Software Certification Laboratories
To Be or Not to Be Liable?

Jeffrey Voas
Reliable Software Technologies

Software certification laboratories (SCLs) will potentially change the manner in which software is graded and sold. However, a main issue is who is liable when, during operation, certified software acts in a manner that the SCL certified was not possible. Given software's inherently unpredictable behaviors, can SCLs provide precise-enough predictions about software quality to reduce their liability from misclassification to a reasonable level? SCLs' survival and effectiveness largely depend on solving the issues of certification liability.

If you visit a doctor's office, you will often hear terms such as "independent laboratory," "second opinion," "additional tests," or "colleague consultation." What these amount to is a doctor getting another party or process involved in a diagnosis or treatment decision. Doctors use outside authorities, in part, to reduce the risk of malpractice. The more consensus built with respect to a particular course of action, the more due diligence has been shown. And more parties are culpable if something goes wrong. For instance, if a medical laboratory returns a false diagnosis that a tissue sample is cancerous and the doctor begins treatments that were not necessary, the doctor can ascribe some or all of the liability for this mistake to the laboratory. The added costs from spreading liability around in this manner are one reason for the cost increases in health care. Each extra opinion and extra test increase patient costs, because each care provider is a malpractice target.

A Demand for Independent Certification

In the software world, a similar phenomenon is being observed. Software producers and consumers are more frequently demanding that independent agencies certify that programs meet certain criteria. Vendors prefer to not be responsible for guaranteeing their software, and software consumers want unbiased assessments that are not based on a sales pitch. Incredible as it may seem, vendors, who typically "cut all corners" in costs, are willing to pay the costs associated with placing this responsibility on someone else.

Because of the demands for SCL services, business opportunities exist for organizations that wish to act in this capacity. By paying SCLs to grant software certificates, independent software vendors (ISVs) partially shift responsibility onto the SCL for whether the software is "good." The question is whether this method of liability transfer will be as successful in software as it has been in health care. As we will discuss, if SCLs set themselves up right, they can build more protection around themselves than you might think, leaving the ISV holding a "hot potato."

There are several relatively obscure SCLs in existence today, e.g., KeyLabs, which handles applications for 100 percent pure Java. Other than these small, specialized laboratories, the next closest match to an SCL (conceptually speaking) is Underwriter's Laboratory (UL). UL certifies electrical product designs to ensure that safety concerns are mitigated. Rumors are that UL is interested in performing SCL services, but to my knowledge, UL has not yet become an SCL.

Commercial software vendors are not the only organizations that see the benefit of SCLs. NASA felt the need for standardized, independent software certification for the software they write and purchase. NASA now has an SCL—the Independent Verification and Validation Facility in Fairmont, W. Va. Intermetrics is the prime contractor at the facility, and their job is to oversee the certification process and provide the necessary independence. This SCL provides NASA with a common software assessment process over all software projects (as opposed to each NASA center performing assessments in different ways). The NASA facility certifies software developed both by NASA employees and NASA's contractors.

The beauty of having SCLs is that they provide a quasi-fair "playing field" for all software vendors—each product is supposed to be given equal treatment. The issue is that when software fails in the field, and an independent party provided an assessment that suggested that the software was good, does the independent party bear any responsibility for the failure?

Who Is Liable When Software Fails?

Who is liable when certified software fails—the ISV, the SCL, both, or neither? More specifically, how is liability divided between these groups? First, the question of how much liability, if any, can be placed onto the SCL will be addressed. By figuring out the liability incurred by an SCL for its professional opinions, we can determine how much liability is offloaded from the ISV.

Limiting Liability

SCLs stand as experts, rendering unbiased professional opinions. This exposes SCLs to possible malpractice suits. Schemes to reduce an SCL's liability include insurance, disclaimers on validity of the test results, and SCLs employing accurate certification technologies based on objective criteria. Of these,

the best approach is to only certify objective criteria, and avoid trying to certify subjective criteria.

