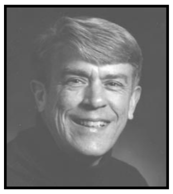

Robert L. Glass

The Loyal Opposition

Defining Quality Intuitively

This is the decade of software quality. In the 1980s we focused on trying to increase productivity; in the 1990s we strive to create higher-quality products.

Alphabet Soup

Trouble is, we haven't been very good at achieving either goal. For all the "breakthrough" claims made by advocates of those 1980s methodologies and techniques, productivity didn't improve much. Each so-called breakthrough concept—4GLs, CASE, RAD—gave us at best a 5- to 30-percent improvement. Sure, productivity increased, but it came nowhere near making good on the "order of magnitude" claims that marketing departments touted and practitioners longed for.

Now it's quality's turn. There's a new alphabet soup of quality improvement techniques floating around, everything from CQI to SPC to TQM. But I've seen little evidence that any of *these* will be our hoped-for breakthrough, either. Although their champions make the same optimistic claims all over again, we in the software field are finally getting wise to them. We've begun to suspect that the Holy Grail of software quality is about as attainable as every failed silver bullet we've been teased with in the past.

Attribute Stew

Progress in improving software quality would be perilous even if all those silver bullets *had* hit their target: We've never agreed on what quality means! Many quality gurus talk about things like satisfying customers and meeting requirements, but for some serious reasons those long-cherished definitions make no sense. All hope is not lost, however: I finally came across one that does.

Although I'd love to take credit for the equation that follows, I didn't invent it. A gentleman from Computer Sciences Corp. presented it at a NASA-

EDITOR: Robert L. Glass • Computing Trends • rglass@indiana.edu

Goddard Software Engineering Workshop some half-dozen years ago. I apologize for not remembering his name, and will supply it in conjunction with a future edition of Loyal Opposition if someone will pass it on to me. The definition contained in this equation makes so much sense we can trash all the others:

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

Notice that the equation defines quality implicitly rather than explicitly. That is, it makes quality an attribute of the thing actually being defined: user satisfaction. That's important. Many people—including the pioneering quality gurus—assert that quality is equivalent to user satisfaction. Although the two are related, they are not the same.

McDonald's hamburgers perfectly match user requirements. But quality dining? Not even McDonald's claims that.

Further, quality is separate from the "compliant product" attribute. Nor does it have anything to do with budget and schedule considerations. Just in case this separation is not obvious, consider a few examples:

♦ McDonald's produces hamburgers that perfectly match user requirements and satisfy expectations for affordability and timely arrival. But quality dining? No one, not even McDonald's, claims that.

♦ The Jaguar automobile of years past boasted a cult of loyal owners generally content with its cost and delivery schedule and satisfied that it met their needs. But, due to the car's abominable electrical system, no one ever claimed the Jaguar was a top-quality vehicle. Owner loyalty persisted despite the car's quality, not because of it.

♦ Most people who have installed the SAP software megapackage are pleased with its performance. It satisfies its customers, it meets the requirements clearly established for it, and most people would say it exhibits high quality. But articles reporting its extremely high cost and oft-interminable installation time reveal that despite its quality, it sometimes fails to meet cost and schedule needs.

So the equation I present satisfies intuitively: few would disagree that it correctly captures an important set of relationships.

Standards Fare

Okay, so quality relates to the equation's other concepts, but it is not actually defined by it. Quality isn't "user satisfaction," "compliant product," or even "delivery within budget and schedule." So what is it?

Here we can fall back on some time-honored definitions. The ISO 8402-1986 standard definition of quality, for example, states that "Quality equals the totality of features and characteristics of a product or service that bear on its ability to satisfy stated and implied needs."

Well, that's pretty straightforward. But it's also fuzzy. What *kinds* of features and characteristics? What *sort* of stated and implied needs?

Fortunately, we can reach back even further in time to find a better definition. Barry Boehm, in a 1970s book (*Practical Strategies for Developing Large Software Systems*, Addison Wesley Longman, 1975), defined quality as a set of attributes:

Quality = portability + reliability + efficiency + human engineering + understandability + modifiability + testability

Some call these the "ilities" of software quality.

As you might expect, others disagree with even this level of definition. One camp, for example, views quality as being purely about errors. They would limit our definition to one of the ilities above, reliability. But this defect-focused definition of quality is extremely limited—and limiting. A product that contains few defects but is virtually unmaintainable, for example, could not by even the furthest stretch of imagination be considered high quality. A software product that never failed, but took 24 hours, 24 minutes, or even 24 seconds to perform its task would also find the high-quality label elusive.

Another source of disagreement concerns the ilities list itself. For example, one software guru refuses to acknowledge that "portability" is an element of software quality. For many applications, he is right. We must bear in mind, however, that different software products call for different ilities mixes, and sometimes one or more can be ignored or de-emphasized. But there is more going on than that. The set of quality attributes in the preceding equation is software-specific—that is, it is carefully chosen to be about software. In other fields, for exam-

Continued on page 107

Continued from page 104

ple, fit and finish are important quality attributes. But fit and finish have almost nothing to do with software. Analogously, portability is desirable when the software product must run in several different environments, but it is rarely a quality attribute of any other product.

What can we conclude from this? That no universal definition of quality at the detail level, that is independent of the product in question, exists. To measure this particular column's quality, for example, would require that you assess its readability, content value, and writing style. Yet those quality attributes are irrelevant to assessing the quality of a hamburger, a Jaguar, or an SAP installation.

A La Carte

The traditional definitions are wrong. Quality is not about user satisfaction, product conformance, or costs and schedules—nor is it solely about defects. Instead, there is a well-defined, intuitive relationship between quality and those other product traits, one that clearly distinguishes among them. Further, the detail-level definition of software quality concerns the attribute set that each product should have, a collection that must be prioritized differently for different project types.

Do you know what's discouraging? It's not often that we feel we've discovered *the* answer to a question so important that we know nearly everybody cares about it. This is one of those rare cases. I wish everyone would belly up to the Loyal Opposition bar and buy into what I'm saying. But they won't. This particular insight—like all the others proliferated through the pages of *IEEE Software* and its professional brethren—will be ignored. Six months or a year from now, we'll still see discussions of software quality that address customer satisfaction, defect counts, or some other piece of nonsense.

Sigh. I suppose being ignored will always be a problem for contrarians like me. That's the essence of being in the loyal opposition. After all, if everyone *did* buy into what I'm saying, I would become part of the mainstream—which is, I suppose, contrarianism's Catch 22. ❖