Chapter 19

Quality Concepts

Slide Set to accompany

Software Engineering: A Practitioner's Approach

by Roger S. Pressman

Table of Contents

19.1 What is Quality?

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

For software development,
quality = quality of software + quality of development

- User satisfaction =
  compliant product (w.r.t. functionality)
  + good quality
  + delivery within budget and schedule

19.2 Software Quality

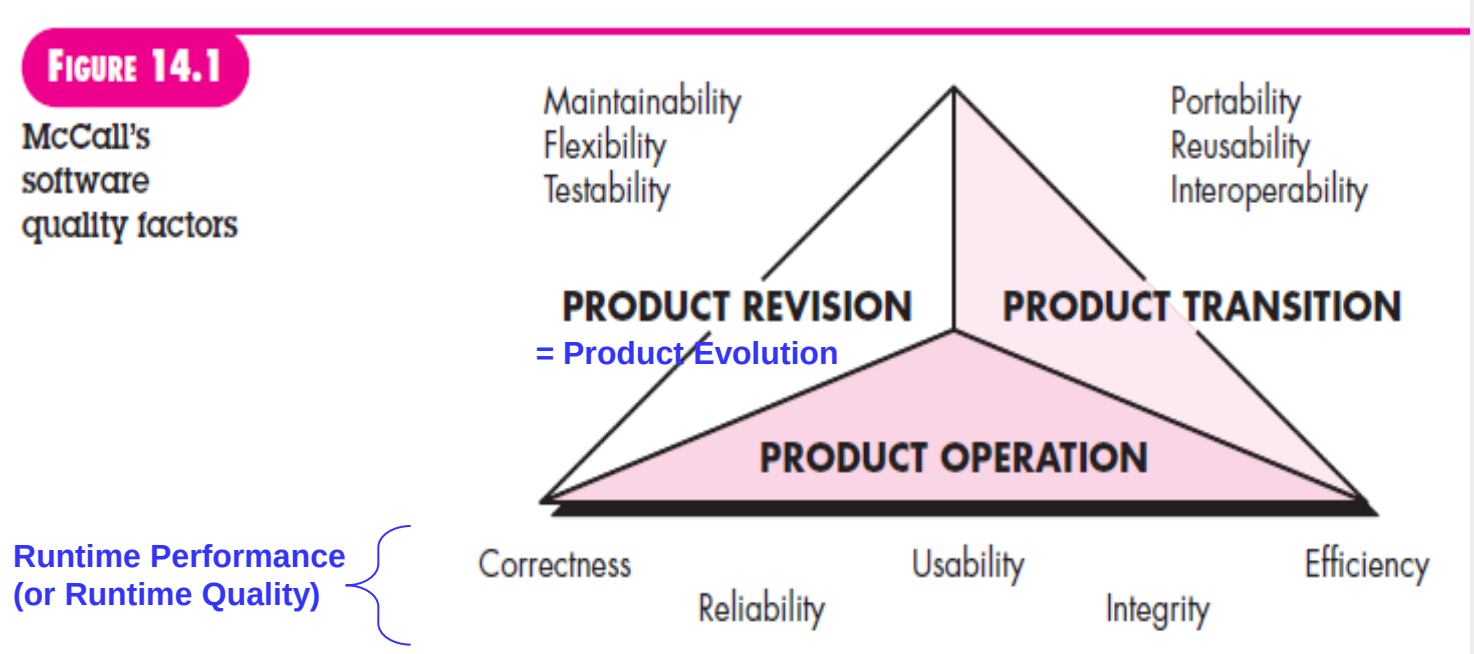
- Software Quality and Software Process
  - An *effective software process* establishes the infrastructure that supports building a high quality software
    - Framework activities allow the developer to analyze the problem and design a solid solution with quality
    - Umbrella activities such as change management and technical reviews have as much to do with quality
      - The management aspects: create the checks and balances that help avoid project chaos.