Subjective vs. Objective Criteria

Different software criteria can be tested by SCLs, spanning the spectrum from guaranteeing correctness to counting lines of code. Subjective criteria are imprecise and prone to error. Objective criteria are precise and less prone to error. For example, deciding whether software is correct is subjective because of the dependence on the precise definition of "correctness." SCLs should avoid rendering professional opinions for criteria that are as contentious as this.

Instead, SCLs should assess objective software characteristics such as exception handling calls and lines of code. Testing for objective criteria is not rocket science. Troubles will begin, however, when an SCL tries to get into the tricky business of estimating a subjective criterion such as software reliability. By only certifying objective criteria, the chances of inadvertent favoritism for one product over another is reduced.

The ICSA Certification Approach

The International Security Computer Association (ICSA) is a for-profit SCL that has taken an interesting approach to the liability issue. They use industry consensus building. ICSA only certifies that specific known problems are not present in an applicant's system. This is an objective criterion. Their firewall certification program is based on the opinions of industry representatives who periodically decide for which known potential problems the software should be checked. Over time, additional criteria are introduced into the certification process. This adaptive certification process serves two purposes: it adds rigor to the firewall certification process and produces a steady stream of business for the ICSA. To further reduce liability, ICSA does not claim that their firewall certificate guarantees firewall security.

ISVs' Liability Concerns

ISVs have a different liability concern, particularly when their software fails in the field. For example, suppose an SCL says that an ISV's software is "certified to not cause problem X." If the software causes problem X and fails, and the ISV faces legal problems, can the ISV use their SCL certificate as evidence of due diligence? Can the ISV assign blame to the SCL? The answer to the first question is "probably," and the answer to the second question depends on what "certified to not cause problem X" means. If the certification was based on objective criteria and the process was performed properly, the ISV probably cannot blame the SCL. If the process was improperly applied, the SCL may be culpable. If subjective criteria were applied, the answer is unclear.

If the SCL used consensus building to develop their certification process, the question that may someday be tested in the courts is whether abiding by an industry consensus on reasonable criteria protects SCLs from punitive damages. Generally speaking, as long as a professional adheres to defined standards, punitive damages are not administered. Professions such as medicine, engineering, aviation, and accounting have defined standards for professional conduct.

Lack of Professional Standards

Software engineering has never had such standards, although several unsuccessful attempts to do so have been staged. Also, there are no state-of-the-practice rules to determine if code meets professional standards. For example, the title "software engineer" is legally invalid in 48 of 50 states. In these states, the title "engineer" is reserved for people who have passed state-sanctioned certification examinations to become professional engineers [1]. Because the software engineering field does not have professional standards, it could also be argued that the actions of organizations such as the ICSA are laudable.

Because software engineering has no professional organization to accredit its developers, the approach taken by the ICSA could also be argued in a court of law to be state of the practice. If argued successfully, software developers whose software passed the certification process could expect to avoid punitive damages. But if these state-of-the-practice standards are deliberately weak, even though consensual, satisfaction of the standards may fail to persuade a jury. It is widely held by the public that the policy of industry self-regulation has failed. When those being forced to comply are those making the rules, are the rules trustworthy? Challenges in the courts could be foreseen, claiming a conflict of interest. This would invalidate claims that consensus-based standards sufficiently protect customers.

However, the commercial aviation industry is an example of the successful application of industry-guided standards. Rigorous software guidelines in the DO-178B standard were approved through industry and government consensus. These software safety guidelines remain the most stringent software certification standards in the world. It is clear that the Federal Aviation Administration's influence played a role during the formation of these standards.

There are self-correcting mechanisms that work to some degree in self-policing industries. If an industry such as air travel failed to police itself, it would lose so much favor with its customer base that the entire industry could fail.

