Software Quality Journal **Vol. 11 No. 1, 2003**

Editorial: The Illusive Nature of Quality

> "Quality … you know what it is, yet you don't know what it is. But that's self-contradictory. But some things *are* better than others, that is, they have more quality. But when you try to say what the quality is, apart from the things that have it, it all goes *poof!* … But for all practical purposes, it does exist. … Why else would people pay fortunes for some things and throw others in the trash pile?" (Pirsig, 1974, pp 163-164)

Years ago, when I was studying the relationship between software metrics and measurement theory, my colleagues and I discovered that developing a single measure that encompasses many of the quality "ilities" --- reliability, maintainability, understandability, testability, fault-tolerance etc, which often conflict, would only produce a measure that could not distinguish between most programs. At that time, the book *Zen and the Art of Motorcycle Maintenance* by Robert Pirsig seemed relevant. When I read Jeff Voas's paper "Trusted Software's Holy Grail" published in this issue, I was reminded of Pirsig's book. I couldn't find my old copy, so I picked up a new one --- it's still in print and ranks fairly high in the Amazon popularity rating --- and flipped through it to see if I would still think that it is relevant.

Pirsig's book, a fictional account of a cross-country motorcycle trip, describes both the many components that go into motorcycle design, and the many maintenance needs and activities required to have a high-quality, functioning motorcycle. Much of the book is a discussion about the elusive notion of "quality". A few more quotes show why I find this book fascinating.

Pirsig argues that you cannot judge quality from external behavior examined at a single point in time:

> "to neglect the past and future for the present is bad Quality indeed. The motorcycle may be working now, but when was the oil level last checked?" (Pirsig, 1974, p. 223)

An analog to software quality would be to assess quality in terms of only black box, functional system testing conducted just before release (one point in time). Such limited testing is clearly not enough to demonstrate quality. You need to know if all of the parts have been tested, and if they can be maintained and adapted in the future.

After a long description of the components of a motorcycle, Pirsig writes

> "that the motorcycle, so described, is almost impossible to understand unless you already know how one works"(Pirsig, 1974, p. 65).

Thus, the notion of understandability is based on the knowledge of the observer. The maintainability of a software system depends in large part on who does the maintenance. For example, the original developers or others with intimate knowledge of a system's design will have a much easier time making modifications than someone with little knowledge or experience with the system.

On the difficulty of partitioning a system into components, Pirsig writes:

> "No two manufacturers ever split it up quite the same way and every mechanic is familiar with the problem of the part you can't buy because you can't find it because the manufacturer considers it a part of something else." (Pirsig, 1974, p. 66)

One person's decomposition may differ from another's. There may not be a clear primary decomposition that represents the best way to partition a system into components. We can end up with many different views or slices of a system:

> "It is important to see this knife for what it is and not be forced into thinking that motorcycles or anything else are the way they are just because the knife happened to cut it up that way." (Pirsig, p. 66)

A system can end up with many components with multiple functionalities based on different views of a system, thus limiting the cohesion of components. The primary motivation for aspect oriented programming and design methods is to use abstractions for specifying, designing, and programming the concerns that are not expressed gracefully by the primary decomposition. Aspect oriented programming has many of its own unique problems. But that's another story.

Pirsig's intention is to demonstrate that a complex entity, mechanical or virtual, cannot be completely understood by understanding all of the parts and how they go together. Also, it cannot be understood without understanding the parts and how they fit together. And, there are perhaps an unlimited number of slices or views of a system. There is a synergistic effect --- the whole is more than the parts, and we cannot ever quite understand, or measure quality precisely. Clearly, the illusive notion of quality, and the difficulty of quality assessment is not problem that is unique to software.

We are stuck with the job of assessing quality, even though it is not possible to do it completely right. This is our challenge.

Finally, re-reading a book can bring new, unexpected insights. For example, I found a possible source for the phrase popularized in the Seinfeld television show:

> "Yah-da, ya-da, yah-da, yah-da, yah" (p. 65)

References

Pirsig, Robert M. 1974. *Zen and the art of Motorcycle Maintenance,* Bantam Books, New York, 1974.

Voas, J. 2003. Trusted Software's Holy Grail, *Software Quality Journal* 9(2):77-78, June 2001.