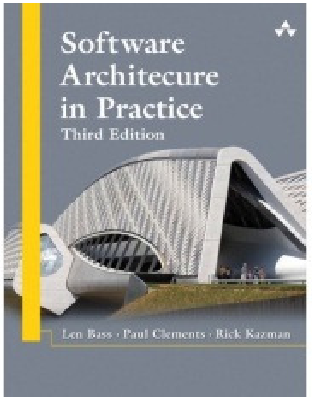
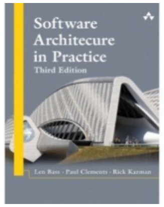


Chapter 12:
Other Quality Attributes

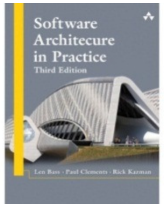
© Len Bass, Paul Clements, Rick Kazman,
distributed under Creative Commons
Attribution License



Chapter Outline

- Other Important Quality Attributes
- Other Categories of Quality Attributes
- Software Quality Attributes and System Quality Attributes
- Using Standard Lists of Quality Attributes
- Dealing with “X-ability”
- Summary

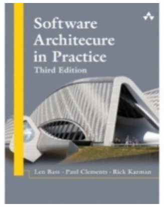
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Other Important Quality Attributes

- Variability: is a special form of modifiability. It refers to the ability of a system and its supporting artifacts to support the production of a set of variants that differ from each other in a preplanned fashion.
- Portability: is also a special form of modifiability. Portability refers to the ease with which software that built to run on one platform can be changed to run on a different platform.
- Development Distributability: is the quality of designing the software to support distributed software development.

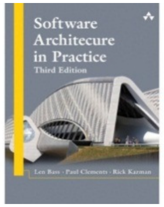
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Other Important Quality Attributes

- Scalability: Horizontal scalability (scaling out) refers to adding more resources to logical units such as adding another server to a cluster. Vertical scalability (scaling up) refers to adding more resources to a physical unit such as adding more memory to a computer.
- Deployability: is concerned with how an executable arrives at a host platform and how it is invoked.
- Mobility: deals with the problems of movement and affordances of a platform (e.g. size, type of display, type of input devices, availability and volume of bandwidth, and battery life).

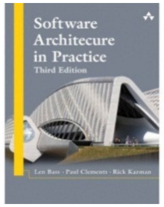
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Other Important Quality Attributes

- Monitorability: deals with the ability of the operations staff to monitor the system while it is executing.
- Safety: Software safety is about the software’s ability to avoid entering states that cause or lead to damage, injury, or loss of life, and to recover and limit the damage when it does enter into bad states. The architectural concerns with safety are almost identical with those for availability (i.e. preventing, detecting, and recovering from failures).

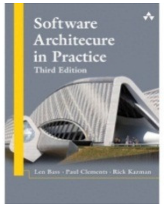
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Other Categories of Quality Attributes

- Conceptual Integrity: refers to consistency in the design of the architecture. It contributes to the understandability of the architecture. Conceptual integrity demands that the same thing is done in the same way through the architecture.
- Marketability: Some systems are marketed by their architectures, and these architectures sometimes carry a meaning all their own, independent of what other quality attributes they bring to the system (e.g. service-oriented or cloud-based).

© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License

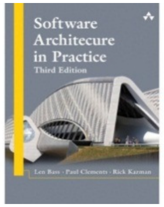


Software Architecure in Practice
Third Edition
Len Bass · Paul Clements · Rick Kazman

Other Categories of Quality Attributes

- Quality in Use: qualities that pertain to the use of the system by various stakeholders. For example
  - Effectiveness: a measure whether the system is correct
  - Efficiency: the effort and time required to develop a system
  - Freedom from risk: degree to which a product or system affects economic status, human life, health, or the environment

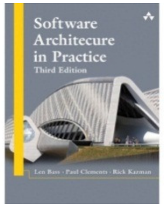
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



