The Three Pillars of Agile Quality & Testing

Bob Gallen & Mary Thorn

September 2014

EuroSTAR
Software Testing Conference

Is This Book Right For Me?

INTRODUCTORY

Introductory content is for software testing professionals who are relatively new to the subject matter of the ebook. Introductory ebooks typically feature an overview of an aspect of testing or guidance on understanding the fundamentals of the subject at hand.

INTERMEDIATE

Intermediate ebooks are for software testers who are already familiar with the subject matter of the eBook but have limited hands-on experience in this area. Intermediate level usually focuses on working examples of a concept or relaying experiences of applying the approach in a particular context.

ADVANCED

Advanced ebook content is for software testers who are, or intend to become, experts on the subject matter. These ebooks look at more advanced features of an aspect of software testing and help the reader to develop a more complete understanding of the subject.

Abstract

Mary Thorn and Bob Galen are working on a new agile testing book to be published in Q4 2014 – Q1 2015. It will focus on a framework for guiding organizational and team transformation from more traditional quality and testing strategies to those more conducive to the agile methodologies.

We've found that this transition to be:

1. It's difficult and challenging under the best of conditions;
2. Often it has a development or technical-centric quality and testing is "left behind";
3. Often the quality and testing folks don't understand how to drive the strategic change;
4. And most often, there is an imbalance in the applied practices that inhibits true agile performance and delivery.

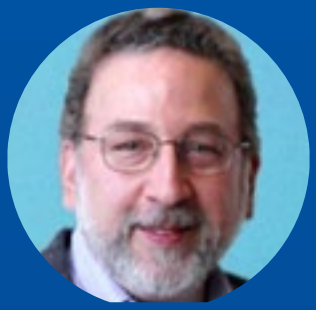
We are including the Three Pillars Introduction chapter and Three Pillars Applied chapter in this eBook as a means of introducing you to the pillars and giving you sufficient ideas to start changing your agile strategies. We think it will immediately change the way you look at quality and testing tactics and strategies when you're trying to "go Agile".

Remember that these are draft chapters. If you want to connect to our efforts in delivering the whole book, then you can join Bob's newsletter here – http://eepurl.com/IAxTD

And we'll keep you updated on the books progress towards publication. You might even get the chance to serve as an early reviewer by emailing us: bob@rgalen.com

Stay agile my friends,
Bob Galen
Mary Thorn

Biography



Bob Galen an agile methodologist, practitioner, and coach based in Cary, NC, Bob Galen helps guide companies in their adoption of Scrum and other agile methodologies and practices. Bob is a principal agile evangelist at Velocity Partners, a leading agile nearshore development partner; president of RGCG; and frequent speaker on software development, project management, software testing, and team leadership at conferences and professional groups. He is a Certified Scrum Coach, Certified Scrum Product Owner, and an active member of the Agile and Scrum Alliances. In 2013 Bob published Scrum Product Ownership-Balancing Value from the Inside Out. Reach him at bob@rgalen.com.

Mary Thorn a QA director at ChannelAdvisor in Morrisville, NC, Mary Thorn has a broad testing background that spans automation, data warehouses, and web-based systems in a wide variety of technologies and testing techniques. During her more than seventeen years of experience in healthcare, HR, financial, and SaaS-based products, Mary has held manager and contributor level positions in software development organizations. She is a strong leader in agile testing methodologies and has direct experience leading teams through agile adoption and beyond.

Table of Contents

1.1 INTRODUCTION

A few years ago I entered an organization to do some agile focused coaching and training. From the outside looking in, it was a fairly mature agile organization. They had internal coaches in place, had implemented Scrum and been also leveraging Extreme Programming technical practices for a couple of years, and seemed to be fairly committed and rigorous in their application of the methods.

It was a global financial firm that delivered their IT projects via highly distributed teams. There were a wide variety of tools in place, both for Application Lifecycle Management (ALM), software development, and software testing support. In my first few days, everything I encountered, while albeit in superficial detail, just felt like a mature agile organization and I was pleasantly surprised. Heck, I was impressed!

