From Waterfall to Agile – How does a QA Team Transition?

Megan Sumrell
Misys Healthcare Systems
m.sumrell@yahoo.com

Abstract

We had been developing software with a strict waterfall mentality for several years. Our testing cycle was broken up into distinct phases for functional, system and regression testing. One phase did not start until the previous one was completed. Our testing team had always been a real asset to the organization. The entire team was a highly skilled group of automation testers using traditional, expensive off the shelf automated testing software. As we switched over to agile development using the Scrum framework, we really struggled with how to incorporate our traditional automated testing strategies into the agile world. Is testing in a waterfall project really that different then testing in an agile project? Yss!

Project Overview

Our organization had been developing software for a few years with a waterfall methodology. Our Product Managers wrote extensive requirement documents; our developers built a robust application from those documents, and our QA team diligently tested the application before it went to our customers. However, we often times would miss the mark on what our customers wanted and would spend considerable time over-developing new features. We were also spending a significant amount of time testing our application. Often times this testing would uncover an issue that could take significant amount of time to fix and re-test. We knew we needed to refine our development process so that we could get new features to market quicker and get customer feedback earlier in the process. We also needed to start testing early in the development process to avoid large re-work efforts before deployment. If we could do this, then we could deliver what our customers really wanted in a more timely fashion and not waste time developing features that they didn’t need.

Our Story

Our development organization realized that we needed to move away from lengthy 6 month waterfall practices into more agile practices. After researching several development methodologies, attending training, etc we decided that the Scrum framework seemed like a good fit for our organization. No one had any real world experience with agile development but we were eager to try anyway. We were an organization that was used to very informal training that typically consisted of reading a book in your spare time and then training others. So, two employees on our pilot scrum team attended a Scrum Master Certification class and several others read every book we could get our hands on relating to Scrum and agile practices. Throughout this initial learning phase, I left every workshop and finished every book still not really understanding how our QA team would transition into
Scrum and what their role would be on a
Scrum team. I clearly understood test
driven development, saw demos of
FitNesse, and was completely on-board
with the concept that full regression tests
could be done daily instead of at the very
end of a project. With our pilot tram
ready to start, we launched into our first
sprint planning meeting. The QA
engineers on the team added tasks to the
sprint backlog along with the developers.
On paper, we looked ready to go.
However, when we started actually using
Scrum, we were 3 sprints in and really
struggling with getting our testing
completed during the sprints. We were
doing 4 week sprints but would then take
an additional 1 week to perform
regression testing after the sprint was
done. We also were not able to get any
real test automation done during the
sprint so our regression testing cycles
were taking longer with each sprint due
to the added features to manually
regression test. We knew we needed to
start testing early in the sprint but
without completed code, we were not
able to do this. We were unable to start
test automation with the tools we were
using unless we had a stable UI. The
rest of the team members were unable to
see the test cases and help execute them
because they did not have a very
expensive license of the testing software
we were using. We had very skilled test
automation engineers on the project and
still couldn’t seem to make any real
progress with getting automation started
eyard in the sprint.
The team was getting more and more
frustrated with the lack of progress on
the agile testing. A common statement
shared by all team members in our
stand-ups and retrospectives was “well,
that just won’t work in our environment
– we are too complex.” I found it hard
to believe that we were so complex that
we couldn’t get the testing rolled into
our sprints. So, we finally engaged two
agile consultants to come on-site for a
few days of coaching.
As part of the coaching effort, we
identified a few key areas to focus on.
One of the major areas of focus was to
have a clear roadmap for how to get our
QA practices working on an agile
project. We knew we needed to make
some changes but until this point, we
really didn’t understand that the QA
team was not the only team that needed
to make changes in order for agile
testing to be successful. The Product
Owners and Developers also played a
big part in achieving agile testing. The
Product Owners needed to participate in
defining the acceptance tests prior to the
sprint planning meeting. We had been
starting each of our sprints with no
acceptance criteria and no clear roadmap
for the testers to follow during the sprint.
We were also given a demo of FitNesse
and talked through how a tool like this
fits into a Scrum team.
At the end of the first consultant visit,
we identified 5 major changes that we
needed to make to better align our team
for agile testing:

1. Traditionally, we had a lot of
   business logic embedded in the
   presentation layer of our
   application. This meant that we
   were unable to begin working on
test automation without the UI
   because the business layer was
   not accessible without the
   presentation layer. In order for
   us to start test automation before
code was completed, the
developers we going to have to
really separate the business layer
out of the presentation layer.
This way, we could start utilizing
FitNesse instead of the heavyweight automation tools we were used to using.

2. Developer unit testing would be critical for agile development to be successful. Traditionally, our QA engineers had been spending at least 40% of their testing time doing unit testing. Our development team didn’t have any formal unit testing and QA was having a hard time focusing on functional and acceptance tests because we were busy testing code at a unit level.

3. The skill set for our QA Engineers would need to shift. Our QA Engineers were all very senior automation engineers using Mercury QTP and Test Director. They were all writing test scripts to test sophisticated data validation and workflows. Now, we were going to need our QA Engineers to start writing fixtures in C# to access the business logic directly.

4. We had a viable way to get rid of our very expensive testing tools. We were spending about $25,000 a year on software maintenance and $12,000 per new QA hire for testing software. Instead, we could leverage open source technology to build out our automation scripts. Due to the significant amount of scripts we had in QTP, we would need to keep some licenses to run legacy scripts but could stop buying new licenses for new hires.

5. Our distinct roles and responsibilities for QA, PM, and Dev would need to change. Quality could no longer be solely owned by the QA team. The entire team needed to participate in the definition and execution of the acceptance tests. Now, we could begin to use anyone on the team with in-depth product knowledge and literally no automation experience to help us write test cases prior to the sprint even beginning.

CONCLUSION
Out transition to agile development was certainly not a smooth one – especially for the QA team. There are several things we could have done better to ensure an easier, less frustrating transition. There are dozens of lessons we learned that I could list out. Instead, I will list out the top 6 lessons we learned around agile testing:

1. Get help from experts early. Don’t assume that your team knows enough to determine what will and will not work for you organization. Whenever you hear the words, “that won’t work here,” don’t assume it is true. Engage a consultant or Agile Coach a few months into rolling out Scrum.

2. Don’t assume that the roles and responsibilities your team members have had will stay the same in an agile team. Be prepared to adjust and redefine your team roles.

3. Everyone owns the quality of your software, not just the QA team members. Every member on an agile team plays a significant part of ensuring the quality of software. Testing begins long before the testers start writing automated scripts.

4. Start using FitNesse (or another lightweight automated testing
tool) from the start. If you think it won’t work in your environment, figure out why. If you are willing to make some changes to how you develop your products, you should not need to use UI based automated testing tools.

5. CM is a critical piece for agile testing to run smoothly. If you don’t have an automated build and deploy system running for development and QA, focus on getting this running as soon as possible. QA needs to get working builds from development as needed. Testing needs to start day 1 of the sprint.

6. Don’t expect your transition from waterfall to agile to be 100% smooth and successful overnight. Take baby steps and celebrate the accomplishments along the way.