

INDEPENDENT SOFTWARE MEASUREMENT'S ROLE IN THE LIABILITY PUZZLE

Jeffrey Voas
Reliable Software Technologies
Suite 250, 21515 Ridgetop Circle
Sterling, VA 20166 USA
Tel: 703.404.9293
Fax: 703.404.9295
jmvoas@rstcorp.com

Abstract

Software measurement has long been the main way by which an organization compares itself against the prevailing “common wisdom” in software quality. For example, metrics can be applied to assess structural complexity, a characteristic of which more is considered undesirable. While self-measurement still continues to be widely applied, independent software measurement by private certification authorities is becoming increasingly sought. This paper will look at the state-of-the-industry in independent software certification, proposed software liability laws, and how the emergence of Off-The-Shelf (OTS) software increases the need for independent software measurement.

1. INTRODUCTION

Demands for independent organizations to certify that software programs meet certain criteria are becoming more frequent [6]. These demands are coming from both software producers and consumers and are a direct result of the emerging software component marketplace. That marketplace has evolved because of the economic advantages of reusing shrink-wrapped software.

Recognize that even the best programmers only churn out 10 lines of code per day, which for systems such as cellular phones (that now have around 300,000 lines of code in them), has made custom software development very expensive [4]. Thus, organizations now recognize the cost effectiveness of software reuse.

A problem exists, however, when the code being reused was produced by another organization. The problem is that the adopter of the software will likely know little (if anything) about the development practices employed by the supplier. Thus, other than contractual or verbal guarantees,

the adopter is at the complete mercy of the supplier.

This “unknown quality” situation is fueling demands for independent software certification. The key reasons why I believe independent certification will become a regular part of the software industry are: (1) component vendors prefer not to be responsible (liable) for guaranteeing their own software’s quality, (2) software consumers want unbiased, independent assessments, and (3) the costs of performing certification can be reduced if organizations that specialize in software measurement are employed.

The advantage of having Software Certification Laboratories (SCLs) is that they provide an “unbiased playing field” for all software vendors—each product can receive equal treatment. To enable SCLs to be unbiased, standardized software measures whose interpretations are unambiguous must be provided.

2. INDEPENDENT CERTIFICATION IS ALREADY OCCURRING

Curiosity and speculation concerning the idea of having private, non-government SCLs is growing. And in several cases, this curiosity has already resulted in independent, for-profit SCLs. Examples include: KeyLabs, VeriSign, the International Computer Security Association (ICSA), and the Software Testing Assurance Corporation. We will now briefly discuss each of these organizations and the different types of software certification services that they offer.

Recently, the 100% Pure Java certification process was introduced by Sun Microsystems. Sun Microsystems had a public competition in which different organizations competed for the right to be the sole SCL for the Java language. KeyLabs (<http://www.keylabs.com/>) won this competition. The 100% Pure Java certification process is simple: vendors ship their Java source code to KeyLabs who then tests to see that the code meets the Java language specification. This certification process only tests for language purity. KeyLabs also performs Windows compliance testing and Novell compliance certification testing.

VeriSign (<http://www.verisign.com/>) is an organization that provides digital certificates that software components were indeed produced by the parties that claim to have originated them. VeriSign offers various classes of certificates. For example, a VeriSign’s Class 1 certificate costs around \$20 and simply states that the named individual was responsible for producing a component or application. This certification process only validates the origin of software. Note that neither KeyLab’s certificate nor VeriSign’s certificate make any claims about the quality of the software. KeyLabs certificate guarantees that the software is written in Java, and VeriSign’s certificate simply authenticates the software’s origin. Neither of these certification schemes employ software measurement.

In contrast to KeyLabs and VeriSign, the International Computer Security Association (<http://www.ncsa.com/>) and the Software Testing Assurance Corporation (<http://www.STACorp.com/>) offer certificates that do say something about the quality of the software. To do so, these certification schemes are based on more traditional measures of quality. The ICSA certifies security-related characteristics of a variety of software products: anti-virus

software, firewalls, cryptography products, and Internet access control software. In the firewall and virus certification processes, sets of known vulnerabilities and viruses (from their zoo) are used, and the products undergoing certification are tested against these known vulnerabilities.

