Software Testing Productivity Tools–It's not about the Tool

By **Philip Lew**, CEO, XBOSoft

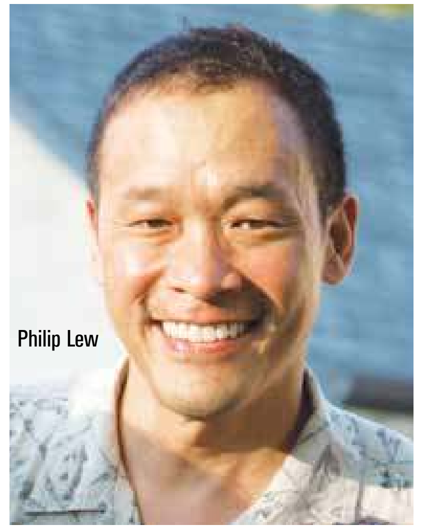
With falling labor productivity growth, as reported in the latest economic data by the federal government, who's to be blamed? What is happening to our productivity? As a whole, the fact is that we are no longer seeing the huge increases in productivity through innovation that we have seen in the past. And the same applies to the domain of software testing.

First, let's examine what it means to be more productive. Productivity means doing the same thing but doing it better and/or faster. In simple terms, achieving a greater or the same amount of output by giving the same or reduced amount of input, respectively. For software testing, how do we test more with the same amount of time? Given this, there are two main areas to increase productivity when it comes to software testing: test management and test automation.

For software testing, there is a plethora of productivity tools, all designed to help us be faster and better. The two main categories for pure speed-related productivity gains are in the areas of test management and test automation. In regard to test management, many fail to realize that these tools come with so many bells and whistles, some worthwhile and some not, that you lose focus on what's most important to your organization and how to integrate them into your processes. Many view software test automation as the ultimate tool or magic bullet for increasing software testing productivity. It makes complete sense–let machines do the work. Machines don't get tired, can work all night, and don't make mistakes. However, many have problems in successfully implementing software test automation.

Going through all the tools in each area would be exhausting and I might offend someone who didn't get included. So, in this article, we won't list out or endorse any particular tools but provide general guidelines in selecting and using them better. Here we've listed some of the most common issues that block implementation success.

Misguided Expectations: When tool vendors come into your office and pitch their products, they talk about all the bells and whistles. But, just as you would run an Agile project, think about the most valuable functions you'll actually be using. What is the minimum viable



Philip Lew

product for you? Make sure that it does those things well, and start from there. Don't construct a grand plan to complete implementation in 30 days. Do it in phases starting with the MVP and build with each phase. When you consider purchasing and implementing the tool, make sure you'll be happy with just implementing the MVP because that might be all you get.

Lack of Training: When you are replacing an existing work process that's been done for years with a new adaptation of it, complemented by a tool, you need to make sure that the tool is capable. Most tools are, but they need some configuration. Implementing a tool right out of the box may not fit your process and then the team will say it doesn't do what it's supposed to do. Most vendors will give training on the tool, so make sure everyone that will be using it gets training, and assign a system administrator or domain expert for the tool in your organization.

Tool-First Approach: If you get caught up in the tool-first approach, you might pick the "best" tool, but for who? The tool should fit into your existing processes, or if you decide you want to optimize your existing process, but don't design your process around the tool. It's best to examine your process, noting problems or bottlenecks, and then find a tool that supports it after you optimize your process. Sometimes changing the process and implementing the tool can be done at the same time. You can buy the tool, install it, and possibly get some productivity gains from out-of-the box configuration, but do keep in mind that you'll more than likely need customized configuration to fit your process.

Piecemeal Implementation: Most of the time, as with any process, there are many parts. If you implement a tool that helps productivity in one or several parts of the process, it will have to be integrated with the rest of the process. Let's say you implement a test automation tool.

•How do the tests that are automated be managed with the tests that are done manually?

•How will you ensure that the combination of manual testing and automation covers what you want?

•How will the reports from the manual testing be complemented by the reports from automated testing?

> The two main categories for pure speed-related productivity gains are in the areas of test management and test automation

•When a test becomes automated, how will you know that it should not be done manually? Are these done by the same person? Probably not.

These days, many test management tools tout that they help you tie your requirements methodically into your tests, and since it's been shown that many software defects originate in requirements, it makes sense to do this. However, how will your business analysts work within your process and tool implementation? Maybe they have their own tool for use cases. Integration with other tools and working within an optimized process is critical.

Stop and Go: When you decide to invest in a tool, you have to have follow-through. Most of the time, your team will already have an established way to get the job done. I've seen teams use spreadsheets for test cases and purchase a test management tool, but because the spreadsheets are so ingrained in their work processes, they just keep doing it that way. The tool then sits there with licenses paid for, either idle or underutilized. Then, management will scream, "Why aren't you using the tool we just invested in?" You might answer that it's too much trouble and that you're faster with the spreadsheets. To implement a tool, especially with an established process and team, there needs to be total buy-in from the team that they are making the decision to get the tool and use it to help them. Other situations may arise where you implement just the basics but don't really get the advanced features and functions of the tool. In this case, you might make some productivity gains, but they are nowhere near what you could if you invested the time into training and customizing the configuration specifically for your process.

This list of pitfalls to avoid is by no means exhaustive, but hopefully alerts you that implementing any productivity tool (not just in software testing) needs careful thought regarding integration into your organization and its processes. As you can see, it's not the tools themselves that can lead to the greatest productivity gains, but how you use the tools, how you introduce them into your organization, and how you follow through to optimize their use. CR