

# **CSE 4621 / SWE 5620**

## **Software Metrics**

# **Measuring External Product Attributes: Software Quality**

## **Chapter -10-**

**Khaled Slhoub, PhD**

- Principal objective of software engineering is to **improve** the **quality of software products**.
- Quality, like beauty, is very much in the eyes of the beholder.
- Software quality is a multidimensional concept:
  - Fitness for purpose
  - Conformance to specification
  - Degree of excellence
  - Timeliness
  - Reliability
  - ...




- From a measurement perspective, we must be able to define quality in terms of specific software product attributes of interest to the user
- So, we need to know how to measure the extent to which these attributes are present in our software products
- Many practitioners and researchers measure and analyze internal attributes because they may be predictors of external attributes.

**Quality is really a composite of many characteristics**

- General Structure

User Oriented

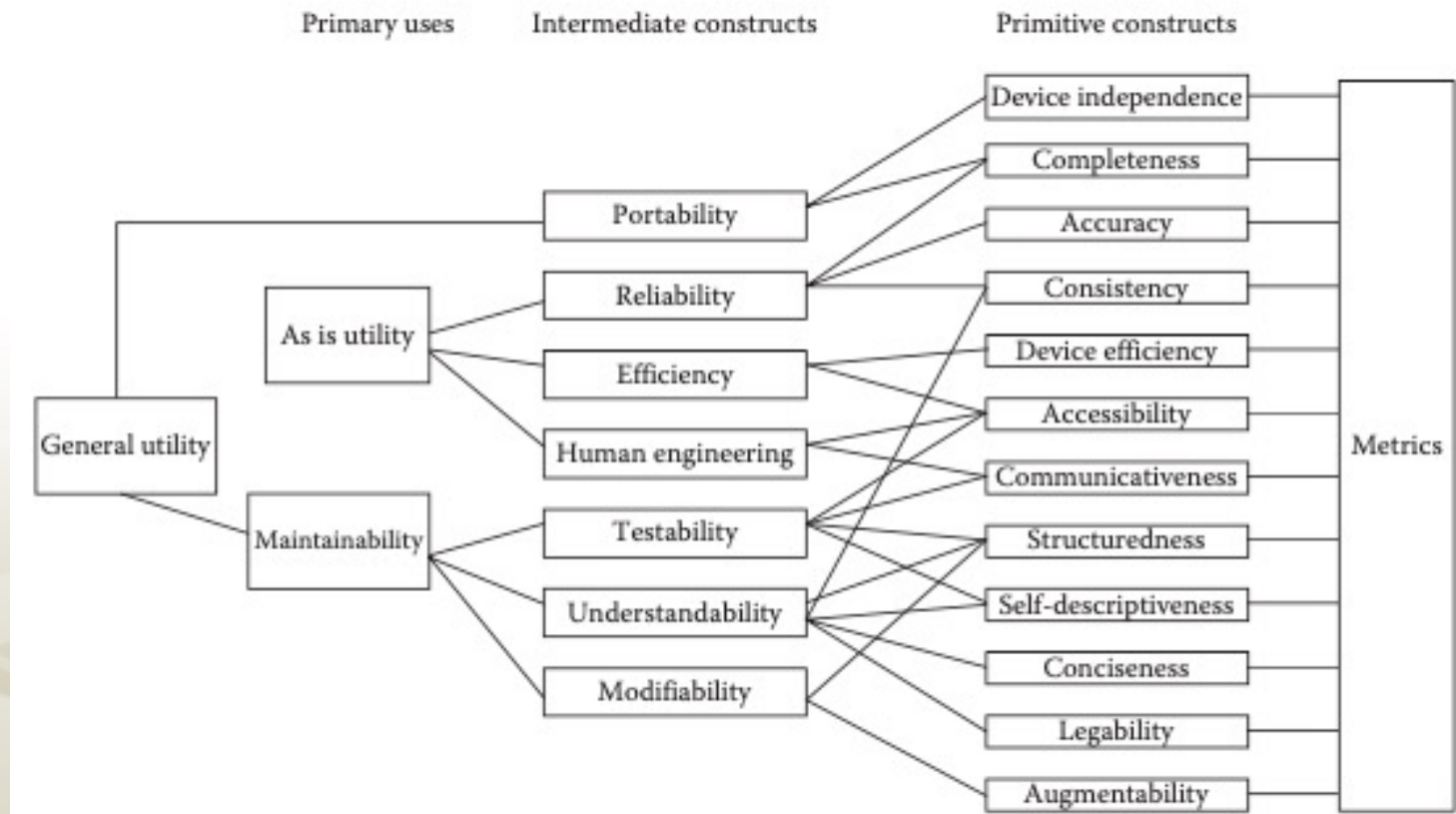


Software Oriented



- Boehm's Model

- focus on the final product ((usually the executable code)
- quality defined from the user's perspective



