Test Automation Politics 101

by Jonathan Kohl www.kohl.ca

Starting out in test automation is challenging. We usually talk about automation design, using test frameworks, and the details of automating tests themselves. We rarely talk about the people we work with, and how their ideas about test automation can be even more challenging. As a rookie test automator, I was surprised by resistance to our test automation efforts. I often wished there was a kind of handbook that pointed out areas of potential resistance so I would be surprised less often. In the absence of a handbook, here are some pointers based on my own experience.

Challenging Delusions of Grandeur

How we think about the tools we use and choose colors our impression of software projects. If we project our imaginations onto the capabilities of test automation tools, they often don't live up. Early in my career, I found myself in a meeting with a development manager who had decided to purchase a tool costing tens of thousands of dollars. We on the test team had little input into the purchase decision, but we were expected to use the tool to help create "defect free software", "speed up release cycles" and get our software to market at just the right time. We found out that the combination of a smooth-talking salesman and the imagination of the development manager had helped her form an odd perception of what the tool could do for us. She thought the expensive test automation tool would run on its own, create its own tests, and report test results, like some sort of intelligent robot or automaton. When we showed he what the tool could actually do, and that even its record/playback engine didn't work with our software, she got angry with us.

Initially, we were blamed for using the tool incorrectly, but a follow-up audit by programmers verified our claims. So, we were ordered to make the tool work anyway: "We don't believe in the impossible in *this* shop!" and furthermore, "the tool was expensive!" Eventually, she had us quietly replace it with something more suitable.

When decision makers pin their hopes on a particular tool as a simple solution to difficult problems, sales people, and sometimes overly-zealous technical people are more than happy to agree. No matter that the technology is old, the design is poor, the development team lacks skill, or that schedules are far too aggressive, test automation will save us! When you are the bearer of bad (but realistic) news – that the tool is there to help, not to rescue –don't be surprised if the decision maker feels disappointed. After all, they probably had to sell this to a purchase manager. They probably feel vulnerable and afraid of looking foolish. The best way to deal with this is to be kind, empathetic, but firm. Always reach for evidence to prove or disprove ideals, and avoid trying to appeal to their emotions.

Acquiring Tools

Determining a useful tool that meets your needs can be difficult and fraught with resistance. There are many tool floggers out there, both proprietary—we buy from a vendor—and free, open source—we download from a website for free—types. Neither group is immune to making wild claims and appealing to the emotions of decision makers. If you talk to a team member who favors one type over another, be prepared for resistance if you are looking at something contrary to their preference.

Tools rarely come out of the box meeting all of our unique needs. There is so much variation in software development tools, technologies and implementations that it is next to impossible to create a tool that is suitable for everyone in every situation. One of my colleagues says any tool will meet about 60% of your needs. The other 40% can be addressed through custom development and using other tools. It's not a bad rule of thumb, as those who try to convince us their tool is *the one* we should use don't tend to think that way.

Originally published in Automated Software Testing magazine, November 2009

Vendor or Open Source Project Pressure

Interest in open source tools has exploded over the past few years. These tools often offer effective alternatives to their expensive, proprietary counterparts. Some open source tools are a bit weak or difficult to use, and some proprietary tools can over-simplify test automation. Most tend to sit somewhere in the middle.

While we all have biases and preferences, the truth is that there are perfectly useful tools from both proprietary and open source projects. For example, I have used, contributed to and support open source tools, but I also use proprietary tools. However, if you talk to an open source zealot, they will rant against the evil corporations that make proprietary tools. Some proprietary tool vendors love to bash open source tools as being "unprofessional" or not ready for enterprise organizations. In reality, there are strengths and weaknesses with both categories.

These biases and preferences move from outside the organization to within our own teams. Some teams will refuse to even consider an open source tool. I've heard to comment, "who will we sue?" if an open source tool is selected by the team. This seems like an absurd way to determine a purchase, but it's not uncommon. Other teams refuse to consider proprietary tools, even though many are more usable and have better error handling and technical support than their open source counterparts. My approach has become much more pragmatic: if it works, and we can afford it, use it.

Don't underestimate the power of personal relationships with vendors ("if you help me sell this, I'll help you") or open source projects ("contribute to our project, and get your company to pay for it!") This type of relationship can be a source of resistance towards decisions that potentially threaten a stake-holder's relationship with someone else.

On one project, we were puzzled that a purchasing manager kept rejecting our tool decision, insisting we buy a different tool. Then, after that tool was purchased and when it didn't work in our environment, we found out that the purchase manager was friends with the tool salesman.

On another project, we felt that an open source tool was being forced on us. When we dug deeper, we realized a senior programmer was friends with one of the tool founders and wanted to work on the open source project during working hours work time.