For the purposes of this introduction, my observations will shift to be quality, testing, and tester centric.

1.2 TOO NARROW A FOCUS

One of the things I noticed is that the firm had gone "All In" on Behavior-Driven Development (BDD) leveraging Cucumber as the tooling framework. They had invited in several consultants to teach courses to many of their Scrum teams and everyone got "test infected"[1]. Teams were literally creating thousands of BDD level automated tests in conjunction with delivering their software. From their perspective, there was incredible energy and enthusiasm. Everyone contributed tests and they measured the number of increasing Cucumber tests on a daily basis.

However, a few days into my coaching, I was invited to a backlog grooming session where a team was writing and developing their user stories. What I expected was to simply be an observer. What actually happened is that I quickly realized the team didn't know how to write a solid user story. They could barely write one at all. On their request, I ended up delivering an ad-hoc user story writing class for them. Afterwards, the team was incredibly appreciative as they started to understand the important place that solid story writing held in the agile development lifecycle.

Over the next few days, I realized something very important. The organization was at two levels when it came to their agile quality and testing practices. Either they were all in or they were unaware of or under-practicing specific techniques. For example, they were all in on BDD and writing automated Cucumber tests and on Continuous Integration. However, they struggled mightily with simply writing user stories and literally had no clear or consistent Definition of Done.

This "See Saw effect" of focusing on a hand-full of practices was doing them a disservice. Why? Because it's actually the interplay across practices that largely influences the effectiveness of your agile testing and the product impact of your quality practices. I prepared a view for them to illustrate the balance that I think it critical in your agile quality practices. I called it The 3 Pillars of Agile Quality &

1. To use a term coined by Elizabeth Hendrickson – http://testobsessed.com/

Testing, and I began to coach a much more nuanced, broad, and deep approach for their teams.

While this is more of a strategic play with a longer-term focus, the discussions and the changes that the model drove had an immediate impact on the organization. I want to share the "Pillars" in this book in the hope that it will help your agile quality strategy development too.

In fact, I want to introduce my "partner in crime" at this point. Mary Thorn and I have both experienced the above scenario in multiple agile coaching engagements. We've seen the effect that imbalanced or non-existent agile test strategies do to organizations adopting agility. In a word, it's not pretty! Trying to bring some rigor to agile testing and quality strategies is our motivation for the book and we'll be sharing some hard fought lessons with you along the way.

While I'm the primary author for the book, Mary will be weighing in more than occasionally with what we'll be calling "Mary's Corner". Here she will share stories and examples from her experience that complement the Three Pillars. Believe it or not, Mary will not always agree with me, so you'll often get some contrasting opinions and advice. Our hope is that book will be that much better for it.

1.3 THE THREE PILLARS OF AGILE QUALITY & TESTING

The driving force behind creating the Three Pillars is organizational quality imbalance. As I travel in my coaching and consulting, I see imbalanced initiatives again and again. Quite often, as in the case with my introductory story, technology or tooling was an initial push or focus, but even then not completely in a balanced fashion. For example, not focusing on Continuous Integration in parallel with automation.

Initially it was unclear to me how to create a model that would help my clients and the agile community at-large. But, eventual I saw enough repeated patterns that I came upon the following three critical areas or 'Pillars'. Here I categorized crucial tactics, strategies, and techniques that help agile teams create a broad and deep supportive structure towards their product quality and testing activities:

Note: we'll explore all of the ideas mentioned in the other pillars below in each associated chapter…so please be patient.

1. **Development & Test Automation:** This pillar is the technology-side of quality and testing and it's not simply focused towards testing and testers. It includes tooling, execution of the Automation Test Pyramid, Continuous Integration, deep use of Extreme Programming technical practices, and support for ALM distributed collaboration tools.

   Often it's the place where organizations gravitate towards first—probably because of our general affinity towards tools and technology solving all of our challenges. An important way to think about this pillar is that it is foundational,

in that the other two pillars are built on top of the tooling. And often, organizations underestimate the importance, initial cost, and ongoing costs to maintain foundational agility in this pillar. Investment and focus is an ongoing challenge here; see Mary's Corner below.

