Software Quality Assurance
Lecture 3

Software Quality

Quality

“Quality is never an accident; it is always the result of intelligent effort.”

— John Ruskin

Software Quality

- What is software quality?

➔Many different answers, depending on whom you ask, under what circumstances, for what kind of software systems, and so on…

- Alternative question: "What are the characteristics for high-quality software?"

➔ Need to examine different perspectives and expectations of user as well as other people involved with the development, management, marketing, and maintenance of the software products.

Quality: Perspectives and Expectations

- General: “good” software quality
- Perspectives:

  People/subject’s view, software as object
- Expectations: quality characteristics & level

Quality Perspectives

- Perspectives: subject and object
- **Subject**: *people's* perspectives
  - External/Consumer: customers and users
  - Internal/Producer: developer, testers, and managers
  - Other: 3rd party, indirect users, etc.
- **Objects** of our study:
  - *Software* products, systems, and services

Quality Perspectives

- **External** View → mostly sees a software system as a **black box**, where one can observe its behavior but not see through inside.

- **Internal** View → mostly sees a software system as a **white box**, or more appropriately a clear box, where one can see what is inside and how it works.

Quality Expectations

- Expectations from different people:
- External/Consumer expectations:
  - “good enough” for the price
    1) Fit-for-use, doing the “right things”
    2) Conformance, doing the “things right”

    → Validation and verification (V & V)
  - Customer vs. user (price?)
- Expectations for different software:
  - General: functionality & reliability
  - Usability: GUI/end-user/web/etc.
  - Safety: safety-critical systems

Quality Expectations

- Internal/Producer expectations:
- "good enough" for the cost
  - Mirror consumer side
  - Functionality & correctness via V&V
- Cost: developers vs. managers
- Service related: maintainability
- Interfacing units: interoperability
- 3rd party: modularity

Five Views of Software Quality

In Kitchenham & Pfleeger (1996):

1) Transcendental view
2) User view
3) Manufacturing view
4) Product view
5) Value-based view

Transcendental view

1) Transcendental View ==> seen/not-defined

- 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

2) **User view** **==>** fitness for purpose/meeting user's needs

- Quality concerns *the extent to which a product meets user needs and expectations*.
- Is a product fit for use?
- 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, e.g. *usability, reliability*, *efficiency, testability*

Manufacturing view

3) Manufacturing view ==> Conformance to process standards/requirements

- This view has its genesis in the manufacturing industry – auto and electronics.
- Key idea: Does a product satisfy the requirements?
  - Quality is seen as conforming to 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.

Product view

4) **Product view ==>** Focus is on inherent characteristics in product

- *Hypothesis: If a product is manufactured with good internal properties, then it will have good external properties.*
- 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.*
- The product view of quality can be assessed in an objective manner.
- 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***.

Value-based view

5) Value-based view: Customers' willingness to pay for a software

- Value-based view 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.

Why Measuring Quality?

- Measuring quality **==>** Measurement allows us to have a quantitative view of the quality concept.
- *What are the reasons for developing a quantitative view of quality?*
  1) Baseline: Measurement allows us to establish baselines for qualities.
  2) Quality improvement based on cost: Organizations make continuous improvements in their process models –and an improvement has a cost associated with it. Measurement is key to process improvement.
  3) Know the present level for future planning : The needs for improvements can be investigated after performing measurements.

McCall's Quality Factors and Criteria

- The concept of software quality and the efforts to understand it in terms of measurable quantities date back to the mid-1970s.
- McCall, Richards, and Walters were the first to study the concept of software quality in terms of quality factors and quality criteria.
- McCall, Richards, and Walters studied the concept of software quality in terms of two key concepts as follows:
    - *Quality factors*
    - *Quality criteria*