Another consideration is that while purchasing managers talk about saving money where we can, they are sometimes under pressure to spend all the money in their budgets before the end of the fiscal year. While they may say they want the selected tool to be cost-effective, they may also see an expensive tool purchase as a source of pride, and steer the purchase towards that.

One team I worked with was shocked to find their cost-effective choice denied in favor of purchasing the one of the most expensive tools on the market. They ended up having to use a free, open source too to fill in the gaps in what that expensive tool could do. While decision makers were pleased with the results, they upset with the use of cheap tools and wanted the team to "make this (expensive) tool work as well as their (free) tool."

Record/Playback vs. Development Library

Another controversial schism in the test automation world is the argument over using so-called "record/playback" testing tools versus using building a custom test harness around a test library. A lot of record/playback systems were sold as a way for non-technical people to take charge of test automation. While they worked well for some projects, on others they failed miserably.

For example, one senior manager I know thought it would take little effort to create a large, effective test automation framework. The test automation specialist subsequently discovered that they would have to create all the test scripts by hand. Difficulties arose. Evidently, decision makers were taken advantage of by aggressive sales tactics that promised the world for little effort. Technical people were stuck with some weird tool that didn't do what anyone expected without a lot of frustrating work.

As a result, many test automation specialists have grown to loathe record/playback and now discourage using it at all. The truth is, there are some projects where record/playback can be used as an effective automation

strategy. It can work in standard, simpler applications that don't change a lot. I've personally looked at it for lightweight regression automation on projects where the user interface doesn't change much. We used it to make sure that basic workflows through the application work from build-to-build, and we didn't mind throwing the scripts away and re-recording when the application changes later on down the road.

If you suggest record/playback may not work with your application to a purchasing manager, or suggest that you might want to use record/playback to an automation veteran, expect resistance.

Implementation Challenges

Once you get past the acquisition phase, implementing a test automation strategy and design can be difficult. It seems that everyone has an opinion on how test automation should be done, whether they are influenced by claims they have read, or by inventions of their own imaginations. Sometimes you wonder where the ideas come from. Here are some common sources.

Process Idealism

Back in the 1990s, process salesman and tool vendors often told test automation specialists: "You should have 100% test automation." The thinking was that if all tests were executed by a tool, it would be faster, cheaper, more reliable, and more effective than having human testers do it. Many of us tried this out and learned that there are some types of tests that are better suited to tools and others that are impossible to be run by anything but an intelligent, skilled human. One shop I visited bought several tools in the attempt to reach the goal of 100% test automation. These tools are now living out their useful days as door stops for the test lab. By the early 2000s, test teams were looking at automation more strategically, and trying to harmonize manual and automated testing efforts. This is good – utilize the skills of humans, and use tools to help them do a better job.

Enter the Agile movement, chiefly the Extreme Programming community. Once again, the "Automate 100% of tests" mantra appeared (much to the chagrin of testers who had lived through it before and felt we had progressed). This over-simplified idea that all testing could and should be automated has gained currency once more. Once again, we're seeing what we used to see ten years ago. Since automated test tools can't observe, think, evaluate or change direction, important bugs get missed if human tester eyes aren't also involved.

One Scrum/XP team spent an enormous effort to automate all the tests in their test tracking tool, only to have obvious and costly bugs creep into a production system. Instead of then blending manual and automated testing, the test automators were ordered to make their automated tests more effective. Over time, the amount of test automation code developed to support this effort far eclipsed the lines of code in the software that was being delivered to customers. The test automation software became buggy, brittle and suffered from architectural problems just like, well, any other software development effort does. Unfortunately, the small test team responsible for the care, feeding and maintenance of this massive automation stack couldn't keep up. Since it didn't bring in revenue, it had fewer resources available to it.

Instead of taking a pragmatic view and evaluating the effectiveness of the tool in how it helped the team be more effective, this team decided to put the ideal above the actual results they were realizing. Things got worse when they decided to integrate functional tests into their continuous build system. The team's wakeup call finally came when their test environment and test automation met their continuous integration and "100% automation" ideals, but were so different from production, that they missed finding catastrophic errors that minor manual testing revealed after the fact.

The power of ideals of how things "should be" are powerful and cause resistance when challenged, particularly when beliefs about processes are deeply held and widespread.

Punished for Thinking outside the Box

Test automation folklore tends to focus on trying to automate what human testers do (called regression test automation). Yet, if you look at mission-critical software and how it is tested, you'll find they make a use of simulators and emulators. Since the software *has* to

work, their test automation tends to focus on simulating different conditions the software might encounter, checking that it can handle different conditions properly.

When one of my colleagues advocated using test automation resources to create much-needed simulators, he encountered a great deal of resistance. The testers expected a record/playback regression test framework or tool. Meanwhile, the programmers were dead set on the extensive use of unit testing with mock objects, and some sort of simple table-based user acceptance testing, such as a tool like FIT.