Finally, this pillar is not centric to the testing function or group. While it includes test tooling and automation, it inherently includes ALL tooling related to product development across the entire agile organization. It provides much of the 'glue' in cross-connecting tools and automation towards efficiency and quality.

2. **Software Testing:** This pillar is focused towards the profession of testing. Towards solid testing practices, not simply agile testing practices, but leveraging the teams' past testing experience, skills, techniques, and tools. This is the place where agile teams move from a trivial view to software testing, which only looks at TDD, ATDD, and developer-based testing; towards a more holistic view of quality and testing.

   It's a pillar where functional and non-functional testing breadth and depth is embraced. Where exploratory testing is understood and practiced as a viable testing technique. It's where non-functional testings' breadth and requirements are fully understood and applied to meet business and domain needs, including: performance, load, security, and customer usability testing.

   By definition, here is where testing strategy resides, where planning and governance sits, and where broad reporting is performed. To be clear, I'm not talking about traditional testing with all of its process-focus and gatekeeper mindset. Instead, I'm talking about effective professional testing, broadly and deeply applied within agile contexts.

3. **Cross Functional Team Practices:** Finally, this pillar is focused towards cross-team collaboration, team-based standards, quality attitudes and importantly towards building things right. Consider this the soft skills area of the Three Pillars, where direction is provided for how each team will operate. You could also consider them "rules of engagement".

   For example, this is the place where good old-fashioned "reviews" are valued. This would include pairing (across ALL team members), but also slightly more formal reviews of architecture, design, code, and test cases. It's a place where inspection is valued and performed rigorously as established by the teams' Definition of Done (DoD). Where refactoring of the code base and keeping it "well kept" is of prime importance.

Speaking of DoD, this is the pillar where cross-team physical constraints, conventions, and agreements are established. But more importantly than creating them, it's where the team makes commitments to consistency and actually "holding to" their agreements. And those agreements (or promises) include those to the customer for delivering high-value, high-quality, and high-impact features that truly solve their problems.

I'm sure many readers will have differing opinions regarding the Three Pillars. Why three for example, why not four or five? Or why was this practice placed here vs. there? Or you've forgotten "this" – does that mean it isn't important?
I've seen this same type of over analysis occur in the 4 Quadrants of Agile Testing that Lisa Crispin and Janet Gregory have shared in their book. An important point to emphasize about the Three Pillars is to not get too hung up on the model. It's not intended to constrain you or to serve as the definitive view of agile testing. It's simply a model.

Ultimately my intent is to get you thinking about your agile testing, and to help guide you towards more balanced strategies. Please don't take it too seriously. It's a thinking tool and a starting point. Yes, I think it's sound, but I want you to make it context-based to YOUR context.

MARY'S CORNER

Bob, I don't think you've emphasized enough how important it is to establish a solid Test Automation framework as part of every agile adoption effort. Yes, there needs to be balance across the Three Pillars. But there also needs to be a foundation in technology-based infrastructure in order to truly perform as an agile organization.

This would include initial and ongoing investments in:

- Automation infrastructure
- Test Automation
- Test environments (virtualization, data, production-level mirrors)
- Continuous Integration and/or Continuous Deployment
- Development Support tooling
- Test Support tooling

Too many firms trivialize these investments and are unwilling to pay for the initial, ramp-up costs and the ongoing maintenance costs. As well as the trade-offs between this sort of investment vs. customer - facing features.

You don't need to have all of this overnight, BUT you need to be relentless in the pursuit of technical infrastructure support.

Figure 1 is an overview of the types of activity and focus within each pillar. This is a high-level view and there are important nuances that are missing—mostly due to a lack of space. That's why we'll be drilling into each of these pillars later on in subsequent chapters.