The current certification process offered by the Software Testing Assurance Corporation (STAC) involves making sure that all Y2000 software fixes are subjected to a standard set of testing techniques. This standard requires coverage testing and functional testing (in order to ensure that code modifications are exercised as well as functional tests that employ dates beyond 2000). Because of future catastrophes that may occur as a result of organizations not getting their software fixed in time, this effort has the backing of several Wall Street organizations as well as some members of the insurance industry. The main benefit of undergoing this certification process is that insurance providers can justify offering reduced business insurance premiums for those organizations that have tested their Y2000 fixes. While this Y2000 certification is the only software certification currently offered by STAC, STAC has plans for additional certification standards.

But commercial software vendors are not the only organizations that can benefit from the idea of independent software certification. Back in the early 1990s, NASA felt the need for standardized, independent software certification for both the software they write and purchase from contractors. So NASA built their own SCL—the Independent Verification & Validation facility in Fairmont, WV. Intermetrics (<http://www.intermetrics.com/natlprio/assure.shtml>) is the independent certification organization at the Fairmont facility. By creating an SCL, NASA enjoys having access to uniform software assessment that can be applied to all of NASA's software projects (as opposed to each NASA laboratory performing whatever software measurement they prefer).

3. CERTIFICATES AND LIABILITY

Reduced insurance premiums, component vendors preferring to not be responsible for guaranteeing their own software's quality, software consumers wanting unbiased, independent assessments, and lower certification costs are all reasons that independent certification will become a viable industry. Two additional reasons include: (1) the fear of being sued over defective software, and (2) distrust of shrink-wrapped COTS and third-party software. We will now discuss these in more detail.

In the United States, the fear of being sued exists in any contractual endeavor. Buying and selling software is no different. Legal liability cases pertaining to software have already been contested in the courts. In 1986, the first American court case to ever result in a software development company being found guilty of software engineering malpractice occurred [1]. Another interesting American case also occurred in 1986 which involved whether enough software testing had been performed. The case of Shapiro Budrow and Associates Inv. v. Microdata Corp [1986 84 Civ.3589 CBM] occurred because the seller of the software (the defendant) claimed that the software had been "totally tested" yet the software did not work as advertised for the plaintiff. As it turned out, the software had only been tested in-house and had never been operationally tested in the field. Even though the software failed, the seller prevailed because the plaintiff failed to prove that "totally tested" meant "field tested." Therefore, when the seller said "totally tested", the seller's

in-house testing could legitimately be construed as “totally tested.”

Software malpractice is a great concern for every software supplier. Even system integrators who simply license COTS software could find themselves as co-defendants because of integrating defective software into their systems. If damages are incurred from these composite systems, charges against integrators could be negligence to adequately validate the code that they acquired.

Clearly SCLs have the potential to aid vendors and consumers in avoiding such ugly legal situations. Vendors can reduce their punitive liability by voluntarily allowing qualified auditors to independently validate quality claims about the products. This is because vendors who show this willingness to allow a second set of eyes to look for problems have taken an additional voluntary extra step towards due diligence. Further, consumers will be able to communicate directly with the SCL to find out what software measures were taken and what results were calculated (as opposed to getting that information from the vendor). Doing so shows due diligence on their part as well; it shows that their concern resulted in their reaching out to the SCL for information.

SCL certificates play a similar role as to what already occurs in government-regulated industries. In those industries, regulatory compliance with certain standards is used as a shield against punitive liability. For example, if an airplane were to crash due to faulty software, but that software manufacturer correctly applied all of the required FAA certification processes in standard DO-178B [3], then the manufacturer can show that they adhered to the “best practices” for their industry. Clearly the supplier of the software will pay damages for the loss of life and aircraft, but the punitive damages for negligence are more likely to be reduced or waived.

3.1 Uniform Commercial Code Article 2B

So far, we have discussed the role of SCLs in reducing vendor and consumer software liability. Interestingly, vendors may be slated to receive an even greater “liability reducer” than access to SCLs: Article 2B of the *Uniform Common Code* [5]. If this addition is approved in its current form, vendors may have a reduced interest in independent certification. We will now discuss why.

In the United States, the Uniform Common Code is the series of laws governing all financial transactions that are not covered by written contracts. For example, you assume that when you buy a banana in a grocery store there is a banana under the skin. If there is not, the Uniform Common Code provides you with the right to demand your money back.

