# Comments on Software Quality

**by**
**Watts S. Humphrey**
**Fellow, Software Engineering Institute**
**Carnegie Mellon University**
**Pittsburgh, PA**

**Summary**

This paper reviews the software industry's current approach to product quality. It also discusses industry trends and what will likely happen in the future. In particular, I describe why current software industry practices do not responsibly address the public's need for quality products. If current trends continue, this industry stance will likely lead to public demands for change. In closing, I recommend a UCC strategy for addressing this issue. In a final appendix, I briefly note my background so the reader can judge my qualifications for commenting on this subject.

## The current state of software quality

The commonly held view of the current state of software quality is best described as "It is impossible to remove all defects from software products." This statement, while widely accepted, is incorrect. Many defect-free software products have been developed.

A more accurate way to state the current situation is that "It is generally impossible to prove that all defects have been removed from a software product." The reason for this is that many software products are extraordinarily complex. It is thus not possible to exhaustively test these products. Thus, if one relies exclusively on testing, it is truly impossible to demonstrate that a software product is defect free.

## Industrial quality principles

The software industry has had a troublesome history of delivering defective products long after they were committed and for far higher costs than planned. There is, however, no reason for the software community to follow different quality principles from those commonly used in other technologically intense industries. While software development and high-technology manufacturing differ in many ways, there is now ample evidence that many of the same quality principles apply to both.

Modern commercial aircraft, for example, are very complex hardware and software systems. One cannot definitively prove that all design and manufacturing defects have been removed from a large aircraft. This, however, does not prevent aircraft manufacturers from warranting their products or from taking full responsibility for product quality. Many other industries produce high-quality products and take full responsibility for any resulting defects.

No technologically intense industry, other than software, relies exclusively on product testing to remove defects. It has been known for over 20 years that testing is an expensive and ineffective way to eliminate defects from any product, including software.

When leading manufacturers in other industries strive for high quality, they focus on the development and manufacturing processes. They use defined and measured procedures and established statistical

methods. Many organizations routinely produce products that are, for all practical purposes, defect free. While they do not guarantee that their products are defect free, they do provide warranty and support services to minimize their customers' damage and inconvenience. They recognize that defects are not acceptable and, to the extent they can, they bear the full costs of fixing or replacing defective products.

**The state of the art in software quality**

Software suppliers do not generally take responsibility for the defect content of their products. They often even ship products that contain known defects, and they commonly charge customers for a significant part of the costs of fixing these defective products. The public is increasingly aware of and unhappy with these practices. Software is routinely blamed for common problems in almost any industry that serves the public, and the public has come to expect software to perform badly.

When an industry gets a reputation for poor-quality products, it is in a risky position. Almost any serious problem could trigger a severe public reaction.

The typical argument for current software practices is that no one does any better. This, however, is wrong. A growing number of software organizations now follow sound management and quality practices to consistently produce high-quality products. They do this by improving the maturity of their processes and using sound engineering methods. These methods have been proven in other industries, and their benefits are now widely recognized for software [Brodman, Butler, Dion, Ferguson, Goldenson, Hayes, Herbsleb 1994, Herbsleb 1996, Herbsleb 1997, Humphrey 1989, Humphrey 1991, Humphrey 1995, Humphrey 1996, Kaplan, Lawlis, Paulk, and Wohlwend].

**The public's need**

As long as software was principally used in scientific or financial applications, and as long as lives and public well being did not depend on software, software-development practices could be undisciplined and still cause little harm or disruption. As software applications become more critical, we face a key question: "Will the software industry address these quality needs voluntarily, or must the changes be forced?"

What the public increasingly needs, and will soon demand, is the ability to easily distinguish between quality software products and defect-prone and unsupported work. They could readily understand the difference if products were clearly labeled. Public demand for quality products would then, in time, lead to enriched competitive offerings.

As long as the entire software industry insists that quality is not their responsibility, quality cannot be a competitive issue, and the public's interests cannot be served. It will then only be a matter of time before some event demonstrates that the software industry's current software quality position is intolerable. When poor software quality poses major economic or life-threatening problems, we can expect a public outcry. And when the general public is concerned, we can expect an almost instant political reaction.