19.2.1 Garvin's Quality Dimensions(SKIP)

- David Garvin [Gar87]: Rather Old View
  - **Performance Quality.** Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?
  - **Feature quality.** Does the software provide features that surprise and delight first-time end-users?
  - **Reliability.** Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?
  - **Conformance.** Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?

- **Durability.** Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?
- **Serviceability.** Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?
- **Aesthetics.** Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.
- **Perception.** In some situations, you have a set of prejudices that will influence your perception of quality.

19.2.2 McCall's Quality Factors



- Product transition – adaptability to new environments

19.2.3 ISO 9126 Quality Factors

- Functionality
- Reliability
- Usability
- Maintainability
- Portability

- ISO 9126 – 1991, 2001 (Old)
  - Replaced by ISO 25010 (2011) (New)

ISO 25010 - **2008**

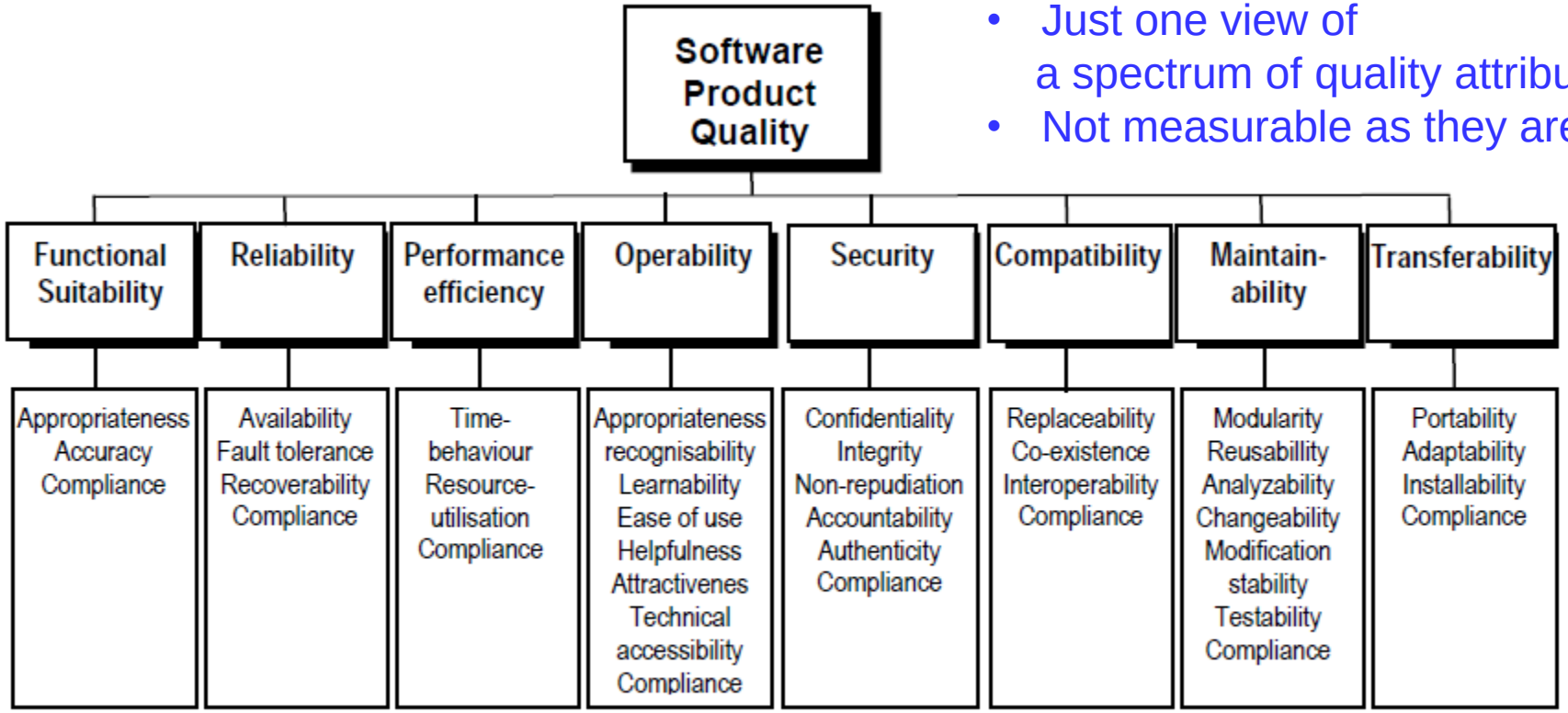


Figure 5 – Software product quality model

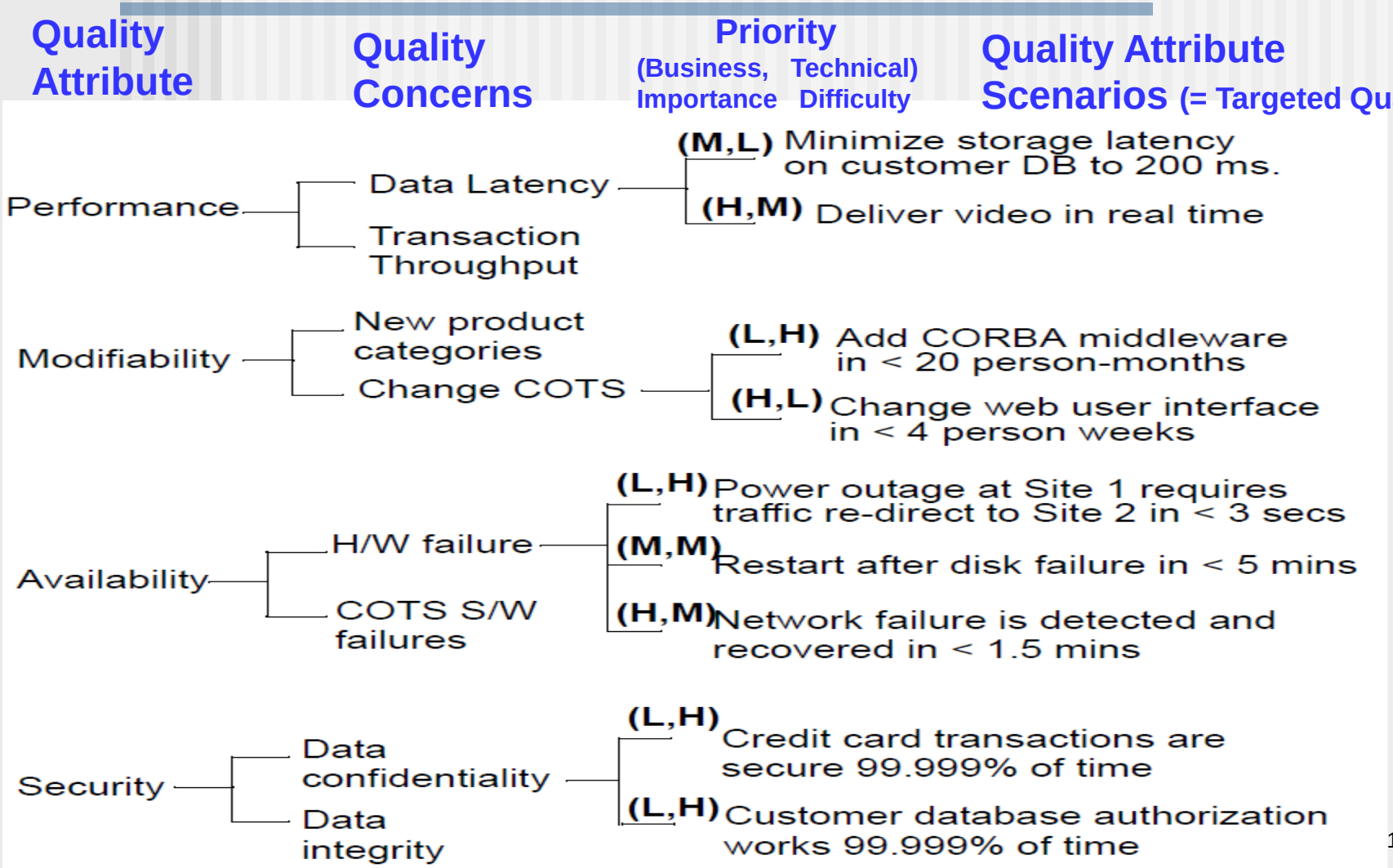
- Just one view of a spectrum of quality attributes
- Not measurable as they are!