Article 2B is a planned addition to the Uniform Common Code that handles “transactions in information.” It handles the laws for transactions relating to the “copyright industries.” While it sounds reasonable to finally have laws governing what is a fair legal transaction in software, Cem Kaner pointed out that Article 2B has little protection in it for the consumer and instead is full of vendor protection. He points out that the consumer will ultimately suffer from bad software and not the vendors [2]:

“Article 2B denies the mass-market customer most remedies, even a refund of the fee charged by the publisher for calling to report problems. Customers are entitled to

refunds only for "material" breaches of contract, and 2B redefines "material" to be narrower and less inclusive than the Restatement of Contracts. Article 2B denies the remedies even when the customer's damages are caused by a defect that was known to the publisher before the product was released for sale, that the publisher chose not to fix and chose not to warn the customer about.

Article 2B lets publishers pick what law will govern their sales and where customers can sue them. There are no geographical restrictions and no requirement of any relationship between the law, the forum and any party or aspect of the transaction. Small claims court actions will be unavailable (when exclusive jurisdiction is given to a higher-level court) or prohibitively expensive (in a state or country far, far away). Article 2B is a forum shopping gone wild, and available only to the publisher. There are slight restrictions on this if the customer is a "consumer" who uses the software for strictly non-business, non-professional purposes (a teacher who does research or writes assignments at home is not acting as a consumer, nor is the unemployed secretary who tries out some home-based network marketing scheme and uses a computer to manage her mailing lists and print fliers.) Few customers with meritorious cases will have the ability to bring a lawsuit against a publisher.

Article 2B settles the Battle of the Forms (the traditional problem of conflicting terms and expectations set between the publisher and the customer) by creating a new set of procedures that will ensure that the publishers wins the battle. The publisher gets the last shot, in the "mass-market license agreement," which the publisher need not even make available to the customer until after the customer has paid for the product, taken it away, and started installing it on his computer. With extremely few exceptions, all of the terms in this "license" "agreement" will be fully enforceable against the customer as if he had reviewed, discussed, and signed a paper contract before the sale."

Article 2B is not yet law. The November 1, 1997 draft of Article 2B [5] was supposed to be brought before the State Legislatures in September of 1997 but was not. It is unknown whether this will occur in 1998 either. (Article 2B is rumored to be highly controversial among the lawyers that are drafting it.)

Although vendor support for SCLs could wane if the current language in Article 2B is approved, SCLs still have a vital role: *consumer protection*. If nothing else, Article 2B should force consumers to realize that their legal protection is woefully inadequate under Article 2B. They will hopefully then demand tougher software measures from SCLs. That is, consumers could opt to use SCLs as a way to counterbalance the lack of consumer protection in Article 2B.

3.2 States Limiting Liability

In addition to Article 2B, we see other vendor-oriented (and thus consumer unfriendly) legislation on the horizon. California Assembly Bill 1710, introduced January 28, 1998, if passed, will amend the California Civil Code to limit recovery for damages resulting from the Y2000 date transition. If this bill becomes law, in a lawsuit for a Y2000 defect, the plaintiff could recover only the costs of (1) damages that resulted from bodily injury, and (2) fixing the bug. The bill states:

“(a) Notwithstanding any other provision of law, in any action to recover damages resulting directly or indirectly from a computer date failure, including any action based on an alleged failure properly to detect, disclose, prevent, report on, or remediate a computer date failure, the damages that may be recovered shall be limited to either or both of the following, according to proof:

(1) Any damages resulting from bodily injury, excluding emotional injury, to the plaintiff proximately caused by the defendant’s conduct.

(2) Any costs reasonably incurred to reprogram or replace and internally test the relevant computer system, computer program or software, or internal hardware timer, to the extent those costs are incurred as a proximate and direct result of the defendant’s conduct.”

Here again, we see the purchaser of defective software ultimately having little legal recourse.

4. WHAT TO MEASURE IN SOFTWARE

To play the role of the consumer protection advocate, SCLs must measure software characteristics that provide accurate and meaningful information about the software’s quality. For example, telling the consumer that there are 1 million lines of code in a program is not very meaningful. It would be far more meaningful to tell the consumer that those 1 million lines of code are highly reliable, well tested, and efficient.