3 Pillars of Agile Quality		
Development & Test Automation	**Software Testing**	**Cross - Functional Team Practices**
> Pyramid - based Strategy: (Unit + Cucumber + Selenium) > Continuous Integration > Attack technical infrastructure in the Backlog > Visual Feedback - Dashboards > Actively practice ATDD and BDD	> Risk - based testing: Functional & Non - Functional > Testing planning @ Release & Sprint levels > Exploratory Testing > Standards - checklists, templates, repositories > Balance across manual, exploratory & automation	> Team - based Pairing > Stop - the - Line Mindset > Code Reviews & Standards > Active Done - Ness > Aggressive Refactoring of Technical Debt > User Stories, "3 Amigo" based Conversations
> Whole team ownership of "Quality" > Knowing the right thing to build; and building it right > Healthy - agile centric metrics > Steering via: centre of excellence or community of practice > Strategic balance across 3 pillars; assessment, recalibration, and continuous improvement		

Figure 1, High-level View of the Three Pillars

1.4 THE FOUNDATIONS OF THE PILLARS

There are some key concepts that permeate through the model and across the Three Pillars. They are first concepts that are crosscutting in nature—serving as a foundation. But they're also things that, at least in my estimation, are ongoing goals rather than things you achieve overnight. Instead, you should be constantly focusing on, investing in, talking about, and evolving them within your agile transformation.

I want to explore each one in a bit of detail, because frankly they are that important.

WHOLE-TEAM OWNERSHIP

I encounter many, many teams in my agile coaching. I also interact with many other coaches. So my sample size in this case is relatively large. We all talk about breaking down functional silos when forming and operating in agile teams – particularly the "wall between" developers and testers. But the reality is that in most agile organizations today, everyone tries to be a team, but there are still functional silos.

As you move into implementation of the Three Pillars, a constant emphasis needs to be placed on the TEAM in your formation of teams, planning, and in your day-to-day

conversations. You have to move from "development" plus "testing" to the "team" in everything you do and say.

One of my key measures of whether your operating as a team is: how actively, willingly, and happily the developers dive into testing during a sprint. If you have this sort of behavior, where they view their jobs beyond their functional silo of writing code, then you might be on the cusp of true whole-team ownership.

I've found that this is mostly a led practice. That is, the leadership team within the organization deemphasizes each functional silo and reemphasizes the whole-team view. Not only do they do that within their teams, but they also do it by creating a leadership partnership across their own functional silos. This makes it clear to their teams that cross-functional engagement is the new team model.

MARY'S CORNER

Bob, whole team ownership is a tricky one. From a leadership perspective when I have come into organizations there is always a consensus that "Yes" each Scrum team owns quality. But when the rubber meets the road and there is a severity-one issue post-production, who is the first one blamed, the testers!

At that point I often realize who within the organization and who on the Scrum team believes in whole team quality. What I always do though is try and create a partnership with the Scrum Master, the Product Owner and the Development managers on every Scrum team. When we have a severity-one issue I ask that the Scrum team hold a retrospective just on that issue.

I often coach all senior leadership that QA cannot guarantee quality and we cannot find every defect. We hope that we shine a light on the worst issues and that the business can make risk-based decisions whether to release the software. But quality is built in from the grooming sessions to the implementation and then to testing. At any point one of those things could break down and it is the responsibility of the team to hold each other accountable.

KNOWING THE RIGHT THING TO BUILD; AND BUILDING IT RIGHT

This should become the mission statement across everyone who has QA or Testing in his or her role descriptions. Clearly the "build it right" alludes to the professionalism, craftsmanship, and integrity of the entire team. When they attack user stories, do they do a complete job without cutting corners? Are the stories well and thoughtfully designed? Are they crafted according to the teams' Definition of Done?

And is all of this done consistently – story after story? Testers can make such a difference by becoming a quality champion within their teams and helping keep everyone focused on these goals.

But that's sort of a "back of the line" responsibility—checking and verifying.

What about assuring that the team is building something that will actually solve

the customers' problems, perhaps going beyond what they asked for, but providing what they truly need. This is another area of focus for the tester in agile teams. I sometimes call it – moving to the front of the line. Being more directly involved with the customer and helping to define user stories (and refine them) so that the resulting functionality truly provides value to and delights the customer.

