Agile Software Development and Quality Management: Finding the Fit

If you’ve ever tried to get your mind around the agile software development process from a Quality Management perspective, you may have found yourself both amazed and perplexed. While complex software products and enhancements are created with great expediency, how is it that detailed requirements are not needed to begin writing software? Why is there so little documentation? With the complexity of software products on the increase, you would think that requirements gathering would become similarly complex. Add to this the question of why are we starting to have quality problems?

A Yearning for Learning

It’s all about rapid cycles of learning. Waterfall-weary writers of complicated code concluded that efforts to document detailed requirements inhibit their craft and waste valuable time. Better to take a general description of what users think they want and refine their vision through mass collaboration and rapid cycles of learning. Agile software teams organize on their own to create innovative software solutions that address wants and needs as expressed by real customers. They do it by rapidly evolving requirements through informal learning cycles. The cross functional teams converge on an optimal solution that minimizes non-value or lesser value-added activities. Just because a software user can envision an enhancement doesn’t mean that their thought is fully formed. This approach allows both customers and developers to cultivate their thoughts as they progress through the development cycle. The progressive experience often results in a change to the customer’s initial vision of the system – and change is encouraged! In the best scenarios real customers provide their opinions and feedback during the process. Agile software teams are free to follow the best path toward addressing and delighting their customers just as a soccer team collectively moves the soccer ball down the field and scores. At the end of a development cycle the teams reflect on what went well and what didn’t and take note for future reference. Teams are given the environment and support they need and are trusted to get the job done.

Too Good to be True?

The above scenario may sound ideal, especially if you have ever been in a position where circumstances did not allow you or your team to put forth best efforts. Let’s look a bit further while noting that the soccer team doesn’t *always* score as they
drive the ball downfield.

Agile teams are interested in minimizing activities that do not generate software and prefer to run calendar-based cycles for deliverables that are eerily similar to a production line schedule. The primary measure of progress is working software. Agile teams make their own commitments to produce code within pre-defined development cycles. When the schedule must be met there are three variables left to adjust – scope, resources and quality.

An agile team’s ability to accurately predict what they can accomplish is key in meeting their promised deliverables. Development of scope is often quite visible as it appears on public tracking boards in a team’s work area. “Tribal knowledge” reigns in this type of environment where most interactions are face-to-face and adding a new developer (or other resource) into the mix can actually slow the progress of the team. What is not so visible is the quality of the code generated during development. Is it alright to simply log software defects that the team cannot address during the development cycles? If so, you can revisit the issue later and address the defects. But what about programming done under time constraints? There is not sufficient time available to understand why a block of code isn’t working as it should. Short term patches or fixes are applied in an effort to meet the schedule when it’s not completely clear what requires this action in the first place. How many iterations of this behavior can go on before we get surprised in testing or even worse, production?

**More, Faster**

Understandably, the drive to innovate and stay one or more steps ahead of the industry is a powerful and necessary force. However, the reality is that more complex software and business models can generate more complex business and technical problems. Add to the equation rapid-fire development teams that can deliver software versions faster than customer support can come up with the right answers and you have a recipe for imminent disaster. If technical debt is increasing at an unknown rate, it is only a matter of time before defects surface that can shake customer confidence to the core. Is it not true that the development process is part of the larger business system used to meet the mission of the organization? How then do you configure your organization to support a high-performance agile development process while maintaining the spirit and integrity of quality management? Easy! Just remember a few basic points.
It Starts at the Top

A quality culture is driven by the attitudes of senior management. Everyone who touches source code has to make choices about the quality of their work effort. Collectively, development teams harbor attitudes that influence their everyday decisions. If senior management does not take quality seriously, neither will the staff that reports to them. The level of quality should be driven by a quality policy and operationalized through internal standards, procedures and audits. It’s clear that agile development teams are responsible for the quality of their work but its management’s responsibility to demonstrate the discipline needed to reach and exceed required levels of quality demanded by the marketplace.

Voice of the Customer

Because today’s development teams rapidly turn out functionality and other software improvements, it is important to thoroughly understand what your collective customer base values. A Voice of the Customer (VOC) system must be in place to provide direction and balance to each new development cycle. It is important to assure that enhancements and other changes are viewed as improvements so that some customers don’t lose functionality they value. Otherwise, you will be rapidly getting yourself into hot water rather than enhancing market share.

While customer focus and involving real customers in the development process is beneficial, there are other stakeholders. Beyond software functionality there are issues of project costs and length of development efforts. Also of concern is maintainability and supportability of the software product itself. It is a clear requirement that software developers efficiently address true market needs. However, the economics have to work for there to be long term viability of the business. Does the agile practice address the issues that drove its selection in the first place? What does the cost/benefit analysis look like?

All Work is a Process

Hiring smart people is a great start but they have to know what to do - particularly as companies apply agile software development principles in different ways. Process ownership and the responsibility for improvement must be established and integrated into the fabric of the organization. Even though the principle of *simplicity* from the Agile Manifesto calls for maximizing the amount of work not done, it can’t be used as an excuse for not doing the work needed to support individual and team learning and development. Process documentation and standards go
a long way in rapidly getting and keeping everyone on the same page while it supports ongoing team learning and improvement. Groups that take the time to develop concise process documentation, including process workflows and narratives, have shown us that documentation enables new team members to get up to speed faster. Process documentation also acts as a reference point when not so new team members want to brush up on an area they haven’t worked in for a while.

**Survive or Thrive?**

Implementing an agile software development process presents unique challenges, but there is good news. As agile software development teams embrace change and use it to their advantage, a positive example is set for the rest of the organization. Continual improvement calls for the ongoing matching of the voice of the process to the voice of the customer and applies to all functions of an organization. Your Finance or Human Resources Team can deploy improvements to their processes every eight weeks taking input from the firm’s employees and other stakeholders. Organizations that can continually improve themselves are capable of greater levels of innovation and long term success.

*Strait & Associates is committed to working with you to enable success within your organization.*

Contact us at info@straitassoc.com or call 888-700-5695