- McCall's Model (developed for the U.S. Air Force)

**McCall quality model is organized around three types of Elements:**

- **Quality Factors** (To specify):

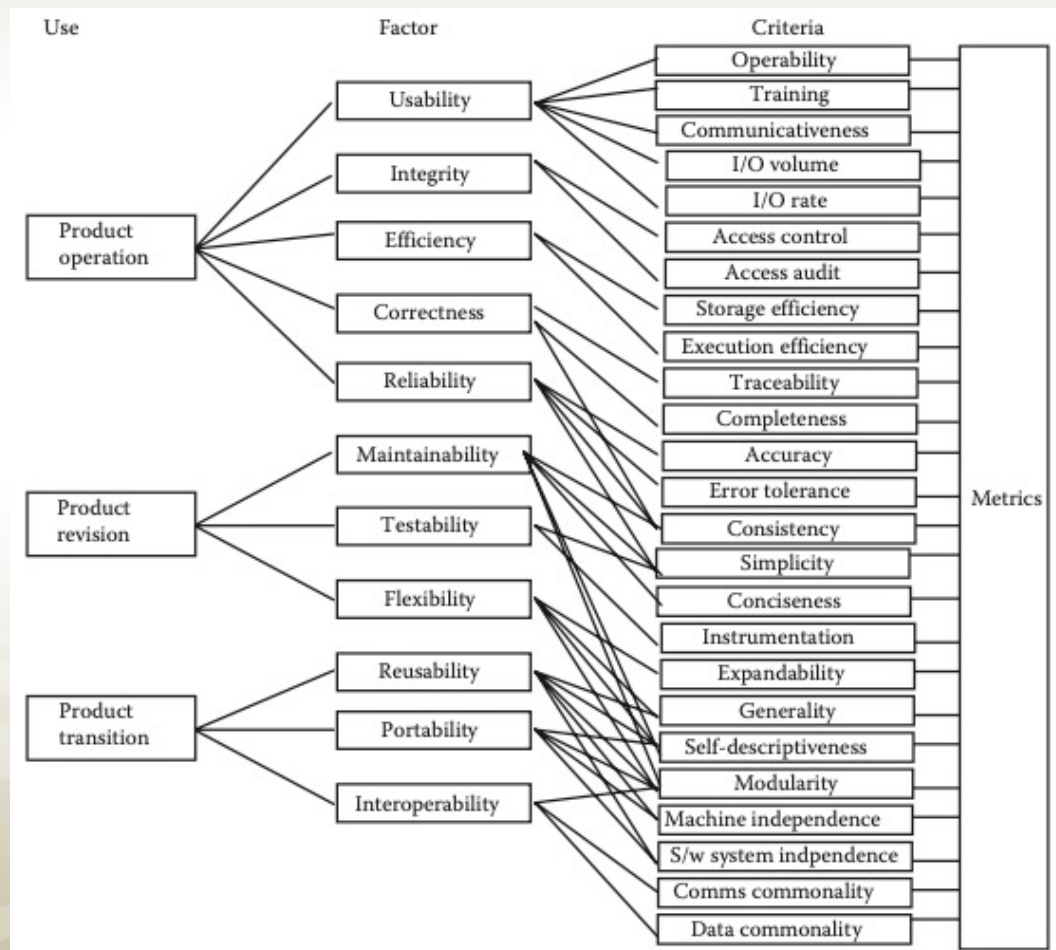
They describe the external view of the software, as viewed by the users.

- **Quality Criteria** (To build):

They describe the internal view of the software, as seen by the developer.

- **Quality Metrics** (To control):

They are defined and used to provide a scale and method for measurement.





- IBM's Model

- Measures user satisfaction in eight factors
- Some of these factors conflict with each other, and some support each other.