Limiting ISV Liability

Possibly the best defense for any ISV is the use of disclaimers, not reliance on an SCL. There is a perverse advantage to disclaiming one's own product. The less competent an ISV portrays itself to be, the lower the standard of professionalism to which it will be held. Taking this principle to an extreme, we might suggest that a disclaimer be included in a comment at the top of each program stating, "This software was developed by incompetent people trying to learn how to program, and it probably does not work." The degree to which this tongue-in-cheek disclaimer reflects reality is a sad commentary on the state of our industry. But until more

cases are tested in the courts, no one knows how much protection software disclaimers will afford.

Article 2B

There is one more interesting development that has occurred: a draft of Article 2B of the Uniform Commercial Code (UCC) (which pertains to computers and computer services) was released Nov. 1, 1997 [2]. Article 2B will play an important role in defining software warranties. Article 2B will only serve as a model template, and each state in the United States will be responsible to modify it to their standards before adopting it as law. Further, Article 2B has the potential to relax the liability concerns that might force an ISV to use a certification laboratory. This could turn out to be a disaster for those parties most concerned with software quality.

Conclusion

Before we can determine what role SCLs will play in software liability, we must wait for more cases to be tested in court to see to what standard of professionalism ISVs are held. If the criteria for which SCLs test are not meaningful, SCLs will find that neither developers nor consumers of software care about the certification process.

For SCLs to succeed, it also is imperative that they employ accurate assessment technologies for objective criteria. If SCLs do this, malpractice suits against them will be difficult to win unless they mishandle a particular case or make false statements.

This article is entitled "Software Certification Laboratories: To Be or Not to Be Liable" because until these hard issues are resolved, it is hard to measure the degree of liability protection afforded an ISV by hiring the services of an SCL. Nonetheless, if SCLs can measure valuable criteria (and by this I do not mean "lines of code") in a quick and inexpensive manner, SCLs have the ability to foster greater software commerce between vendors and consumers. This could move an SCL certificate from being viewed as a tax to a trophy. ◆

About the Author

Jeffrey Voas is a co-founder of and chief scientist for Reliable Software Technologies and is currently the principal investigator on research initiatives for the Defense Advanced Research Projects Agency and the National Institute of Standards

and Technology. He has published over 85 refereed journal and conference papers. He co-wrote *Software Assessment: Reliability, Safety, Testability* (John Wiley & Sons, 1995) and *Software Fault-Injection: Inoculating Programs Against Errors* (John Wiley & Sons, 1997). His current research interests include information security metrics, software dependability metrics, software liability and certification, software safety and testing, and information warfare tactics. He is a member of the Institute of Electrical and Electronics Engineers and he holds a doctorate in computer science from the College of William & Mary.

Reliable Software Technologies
21515 Ridgetop Circle, Suite 250
Sterling, VA 20166
Voice: 703-404-9293
Fax: 703-404-9295
E-mail: jmvoas@rstcorp.com

References

1. Jones, C., "Legal Status of Software Engineering," *IEEE Computer*, May 1995.
2. UCC Article 2B (Draft), November 1997, the American Law Institute and the National Conference of Commissioners on Uniform State Laws.

CMM, from page 20

time and within budget. Best of all, we achieved exceptional customer satisfaction, which is, after all, what counts. ◆

About the Author

Rita Hadden is the information systems performance practice leader at Project Performance Corporation. She has provided leadership, coordination, and coaching on more than 70 software projects for more than 35 organizations.

Her software engineering and management experience includes 28 years working with multiple teams of developers and managers. She has successfully managed cross-platform information system projects for the private and public sectors. She is an acknowledged leader in industry best practices, software process improvement, and corporate culture change. She has helped organizations worldwide mature their software capabilities and meet their business objectives. She is certified by the SEI as a lead software process assessor.

Project Performance Corporation
20251 Century Boulevard
Germantown, MD 20874
Voice: 301-601-1810
E-mail: rhadden@ppc.com.