At the end of the day, we want our customers to be raving fans as our teams. Right?

HEALTHY AGILE METRICS

Several times in the book we will explore what "healthy" agile metrics look like. You see some discussion in the Pillar 2 and Strategy chapters. But beyond that, I want you to be constantly vigilant in your metrics. There are two primary considerations—

- Agile Metrics are Different! When you begin your agile transformation you want to move from functional metrics to team-based metrics. It's a fundamental shift that is incredibly important, but far too many organizations ignore or resist it.
- Avoid Metrics Dysfunction! This has little to do with "agile" and much more to do with defining sound metrics. We've had a habit in our industry of measuring the wrong things. Robert Austin wrote a very good book that explored Metrics Dysfunction[2] and the inherent

I want you to view your shift towards agility as a place to start fresh with what you measure, how you measure, and how you react to measures. Please keep this in the back of your minds as instantiate your agile teams and begin to instill a Three Pillars view towards your quality and testing.

2. Robert Austin published the book: Measuring and Managing Performance in Organizations in 1996. It's still one of the best books that explore metrics. It starts from the perspective of examining metrics dysfunction and then explores the factors highlighting more healthy measures.

MARY'S CORNER

Bob, this topic alone keeps me employed. Can we just change the world to ask the question, "What does good look like for your organization"? and "Are your customers happy"?

For the past five years I have been in more meetings about creating better metrics for agile teams then the previous ten years combined. At the end of the day it's hard for those above to trust the team, that 30 points is their velocity, that 60% might be good enough for their code coverage, that 80% of their regression test suite is automated.

They want more numbers, to be honest they just want their waterfall metrics, it made them feel warm and cozy because you had objective measures. If this was an easy problem to solve there would be books written on it and people would be happy with "agile metrics". The problem is, it is not easy, and what is important to one organization won't be important for the next.

What I like about the Three Pillars is that is my scorecard, my metrics to say, this is what I think "Good Agile Testing" looks like.

AN ASIDE: CLASSIC MEASUREMENT DYSFUNCTIONS

- If you measure and compensate/reward developers by the Lines of Code they produce you will get…LOTS of lines of code, but might miss value, design, craft, quality, maintainability, etc.
- If you measure and compensate/reward testers by the Number of Tests (coverage) they run, you will get…LOTS of tests running over and over, but missing risk, intentionality, new tests, exploration, thoughtful coverage, etc.

STEERING

In the foundation I also identify the notion of an Agile Centers of Excellence or Communities of Practice. From my perspective, both of these are mechanisms for "steering" your agile adoption and transformation. The emphasis here is on the steering of things.

As far as steering, I've often found two common problems. First, organizations don't do enough general steering of their adoptions. Sure they might pull a "committee" together that reads a few books and meets every two weeks, but that isn't steering from my point of view. And it's uninformed. So the steering not only needs to occur, but it needs to be led my experienced agile coaches.

Many organizations are reluctant to seek external, expert advice. Sometimes it's budgetary constraints, but in my experience it's more often a reluctance to ask for help, preferring to implement agility on their own. Often you hear the mantra of – it's so simple, of course we can do it.

The second key problem is that the quality and testing organizations are rarely an equal partner in the agile transformation. Instead they are often along for the ride

with the technology or development organizations who driving the majority of the agile strategies.

While I don't have a problem with development doing some of it, the Three Pillars is truly predicated on their being a whole-team; not only at the agile team level, but also at the leadership team level. That means that QA and test leadership needs to be an integral part of guiding and steering the organizations evolution.

Key areas that come to mind as part of this steering include:

- Test Automation & CI or CD
- Test Environments
- Process Artifacts
- Definition of Done
- Measures
- Agile Testing Practices
- Continuous Improvement
- Customer Engagement

CROSS-CUTTING CONCERNS

You'll notice at the base of the 3 Pillars a foundational layer that captures crosscutting concerns that are central to the effective balance of the pillars. For example, the mantra of:

Responsibly building the right things and building them right…

Is central to the quality proposition that is inherent in high-performance agile teams. The teams need first to focus on the client challenges, interact with them and provide solutions that appropriately meet those needs. And in addition, those solutions need to be sound, well crafted, and of high quality. They need to simply work.

