models focus on the same abstract entity (*software quality)*, there are many similarities between the two models.
- What is called *quality factor* in McCall’s model is called *quality characteristic* in the ISO 9126 model.
- The following high-level quality factors/characteristics are found in both models: reliability, usability, efficiency, maintainability, and portability.

McCall’s quality model VS ISO 9126 model

- There are several differences between the two models:
  - The ISO 9126 model emphasizes characteristics *visible* to the users, whereas the McCall model considers *internal* qualities as well.
  - In McCall’s model, one quality criterion can impact several quality factors, whereas in the ISO 9126 model, one subcharacteristic impacts exactly one quality characteristic.
  - A high-level quality factor, such as testability, in the McCall model is a low-level subcharacteristic of maintainability in the ISO 9126 model.

Module 2 - Content Outline (Agenda)

- ◆ Software quality
- ◆ McCall’s quality model
- ◆ ISO 9126 Quality Characteristics
- ➔ISO 9000:2000 Software Quality Standard

ISO 9000:2000 standard

- There are three components of the ISO 9000:2000 standard:
  - *ISO 9000* : Fundamentals and vocabulary
  - *ISO 9001* : Requirements
  - *ISO 9004* : Guidelines for performance improvements

ISO 9000:2000 Fundamentals

- The ISO 9000:2000 standard is based on the following eight principles:
  - **Customer Focus**
  - **Leadership**
  - **Involvement of People**
  - **Process Approach**
  - **System Approach to Management**
  - **Continual Improvement**
  - **Factual Approach to Decision Making**
  - **Mutually Beneficial Supplier Relationships**

ISO 9001:2000 Requirements

- Briefly describe five major parts of the ISO 9001:2000，that is, part 4-8
  - ***Part 4: Systemic Requirements***
  - ***Part 5: Management Requirements***
  - ***Part 6: Resource Requirements***
  - ***Part 7: Realization Requirements***
  - ***Part 8: Remedial Requirements***

Module 2 - Review

- What is Quality?
- Who are the Stakeholders?
- What is a Defect?
- What are Dimensions of Quality?
- What are the McCall’s quality factors and quality criteria?
- What are the ISO 9126 quality characteristics?
- What are the differences of McCall’s quality model and the ISO 9126 quality model?