Shorter Interval Projects Are Trending, But What Does That Mean For Testers?

By Michael Murphy, Test Consultant
June 17, 2014

I find it interesting that software development projects are generally being planned and executed with shorter time frames than before. Software development projects that were once taken on as a single large project are now being broken down into smaller sub-projects. Project releases that might have taken a year or longer are now being rolled out more frequently in smaller increments.

> "planned and executed with shorter time frames than before"

So what is driving this trend and what does it mean for software testers?

Thinking in Incremental Improvements

How do you approach a task like searching online? What is your strategy? If you decided to search using the same strategy as some traditional software projects, like Waterfall, you might spend some time in advance planning your search terms. You might consider what you know about your search topic and how Internet search works. Once you have completed the planning phase, you might proceed to execute your search. Confident that careful upfront planning means your search term was ideal, the final step is to go through the results pages until you find what you are looking for.

> "This type of incremental improvement comes naturally to us"

No one searches like that. Typically, only a minimal amount of consideration is put into a search term, and rarely would we look at the results beyond the first or second page. Instead of looking through pages of results, we would typically prefer to refine and improve the search terms and then search again. This type of incremental improvement comes naturally to us and people seem to think it works better. What if we applied this concept to software development projects and settled on only the basic requirements and then refined them later rather than plan out every detail we can think of? This is Agile.

Thinking in Short Intervals

Attention spans may be shorter than they were before. Have shows like Sesame Street trained us to think in smaller chunks rather than having to absorb a huge concept? How do you figure out

the best way to do something new? Is planning everything out in advance important, or is it sufficient to only understand the overall big picture? Try the Lemon Toss[1] game and take notice of what your natural approach is.

projects:

- Game rules vs. software requirements
- Scoring points vs. acceptance criteria
- Game option A or B vs. Waterfall or Agile

Lemon Toss

* The way to play the game is to have everyone toss lemons at each other
* Best played with 8+ people and 12+ lemons
* Have a whiteboard handy if you feel like writing anything down

The rules

- If you drop a lemon, you can't pick it up until the end of the round
- You must decide how many lemons to start with in advance
- You may not toss the lemon to the person directly beside you
- No hand-to-hand passing; you must toss the lemons
- You have 10 minutes of total planning time and 10 minutes of total playing time

Scoring Points

- Every time a lemon has been tossed by every person = 1pt
- At the end of the round, if you are holding the lemon you started with = 1pt

Game play options

A. Plan for 10 minutes, then play for 10 minutes
B. Plan for 2 minutes, then play for 2 minutes. Repeat 5 times.

Do you think you and your team would score higher with option A or B? Would you be surprised that more people would feel more comfortable with option B? Breaking the game up into smaller chunks and having the ability to revise your strategy throughout is becoming more typical of how a software development project is typically set up. There are many more parallels between this game and software

More people find it natural to think in short intervals and to improve incrementally. It's easier to take on a large project by breaking it into smaller manageable chunks and, indeed, this appears to be a trend in how software projects are being structured. The overall big picture of the project is important, but working out the details for every phase before starting the work is not as natural. That would be like

planning your search terms in advance, or playing the lemon game with option A. It goes against how we naturally would think.

> "Thinking in shorter intervals with incremental improvements"

How we think naturally goes hand-in-hand with shorter projects and Agile methodologies. It's all the same thing: Thinking in shorter intervals with incremental improvements. So, as the trend continues and large projects are broken into smaller sub-projects, what are some things to consider?

A 9 month project today was a 12 month project before. Why? Because it took 12 months last time and, because of Agile, we think it is going to go faster this time. It seems that Agile is set up to help us move faster because we can negotiate the scope and schedule with stakeholders sooner. However, though we choose Agile for speed that is not what we usually end up getting. What we do get is a clearer picture of why it isn't going faster and a better understanding of what needs to be prioritized or cut.[2]

So What Does This Mean for Software Testers?

As testers adapt to shorter projects, does anything change from our perspective? Testers, just like programmers, business analysts, and project managers, might need to adapt to new technologies and become more comfortable with new concepts to meet the demands of shorter schedules. Communication on shorter projects might have different needs as well, but the basics of designing good tests remain as important as ever. One approach to testing that becomes especially useful in Agile is to test around the changes, whether they are changes in code, features, or customer expectations. Testing around the changes can help make the process manageable as it goes faster. Using test automation can be a key part to managing those changes over the multiple sprints in a project and at an earlier stage in the software development lifecycle.

> "the basics of designing good tests remain as important as ever"

However, are we in danger of losing something important as we move toward shorter projects? Try playing the lemon toss game again, but play with option A. You might find that with the amount of planning time given that you find it useful to write out your strategy for everyone to see, especially if the group you are playing with is larger. Compare this to option B where most of the communication is verbal and the urge to write down changes to your strategy between rounds is low. Overall, the "Agile option", option B, may be more natural to you and you might end up scoring more points, but what will happen once the game is over? Will the next team be able to pick it up in six months and benefit from all of your team's lemon tossing insights?

That is food (or lemon) for thought.

[1] Adapted from Boris Gloger, The Scrum Ball-Point Game http://borisgloger.com/2008/03/15/the-scrum-ball-point-game/
[2] Lanette Creamer at 2014 Quality in Agile Conference - Small Agile Projects (Delivering Quality in 3 weeks to 9 months) - http://agilevancouver.ca/index.php/events-in-2014/2014-quality-in-agile/sessions

About The Author

Michael Murphy is a Software Tester in Vancouver, BC. As an IT professional with over 13 years of experience, Michael is an IT jack-of-all-trades who has worked in several challenging, fast-paced and time-sensitive environments. A strong supporter of open source and web technologies, Michael has spoken at conferences on agile methodologies. He has worked in many industry verticals and has helped deliver projects in the financial, e-commerce, healthcare, transportation, technology and airline industries.