Are You Ready for the Test Automation Game?

How to make automation choices in the best interest of your business

by Kerry Zallar

Which way should we go? Christopher Columbus had a clear answer to that question when his project team set sail from Spain in 1492. His intention was clear: sail to the West to find an easier way to get to the East. Along the way, however, he ran into some unexpected land masses that weren't on his map. His intentions may have been clear-cut, but his expectations were slightly skewed from reality.

And so it also goes for some of us who intend to launch a test automation process.

Test automation can be an effective route to success for some, but others never seem to reach their intended destination. Like Columbus, their expectations can be quite different from the experiences they encounter on their journey, and teams can become stalled. Those who have been around the test automation block a few times will tell you that's not unusual; many of them would say they've had to modify their original perspective as they've learned their craft.

QUICK LOOK

- Choosing a large- or small-scale approach
- Things to consider before automating
- Maximizing your investment payoff

As you chart your course, realize that even people in the same organization may have widely different visions of how test automation should proceed. You may see test automation as a true software development effort, not unlike other software development endeavors. Some managers, on the other hand, may still have visions of capture/replay scenarios in their heads when they think of automation. Bridging these different perceptions of test automation, its capabilities, and its benefits, will improve the chances of

your software development team meeting its testing objectives.

What Will Automation Mean for You?

Before your organization starts to navigate through its automation choices, it's a good idea to examine just what's involved in the process. Here we'll look at five important factors that managers need to consider as they gauge whether their team is ready for test automation.

1. Test automation is software development.

One of the most important concepts for everyone to agree on is that test automation *is* software development. Test automation is not just throwing a tool into an existing process to make it go faster.

Unfortunately, I've seen that expectation acted out all too often with testers. Testers are extremely busy performing test management, test design, and test execution—and aren't able to keep up using manual testing techniques. Then someone, perhaps even one of the testers, suggests they bring in a tool to automate their testing job. Not realizing the scope of the test automation effort and how best to proceed, the first thing that gets focused on is purchasing a test automation tool, usually referred to as a "capture/playback" tool. Some evaluations are done, and testing tools are purchased. As a result, the testers have now raised Management's expectations that testing will be done more quickly. In reality, there's even more work to do than ever before, with different job requirements and perhaps mismatched skill sets.

The "capture/playback" view of test automation is a problem, in that many managers believe that you can use automation tools to just capture tests while they're running, and then execute them at any time using the playback feature of the tool. To their credit, most of these automated tools will allow you to do this. But relying solely on this feature invites problems.

While the capture/playback scenario makes for a nice tool demonstration, experience has shown that trying to maintain dozens or hundreds of capture/playback scripts over time can be a nightmare. As the application under test undergoes changes in its functionality, trying to modify the two hundred corresponding captured test scripts is difficult and inefficient. Human nature being what it is, if this maintenance effort is too complicated or painful, it's likely it won't be done—and your investment will be lost.

If you as a manager are going to be fully supportive of your organization's efforts—and fully informed in the resource decisions you'll have to make—it's important to understand what your teams are tasked with, and how automation both simplifies and complicates that work.

The good news is that managing an automation effort isn't a completely different beast from the other projects you've overseen in your organization. The same *basic fundamental software development best practices* concepts that you're used to applying to other software development efforts apply just as well to test automation. This means that effective automation requires planning, logical and modular code designs, standardization, construction, configuration management, documentation, and yes, even testing. It also requires matching the right resources with the right skill sets in order to succeed at the job of test automation.

2. Test automation is a long-term investment.

Every dollar, lira, or yen that a company spends for testing is an investment. This investment makes financial sense as long as the costs of testing are less than the potential costs of (a) supporting defective software in production, (b) engineering maintenance releases to fix problems in production, and (c) losing business due to user or customer dissatisfaction. It's smart to view test automation as just another part of that investment.

Test automation adds two unique aspects to the test investment picture. First, like any other engineered product, it has to be built before it can be used—and that building will involve some up-front costs.