CROSSCUTTING STRATEGIES

Beyond the individual pillars, the value resides in crosscutting concerns. Let's go back to the original story to help make this point. Our client was advanced in BDD practices, but struggling with user story writing, or even understanding "the point of" the user story. Here they should have made the following cross-pillar connections:

- **In Pillar #1** – behavior-driven Development (BDD) and Acceptance Test-driven Development (ATDD) are mostly technical practices. They focus on articulating user story acceptance testing in such a way as to make them automatable via a variety of open source tooling. Unfortunately they have an underlying assumption that you understand story development and are connecting the acceptance test TO the customers' needs and value.
- **In Pillar #2** – one thing we didn't mention in the story was that every team had a different view towards what a story should look like and the 'rules' for writing effective stories. There were no norms, consistency guidance, or even solid examples. A focus on the Software Testing aspects of pillar two would have established these practices, which would have helped their teams.
- **In Pillar #3** – an important aspect of the user story that our client failed to realize was the conversation-part of the story. If you reference the 3-C's[3] of effective story writing as a model, one of the C's is having a conversation or collaborating around the story. It's the most important 'C' if you ask us. It's where the "3 Amigo's" of the story (the Developer(s), the Tester(s), and the Product Owner(s)) get together and leverage the story to create conversations that surround what customer problem are they trying to solve.

Do you see the pattern in this case?

You can't effectively manage to deliver on agile quality practices without crosscutting the concerns. In this case, effective use of user stories and standards, plus BDD and automation, plus the conversations needed to cross all Three Pillars. It requires a strategic balance in order to implement any one of the practices properly.

As together you and I explore each of the pillars, I'll try to continuously "connect the dots" across them. But I hope that you start doing that on your own as well.

Now onto the Three Pillars…

3. I believe the 3-C's were coined by Ron Jefferies about the characteristics of User Stories. The C's are:
   1. C – Card (front of card; As a, I want, So that)
   2. C – Confirmation (confirmation or acceptance tests)
   3. C – Conversation (collaboration)

It's often said that the Conversation is the most important C.

2.1 INTRODUCTION

I've been waiting to dig into this chapter for quite a while. This is the where the real value of the Three Pillars resides; considering cross-cutting values while crafting your agile testing adoption strategy. So what are the dimensions of this strategy? In order to make the point, perhaps we can explore three anti-patterns of effect agile testing adoption. I see all of them fairly often "in the wild" of agile adoption and think that establishing what not to do might be a good way to begin this chapter.

ANTI-PATTERN: TESTING GOES AWAY

In this case the strategy is that you need little to no testers or narrow testing within the agile methods. The intent, as I've mentioned before, comes from the early days of the methods, mostly because of their developer-driven emphasis. The other driver for it is the "generalist" view that many have in the agile community. That instead of specialized skills, everyone can do every job within the team. This not only impacts testers, but also development roles – for example, front-end vs. backend development. Those views still percolate today and cause many teams to literally forget about the testing team when they're "going Agile".

Now I'm not implying that the testers are fired, although that often happens. Instead I'm simply saying that testing is marginalized even more than it normally is. Processes are dropped and testing becomes functionally driven at a story level. Automation may be created, but it is entirely developer driven. Test leadership is stuck trying to guide their testers through a sea of agile so that they "do work" but make as "few waves" as possible.

There are many examples of companies who physically reframe the tester role, essentially removing it, as part of their agile adoption: Google, Microsoft, and Facebook are well-known examples. I consider this strategy (or lack thereof) an anti-pattern, but it does set the stage for what not to do. Let's explore another.

ANTI-PATTERN: TESTING GOES ON

Another quite frequent strategy is changing virtually nothing. This often happens in regulated or mission-critical environments. The logic is generally that all of the testing practices, tactics, measures, and approaches must stay the same. Then there is a litany of reasons why – for the regulators, for the auditors, for the Project Management Office, because of accounting practices, to support the "Software Process" are some reasons often given.