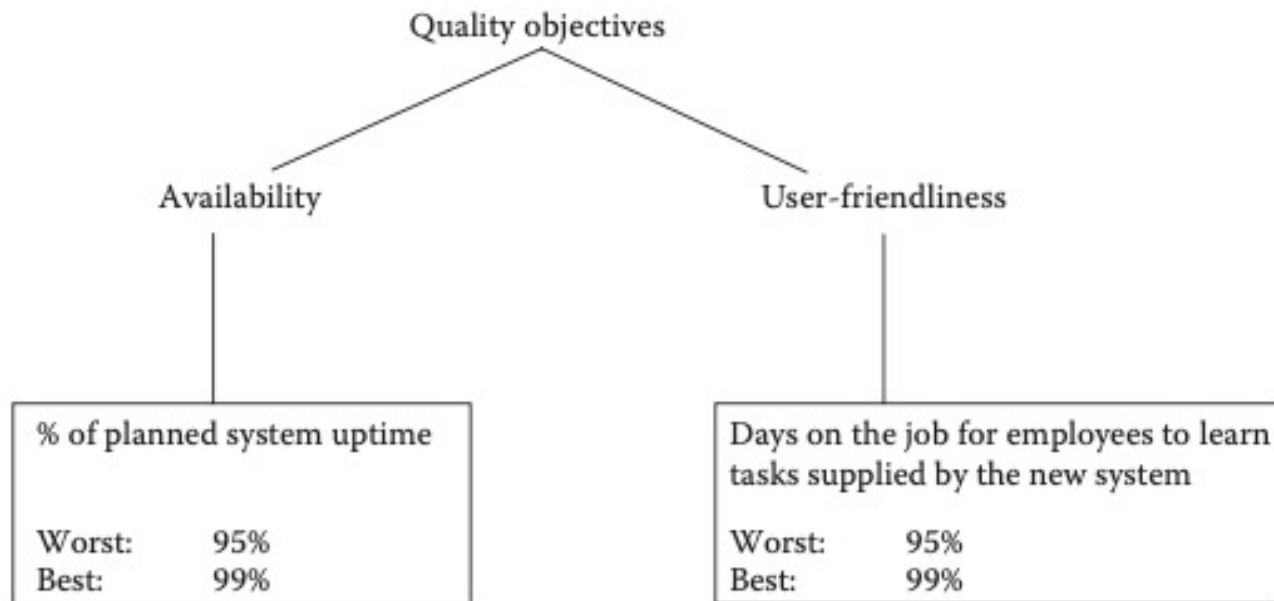
	Capability	Usability	Performance	Reliability	Instability	Maintainability	Documentation	Availability
Capability								
Usability								
Performance	●	●						
Reliability	●	○	●					
Instability		○	○	○				
Maintainability	●	○	●	○				
Documentation	●	○				○		
Availability	●	○	○	○	○	○		

●: Conflict One Another

○: Support One Another

Blank: Not Related

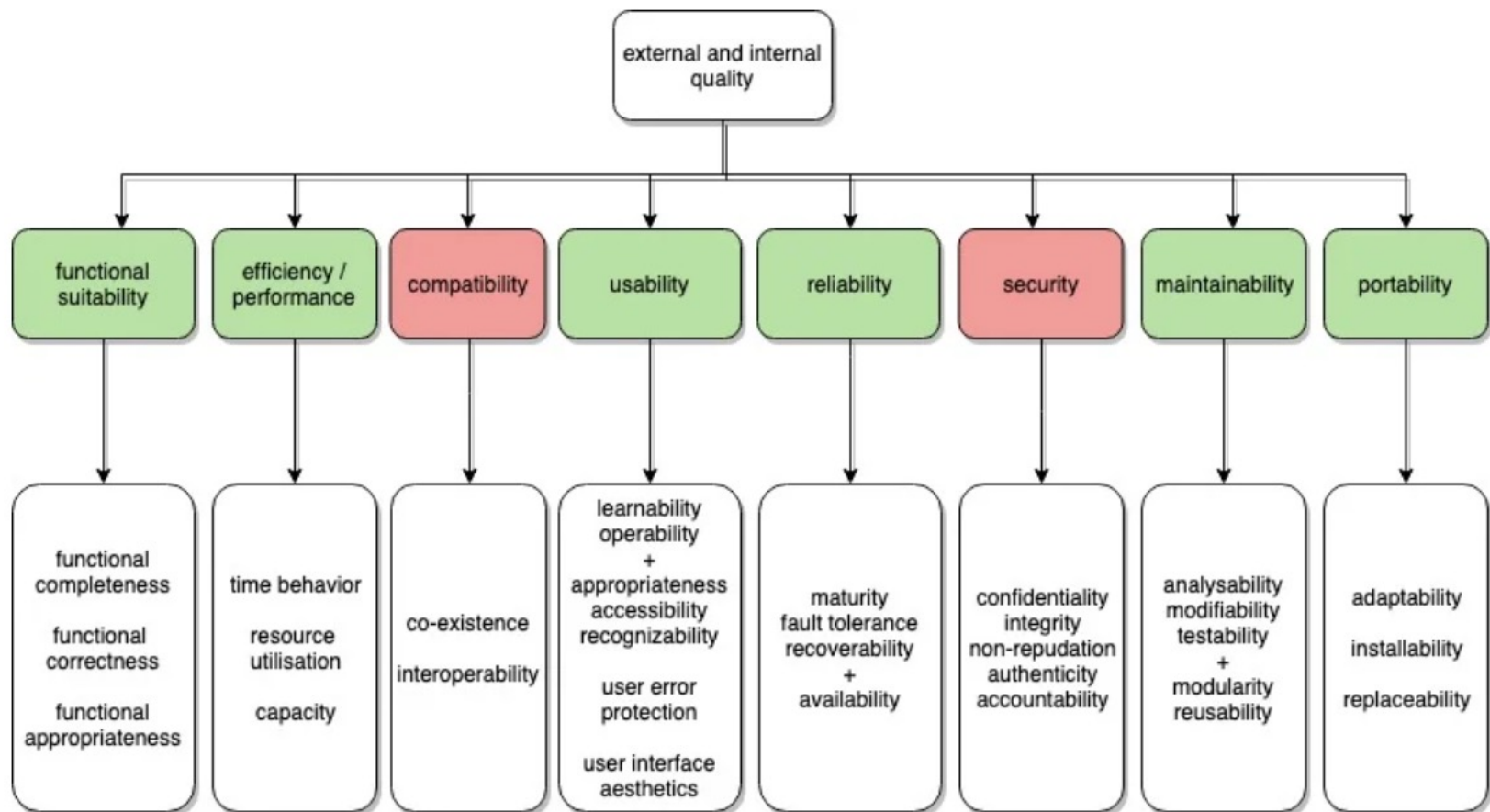
- Gilb's Method
  - Define-Your-Own Quality Model
  - Design by measurable objectives