But doing so puts the SCL into a tricky legal quagmire of its own. Problems arise if an SCL certifies a program when it should not have or vice-versa. Therefore the key to sufficient legal protection as a SCL is three-fold:

1. Only provide objective measures of quality if possible. For example, if you are assessing what code coverage was achieved, simply state what parts of the code were exercised during testing and which parts were not. Do not try to interpret the results and say that those parts that were not exercised did not need to be. That could easily be debated in a court of law.
2. Ensure that any subjective code measure (such as reliability estimation) is labeled with a “relative” interpretation as opposed to an “absolute” interpretation. For example, if a component’s mean-time-to-failure (MTTF) scores decrease from 100 (per unit time) to 10, that suggests that the software is becoming less reliable, *relatively* speaking. It would be foolish to believe that the software will *absolutely* not fail until 10 units of time have passed, and so the SCL should not suggest such. Also, if an SCL provides subjective code measures such as reliability estimation as an offering, they will need to precisely define all parameters used during the testing (e.g., the test distribution, the particular reliability model used along with any predetermined parameters, the number of test cases, etc.) from which those estimates were found.
3. Do not make certification into an all or nothing proposition. Simply provide valid results from a set of software measures. This places the onus on the consumer to decide whether the

actual measurements the SCL published for the product are satisfactory for their needs. This avoids the problem of granting certificates when they should not be granted and vice versa (as long as the results the SCL publishes for the product are correct).

In summary, we believe that SCLs can successfully measure component quality using subjective and objective measures. All measures should be provided without subjective interpretation. Let the consumer decide what those results mean.

5. SUMMARY

Capturing metrics for the sake of collecting uninterpretable numbers has long been an accusation against the software metrics community. Critics, mainly in the formal methods communities, have long considered software measurement useless. But software measurement is gaining in interest as volumes of software with unknown quality continues to enter the market.

Because we have no professional codes of behavior for software developers, it is difficult (although not impossible) to charge a developer with negligence. Developers do not take professional exams, and therefore most employers can only decide whether to hire a developer based on references, code samples, and claims about past experience. Therefore, a software development organization takes liability risks each time it hires a developer. Fortunately, SCLs will provide such organizations with an opportunity to demonstrate that producing quality software was a priority for that organization. This clearly helps the case of any organization that finds itself as a defendant.

Whether Article 2B or legislation like California's Assembly Bill 1710 will cause vendors to feel that independent certification is unnecessary is unclear. But simply having these "potential" laws waiting in the wings must make software consumers pause and consider that SCLs may someday be their only source for accurate quality measurements for Off-the-shelf and third-party software.

Acknowledgments

This work has been partially supported by DARPA Contracts F30602-95-C-0282 and F30602-97-C-0322, National Institute of Standards and Technology Advanced Technology Program Cooperative Agreement Number 70NANB5H1160, and Rome Laboratories under US Air Force Contract F30602-97-C-0117. THE OPINIONS AND VIEWPOINTS PRESENTED ARE THE AUTHOR'S PERSONAL ONES AND THEY DO NOT NECESSARILY REFLECT THOSE OF THESE AGENCIES.

References

- [1] Data Processing Services, Inc. v. L. H. Smith Oil Corp., 492 N. E. 2d 314 (Ind.App. 1986).
- [2] C. KANER. Article 2B is Fundamentally Unfair to Mass-Market Software Customers, October 1997. Submitted to the American Law Institute for its Article 2B review.
- [3] FEDERAL AVIATION AUTHORITY. Software Considerations in Airborne Systems and Equipment Certification, 1992. Document No. RTCA/DO-178B, RTCA, Inc.

- [4] S. BAKER, G. MCWILLIAMS, AND M. KRIPALANI. "Forget the Huddled Masses: Send Nerds", *Business Week*, July 11, 1997.
- [5] THE AMERICAN LAW INSTITUTE AND NATIONAL CONFERENCE OF COMMISSIONERS ON UNIFORM LAWS. *Uniform Commercial Code Article 2B (DRAFT)*, November 1997.
- [6] J. VOAS. *Software Certification Laboratories: To Be or Not to Be Liable?* *Crosstalk*, April 1998.