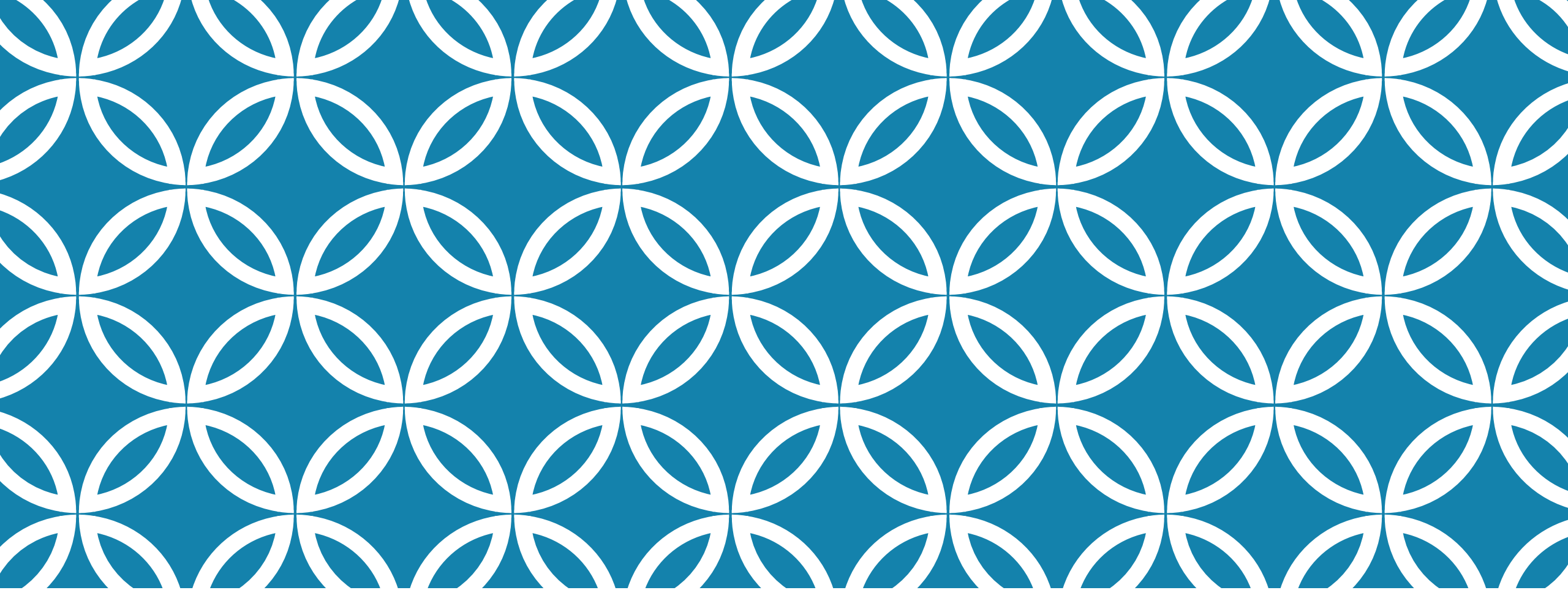


CHAPTER – 17
SOFTWARE QUALITY

OUTLINE OF THE CHAPTER

Five Views of Software Quality

McCall’s Quality Factors and Criteria

The ISO 9126 Quality Characteristics

Summary

FIVE VIEWS OF SOFTWARE QUALITY

Transcendental view

User view

Manufacturing view

Product view

Value-based view

FIVE VIEWS OF SOFTWARE QUALITY

Transcendental view

- Quality is something that can be recognized through experience, but not defined in some tractable form.
- A good quality object stands out, and it is easily recognized.

User view

- Quality concerns the extent to which a product meets user needs and expectations.
- Is a product fit for use?
- This view is highly personalized.
  - A product is of good quality if it satisfies a large number of users.
  - It is useful to identify the product attributes which the users consider to be important.
- This view may encompass many subject elements, such as *usability*, *reliability*, and *efficiency*.

