Andy Chou on ASQA:
Automated Software Quality Assurance Really Matters (3)

by M. E. Kabay, PhD, CISSP-ISSMP
Associate Professor of Information Assurance
School of Business & Management
Norwich University, Northfield VT

In this, the third part of an interview about automated software quality assurance (ASQA), Andy Chou< http://www.linkedin.com/pub/andy-chou/0/88/149 >, Chief Scientist & Co-founder of Coverity < http://www.coverity.com/ >, finishes with some interesting case studies about real-world application of ASQA.

6) Have you ever seen an absolute disaster in the implementation of ASQA? Can you tell the readers what happened (without giving embarrassing identifying details) and the lessons you and your colleagues learned from the experience?

None of the challenges to automated software integrity analysis are fundamental or insurmountable, but sometimes people don’t understand the technology, what it can do for them or how to deploy it. This might mean that they make poor decisions when implementing it or helping developers understand how to use it.

We recently wrote a paper for the Communications of the ACM 53(2):66-75 on the challenges we faced bringing Coverity and automated integrity testing to market, “A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World.”< http://cacm.acm.org/magazines/2010/2/69354-a-few-billion-lines-of-code-later/fulltext > We write about these challenges and the mind-set change it takes to be successful.

Some of the key findings we discussed in that paper about the differences between a research lab and the real world include the following:

- The volume of tests in real-world applications of ASQA is orders of magnitude greater than in the product-development lab. The number of language dialects, programming styles, bugs, and false positives all go up when testing real-world software.
- Users don’t necessarily have the same perspective as ASQA tool builders; they may interpret error messages very differently from the developers’ intentions. The product must provide rapid processing and clear results that can be understood with minimal training.
- Sometimes programmers or system managers actually stop the ASQA tool from testing specific parts of their code, making the results untrustworthy and incomplete.
- Differences in the technical platforms used to generate compiled code may not be compatible with the ASQA tool, including even radical differences in the interface used by the programming teams (i.e., graphical user interfaces vs command-line interfaces).
- Company policies may forbid even harmless changes to a production sequence, making a specific ASQA tool unusable because of minor incompatibilities with the operating environment.
- Compilers that don’t reject illegal source-code constructs as defined by language standards can produce unparsable code that the ASQA tool cannot test – and cause conflict with the programmers who define their code as, say, C++ if it is accepted by their C++ compiler no matter how illegal their constructs are.
Many sites involving safety-critical systems are stuck with ancient compilers because it’s too expensive to recertify the safety-critical software every time a compiler changes version. These old compilers – sometimes decades old – cannot be purchased at all by the ASQA maker and so are very difficult to include in ASQA-tool testing.

Programmers may insist that the bugs found in their code are not bugs – including for example treating buffer overflows as normal!

The organizational culture may cause people to dismiss many bugs as unimportant because they don’t have any direct effect on themselves.

7) Where do you see ASQA heading in the coming years? What’s next on the horizon in terms of ease of use, cost-effectiveness and speed?

The next few years we’re going to see a huge focus on the software supply chain. Companies are beginning to realize that they need to understand the integrity of their software regardless of whether they develop the software themselves or integrate software and components from third party suppliers or open source. I think we’ll see more and more companies measuring and rating the integrity of the software and components they receive from their supply chain.

People don’t realize the risk they take in shipping software. For instance, Toyota<http://www.google.com/hostednews/ap/article/ALeqM5jiTG7SuUsaqyE6b09GPluAfU5blewD9GTVER00> is a case where users start to wonder, what’s in there? Will it work? In the coming years, I believe awareness will grow. There will be more software failures. As software becomes increasingly pervasive we will see systems breakdown – software integrity will become a business and regulatory priority, managed at the highest levels.

***

Andy Chou<http://coverity.com/html/about_leadership.html>, PhD is co-founder and Chief Scientist of Coverity, Inc.<http://coverity.com/html/about_fact.html> He is responsible for advancing source code analysis technology at Coverity as well as furthering the state of the art in software integrity industry-wide.


Copyright © 2010 Andy Chou & M. E. Kabay. All rights reserved.

Permission is hereby granted to Network World to distribute this article at will, to post it without limit on any Web site, and to republish it in any way they see fit.