How To Test Your Application, Generally Speaking

By Scott Gamron

Scott is BASIS' Quality Assurance Supervisor. He has 18 years of experience as a programmer and developer, five of which have been in software testing.

So you want to test your application. I'm glad to hear it. Most people test their application in one fashion or another. Some of the various ways I've seen are

- The developers test the application
- Technical Support tests it
- The entire company has a testing party and tests it
- The customers test it
- Whoever has time to look at the application tests it, etc., etc.

I'm sure you can come up with a few of your own. While these are all noble ideas, they don't necessarily ensure that the customer gets the application with as few defects as possible. What I'd like to share with you are some basic issues surrounding the question, "How do I test my application?"

Let's talk about quality assurance (QA). According to Webster's dictionary, the definition of quality is *1. Degree of excellence; 2. high social status*. The definition of assurance is *1. Certainty; 2. self-confidence.* According to these definitions, QA is ensuring that the *degree of excellence* is met, with *certainty*. Well, to whom does this apply? This applies to the entire company, including the janitor cleaning the office. What this means is that some lines need to be drawn between QA at a company level versus QA at the software level. People tend to think that because you are in the "QA department" you are responsible for assuring everything.

Here is a more appropriate definition of QA as it applies to software testing. This definition comes from the Institute of Electrical and Electronics Engineers (IEEE) software engineering terminology glossary: *Quality assurance is a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.* So don't get roped into doing more than you are capable of, like ensure that the lunchroom is cleaned according to specs...

Defect Tracking

OK, so you're looking at this application and you have questions or find something drastically wrong. How do you get this information or question to the developer? Also, wouldn't it be helpful if these issues were written down somewhere, just in case the issues comes up again in the future? This is the job of a defect tracking system. Call it what you want: defect tracking system, QA memo system, incident tracking system, trouble report system, etc. It's critical to establish a system to communicate to the developer and to keep a record of all issues. It could be as simple as a pad of paper. The trick is to keep it simple yet thorough enough to be useful in the future.

Manual Testing

Next we look at the product. What do we test? This is where a good specification comes in handy. Review each item in the specification and design a series of tests to test the functions of each item. This is a test plan. You write many test plans, and there, you have a nice test bed.

What?! You don't have a specification?! Don't lose faith. You can still test the product. Probably not as thoroughly and with as much *certainty* as you'd like, but it'll be better than no testing at all. The key is to go through the entire application and identify various areas or modules. Try to break these modules to fairly small chunks, so they'll be easy to maintain in the future. Now generate test plans from the individual modules. You could also go through the user manual (if you have one) and generate test plans accordingly. As an example of how to structure your tests, you may want to break down your test plans according to the menu structure of the application. Some sort of structure is what you need because the main thrust of all of this is to end up with something that you, or someone else, can repeat. Trust me, with any test you perform one time, you'll probably get to perform it at least 10 more times, so if you have what you're trying to do written down, the task becomes a lot easier. This leads us into automation.

Automated Testing

This is a huge timesaver. Imagine a machine running all these manual tests at night. Think of all the time and effort saved, which equates to money saved. This is what automation is all about.

Software automation tools exist that allow you to record all your keystrokes and mouse movements, and play them back. Here at BASIS, we use SQA Robot by Rational Software, but there are many others. You essentially run the manual test with the software automation tool running, recording all of your movements. For example, suppose you want to check the content in a list box. No problem. Just start the recording tool and when you get to the point to check the list box, use the tool to capture the state of the list box, then continue on. When you play the script back, it'll ensure that the list box is in the same state that it was when you recorded it. Then the tool will tell you what test cases passed or failed. Very slick.

It usually takes longer to go through and do this initial recording while you test. But in the long run, every time you have to run this test again, the tool can check the same item in the same way over and over more accurately and faster than you can. From there, you build your test suites from these scripts. But it is critical that you structure your automated tests properly; otherwise, you'll spend a lot of time and effort maintaining the tests. Which brings us to the next topic, overall structure.

Overall Structure

This is where you tie everything together. You need some way of displaying and distributing information about defects, tests, results, etc., to certain people. And you need to be able to deliver accurate estimates to management on how long it takes to test. This is what I'm calling the overall structure of a QA department. If you don't have streamlined procedures for doing this, you'll never have the time you need to do the actual testing with *certainty*. Well, there are many methods. Maybe you have a binder with all the information in a central

location. Maybe you have Microsoft Word documents archived on a certain machine in your organization. Maybe you maintain an intranet page with all this information. This last method is by far the most efficient because everyone can easily access the information and track what's happening with defects, tests, results, notes, etc., as it evolves. The point is that without an efficient way of distributing this information, everything else is essentially useless, except to you.

OK, so there is a whole lot of stuff I haven't covered. What I've talked about in this article is just the beginning of testing your application. You'll run into all kinds of other issues as you get into it. I'd like to steer you toward some reference materials. A couple of books I'd recommend are:

Handbook of Software Quality Assurance, by G. Gordon Schulmeyer & James I. McManus (Prentice Hall, ISBN: 0-13-010470-1). This book is geared toward overall quality of software as opposed to the specifics of testing software. It'll give you a lot of good ideas about how to build quality into your application, as well as how to quantify quality.

Testing Computer Software, by Cem Kaner, Jack Falk & Hung Quoc Nguyen (International Thomson Computer Press, ISBN: 1-85032-847-1). This book specifically explains how to test your software, set up a defect tracking system, find bugs, etc. This is a must-have for anyone interested in testing computer software.

You'll also find a ton of information pertaining to software quality at these sites:

- Cem Kaner has lots of stuff at www.kaner.com
- American Society of Quality (ASQ) at www.asq.org
- Quality Assurance Institute (QAI) at www.qaiusa.com
- Software Quality Engineering (SQE) at www.sqe.com
- The Software Testing Institute (STI) at www.ondaweb.com/sti

Scott will be delving deeper into software testing at TechCon99 in September. He welcomes feedback and questions via e-mail at sgamron@basis.com.