High-Tech Warranty Project - - Comment 994413

By
Stephen E. Cross
Director, Software Engineering Institute
Watts S. Humphrey
Fellow, Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA

Summary Position

1. Software products should be warranted by their suppliers in the same way as other products. There is no technical reason to provide special provisions for software vendors and there is increasing evidence that software users need improved protection. The common reasons given for having special warranty provisions for software and our summary responses to these reasons are as follows.

    1. Quality software cannot be produced. This is demonstrably incorrect.
    2. It costs too much to produce quality software. This is also demonstrably incorrect.
    3. Customers do not want quality. The fact that customers purchase poor-quality products does not mean that they do not value quality, just that they do not have a choice.
    4. Software is a vital industry that should be protected. No industry that persists in undisciplined or poor-quality practices should be protected.

Background

2. Software technology is currently used to provide the increasingly sophisticated functions of commonly used products. The reasons for using software logic to replace hardware logic are that it costs less, it does not wear out, it has negligible manufacturing costs, and it is relatively easy to modify and enhance. While software can be stored in many ways, the essential nature of software is the same, whether it is kept in disk memory, semiconductor chips, or CDs. Further, the methods for developing all such kinds of software are essentially the same.

3. While software products are often highly defective, there is no inherent reason for software to be any more defective than hardware of equal complexity. In fact, software is intrinsically more reliable and less defective than hardware: it does not wear out or deteriorate. However, since software is often used for more complex functions than hardware and since increased complexity generally leads to more defects, software often seems more defective than hardware.

4. In spite of its inherent quality advantages, there are three reasons why software products often are currently more defective than hardware of comparable complexity.

    1. The practices generally used to develop software are undisciplined and rarely follow the best known quality methods. These poor practices leave many defects in the products, even after they have been tested.
    2. Since customers currently buy defective software products, the suppliers do not view quality as important.
    3. When quality is not seen as important, suppliers do not measure or manage the quality of their work. When not measured and managed, product quality is invariably poor.

5. While there is currently considerable pressure to absolve software suppliers from liability for defective products, this would not be in the national best interest for three reasons.

    1. The general public will increasingly depend on software and will need more, not less, protection from poor quality products.
    2. By removing the quality incentive, U.S. software suppliers would have even less motivation than today to improve quality practices.
    3. If the U.S were to absolve its software suppliers from legal responsibility for software quality, competing nations would soon learn to build superior products and could ultimately replace the U.S. as the industry leader.

6. The balance of this paper reviews the software industry's current approach to product quality and discusses likely future industry trends. In particular, we describe why the software industry is not responsibly addressing the public's need for quality products and why current proposals for reduced consumer protection are not in the national best interest.

Industrial trends in product development

7. Because of its inherent advantages and the growing sophistication of many products, increasing numbers of common consumer devices use software technology for their most complex functions. For the engineer, the choice of using hardware or software is motivated by the substantial economic advantages of the software technology. The user is not aware of this choice, however, and the product looks and functions identically, regardless of whether the features are implemented in hardware or software.

8. Today, suppliers can offer products as integrated hardware-software units or as separate hardware and software products. This choice is independent of the technology and is made for packaging, distribution, financial, and legal reasons. The identical software could be offered, priced, and warranted as part of the hardware or as a separately priced software product. The assembled product

would appear identical to the user in either case, and that user would suffer identical harm from a defect in either case. If software suppliers were accorded special protection, they would be tempted to separately package the software portions of their products, solely to escape legal liability.

The current state of software quality

9. The commonly-held industry position is best described as "It is impossible to remove all defects from software products." This statement, while widely accepted, is incorrect. Many apparently defect-free software products have been developed.

10. 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. Thus, it is not possible to exhaustively test these products. Therefore, if one relies exclusively on testing, it is truly impossible to demonstrate that a software product is defect free. However, software products are not unique in this regard.

Industrial quality principles

11. In one example, modern commercial aircraft are very complex hardware and software systems. One cannot definitively prove that all design and manufacturing defects have been removed from a large aircraft. However, this 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.

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

13. 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

14. Software suppliers generally do not 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 publicly available service, and the public has come to expect software to perform poorly.

15. The typical argument for the current software practices is that no one does any better. This argument is wrong for the same reasons it was wrong in the 1960s and 1970s for the U.S. automobile industry: better quality was both achievable and economically practical. The software industry is highly exposed to foreign competition for two reasons: other countries have many talented software professionals, and little capital is required to start a software business. Further, with the Internet, national borders, time zones, and geographic barriers are much less important than ever before.

16. A growing number of software organizations now follow sound management and quality practices and consistently produce quality products. They do this by improving the maturity of their processes and by using sound engineering methods. These methods have been proven in other industries, and their benefits are now 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

17. 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 public 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?"

18. What the public increasingly needs, and will likely 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.

19. As long as the entire software industry insists that quality is not their responsibility, quality cannot be a competitive issue, and the public's interest 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 a political reaction.

20. 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. Since the software industry is relatively new and rapidly evolving and since regulatory oversight can severely constrain innovation and increase costs, such oversight should be avoided if at all possible.

A suggested approach

21. What we suggest is the following.

    1. Make it clear that quality is the normal responsibility of the supplier of all products and services, whether provided with hardware or software.
    2. Require software suppliers to stand behind their products.
    3. Permit any software supplier to use no-warranty and no guarantee policies 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. Customers would then know that buying and using such products entails risks and they could make rational choices.

22. When software quality becomes an important consumer concern, competitive market forces will, in time, ensure that the public's needs are addressed. This approach has the following implications.

    1. People should stop talking about software bugs as if they were mere annoyances. They are defects and should be so labeled.
    2. When software products have known defects, they should be recognized as defective products.
    3. Suppliers of defective software products should be expected to fix the defects at their own expense and to minimize or remedy the damages the defects cause.
    4. If the software suppliers do not live up to their obligations, the users should be able to seek legal recourse, just as with other products.

23. While it is true that accepted industry practices are not now generally producing software of the desired quality, these practices are changing. By adopting these recommendations, quality could become an important competitive issue. In time, market forces will motivate the software suppliers to meet the public's needs.

Conclusions

24. There is no technical reason to treat software warranties any differently than those for other products. The software industry may wish to disclaim all responsibility for the quality of their products, but this is tantamount to insisting that the market

change before the industry will. This stance guarantees that the public must first be harmed before the industry recognizes its quality obligations.

25. While it is not clear that public damage can now be prevented, an important first step would be for the U.S. Government to establish a responsible position regarding software quality. Even though this proposed position would not likely cause immediate practice changes, it would establish a responsible public attitude. It would also provide a clear means for the public to distinguish high-quality offerings from all others.

26. This may not seem advantageous to the U.S. software industry today, but the quality suppliers will soon want ways to distinguish themselves from those who are unscrupulous or incompetent. Also, many other 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.