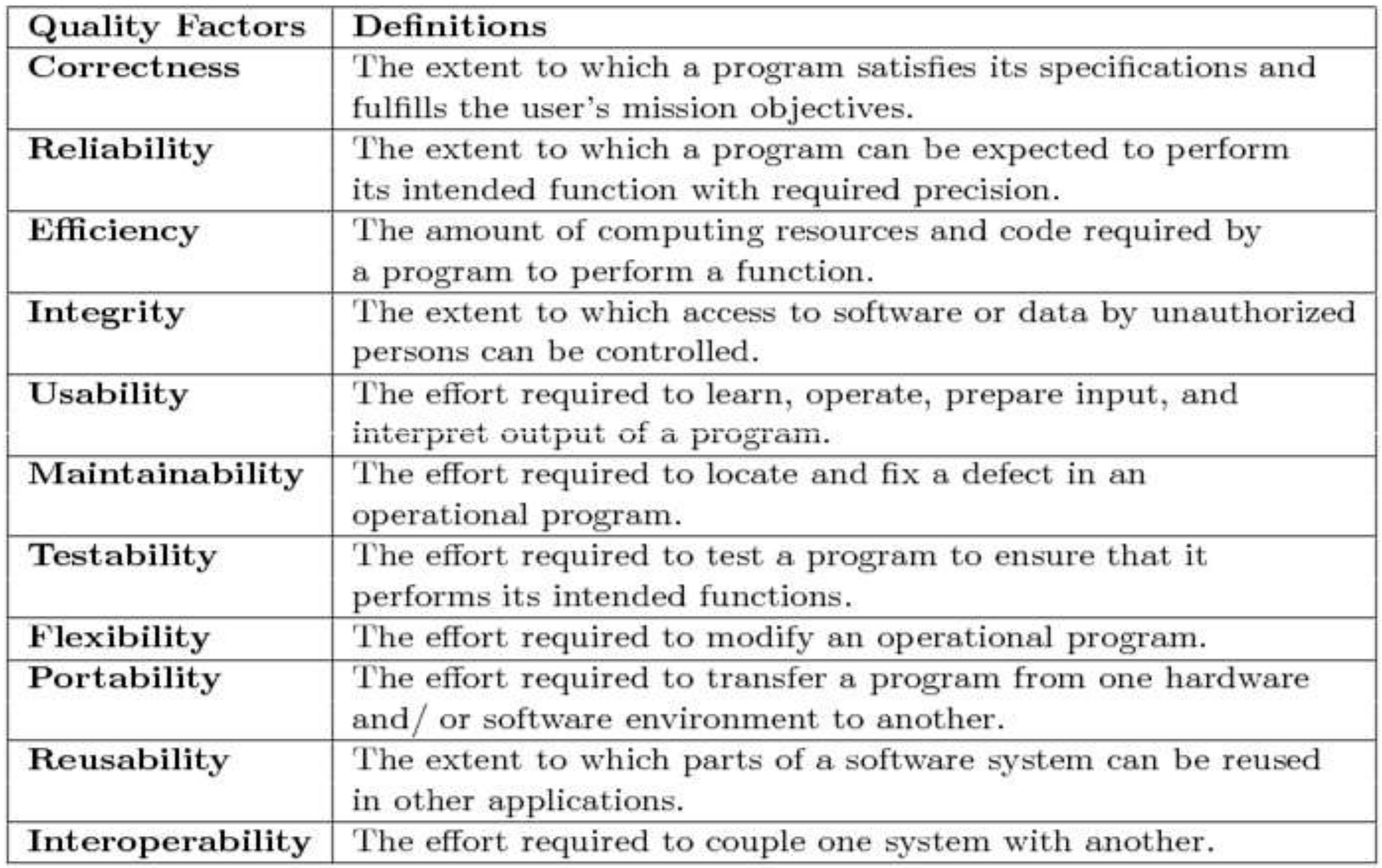
McCall’s Quality Factors and Criteria

- **Quality Factor**: A *quality factor* represents the *behavioral characteristic* of a system.
- Examples:
  - Correctness
  - Reliability
  - Efficiency
  - Testability
  - Portability

McCall’s Quality Factors and Criteria

- **<u>Quality Criteria</u>:** 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* and *23 quality criteria.*

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.

ISO-9126 Quality Framework

- **ISO ==> International Organization for Standardization**
- **ISO-9126**:
  - ISO-9126 is an international standard for the evaluation of software.
  - The mostly influential one in the software engineering community today.
  - Provides a hierarchical framework for quality definition, organized into quality characteristics and sub-characteristics
  - *Six top-level quality characteristics*, each associated with its own exclusive(non-overlapping) sub-characteristics

ISO-9126 Quality Framework

- The ISO-9126 software quality model identifies **6 main quality characteristics :**
  1) Functionality
  2) Reliability
  3) Usability
  4) Efficiency
  5) Maintainability
  6) Portability

ISO-9126 Quality Framework

1) Functionality ==> what is needed?

- 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.
- The sub-characteristics include:
  - Suitability
  - Accuracy
  - Interoperability
  - Security

ISO-9126 Quality Framework

2) Reliability **==>** function correctly

- A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
- The sub-characteristics include:
  - Maturity
  - Fault tolerance
  - Recoverability

ISO-9126 Quality Framework

3) Usability: **==>** effort to use

- 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.
- The sub-characteristics include:
  - Understandability
  - Learn ability
  - Operability

ISO-9126 Quality Framework

4) Efficiency ==> resource needed

- A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
- The sub-characteristics include:
  - Time behavior
  - Resource behavior

ISO-9126 Quality Framework

5) Maintainability ==> correct/adapt/improve

- A set of attributes that bear on the effort needed to make specified modifications.
- The sub-characteristics include:
  - Analyzability
  - Changeability
  - Stability
  - Testability

ISO-9126 Quality Framework

6) Portability ==> one environment to another

- A set of attributes that bear on the ability of software to be transferred from one environment to another.
- The sub-characteristics include:
  - Adaptability
  - Install ability
  - Conformance
  - Replace ability

Other Quality Frameworks

- Adaptation of ISO-9126:
  - Customized for companies
    - e.g. IBM's CUPRIMDSO(Capability, Usability, Performance, Reliability, Installation, Maintenance, Documentation, Service, Overall customer satisfaction)
  - Adapted to application domains
    - Reliability, usability, security for Web
- Other quality frameworks/mega-models besides the ISO-9126:
  - SEI/CMMI: Process focus/levels
  - McCall: Factors, Criteria
  - Basili: GQM (goal-question-metric)
  - Dromey: Component reflects Q-attributes

Correctness, Defect and Quality

- **High quality→ low defect**
  - intuitive notion related to correctness
  - quality problem →defect impact
  - widely accepted, but need better definitions
- **Defect/bug definition**
  - Failure: external behavior → *deviation from expected behavior*
  - Fault: internal characteristics → *cause for failures*
  - Error: *incorrect/missing human action*
  - Defect: error, fault, failure collectively
  - **Bug**: Software problems/defects, never precisely defined
- **Relations**: ***Errors*** ==> ***Faults*** ==> ***Failures*** (not necessarily 1-1)

Summary

- So, what is software quality?
- Software quality may include many different attributes and may be defined & perceived differently based on people’s different roles and responsibilities.

  ==> Many aspects/perspective, but correctness-centered in Software Quality Engineering (SQE)