To gauge the likely consequences, compare the freewheeling practices in the software industry with those in such regulated fields as nuclear power, medical instrumentation, auto safety, or drugs. No software group would voluntarily work under the constraints common in these industries.

**A suggested approach**

What I suggest is the following:

1. Make it clear that quality is the normal responsibility of the suppliers of software products and services.
2. Require software suppliers to stand behind their products.
3. Permit any supplier to continue with the no-warranty and no guarantee policies of today only under specific conditions.
4. Such software products could be offered on an as-is basis with no quality obligations only if they were clearly labeled "AS IS, UNSUPPORTED, AND USE AT YOUR OWN RISK."
5. Buyers would then know that buying and using such products entails risks and they could make rational choices.

When software quality becomes an important customer concern, competitive market forces will, in time, assure that the public's needs are addressed.

**Suggested Uniform Commercial Code Article 2B Changes**

While the industry is unlikely to change merely in response to UCC changes, it is important that current industry practices not be enshrined in the draft code or in the laws of the 50 states. We should make it clear that software suppliers are expected to produce quality products and to stand behind them. While exceptions must be permitted where undisciplined and low quality work is acceptable, such practices should not be the industry norm.

This approach has the following implications:

1. We should stop talking about software bugs as if they were mere annoyances. They are defects and should be so labeled.
2. When a software product has one or more known defects, it should be recognized as a defective product.
3. Suppliers of defective software products should be expected to fix the defects and to minimize or remedy the damages the defects cause.

To address these issues, the following general changes should be made in the current Article 2B draft.

1. Open section 2B-403 (Implied Warranty: Quality of Computer Program) with a statement that, by offering a computer program, the supplier assures its customers that the product has been produced according to established industry practices and that the product is warranted to be substantially free of defects.
2. When defects are found in a software product, the supplier will, at its own expense, fix the identified defects and promptly provide the fixes to the users.
3. These quality obligations remain in force even when merchantability and fitness for purpose are disclaimed.
4. These quality obligations can only be disclaimed by including a warning such as: "THIS PRODUCT IS OFFERED AS IS, AND IS UNSUPPORTED. USE IT AT YOUR OWN RISK." This notice should be clearly visible prior to the product's sale.
5. Unless the products are so labeled, the suppliers could not disclaim liability for the consequences of defects that were known at the time the product was shipped.

While it is true that accepted industry practices are not now generally producing software of the desired quality, these practices are changing. By including these statements in the 2B draft, quality could become an important competitive issue. In time, we would then expect market forces to produce products that meet the public's needs.

I also suggest that the entire article 2B draft be reviewed to see if other changes are needed.

**Conclusions**

While today it may seem rational for the software industry to disclaim all responsibility for the quality of their products, this is tantamount to insisting that the market change before the industry will. This stance guarantees that when the market changes, as it must, the public must first be damaged. This will cause avoidable harm and discredit the present suppliers.

Continuing with this strategy will mean that software quality will inevitably become a hot political issue. Not only will the politicians demand immediate action, but they will not likely look to the industry that caused the problems to recommend solutions. In the past, such conditions have led to protective regulation to guard the public from irresponsible industry practices.

While it is not clear that such a reaction can now be prevented, an important first step would be for the industry to establish a responsible position regarding product quality. Even though the proposed changes would not likely cause immediate practice changes, they would establish a responsible public attitude. They would also provide a clear means for distinguishing high-quality offerings from all others. While this may not seem advantageous to the U.S. software industry today, the quality suppliers will soon want ways to distinguish themselves from those who are unscrupulous or incompetent. Also, many developing countries have made software their highest industrial priority. They are actively pursuing the software business and their principal target is the U.S. market. To maintain a sound competitive position, current U.S. suppliers will need to differentiate their products in ways that are meaningful to their customers. Quality is a proven way to do this.

**References**

Brodman, J. G. and Johnson, D. L. "What small businesses and small organizations say about the CMM. Proceedings of ICSE '94 (Sorrento, Italy, May 16-21).

Butler, K. L., "The economic benefits of software process improvement," CrossTalk (July 1995), 14-17.

Dion, R., "Process improvement and the corporate balance sheet." IEEE Software, 10, 4 (July 1993), 28-35.

