Software Quality

Course Code: CSC4133 Course Title: Software Quality and Testing

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**3**	**Week No:**	**2**	**Semester:**	
Lecturer:	*Prof. Dr. kamruddin Nur, kamruddin@aiub.edu*				

Lecture Outline

- Software Quality
- Quality: Perspectives and Expectations
- Different Views of Software Quality
- Why Measuring Quality?
- McCall’s Quality Factors and Criteria
- The ISO-9126 Quality Framework

Objectives and Outcomes

- **Objectives**: To understand the term software quality, different perspectives and expectations of quality, different views of quality, the need for measuring quality, McCall's Quality Factors and Criteria, The ISO-9126 Quality Framework.

- **Outcomes**: Students are expected to be able to explain software quality and different perspectives and expectations of quality, be able to explain the five views of quality, be able to explain the major six quality characteristics of ISO 9126 model.

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
- **<u>Subject</u>**: *people's* perspectives
  - <u>External/Consumer</u>: customers and users
  - <u>Internal/Producer</u>: developer, testers, and managers
  - Other: 3rd party, indirect users, etc.
- **<u>Objects</u>** 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

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
- 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 **CMMI** 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

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

McCall’s Quality 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.
- McCall et al. identified *11 quality factors* and *23 quality criteria.*

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

Six Quality Characteristics of ISO 9126

- The ISO-9126 software quality model identifies 6 main quality characteristics :
  1) **Functionality** ==> what is needed?
  2) **Reliability** **==>** function correctly
  3) **Usability** **==>** effort to use
  4) **Efficiency** ==> resource needed
  5) **Maintainability** ==> correct/adapt/improve
  6) **Portability** ==> one environment to another

Books

- Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement, by Jeff Tian

References

1. *Software Testing and Quality Assurance: Theory and Practice*, by Kshirasagar Naik, Priyadarshi Tripathy
2. *Software Quality Assurance: From Theory to Implementation*, by Daniel Galin
3. *Software Testing and Continuous Quality Improvement*, by William E. Lewis
4. *The Art of Software Testing*, by Glenford J. Myers, Corey Sandler and Tom Badgett
5. *Software Testing Fundamentals: Methods and Metrics* by Marnie L. Hutcheson