Now there is a "grain of truth" in all of these claims. Typically these environments are less then supportive of the iterative testing tempo that agile requires. They also struggle with the entire agile requirements process. That being said, not changing at all is a recipe for disaster.

My favorite story that illustrates this is from a large organization that I coached a few years ago—

"It was a large Scrum agile instance of 100-120 Scrum teams. At the time, they had a testing organization of about 300-350 testers supporting over a 1000 developers. When the testing leadership decided to "go Agile" they reserved about 300 of their testers to continue running Integration, System, and Regression testing as they'd always done it. Some of these tests were automated, but the majority were manual—approximately a 20/80 split. These tests needed a tremendous amount of "care and feeding".

That left approximately 50 testers, in reality about 30, to be integrated with their Scrum teams. And if you're doing the ratio based math – that was 1 tester for every 3 Scrum teams. Clearly the testers on the Scrum teams were overloaded to the point of ineffectiveness. This was even more troubling because the traditional testing team was moving slowly in their test preparation and execution.

But the insidious problem with this ratio balance was that the agile testers had no time to feed the traditional team the change specifications for new work. They simply didn't have the time to collaborate with the Scrum teams and pass along the information to the regression crew. So over time, the tests became more and more stale or invalid. In order to catch them up, the testers had to rework their tests outside of the iterative model – thus slowing things down.

In the end, this model proved to be 30-40% slower than their Waterfall models in delivering software. They were incredibly out of balance. It turned out that the "right" ratio was nearly the reverse of their initial investment; that is 300 testers sprinkled across the Scrum teams and 50 held in reserve to run the larger-scale testing cycles. Once they switched the ratios and went "all in" to support their Scrum teams, things worked surprisingly well."

MARY'S CORNER

Bob, I know you hate ratios but you have to have them in almost every organization. If you have a new team and no test automation I like 1:2 ratio b/c there is a lot of work to be done from the automation setup perspective. Once you have 200-300 tests or until you have a fairly low maintenance test suite then I like the ratio of 1:3, but that is absolutely as low as I will go.

ANTI-PATTERN: UNBALANCED - DRILL DOWN

I exposed a variant of this anti-pattern in the books' Introduction. The primary focus here is to pick a few agile testing practices and focus on them to the exclusion of everything else. Often this "feels good" and I often hear it defined as a "taking baby steps" pattern. The problem with it is two-fold:

1. Often the organization and teams pick the agile practices that are easiest to implement in their culture. Sometimes these are the right things to focus on, but more often they're not.
2. Balance is lost. Most if not all of the agile principles are inter-locking and interrelated. That implies that it's hard to take a piecemeal approach. Sure, you don't have to accelerate across 20 practices equally, but you do have to create an interlocking strategy…in order to gain the effects of agile.

Here's a snippet from the Introduction that illustrated this anti-pattern—

"One of the things I noticed is that the firm had gone "All In" on Behavior-Driven Development (BDD) leveraging Cucumber as the tooling framework. They had invited in several consultants to teach courses to many of their Scrum teams and everyone got "test infected" [4]*. Teams were literally creating thousands of BDD level automated tests in conjunction with delivering their software. From their perspective, there was incredible energy and enthusiasm. Everyone contributed tests and they measured the number of increasing Cucumber tests on a daily basis.*

However, a few days into my coaching, I was invited to a backlog grooming session where a team was writing and developing their user stories. What I expected was to simply be an observer. What actually happened is that I quickly realized the team didn't know how to write a solid user story. They could barely write one at all. On their request, I ended up delivering an ad-hoc user story writing class for them. Afterwards, the team was incredibly appreciative as they started to understand the important place that solid story writing held in the agile development lifecycle.

Over the next few days, I realized something very important. The organization was at two levels when it came to their agile quality and testing practices. Either they were all in or they were unaware of or under-practicing specific techniques. For example, they were all in on BDD and writing automated Cucumber tests and on Continuous Integration. However, they struggled mightily with simply writing user stories and literally had no clear or consistent Definition of Done."

