Can You MANAGE Quality Into a Software Product?

Is software quality a management topic or a technical topic?

A lot of people see it as the former. If you read a book on software quality, or take a course on software quality, there is a tacit assumption that right after the word "quality" will be found the word "assurance." With that addition certainly quality becomes a management topic.

But I don't see that addition as necessary. I believe that the fundamental problem in software quality is technical, not managerial, and I would like to decouple the word "quality" from the word "assurance."

Now this technology-first approach requires an explanation. Let's take a look at the management and the technology of software quality.

No matter how carefully management plans for and provides for quality processes in software development, I believe that it is no more possible to *manage* quality into a software product than it is to *test* it in. It is well accepted that quality cannot be tested into software (because testing only looks at reliability, one facet of quality, and because testing comes too late in the life cycle to have a preventive effect on poor quality), but it is not at all commonly accepted that quality cannot be managed in. In fact, that is a radical viewpoint.

The reason I believe that you can't *manage* in quality is that quality is a deeply intimate software trait. As we explore what quality *really* is, we see that the injection of quality, and the detection of quality, occur far below the surface of software's facade. How can you put understandability and modifiability, two of quality's attributes, into a software product? How can you find out if they were put there? There are no easy answers to those questions, but the fact is that only a technical person can do the putting and the finding. Understandability and modifiability are so deeply technical that it is simply not assessable by the casual viewer. Management, for better or worse, is in this sense only a casual viewer of the software product. Management in general is not and should not be comfortable with the technical tasks necessary for either instilling or detecting understandability and modifiability.

And those are not the only two examples of technology's key role in quality. Software must be reliable, nearly always. Software must be portable, at least sometimes. Both are quality attributes. Who but a technical specialist can truly insure that software is reliable, or distinguish between portable and nonportable software? Most managers have either atrophied capabilities, or none at all, in these areas.

That is not to say, of course, that management has no role in building quality software. To the contrary, there is an essential role for management to play. Management must construct and maintain a climate in which quality is fostered and nurtured.

What do I mean by a quality climate? One where processes which facilitate quality are enabled and followed. One in which tools to assist in providing quality are procured and used. One in which people who think quality are hired and helped. One in which advocacy of product quality occurs right up there with advocacy of schedule and cost constraint conformance.

It is easier to say those things than to do them. We are passing through an era in which schedule and cost have been the dominant factors in evaluating software management and software products. There are some good reasons for that. Software's schedule and cost performance have often been abysmal. With rampant problems in that area, management has rightfully concentrated on trying to solve those problems.

But there is a danger here. As the pressure of meeting schedule and reducing cost intensifies, it is quality that inevitably suffers. How can we accelerate a late product? Cut back on whatever is happening when the problem is discovered, usually verification and testing. The result is reduced product quality.

The software manager of the twenty-first century must find a new equation for software. It must not simply be

software product = on schedule + within budget

It must instead be

software product = quality + on schedule + within budget

And achieving good results in that more complicated equation requires management commitment to quality as an end goal right up there with schedule and cost.

Fortunately, it is not hard for management to facilitate the construction of quality software. Most technical people fundamentally want to do a good, quality job. In fact, sometimes in the tradeoffs between quality and cost and schedule, technical people lean too heavily on the side of quality! (Steve Jobs, during a 1989 interview, said ". . .our jobs as senior managers are to help people get very clear on the goals . . . and then to get the hell out of their way.") Management usually will find very willing partners in the task of facilitating quality software.

So there it is. In the business of software quality, the technical considerations are much more complex and vital than the management ones. Quality assurance is important, but it is only one avenue to software quality.

Let me try a little exercise with you. I will now say the word "quality." See if you can avoid hastily adding the unwritten additive "assurance."

Quality.

There, that wasn't so hard, was it?