- ISO/IEC 9126 and ISO/IEC 25010 Standard Quality Models
  - universal model for depicting and expressing quality
  - it makes it easier to compare one product with another
  - In the ISO/IEC 25010 standard, software quality is defined to be the following: *“The degree to which a software product satisfies stated and implied needs when used under specified conditions.”*

- ISO/IEC 25010 Standard Quality Models (2011)
  - Was last reviewed and confirmed in 2017



- ISO/IEC 25010 Standard Quality Models (2011)
  - Was last reviewed and confirmed in 2017

In ISO/IEC 25010:2011, *reliability* is defined as the

degree to which a system, product or component performs specified functions under specified conditions for a specified period of time...

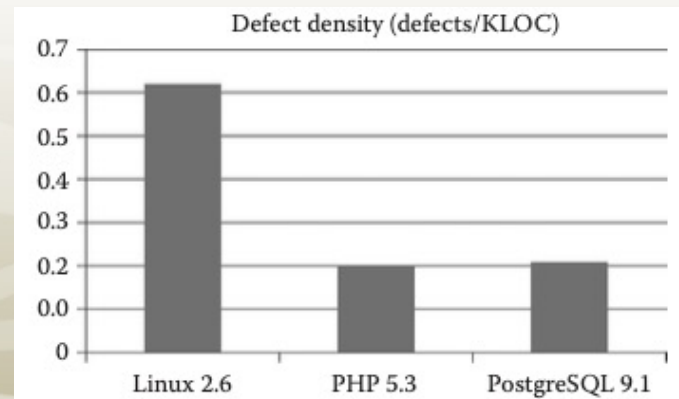
while *portability* is defined as the

degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

- Many software engineers think of software quality in a much narrower sense
- Where quality is considered *only to be a lack of defects*. Here, “defect” is interpreted to mean a known error, fault, or failure
- **Defect Density Measures**

$$\text{Defect density} = \frac{\text{Number of known defects}}{\text{Product size}}$$

The number of defects per thousand source statements (KNCSS) in three well-known open source systems: Linux 2.6 (6849 KLOC), PHP 5.3 (538 KLOC), and PostgreSQL 9.1 (1106 KLOC). The Figure shows how the defect density can vary (Coverity 2011).



- there is no general consensus on what constitutes a defect.
- To use defect density as a comparative measure, you must be sure that all parties are counting the same things in the same ways.
- there is no consensus about how to measure software size in a consistent and comparable way
- Defect density may tell us more about the quality of the defect-finding and defect-reporting process than about the quality of the product itself.
- Even if we were able to know exactly the number of residual faults in our system, we would have to be extremely careful about making definitive statements about how the system will operate in practice.

- Standard software quality measures used at AT&T Bell Laboratories
  - Cumulative fault density—faults found internally
  - Cumulative fault density—faults found by customers
  - Total serious faults found
  - Mean time to close serious faults
  - Total field fixes
  - High-level design review errors per thousand NCLOC
  - Low-level design errors per thousand NCLOC
  - Code inspection errors per inspected thousand NCLOC
  - Development test and integration errors found per thousand NCLOC
  - System test problems found per developed thousand NCLOC
  - First application test site errors found per developed thousand NCLOC
  - Customer found problems per developed thousand NCLOC

- **System Spoilage**

$$\text{System spoilage} = \frac{\text{Time to fix post - Release defects}}{\text{Total system development time}}$$

Tajima and Matsubara described quality improvements in the 1970s at Hitachi in terms of this measure.





- Other software quality measures:
  - Usability Measures
  - Maintainability Measures
  - Security Measures
- **The above will presented by students**