Ferguson, P., Humphrey, W.S., Khajenoori, S., Macke, S., and Matvya, A., "Results of Applying the Personal Software Process," *IEEE Computer*, vol. 30, no. 5, pp 24-31, May 1997.

Goldenson , D. R. and Herbsleb, J. D., "After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success," Technical Report CMU/SEI-95-TR-009, August 1995.

Hayes, W. and Zubrow, D., "Moving On Up: Data and Experience Doing CMM-Based Process

Improvement," Technical Report CMU/SEI-95-TR-008, August 1995.

Herbsleb, J., Carleton, A., Rozum, J., Siegel, J., and Zubrow, D., "Benefits of CMM-Based Software Process," CMU/SEI-94-TR-13.

Herbsleb, J. D., and Goldenson, D. R., A systematic survey of CMM experience and results. In Proceedings of ICSE '96 (Berlin, Mar. 25-30).

Herbsleb, J. D., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M., "Software Quality and the Capability Maturity Model," Communications of the ACM, June 1997, vol. 40, no. 6, June 1997.

Humphrey, W. S., *Managing the Software Process.* Reading, MA: Addison-Wesley, 1989.

Humphrey, W. S., Snyder, T. R., and Willis, R. R., "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991, pp. 11-23.

Humphrey, W. S., *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.

Humphrey, W. S., "Using a Defined and Measured Personal Software Process," *IEEE Software*, May, 1996.

Kaplan, C., Clark, R., and Tang, V., Secrets of Software Quality, 40 Innovations from IBM. New York: McGraw Hill, Inc., 1995.

Lawlis, P. K., Flowe, R. M., and Thordahl, J. B. A correlational study of the CMM and software development performance. CrossTalk (Sept. 1995), 21-25.

Paulk, M. C. and others, *The Capability Maturity Model: Guidelines for Improving the Software Process.* Reading, MA: Addison Wesley, 1995.

Wohlwend, H. and Rosenbaum, S. "Schlumberger's software improvement program." *IEEE Trans. Soft. Eng*. 20, 11 (Nov. 1994), 833-839.

**Appendix A: Watts S. Humphrey biographical sketch.**

Mr. Humphrey is currently and SEI Fellow at the Software Engineering Institute (SEI) of Carnegie Mellon University. He joined the SEI after his retirement from IBM in 1986. While at the SEI, he established the Process Program, led the initial development of the Software Capability Maturity Model (CMM)[SM], and introduced the concepts of Software Process Assessment, Software Capability Evaluation, and, most recently, the Personal Software Process (PSP)[SM] and Team Software Process (TSP)[SM]. The Capability Maturity Model is now used by software organizations throughout the world to guide their process improvement work. It has become the standard used by the U.S. Department of Defense, the British Ministry of Defense, and many other government departments and industrial corporations to evaluate their own capabilities and those of their software suppliers.

Prior to joining the SEI, Mr. Humphrey spent 27 years with IBM in various technical executive positions including the management of all IBM commercial software development. This included the first 19 releases of OS/360. He was also IBM's Vice President of Technical Development, responsible for the

system architecture and software development for the IBM System 370 product line. At one time, while he was IBM Director of Policy Development, he was responsible for IBM's world-wide hardware and software contracts and competitive practices. Most recently, he was IBM's Director of Programming Quality and Process.

Mr. Humphrey holds graduate degrees in Physics from the Illinois Institute of Technology and Business Administration from the University of Chicago. He is an SEI Fellow, a member of the ACM, an IEEE Fellow, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He was awarded the American Institute of Aeronautics and Astronautics Software Engineering Award for 1993 and the SEI Leadership Award in 1997. His publications include many technical papers and six books. His most recent books are *Managing the Software Process* (1989), *A Discipline for Software Engineering* (1995), *Managing Technical People* (1996), and *Introduction to the Personal Software Process* (1997). He holds five US. patents.

---

SM Capability Maturity Model and CMM are service marks of Carnegie Mellon University.

SM Personal Software Process and PSP are service marks of Carnegie Mellon University.

SM Team Software Process and TSP are service marks of Carnegie Mellon University.

---


| Participants' Comments | Issues and Comments: Contents | *2BGuide* HomePage |
|---|---|---|