FIVE VIEWS OF SOFTWARE QUALITY

Manufacturing view

- This view has its genesis in the manufacturing industry – auto and electronics.
- Key idea: Does a product satisfy the requirements?
  - Any deviation from the requirements is seen as reducing the quality of the product.
- The concept of process plays a key role.
- Products are manufactured “right the first time” so that the cost is reduced
  - Development cost
  - Maintenance cost
- Conformance to requirements leads to uniformity in products.
- Some argue that such uniformity does not guarantee quality.
- Product quality can be incrementally improved by improving the process.
  - The CMM and ISO 9001 models are based on the manufacturing view.

FIVE VIEWS OF SOFTWARE QUALITY

Product view

- Hypothesis: If a product is manufactured with good internal properties, then it will have good external properties.
- One can explore the causal relationship between *internal properties* and *external qualities*.
- Example: *Modularity* enables *testability*.

Value-based view

- This represents the merger of two concepts: *excellence* and *worth*.
- Quality is a measure of excellence, and value is a measure of worth.
- Central idea
  - How much a customer is willing to pay for a certain level of quality.
  - Quality is meaningless if a product does not make economic sense.
  - The value-based view makes a trade-off between cost and quality.

FIVE VIEWS OF SOFTWARE QUALITY

Measuring quality

- Reasons for developing a quantitative view of quality
- Measurement of user's view
- Measurement of manufacturing view

Reasons for developing a quantitative view of quality

- Measurement allows us to establish baselines for qualities.
- Measurement is key to process improvement.
- The needs for improvements can be investigated after performing measurements.

FIVE VIEWS OF SOFTWARE QUALITY

Measurement of user's view

- Functionalities can be easily measured.
  - What functionality test cases have passed?
  - Measure of delivered functionality = ratio of # of passed test cases to the total number of test cases designed to verify the functionalities.
- Apply Gilb's technique.
  - The quality concept is broken down into component parts until each can be stated in terms of directly measurable qualities.
  - Example: *Usability* can be broken down into
    - *Learnability*
    - *Understandability*
    - *Operability*

FIVE VIEWS OF SOFTWARE QUALITY

Measurement of manufacturer's view

- Manufacturers are interested in *defect count and rework cost.*
- *Defect count:* The total number of defects detected during development and operation.
  - It is a measure of the quality of the work produced.
  - One can analyze the defects as follows.
    - For each defect, identify the development phase in which it was introduced.
    - Categorize the defects, say, based on modules.
    - Normalize defect count by product size.
      - Defect density: Number of defects per 1000 lines of code.
    - Separate the defects found during operation from those found during development.
- *Rework cost:* How much does it cost to fix the known defects?
  - Development rework cost: This is the rework cost incurred *before* a product is released. This is a measure of *development efficiency*.
  - Operation rework cost: This is the rework cost incurred *when* a product is in operation. This is a measure of the *delivered quality*.

MCCALL'S QUALITY FACTORS AND CRITERIA

Quality Factors

- McCall, Richards, and Walters studied the concept of software quality in terms of two key concepts as follows:
  - *quality factors*, and
  - *quality criteria*.
- A quality factor represents the behavioral characteristic of a system.
  - Examples: correctness, reliability, efficiency, testability, portability, …
- A quality criterion is an attribute of a quality factor that is related to software development.
  - 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.
- McCall et al. identified 11 quality factors (Table 17.1.)

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].

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 Criteria

- A quality criterion is an attribute of a quality factor that is related to software development.
- 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.
- In Table 17.3, we have listed 23 quality criteria.

