Unit IV: Software Testing

Unit IV: Software Testing

CHAPTER 19: QUALITY CONCEPTS

19.1 What Is Quality?

19.2 Software Quality

19.2.2 McCall's Quality Factors

Chapter Preamble

- In 2005, *ComputerWorld* [Hil05] lamented that
  - "bad software plagues nearly every organization that uses computers, causing lost work hours during computer downtime, lost or corrupted data, missed sales opportunities, high IT support and maintenance costs, and low customer satisfaction.
- A year later, *InfoWorld* [Fos06] wrote about the
  - "the sorry state of software quality" reporting that the quality problem had not gotten any better.
- Today, software quality remains an issue, but who is to blame?
  - *Customers blame developers, arguing that sloppy practices lead to low-quality software.*
  - *Developers blame customers (and other stakeholders*), arguing that irrational delivery dates and a continuing stream of changes force them to deliver software before it has been fully validated.

19.1 What is Quality

- *American Heritage Dictionary* defines *quality* as
  - "a characteristic or attribute of something."

- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.

Relationship between parameters as a equation

User satisfaction = compliant product + good quality + delivery within budget and schedule

19.2 Software Quality

- Software quality can be defined as:
  - *An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.*

- **Definition serves to emphasize three important points:**

19.2 Software Quality: a) Effective Software Process

- Establishes infrastructure that supports any effort at building a high quality software product.
- Management aspects of process create the checks and balances that help avoid project chaos.
- Software engineering practices allow the developer to analyze the problem and design a solid solution.
- Umbrella activities (change management and technical reviews have as much to do with quality).

19.2 Software Quality: b) Useful Product

- A *useful product* delivers the content, functions, and features that the end-user desires.
- But as important, it delivers these assets in a reliable, error free way.
- A useful product always satisfies those requirements that have been explicitly stated by stakeholders.
- In addition, it satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

19.2 Software Quality: c) Adding Value

- By *adding value for both the producer and user* of a software product, high quality software provides benefits for the software organization and end-user.
- Software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- User gains added value because software provides a useful capability in a way that expedites some business process.

- The end result is:
  - (1) greater software product revenue,
  - (2) better profitability when an application supports a business process, and/or
  - (3) improved availability of information that is crucial for the business.

19.2 Software Quality: 19.2.2 Mc Call's Quality Factors

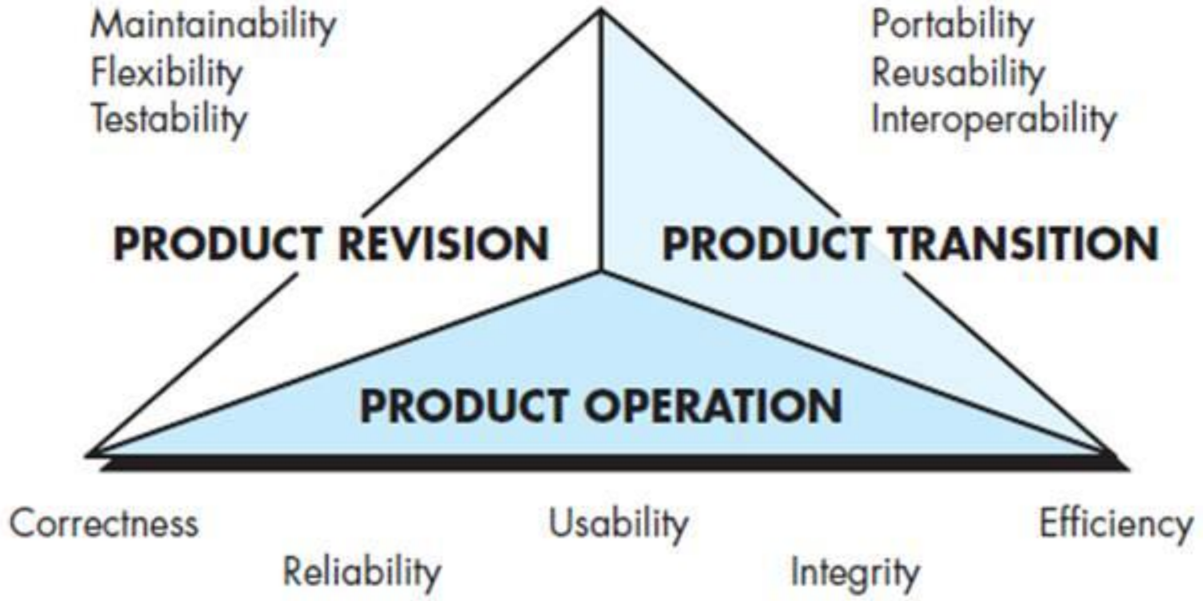
- McCall proposed a useful categorization of factors that affect software quality. These software quality factors, shown in Figure below, focus on three important aspects of a software product:

- **(a) its operational characteristics,**
- **(b) its ability to undergo change,**
- **(c) its adaptability to new environments.**

19.2 Software Quality: 19.2.2 Mc Call's Quality Factors

FIGURE 19.1

McCall's software quality factors



19.2 Software Quality: 19.2.2 Mc Call's Quality Factors

(a) its operational characteristics,

- **Correctness.** *The extent to which a program satisfies its specification and fulfills the* customer'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 its function.
- **Integrity.** *Extent to which access to software or data by unauthorized persons can be* controlled.
- **Usability.** *Effort required to learn, operate, prepare input for, and interpret output of* a program.

19.2 Software Quality: 19.2.2 Mc Call's Quality Factors

(b) its ability to undergo change,

- ***Portability.*** *Effort required to transfer the program from one hardware and/or software* system environment to another.
- ***Reusability.*** *Extent to which a program [or parts of a program] can be reused in other* applications—related to the packaging and scope of the functions that the program performs.
- ***Interoperability.*** *Effort required to couple one system to another.*

(c) its adaptability to new environments.

- ***Maintainability.*** *Effort required to locate and fix an error in a program.*
- **Flexibility.** *Effort required to modify an operational program.*
- ***Testability.*** *Effort required to test a program to ensure that it performs its intended* function.