Second—and most important—you'll need to build in maintenance costs, because your automated tool needs to be useful as long as the application software being tested is supported in production. The *maintainability* of automated test systems and scripts is key to gaining the benefits of investments in test automation.

For those of you interested in crunching numbers to estimate the value of the return on investment (ROI) with test automation, I'd recommend reading a well-written article by Douglas Hoffman on "Cost Benefit Analysis of Test Automation" (available online at www.StickyMinds.com).

3. Assess your resources: people and skills.

A truly effective automation process requires a visionary—someone who can take responsibility for steering the process toward long-term success and return on investment. This may be the current test manager, or it may be someone brought in specifically to manage (and perhaps even build) the test automation system. This person will be the architect of the full test automation effort. They'll be responsible for documenting and communicating strategic plans for implementing test automation in the short term, as well as how it will be maintained in the long run. They will also be responsible for ensuring that test automation is planned, designed, and managed well so that the investments made will pay off over time.

Once you have your point person in place, you'll need to identify other staff with the right test automation development skill sets to pull off this job.

For example, most test automation tools incorporate the use of a programming language of some sort. While most tools have been designed to make writing functions simpler by incorporating easier-to-use interfaces, the end result still ends up looking like what it really is: program code. Your test automators will be those individuals who know how to create and maintain a large number of reusable modules and test scripts. They need to know basic programming language techniques as well as structured programming concepts.

You'll want to look at your testing staff carefully. Do they have these skills? Obviously, this is a significant change in personnel requirements from manual testing. Even if existing testing resources have these skill sets, it's unlikely they'll have time to

work on test automation and also continue to perform manual testing. In fact, I would recommend that someone other than those doing manual testing be identified as the test automators. Otherwise, you introduce the risk of prioritization mistakes. Picture it: as manual testing takes precedence due to project schedules, test automation will end up being shunted to the side. When that happens, test automation won't be kept up, the tool will become "shelfware," and all investment to that point will be lost.

So at some point you will need to make a resource commitment. Don't think of test automation as a "project" that has an end. Instead, compare it to a "product" that will be built and maintained. It should live as long as the application under test needs testing.

4. There's no one-size-fits-all approach.

One of the most important things to remember is that there's no "one-size-fits-all" method for applying test automation. An automation effort depends on the criticality of the software under test, the level of investment you're willing to make, the maturity of the software development and testing processes, and the timeframes in which you expect to see a return on investment.

Starting Large The decision to automate can be a big, bold move that affects the entire organization, as I observed in a company I visited earlier this year. This is a large business with millions of customers, whose Web site provides a presence to their products and services. A non-trivial problem with this Web site could have a large negative impact on its customer base. The risk associated with any problem with this Web site is high, so their management was willing to invest highly in testing it—including investing in test automation. Test automation was important as well because the Web site would be modified frequently to keep up with competition and customer demands. Manual testing efforts would not be able to keep up with the pace of development. This company recognized they did not have the expertise in house to build

PERSPECTIVE

Putting the Pieces in Place

Melissa Mutkoski has worked where tools were tossed as soon as the product changed, where automation was a targeted effort, and where companies were caught off guard due to preconceived notions about how to automate. Put simply, Melissa Mutkoski has seen it all.

Today, Mutkoski oversees automation at Formation Systems, Inc., a company that essentially markets automation. At Formation, she says, the goal is basically to "Automate everything!" While this is a noble pursuit, the challenge lies in the fact that—marketing aside—automation never actually saves time or money. Though it allows you to perform many tests in a short time, thereby significantly enhancing productivity, it also requires you to spend time creating, maintaining, and interpreting the automated test suite. And so it goes that automation is truly an investment in *quality.*

"For those that are just getting started this can be problematic," Mutkoski explains. "As a software manager, you're aware that the financial powers-that-be often view automation as a one-time expense—the purchase and implementation of the tool. When they see you coming back for maintenance money, it can become a tangle."