MCCALL'S QUALITY FACTORS AND CRITERIA

Quality Criteria	Definitions of Quality Criteria
Access audit	The ease with which software and data can be checked for compliance with standards or other requirements.
Access control	The provisions for control and protection of the software and data.
Accuracy	The precisions of computations and output.
Communication commonality	The degree to which standard protocols and interfaces are used.
Completeness	The degree to which a full implementation of the required functionalities has been achieved.
Communicativeness	The ease with which inputs and outputs can be assimilated.
Conciseness	The compactness of the source code, in terms of lines of code.
Consistency	The use of uniform design and implementation techniques and notations throughout a project.
Data commonality	The use of standard data representations.
Error tolerance	The degree to which continuity of operation is ensured under adverse conditions.
Execution efficiency	The run-time efficiency of the software.
Expandability	The degree to which storage requirements or software functions can be expanded.
Generality	The breadth of the potential application of software components.
Hardware independence	The degree to which the software is dependent on the underlying hardware.
Instrumentation	The degree to which the software provides for measurements of its use or identification of errors.
Modularity	The provision of highly independent modules.
Operability	The ease of operation of the software.
Self-documentation	The provision of in-line documentation that explains the implementation of components.
Simplicity	The ease with which the software can be understood.
Software system independence	The degree to which the software is independent of its software environment – non-standard language constructs, operating system, libraries, database management system, etc.
Software efficiency	The run-time storage requirements of the software.
Traceability	The ability to link software components to requirements.
Training	The ease with which new users can use the system.

Table 17.3: McCall's quality criteria [10].

MCCALL'S QUALITY FACTORS AND CRITERIA

Relationship Between Quality Factors and Quality Criteria

- Each quality factor is positively influenced by a set of quality criteria, and the same quality criterion impacts a number of quality factors.
  - Example: Simplicity impacts reliability, usability, and testability.
- If an effort is made to improve one quality factor, another quality factor may be degraded.
  - Portable code may be less efficient.
- Some quality factors positively impact others.
  - An effort to improve the correctness of a system will increase its reliability.
- See Figure 17.1.

