BEST PRACTICE
Agile Panel Discussion

Date: 5/2/08
Expert Panel: Quality Opportunities and Challenges with Agile

Panelists:
Burke Cox, Stelligent Incorporated
Bill Rinko-Gay, Technisource
Lois Zells, Lois Zells & Associates, Inc.

Co-moderators:
Bill Rinko-Gay, Technisource
Nancy Kastl, Kaslen Group, Inc.

The following questions were answered by Burke Cox

Q: What about regression testing? Where does it fit?

In an Agile environment, regression tests are "built-in" to the process because developers and QA are writing automated tests for all new features and changes. When teams use Continuous Integration to run automated builds, these builds can run all of the unit, component, functional and any other types of automated tests with every change applied to the version control repository for the project. As more automated tests are added, they become part of a comprehensive regression test suite.

Q: When do you start to automate test cases in Agile? At iteration level or for regression testing?

Most teams out there are getting very good at automating unit testing. Some are getting good at automating their regression tests, after the fact. We have been promoting the notion of using testing (automated testing) as the driving force for the entire development process. That means that starting from the moment the customer defines the story, everyone focuses on testing the functionality. Using a tool like FIT, we teach our teams how to use tests throughout the entire software development lifecycle.

Q: Is the Agile approach black & white? In other words, can an organization be "pseudo-Agile" due to environmental, cultural or other constraints?

There are Agile advocates out there that would have you believe that such a thing exists, but we don't think so. One of our philosophies revolves around being "practical" and figuring out the right mix. "Agile" means adaptive. Adaptive planning...adaptive process...adaptive people. We have all learned that the monolithic, prescriptive methodologies of the past don't respond well to fast-paced, dynamic development projects. So, our answer to your question is that "pseudo-Agile" is a bad idea.

Q: What is the typical timeframe for an iteration in an Agile project?

It varies. You should hone your iteration length based on your team and environment. However, we like to start with 1 week iterations. 1 week keeps the team moving at a very steady pace. 2 week iterations work well, too, but the team has a tendency to not work as hard the first week and then sprint really fast the second week. Keeping the team working smoothly and consistently provides the best results. Another note here, consider running your iterations from Wednesday to Wednesday. Wednesdays seem to be the least missed day for vacations, illness, travel, conferences, etc. Starting and ending the iteration on Wednesday seems to ensure that most team members will be present.

Q: What type of estimating techniques do you find work best for an Agile project?

We usually do two types of estimates. A "high-level" estimate when a story is first created to help with the overall release planning and prioritization. Then, an iteration estimate during the iteration planning meeting. The high level estimate can take many forms, but we like High, Medium and Low. A good iteration manager will discover through good metrics tracking what "High" means for her team. The same will be true for Medium and Low. They are just *buckets* that represent relative amount of effort. We usually end up with one bucket called "Really High", which probably means the story needs to be decomposed...it's just too big.

For iteration estimates, we like to get the entire team involved and estimate in days. So that nobody influences another person's estimate, we count down to three and then hold up a number of fingers between 1 and 5 (and a bent finger means one half) days. If there are any outliers, we talk about why. So if almost everyone throws a 3, but we get someone with a 1, then we get to have a discussion about why that person thinks it is so easy. Then, we rethrow. It's a fun process. It also seems to drive the "right" type of story discussion, focusing on the important characteristics while skipping the mundane details.

Another play on this concept of story poker, where the project manager describes/reads a story, and then each team member holds up a number between 1 and 10 to represent an estimate of complexity.

We also talk about outliers, discuss concerns and re-estimate.

I tally all the numbers and for the first couple of iterations, we gauge how much complexity the team can "consume" during the 2 weeks. Once we get somewhat consistent, we start baselining our story list based on the historical 2-week consumption model.

For example, if we discover that the team can produce software for 12 story cards, with a total complexity of roughly 50, then I'll try to keep all future iterations to about 50 complexity as well.

Q: Can offshore testing work effectively within a true Agile environment?

In a true agile environment, in my opinion, the testers are there with the developers every day, helping with the stories, helping with the unit testing, helping setup infrastructure and doing a lot of testing-as-you-go through the iteration.

If the testers are out of the office, it's already going to be much more difficult to embed them in the process (much like it would be difficult to be an embedded reporter with a battalion in Iraq when you're actually in Hawaii). One of the driving ideas behind agile, IMO, is the interactions, the closeness, the learning to trust each other's strengths and intuitions. That's not going to happen easily if the testers are not in the same building. It is essentially impossible for it to happen if they're in a significantly different time zone.

Q: What is "real" life Agile meaning? Could you list what "real" Agile should have?

In real life, Agile is about producing working software that customers *actually* want. Real world agile projects give users what they want in a manner that meets their time frame expectations too. That time frame could be rapid or it could be longer, but all the while, the customer is on board and is empowered to facilitate course correcting with the expectations that such a move can have consequences with respect to delivery.

Real agile projects should have a customer, which in some cases is a challenge to define, by the way. This is key. Without a customer or a customer advocate, teams run the risk of delivering something that doesn't have or add value.

Real agile projects also have an infrastructure that facilitates automatically producing working software. Producing working software automatically means running all manner of tests (from unit to functional) and inspections. Producing working software automatically also means handing deployment and promotional issues without relying on complex manual processes.

References

Conference – Quality Engineered Software and Testing Conferences (QUEST) Chicago, 2008