19.2.4 Targeted Quality Factors

- A software team can develop a set of quality characteristics and associated questions that would probe the degree to which each factor has been satisfied.

☛ For example, SEI's Quality Attribute Workshop

SEI's Quality Attribute Tree

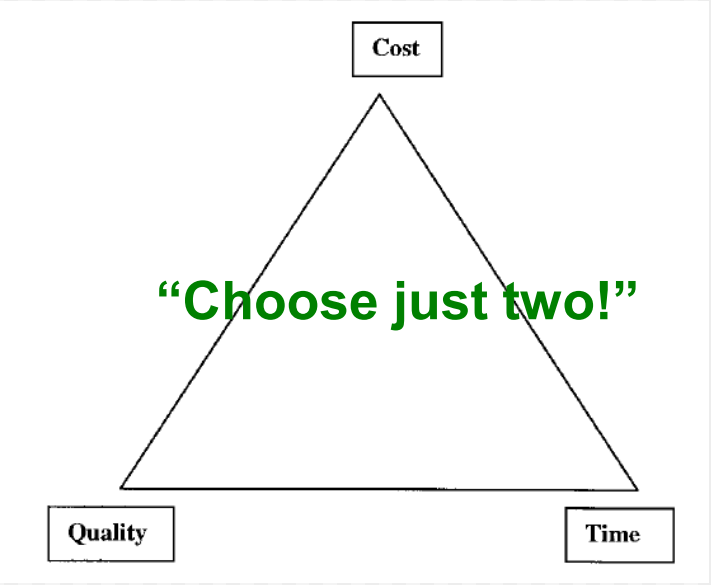


19.2.5 The Transition to a Quantitative View

- Software metrics can be applied to the quantitative assessment of software quality.
- In all cases, the metrics represent indirect measures; that is, we never really measure *quality* but rather some manifestation of quality.
  - The complicating factor is the precise relationship between the variable that is measured and the quality of software.

19.3 The Software Quality Dilemma

- Too low quality - If your software system that has terrible quality, you lose because no one will buy it.
- Too high quality - If you spend extremely large effort to build the perfect piece of software, then it's going to take very long and it will be so expensive to produce that you'll be out of business.
- So people in industry try to get to that magical middle ground where the product is good enough not to be rejected right away but also not to be perfect and not to require so much work. [Ven03]



Skip to Section 19.4

19.3.1 “Good Enough” Software

- Good enough software delivers high quality functions and features that end-users desire, but at the same time it delivers other more obscure or specialized functions and features that contain known bugs.

- Arguments *against* “good enough.”
  - It is true that “good enough” may work in some application domains and for a few major software companies. After all, if a company has a large marketing budget and can convince enough people to buy version 1.0, it has succeeded in locking them in.
  - If you work for a small company , then if you deliver a “good enough” (buggy) product, you risk permanent damage to your company’s reputation.
  - If you work in certain application domains (e.g., real time embedded software), application software that is integrated with hardware can be negligent and open your company to expensive litigation.