MCCALL'S QUALITY FACTORS AND CRITERIA

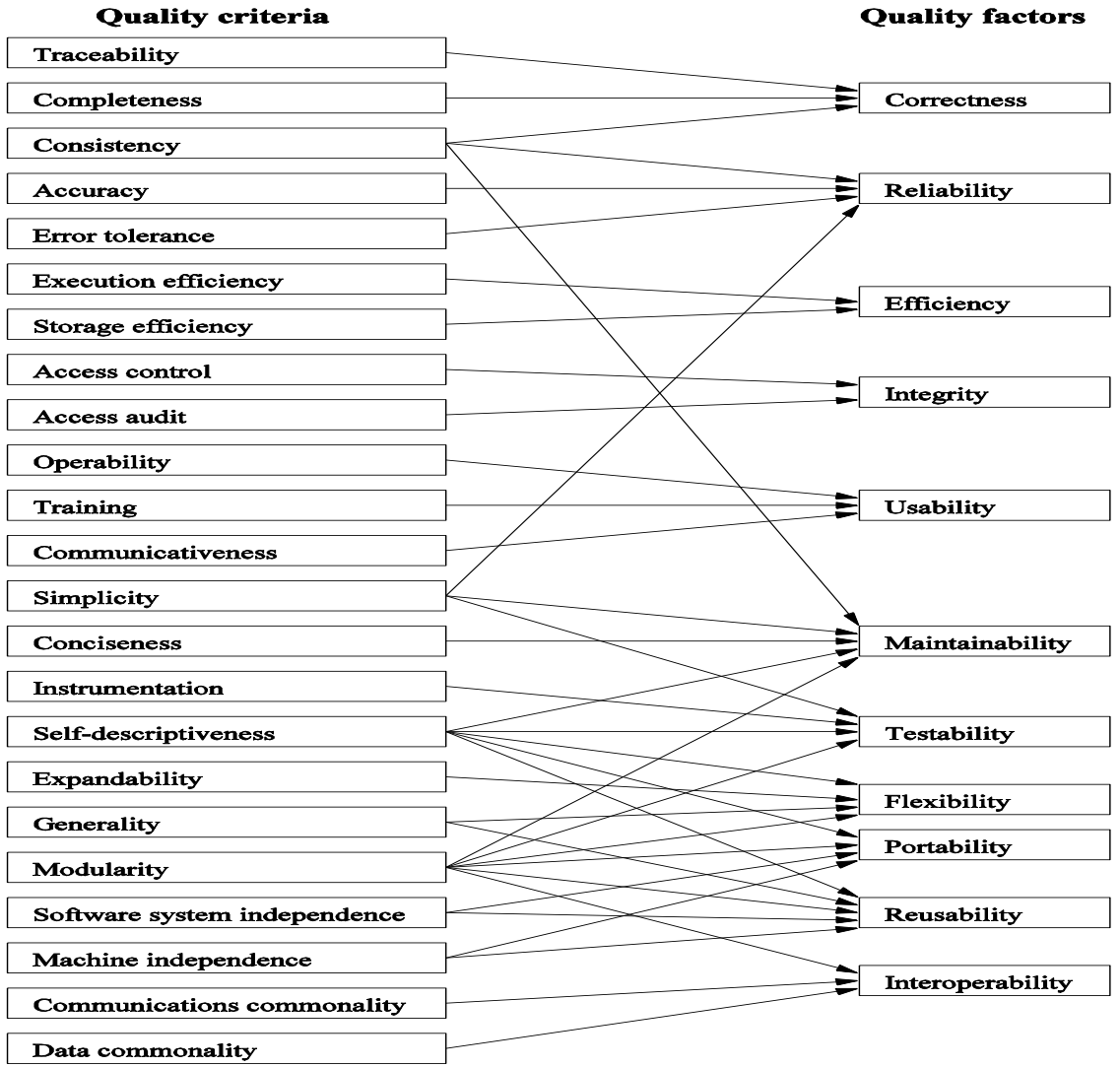


Figure 17.1: Relation between quality factors and quality criteria [10].

THE ISO 9126 QUALITY CHARACTERISTICS

The ISO 9126 document is the product of an international effort.

- ISO: International Organization for Standardization

The document defines six broad quality characteristics as shown in Table 17.4.

The document includes an example quality model (Figure 17.2) that further decomposes the quality characteristics.

- Figure 17.2 is just an example, and not a universal one.
- The 20 sub characteristics of Figure 17.2 have been defined in Table 17.5.

THE ISO 9126 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.

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.)

Table 17.4: ISO 9126 quality characteristics.