Other advice she gives to managers approaching automation is to hire a lead architect before making decisions about what tool to buy and what kinds of resources to devote. It's critical that managers select someone who has had lots of experience and can focus in on the issues related to automation. But finding the person with just the right mix of experience for this job won't be easy. Mutkoski says the profile for this perfect person might look like this:

- **Programming experience.** This is more important than specialized experience in any one tool.
- **QA background.** Ideally, but not necessarily, in test automation.
- **Destructive tendencies.** The ideal person has to have a history of liking to break things.

"It's hard to find people with this hybrid skill set," says Mutkoski. "But what you need is someone who can write code for hours, then know how to test the heck out of something to find the bugs."

The next step is building the right team. "We created a separate team for automation," Mutkoski states. "Then we developed an infrastructure, coding standards, and things they could re-use." She smiles. "We built a serious automation foundation." Mutkoski also claims that any time you have a separate automation team with serious object-oriented programming skills, you're ahead of the game.

In the end, the key to successful automation is clear—lots of planning ahead. The important thing to remember is that in the case of automation, test automation *is* a software development function. With this in mind, Mutkoski advises, it's up to you to take steps to ensure that your automation projects get the right architect, the right tool, and the right team. *—A.S.*

and manage such an automated testing effort, so they brought in a company to architect the test automation effort and maintain it for them on the company's premises. They proceeded to automate on a large scale in those areas where it made sense to automate. This approach was successful—and continues to be—because the outsourcing company maintains existing test scripts and creates new ones. Some manual testing that would take two to three weeks was reduced to between one and two days by automating these tests. While the test automation development is outsourced, employees of the company run the automated test scripts, perform other manual testing, and report test results.

One key to their success was that Management was willing to listen to the outsourcing company as they described their framework for test automation. By communicating expectations up front, they were able to agree on an appropriate investment level that led to success.

Starting Small Not all companies or departments have the resources to outsource their test automation efforts. Nor do most applications have this kind of business risk. For those starting off with test automation, or for those trying to become more successful with it, it may make more sense to start off small, show successes, then grow test automation in those areas where it makes sense to do so. Begin with tests in your test design that are easier to automate and provide quick results. By starting small, you're able to learn how to use the tool, design your test automation better, and show some immediate payback. Some automated tests will take longer to design and build and won't provide immediate payback. These will pay back more slowly as they're executed over and over.

Note that you'll always have a mix of manual and automated testing—regardless of the size of your approach—and you can guide the proportions within this mix to meet your organization's needs. Automation can be a minor portion of your testing efforts until you learn how to best design it for your application under test, acquire appropriate resources, and prove you are able to show that test automation will continue to benefit you in the near and long term. The return on the initial investment in tools and resources, obviously, would take much longer using this approach, but for some organizations it would be a logical path to follow.

5. You need to gauge your maturity levels.

Do your developers and testers throw spit wads at each other? Are you embarrassed by your engineering team's daily antics? Well, that's a personnel maturity problem we're not going to address here. What I would like to talk about instead is the importance of *process* maturity and its relative influence on the success of any engineering effort, including software test automation.

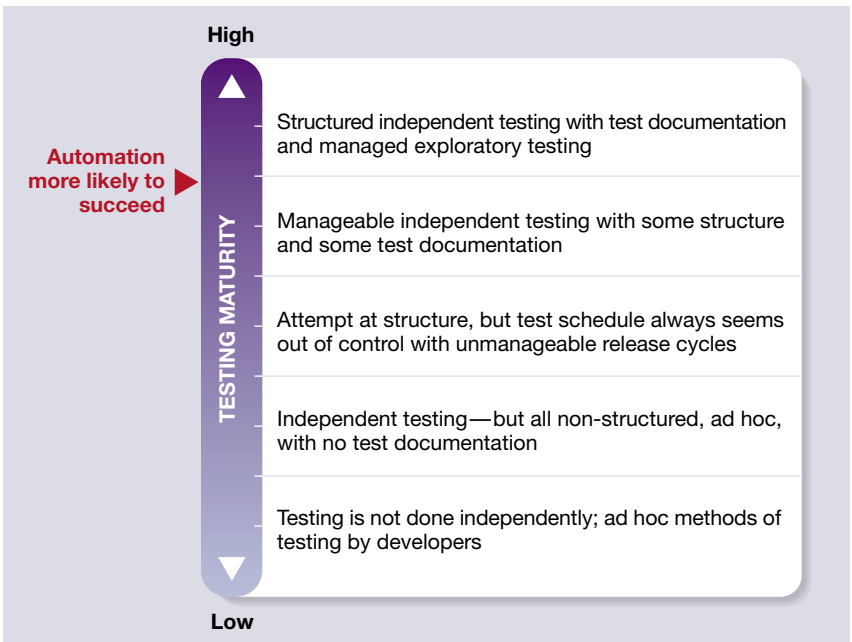
The Standish Group estimates that on average, 31.1% of projects will be canceled before completion, and only 16.2% of software projects are completed on time and on budget. (See this article's StickyNotes for a link to the complete study.)

