Automated Testing a Must for Effective SQA

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

In software quality assurance (SQA), testing plays a crucial role: you can’t assert that a program works until you have tried your best to show that it doesn’t.

Sometimes students express surprise at this formulation of the purpose of testing; beginners often believe that testing is designed to show that a program works. Thus a student will proudly present a program that has been tested using, say, all the expected inputs and that has produced all the expected outputs. The poor child is then astonished when an experienced programmer cheerfully throws unexpected inputs at the program and promptly crashes it.

Any textbook on software development will explain that there are different types and stages of testing in SQA; for example, a common roster of tests includes

- Module / Unit Testing (checking individual pieces of the program)
- Integration Testing (seeing if the individual pieces interact correctly)
- Function Testing (verifying that the program does what it was designed to do)
- System Testing (many aspects – see below)
- Acceptance Testing (end user tests to see if the program conforms to contract)
- Installation Testing (watching to see that the program has been installed properly).

System testing has many aspects, most of which are self-explanatory; for example,

- Facility Testing
- Stress Testing
- Volume Testing
- Usability Testing
- Security Testing
- Performance Testing
- Storage Testing
- Configuration Testing
- Compatibility / Conversion Testing
- Installability Testing
- Reliability Testing
- Recovery Testing
- Serviceability Testing
- Documentation Testing
- Procedure Testing.

As programs increase in complexity, the burden of following scripts to test all the elements of these tests – often repeatedly, since regression testing involves doing all the tests over again every time something is changed – can become wearing on the human beings assigned to the test function. Manual testing is universally considered inadequate for professional systems; it is unstructured, slow, and too dependent on the human tester’s awareness and attention to be reliable. They also usually generate no audit trail and poor or no statistics.
In contrast, structured automated testing systems provide a scripting language to define and edit test instructions, can use databases as a source of input for data in tests, and can handle errors intelligently so that processing does not stop dead the moment an unexpected result occurs. These systems

- Produce consistent, reproducible results;
- Increase test coverage (the proportion of lines of code that are actually executed during the testing process);
- Facilitate easier maintenance of both the source code and the test scripts; document the tests performed and all the results;
- Produce global and detailed statistical reports that help identify problem areas (and possible problem programmers);
- Result in higher-quality software; and
- Cost less to execute that manual tests.

In a specific test I recall from the 1980s when I was directly involved in marketing an automated testing tool, a software development company generated the following results comparing their manual testing results and costs with their trial of the automated testing tool:

- Both testing processes
  - Used 6 people
  - Had 3 test phases per product release
  - Cost $81,00 per release
- The manual process had
  - 3,000 tests per phase
  - 12.5% test coverage (lines of code executed)
  - 15 days elapsed time required per release
- The automated process had
  - 24,000 tests per phase
  - 100% test coverage
  - 5 days elapsed time per release.

Not bad, eh?

For an extensive set of resources on the benefits of automated testing, see the many links available from the Automated Testing Institute<http://www.automatedtestinginstitute.com/>. In particular, there are 329 articles and white papers<http://www.automatedtestinginstitute.com/home/index.php?option=com_alphacontent&view=alphacontent&Itemid=70> available for free download. Topics range from fundamentals through case studies and will be interesting reading for any group involved in software development which has not yet implemented automated testing as part of their overall security assurance process.

***

M. E. Kabay,<mailto:mekabay@gmail.com> PhD, CISSP-ISSMP, specializes in security and operations consulting services and teaching. He is Chief Technical Officer of Adaptive Cyber Security Instruments, Inc.<http://acsi-cybersa.com/> and Associate Professor of Information Assurance<http://norwich.edu/academics/business/infoAssurance/index.html> in the School of Business and Management<http://norwich.edu/academics/business/faculty.html> at Norwich University.<http://www.norwich.edu> Visit his Website for white papers and