THE ISO 9126 QUALITY CHARACTERISTICS

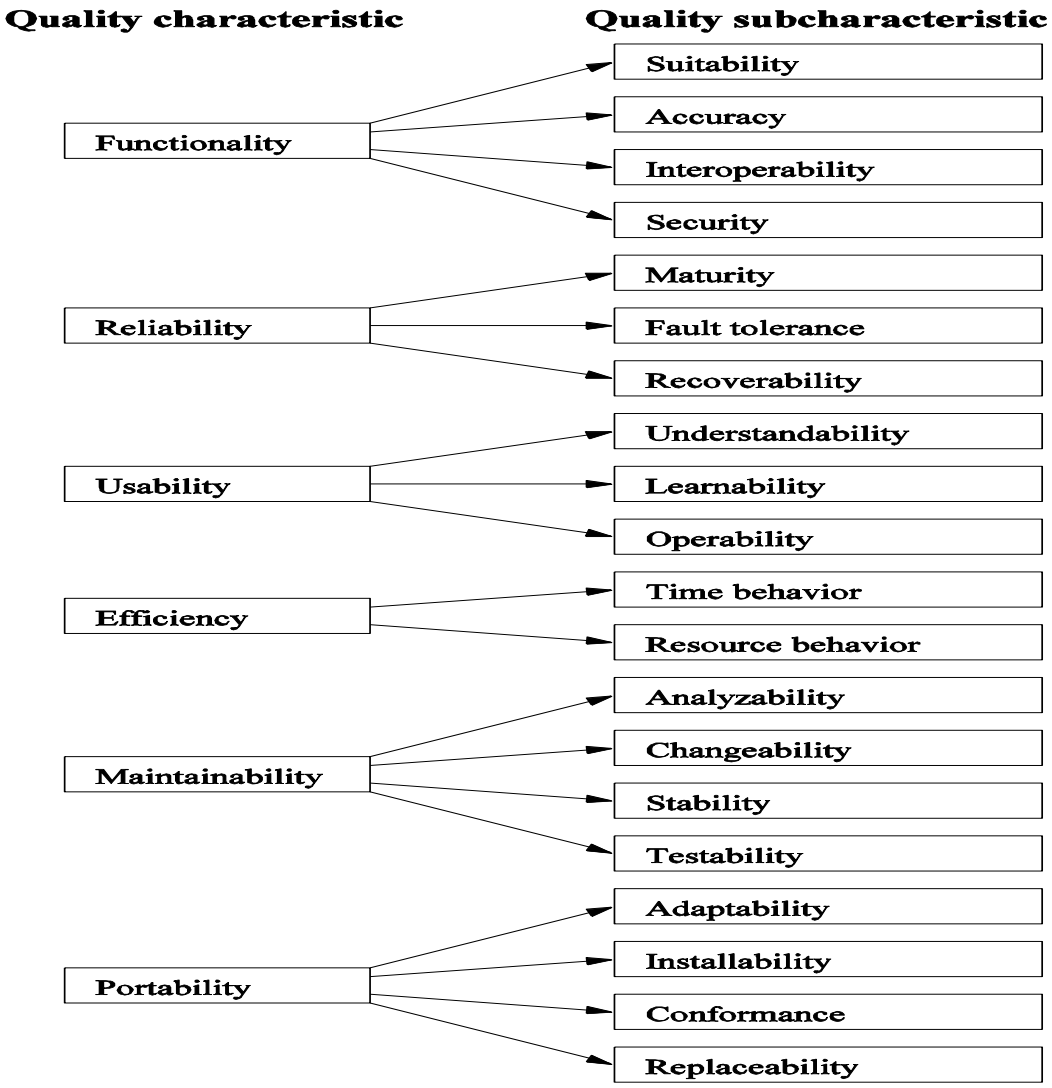


Figure 17.2: The ISO 9126 sample quality model refines the standard's features into sub characteristics [9]. (©[1996] IEEE)

THE ISO 9126 QUALITY CHARACTERISTICS

Suitability: The capability of the software to provide an adequate set of functions for specified tasks and user objectives.

Accuracy: The capability of the software to provide the right or agreed results or effects.

Interoperability: The capability of the software to interact with one or more specified systems.

Security: The capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorized access to confidential information, or to make unauthorized modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.

Maturity: The capability of the software to avoid failure as a result of faults in the software.

Fault tolerance: The capability of the software to maintain a specified level of performance in case of software faults or of infringement of its specified interface.

Recoverability: The capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.

Understandability: The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.

Learnability: The capability of the software product to enable the user to learn its applications.

Operability: The capability of the software product to enable the user to operate and control it.

Attractiveness: The capability of the software product to be liked by the user.

Time behavior: The capability of the software to provide appropriate response and processing times and throughput rates when performing its function under stated conditions.

Resource utilization: The capability of the software to use appropriate resources in an appropriate time when the software performs its function under stated condition.

Analyzability: The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.

Changeability: The capability of the software product to enable a specified modification to be implemented.

Stability: The capability of the software to minimize unexpected effects from modifications of the software.

Testability: The capability of the software product to enable modified software to be validated.

Adaptability: The capability of the software to be modified for different specified environments without applying actions or means other than those provided for this purpose for the software considered.

Installability: The capability of the software to be installed in a specified environment.

Co-existence: The capability of the software to co-exist with other independent software in a common environment sharing common resources.

Replaceability: The capability of the software to be used in place of the other specified software in the environment of that software.

Table 17.5: Quality subcharacteristics of ISO 9126.

