Quality Defined

by Linda Westfall

The quality and software industries have not, and may never, come to a single definition of the term *quality*. For example, the ISO/IEC/IEEE *Systems and Software Engineering—Vocabulary* has the following set of definitions for quality:

1. The degree to which a system, component, or process meets specified requirements
2. Ability of a product, service, system, component, or process to meet customer or user needs, expectations, or requirements
3. The totality of characteristics of an entity that bears on its ability to meet stated and implied needs
4. Conformity to user expectations, conformity to user requirements, customer satisfaction, reliability, and level of defects present
5. The degree to which a set of inherent characteristics fulfills requirements.

Based on his studies of how quality is perceived in various domains (for example, philosophy, economics, marketing, operations management), Garvin [Schulmeyer 1998] concluded, "Quality is a complex and multifaceted concept." Garvin describes quality from five different perspectives:

Manufacturing perspective: Philip Crosby defines quality in terms of conformance to the specification. His point is that an organization does not want a variety of people throughout the development of a product trying to make judgments about what the customer needs or wants. A well-written specification is the cornerstone for creating a quality product. For software, however, this perspective of quality is necessary but may not be sufficient since, according to Wiegers [Wiegers 2003], errors made during the requirements stage account for 40 percent to 60 percent of all defects found in a software project. From another viewpoint, this perspective refers to the ability to manufacture (replicate) a product to that specification over and over within accepted tolerances. W. Edwards Deming talks about quality needing "precision of effort." Before an organization adjusts its processes to improve them, it must let them run long enough to understand what is really being produced. Then it needs to design its specifications to reflect the real process capabilities. While the primary focus of software quality is on the design and development activities, this manufacturing and "precision of effort" quality perspective reminds software organizations that the replication process can not be completely ignored.

User perspective: Joseph M. Juran cites fitness for use as the appropriate measure for quality. Software practitioners can all probably relate stories of software products that conformed to their specifications but did not function adequately when deployed into operations. This perspective of quality not only considers the viewpoints of the individual users but their context of use as well. For example, what a novice user might consider a"quality" user interface might drive a power

user to distraction with pop-up help and warning messages that require responses. What is a secure-enough interface for a software database used for personal information at home might be woefully inadequate in a business environment.

Product perspective: Quality is tied to the inherent characteristics of the product. These characteristics are the quality attributes, also called the "ilities" of the software product. Examples include reliability, usability, availability, flexibility, accessibility, maintainability, portability, installability, and adaptability. Of course, they don't all end in "ility." Correctness, fault tolerance, integrity, efficiency, security, and safety are also examples of quality attributes. The more the software has high levels of these characteristics, the higher its quality is considered to be. The ISO/IEC 25000 *Software Engineering—Software Product Quality Requirements and Evaluation* (SQuaRE) standard provides a reference model and definitions for external and internal quality attributes and quality-in-use attributes. This standards series also provides guidance for specifying requirements, planning, managing, measuring, and evaluating quality attributes.

Value-based perspective. Quality is dependent on the amount a customer is willing to pay for it. This perspective leads to considerations of "good enough" software quality. Are people willing to pay as much for high-quality video game software as they are for high-quality software in biomedical devices or the high-quality software for airplane navigation systems?

Transcendental perspective. Quality is something that can be recognized but not defined. As stated by Kan [Kan 2003], "to many people, quality is similar to what a federal judge once said about obscenity: 'I know it when I see it.'" This perspective of quality takes the viewpoint of the individual into consideration. What is "obscenity" to one person may be "art" to another. What one customer considers good software quality may not be high enough quality for another customer. Tom Peters cites the customer's reaction as the only appropriate measure for the quality of a product. This requires that product developers keep in touch with their customers to ensure that their specifications accurately reflect the customer's real (and possibly changing) needs.

References:

Kan 2003 - Kan, Stephen. *Metrics and Models in Software Quality Engineering*, Second Edition. Boston: Addison-Wesley/ 2003.

Schulmeyer 1998 - Schulmeyer, G. Gordon, and James McManus, editors. *Handbook of Software Quality Assurance*, Third Edition. Upper Saddle River, NJ: Prentice-Hall PTR. 1998.

Wiegers 2003 - Wiegers, Karl. *Software Requirements, Second Edition. Redmond, WA: Microsoft Press.* 2003.