It took a lot of convincing, patience, diplomacy—and a thick skin—but my colleague convinced the team to try out a simulator. They did, and were amazed at how effective it was in helping them create different conditions to test the software against. The blend of running a simulator and manual testing by testers and subject matter experts found all kinds of problems early on in development. This led to creating a better design as the team learned the sources of bugs.

It can be difficult to recommend using automation tools in ways other than mainstream regression testing tools. However, regression testing is only one possible area for us to consider using test automation. We can save time by automating tasks such as deploying builds, monitoring log files or error conditions, or automating the setup for manual tests. However, if you do buck the mainstream, you will make stakeholders on your team nervous. Be prepared for resistance, and make sure you gather data to back up your claims.

This Tool Cost a Lot so Make It Work!

Sometimes, people above us may have staked their reputation or jobs on a decision to use a particular tool. As a result, we're stuck using it. This can cause strange challenges, particularly because most tools are tailored or suited to particular development lifecycles or processes.

For example, if you're working on an Extreme Programming and your record/playback tool states in its user manual that the requirements and the user interface should be frozen prior to running the tool, you will probably have a lot of explaining to do. On an XP team, you probably won't have a frozen user interface or finalized requirements until near the time you ship the software.

Conversely, if you are trying to use an Agile testing tool such as FIT (framework for integrated tests) on a team that doesn't have an architecture to support the tool, you're going to have to have some frank discussions with those people who expect the results like the ones they heard from an Agile team at the last conference they went to.

Reporting Results

Once our tests are running, we spend time reviewing the results and deciding what to do with the information. If the results from your automated tests differ from what other team members or decision makers are expecting, brace yourself for strange behavior. You may even feel like you yourself are being resisted, or not listened to.

Dealing with Resistance

- Don't take it personally
- Use evidence, not emotions
- Don't make people feel stupid if they have overly simplistic or unrealistic ideas about test automation
- Don't blame decision makers for making a poor decision
- Use evidence to choose the right tool or approach – set your emotions and biases aside
- Don't choose a tool or approach until you have evidence to support it is the most suitable
- Don't expect automation to solve all your problems
- Don't play politics and choose a side – you will live with the results of the decision, and political environments change quickly
- Set goals for automation, and demonstrate how the tool and approach is helping the team meet those goals. (Note: test automation itself is not a goal, it is a means to help reach goals)
- Don't put process on a pedestal, strive for meaningful results
- If you use tools unconventionally to create automation value, be patient and demonstrate how they help the team reach goals
- Be patient, resistance fades away in the face of evidence and logic

Originally published in Automated Software Testing magazine, November 2009

Beware of Misplaced Faith

Sometimes, way down deep you know something is wrong with your project, but you don't want to face it. When that's the case, people don't deal well with evidence that tells them something is wrong. We put faith in processes, tools and methodologies with as much thoughtless abandon as any religious fanatic. When our faith is challenged, we resist, at least at first.

Ignoring the Real Results

Our test results are ignored if they differ from prevailing impressions about the project (testing reveals problems when stakeholders expected perfection.) This seems to be particularly common on performance or security testing efforts.

Often, a manager has made claims about the effectiveness of the software to customers or investors. The alternative—that the product doesn't work—is so objectionable to think about, it's preferable to hold onto the delusion before dealing with the real problems.

Blaming the Messenger

The test automator becomes the target of hostility when they are the bearer of bad news: "Sorry that you made these wild claims without evidence, but our tests show that the software can't handle load." At least you aren't ignored, but it is still hard to take, particularly if you are the focus of an emotional outburst. Have courage, speak truthfully and don't take it personally. They will stop resisting once they get used to the idea.

Conclusion

Resistance in test automation is common, but it isn't as bad as it seems in the moment. In the short term, these interpersonal difficulties can be confusing and hard to take, but in the long-term the results of your test automation efforts can far eclipse the pain of the awkward exchanges.

With these experiences now behind me, I often look for the absurd humor in the difficult situations I've been through in the past. To be frank, I couldn't possibly do the testing I do now without test automation tools, and thankfully, the odd emotional or political landmine is now more mere annoyance, like ants at a picnic or mosquitoes at a party. If you are faced with politics on your automation project, stick to your principles, your skills and logic. If you appeal to evidence over emotion and ideals, you'll come out ahead.

Jonathan Kohl

Jonathan Kohl is a testing consultant with Kohl Concepts Inc, based in Calgary, Alberta, Canada. He has been involved with test automation projects for close to a decade. In addition to working with teams helping them solve technical and business problems, Jonathan writes about and speaks on software-related topics. Read more of his work at http://www.kohl.ca/Contact Jonathan @ jonathan@kohl.ca

Originally published in Automated Software Testing magazine, November 2009