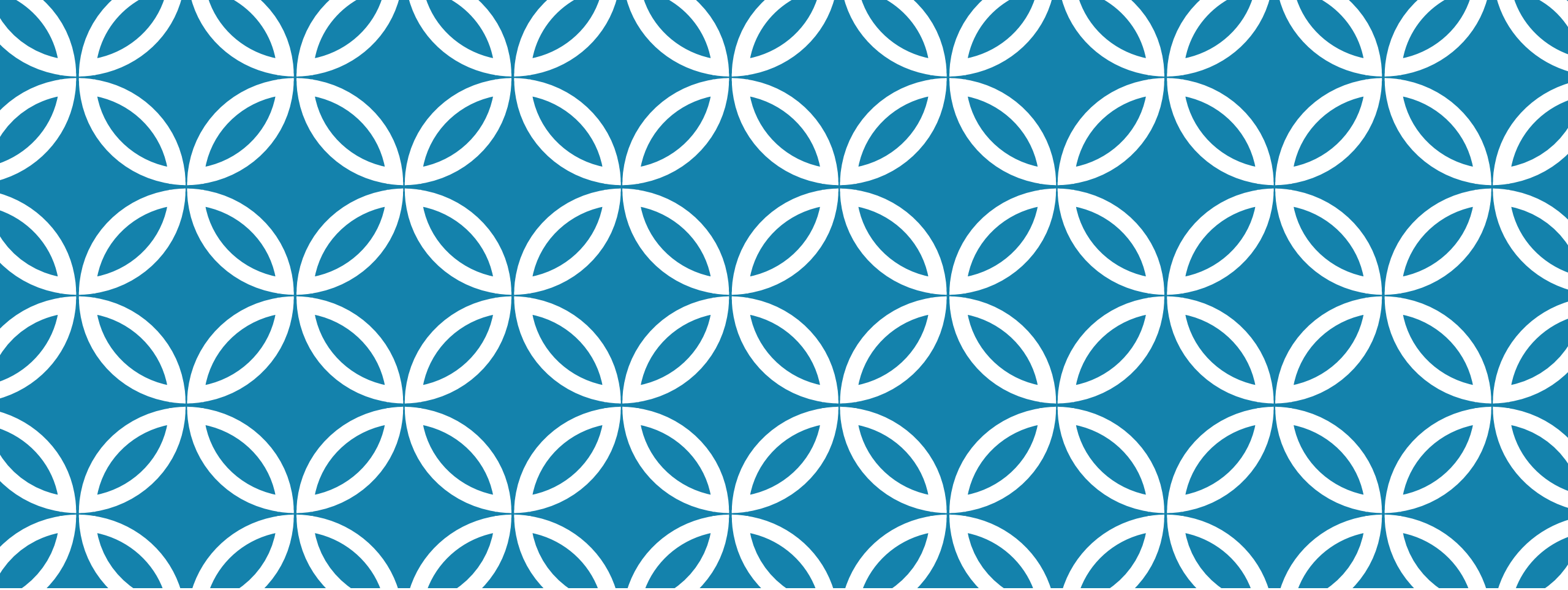
THE ISO 9126 QUALITY CHARACTERISTICS

McCall's quality model vs. ISO 9126 model

- What is called *quality factor* in McCall's model is called a *quality subcharacteristic* in the ISO 9126 model.
- The following quality factors/characteristics are found in both the models.
  - reliability, usability, efficiency, maintainability, and portability
- Differences between the two models
  - The ISO 9126 model emphasizes on characteristics *visible* to the users, whereas the McCall model considers *internal* qualities as well.
  - In McCall's model, *one* quality criterion can impact *many* 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's model is a low-level subcharacteristic of *maintainability* in the ISO 9126 model.

Concerns

- There is no consensus about what high-level quality factors are most important at the top level. McCall, et al. define 11 high-level quality factors, whereas there are only *six* in the ISO 9126 document.
- There is no consensus regarding what is a top-level quality factor/ characteristic and what is a more concrete quality criterion/ subcharacteristic.



THANK YOU!!