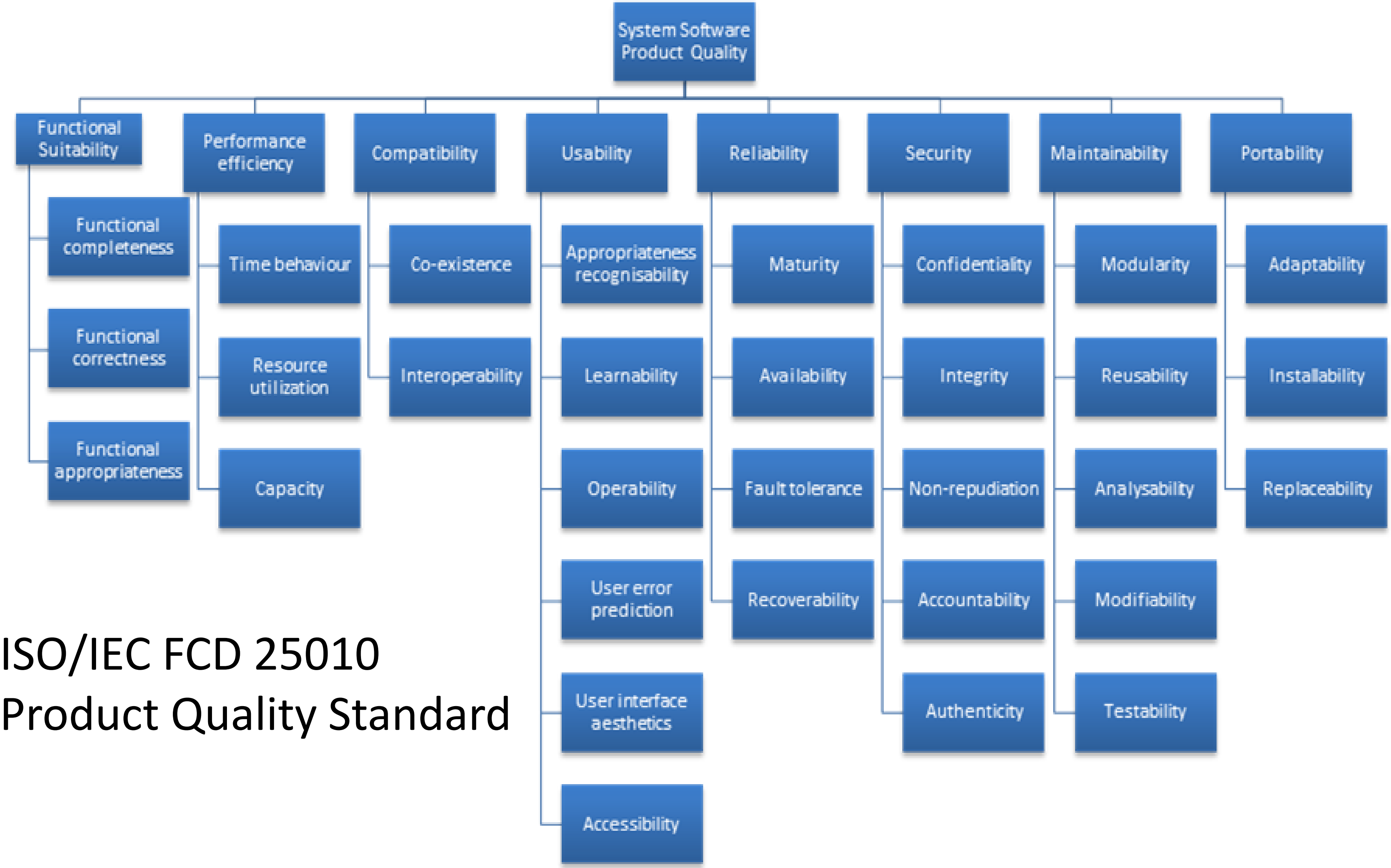
Software Quality Attributes and System Quality Attributes

- Physical systems, such as aircraft or automobiles or kitchen appliances, that rely on software embedded within are designed to meet a whole other litany of quality attributes: weight, size, electric consumption, power output, pollution output, weather resistance, battery life, and on and on.
- The software architecture can have a substantial effect on the system's quality attributes.

© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



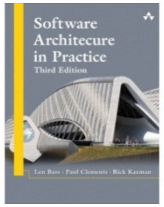
Standard Lists of Quality Attributes



- System Software Product Quality
  - Functional Suitability
    - Functional completeness
    - Functional correctness
    - Functional appropriateness
  - Performance efficiency
    - Time behaviour
    - Resource utilization
    - Capacity
  - Compatibility
    - Co-existence
    - Interoperability
  - Usability
    - Appropriateness recognisability
    - Learnability
    - Operability
    - User error prediction
    - User interface aesthetics
    - Accessibility
  - Reliability
    - Maturity
    - Availability
    - Fault tolerance
    - Recoverability
  - Security
    - Confidentiality
    - Integrity
    - Non-repudiation
    - Accountability
    - Authenticity
  - Maintainability
    - Modularity
    - Reusability
    - Analysability
    - Modifiability
    - Testability
  - Portability
    - Adaptability
    - Installability
    - Replaceability

ISO/IEC FCD 25010
Product Quality Standard

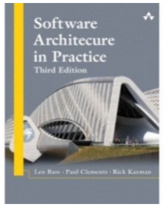
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Standard Lists of Quality Attributes

- Advantages:
  - Can be helpful checklists to assist requirements gatherers in making sure that no important needs were overlooked.
  - Can serve as the basis for creating your own checklist that contains the quality attributes of concern in your domain, your industry, your organization, your products, …

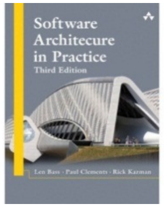
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Standard Lists of Quality Attributes

- Disadvantages:
  - – No list will ever be complete.
  - – Lists often generate more controversy than understanding.
  - – Lists often purport to be *taxonomies*. But what is a denial-of-service attack?
  - – They force architects to pay attention to every quality attribute on the list, even if only to finally decide that the particular quality attribute is irrelevant to their system.

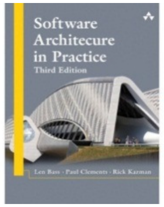
© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Dealing with “X-ability”

- Suppose you must deal with a quality attribute for which there is no compact body of knowledge, e.g. green computing.
- What do you do?
  1. Model the quality attribute
  2. Assemble a set of tactics for the quality attribute
  3. Construct design checklists

© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License



Summary

- There are many other quality attributes than the seven that we cover in detail.
- Taxonomies of attributes may offer some help, but their disadvantages often outweigh their advantages.
- You may need to design or analyze a system for a “new” quality attribute. While this may be challenging, it is doable.

© Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License