Given that test automation *is* software development, what makes you think that your test automation effort will be any more successful?

Before making automation decisions, it's worth taking a sober look at your organization's current procedures. If your application development processes don't incorporate basic fundamental best practices, then test automation is probably not the right solution for providing your users better software. First, if your application development processes don't follow best practices, what makes you think you're going to change the culture of your development organization and begin applying these practices to your test automation system? Second, it's probably more prudent to invest that money in getting educated in development best practices and begin to incorporate them. It's generally cheaper to prevent defects from being introduced than it is to test defects out of a product.

In contrast, if your development environment already incorporates fundamental best practices throughout the development lifecycle, you can increase your chances of succeeding with test automation by leveraging the culture and practices already established within the engineering organization.

Testing Maturity The same maturity requirement applies to the maturity of your *testing* processes, as shown in Figure 1. Do you have independent testing? If you do, is it mostly



ANNIE BISSETT

FIGURE 1 Testing maturity scale

being done in an ad hoc manner or with some structure? Are your tests documented? Automating testing is a challenge if you don't know what tests you're automating, or if the manual testing effort is unpredictable and inconsistent. Achieving the discipline required for maintaining an automated test system could be a significant change for your testing staff. If this level of discipline is new, it's likely that necessary processes won't be followed and eventually the test automation effort will fail. The key point here is that the maturity of your existing manual testing processes will matter when considering test automation.

Release Management Sometimes testers try their best to incorporate common best practices, but their efforts are undermined when software releases are managed poorly. When it's unclear what's to be changed in the release of software being tested, when the requirements are changing too frequently, or when release schedules are determined without input from the testers, testing is like trying to hit a moving and changing target. In that environment, successful test automation would be even more difficult—or even impossible. Good release management practices include disciplined prioritization and communication on the part of everyone involved with the product, including those doing test automation.

Are You Ready?

As a manager with a grasp of The Big Picture, you need to carefully evaluate your organization to determine whether your project or organization has the discipline to apply another software development effort in the form of test automation. In your evaluation, keep a few important points in mind. First, test automation is more than just buying a tool; it's software development that requires an effective development and maintenance process. Second, view it as a long-term investment of money, time, people, and skills—with script maintenance being a key factor in a successful return on your investment.

I've recommended to managers of several groups that they *just weren't ready yet* for test automation—and although it's difficult to measure, I believe they saved a lot of money and time by not diving into test automation prematurely. Before investing thousands of dollars in test automation (or continuing to struggle with your own automation setup, if that's the case), take the time to read other articles and books on the subject (see this article's StickyNotes for further reading).

No matter what your final decision is, there can be considerable value in going through the self-examination steps we've described here. Improving process maturity to the point where automation can be beneficial is a good investment—whether you decide to automate testing right now or not. STQE

Kerry Zallar, *CQA*, *CSTE* (zallar@testingstuff.com), *has been practicing QA and QC since 1987. He works for Bank of America and supports the online site* www.testingstuff.com.

STQE magazine is produced by STQE Publishing, a division of Software Quality Engineering.