

Software Certification Laboratories?

Jeffrey Voas

Reliable Software Technologies

ABSTRACT *Software Certification Laboratories (SCLs) will potentially change the manner in which software is graded and sold. The main issue, however, is who is to blame when a certified piece of software acts in a manner during operation that the SCL certified was not possible?. Given software’s inherently unpredictable behaviors, can SCLs ever provide precise enough predictions about software quality to reduce their liability from misclassification to a reasonable level?*

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 that gets built with respect to a particular course of action, the more *due diligence* has been shown. And the more parties that are then culpable if something goes wrong. For instance, if a medical lab falsely returns a 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 onto 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.

In the software world, a similar phenomenon is being observed. Demands for *independent* agencies to certify that programs meet certain criteria are becoming more frequent. These demands are coming from software producers and consumers. Vendors prefer to not be responsible for guaranteeing their own software, and software consumers want unbiased assessments that are not based on sales pitch hype. 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.

The beauty of having Software Certification Laboratories (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?

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 (like when a doctor orders

a second opinion or another test) for whether or not 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% Pure Java). Other than these small, specialized labs, the next closest organization to what you would think of as a 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 UL has not yet become an SCL to our knowledge.

Commercial software vendors are not the only organizations that see the benefit of SCLs. NASA felt the need for standardized, independent software certification both for the software they write as well as the software they purchase. NASA now has their own SCL—the Independent Verification & Validation facility in Fairmont, WV. 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 assessment in different ways). The NASA facility certifies both software developed by NASA personnel as well as NASA’s contractors.

Our interest in SCLs is in figuring out who is liable when *certified software fails*. The ISV, the SCL, both, or neither? More specifically, we are interested in how liability is divided between these groups? We will first address the question of “how much liability, if any, can be placed onto the SCL?” By figuring out the liability incurred by an SCL for its professional opinions, we can determine how much liability is offloaded from the ISV.

SCLs stand as experts, rendering unbiased professional opinions. This opens up the SCL to possible malpractice suits. Schemes for reducing 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.

Different software criteria can be tested for 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 what “correctness” really means for a piece of software. SCLs should avoid rendering professional opinions for criteria that are as contentious as this. But SCLs should be able to assess characteristics such as whether a program has exception handling calls in it and how many lines of code a program has. Testing for these criteria is not “rocket-science.” Troubles will begin, however, when an SCL tries to get into the tricky business of estimating a criterion such as software reliability. Further, by only certifying objective criteria, the chances of inadvertent favoritism being shown to one product over another is reduced.

The National Security Computer Association (NCSA) is a for-profit SCL that has taken an interesting approach to this liability issue. They use industry consensus building. NCSA only certifies that specific *known* problems are not present in an applicant’s system. This is an objective criteria. Their firewall certification program is based on the opinions of industry representatives who meet periodically to decide what known problems should be checked for.

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 it produces a steady stream of business for the NCSA. To further reduce liability, NCSA adds the standard disclaimer package that their firewall certificate does not guarantee firewall security.

ISV's have a different liability concern, particularly when their software fails in the field. For example, even if an SCL tells an ISV that their software is "certified to not cause Problem X", when the software fails causing Problem X and the ISV faces legal problems, can the ISV use their SCL certificate as evidence of due diligence? Further, 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" meant. If this 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, then the SCL will probably be culpable. If subjective criteria were applied, the answer is unclear.

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

Software engineering has never had such standards, although several unsuccessful attempts to do so have been waged. State-of-the-practice rules that differentiate code meeting professional standards from code not meeting professional standards are non-existent. Consider the fact that the phrase "software engineer" is illegal in 48 of 50 states because the term "engineer" is reserved for persons who have passed state-sanctioned certification examinations to become professional engineers [1]. Because we do not have professional standards, it could also be argued that what organizations such as the NCSA have done is laudable.

Since software engineering has no professional organization to accredit its developers, the approach taken by the NCSA could also be argued in a court of law as 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 satisfy a jury.

The reason for this is because it is widely held by the public that industry policing itself is a failed policy. 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 protected customers. One example of where industry-guided standards have worked quite well, however, is the commercial aviation industry. Here, rigorous software guidelines in the DO-178B standard were approved through an industry/government consensus. Those guidelines for software safety are still the most stringent software certification standards in the world. No doubt the FAA's influence during the formation of these standards played a role here. And it cannot be ignored that an industry such as air travel, if it were to fail to police itself, would lose so much favor with its customer base that the entire industry could fail. So there are self-correcting mechanisms that do work to some degree in self-policing industries.

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 themselves to be, the lower the standard of professionalism to which they 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 persons trying to learn how to program and it probably does not work*. The degree to which this tongue-in-cheek disclaimer actually reflects reality is a sad commentary on the state of our industry. But until more cases are tested in the courts, who really knows how much protection software disclaimers really afford.

There is one more interesting development that will occur in the near future and that is Article 2B of the Uniform Commercial Code (which pertains to computers and computer services). Article 2B will be released in the Fall of 1997, and it will play an important role in defining software warranties. Note that Article 2B will only serve as a model template, and each state in the US will be responsible for modifying it to their tastes 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 lab. This could turn out to be a disaster for those parties most concerned with software quality.

In summary, we are going to have to wait for more cases to be tested in the courts to see what standard of professionalism ISVs are held to before we will know what role SCLs play in software liability. We can say that if the criteria that SCLs test for are not meaningful, SCLs will find that neither developers nor consumers of software care about the certification process. For an SCL to succeed, it is also imperative that the SCL employ accurate assessment technologies for objective criteria. If SCLs do this, malpractice suits against them will be very difficult to win, unless the SCL simply fowls up on a particular case or makes false statements.

This piece is titled "Software Certification Laboratories?", because until these hard issues are resolved, the degree of liability protection afforded an ISV by hiring the services of an SCL is hard to measure. Nonetheless, if SCLs can measure valuable criteria (and by that 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. And this could move SCL certificates from being viewed as taxes to trophies.

References

- [1] C. JONES. Legal status of software engineering. *IEEE Computer*, May 1995.