19.3.2 The Cost of Quality

- Prevention costs include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- Internal failure costs include
  - rework
  - repair
  - failure mode analysis
- External failure costs are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work

Before failure

After failure

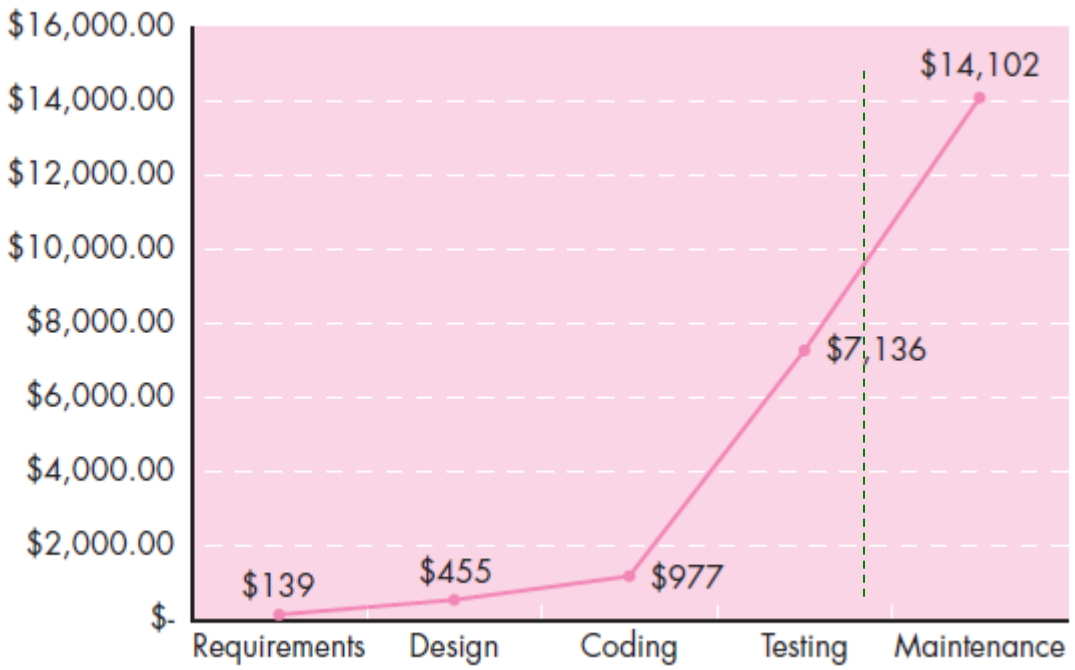
Cost

- The cost to find and repair an error or defect increase dramatically as we go from prevention to detection to internal failure to external failure costs.

FIGURE 14.2

Relative cost of correcting errors and defects

Source: Adapted from [Boe01b].



19.3.3 Risks

- *"People bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right."* SEPA, Chapter 1

- Example:

  Throughout the month of November, 2000 at a hospital in Panama, 28 patients received massive overdoses of gamma rays during treatment for a variety of cancers. In the months that followed, five of these patients died from radiation poisoning and 15 others developed serious complications. What caused this tragedy?

  A software package, developed by a U.S. company, was modified by hospital technicians to compute modified doses of radiation for each patient.

19.3.4 Negligence and Liability

- A governmental or corporate entity hires a major software company to construct a software-based “system” to support some major activity.
- Work begins with the best of intentions on both sides, but by the time the system is delivered, things have gone bad.
- The system is late, fails to deliver desired features and functions, is error-prone, and does not meet with customer approval.
- Litigation ensues.

19.3.5 Quality and Security

- Gary McGraw comments [Wil05]:

"Software security relates entirely and completely to quality.

You must think about security, reliability, availability, dependability—at the beginning, in the design, architecture, test, and coding phases, all through the software life cycle [process]. Even people aware of the software security problem have focused on late life-cycle stuff.

The earlier you find the software problem, the better.

And there are two kinds of software problems.

One is bugs, which are implementation problems.

The other is software flaws—architectural problems in the design. People pay too much attention to bugs and not enough on flaws."

19.3.6 The Impact of Management Actions

- Not only technology decisions but also management decisions can affect software quality
  - Estimation decisions
  - Scheduling decisions
  - Risk-oriented decisions

19.4 Achieving Software Quality

- Critical success factors:
  - Software Engineering Methods
  - Project Management Techniques
    - Quality Control
      - A set of software engineering actions that help to ensure that each work product meets its quality goals
    - Quality Assurance
      - Establishes the infrastructure that supports solid software engineering methods, rational project management and quality control actions.