Now that I've established some "don't do this" strategies; let's explore more effective approaches for morphing your quality and testing strategies towards agile methods. I've already let one key factor out of the bag—believing that a "balanced" approach between your established testing techniques and agile techniques is required.

Let's call it a "balanced transformation" to agile quality. Figure 1 provides an overview of the focus for the Three Pillars, which should help you visualize the balancing act you'll want to take across practices and techniques.

4. To use a term coined by Elizabeth Hendrickson – http://testobsessed.com/

3 Pillars of Agile Quality		
Development & Test Automation	**Software Testing**	**Cross - Functional Team Practices**
> Pyramid - based Strategy: (Unit + Cucumber + Selenium) > Continuous Integration > Attack technical infrastructure in the Backlog > Visual Feedback - Dashboards > Actively practice ATDD and BDD	> Risk - based testing: Functional & Non - Functional > Testing planning @ Release & Sprint levels > Exploratory Testing > Standards - checklists, templates, repositories > Balance across manual, exploratory & automation	> Team - based Pairing > Stop - the - Line Mindset > Code Reviews & Standards > Active Done - Ness > Aggressive Refactoring of Technical Debt > User Stories, "3 Amigo" based Conversations
> Whole team ownership of "Quality" > Building it 'Right'; Building the 'Right' Thing > Healthy - agile centric metrics > Center of excellence or community of practice > Strategic balance across 3 pillars; assessment, recalibration, and continuous improvement		

Figure 2, High-level View of the 3 Pillars

2.2 WHAT DO I MEAN BY "STRATEGY"?

Before I go much further, it's probably best to establish a baseline understanding of what I mean by strategy. When my co-conspirator Mary Thorn and I worked together at iContact, she did a wonderful job of establishing and running with an agile quality and testing strategy within her team. Mary reported to me as the Director of our testing team. We would periodically sit down to discuss the overall agile strategy I envisioned for the organization and then drill into the quality and testing bits that she would need to support.

I had the overall "Big Picture" and I expected Mary to develop a quality-centric view that integrated with our overall organizational plans. We would iterate on this several times until we had a synchronized view between us. Then Mary would align and rationalize her quality goals with her team. Yes, I said it, with her team. She would meet with the entire group and share her vision. However, together they would build a roadmap of specific activities geared towards realizing that vision.

This transition looked very much like the one made from release planning activity as organizations move from Roadmaps to Product Backlogs. But as in release planning, the results went far beyond the plans themselves. What they were creating was

a shared vision, a shared strategy, and a shared view to the path to get there. We considered several steps necessary to establish and execute a strategy. At a high level, I'll explore those steps next.

BEGIN WITH THE END IN MIND – GOALS

As I said, Mary would sit down with me and we would brainstorm the quality and testing goals we wished to establish across our organization. There were three key things that we considered while doing this:

1. **Balanced across the Three Pillars** – At the time we hadn't defined the pillars, so we were going by instinct and good judgment. Now that we have articulated the model, it's easier to see where we might be missing something. But in our practice, we tried very hard to balance across principles, tactics, and across functions, for example, development vs. testing vs. architecture.
2. **Compelling or BHAG** – We wanted the goals to be reasonable and achievable. But at the same time, we wanted them to be compelling. There's the notion of a Big Hairy Audacious Goal or BHAG and that truly represents are intentions towards goal setting. And goals were at different levels; we had annual roadmap goals that were connected to quarterly and annual organizational goals.
3. **Integrated with our overall Agile Transformation Strategy** – This is where Mary and I needed to create a shared vision for our overall agile transformation and then explore the quality and testing bits. The implication here was cross-functional, cross-team, and cross-organizational strategic vision—no silos allowed.

Remember, goals are rarely "for" leadership. Leadership establishes them, connects them, strategizes them and measures them; but the goals are for the team. They should promote action and excitement. They should inspire creativity and innovation towards meeting them; something beyond by-rote execution, but instead providing the inspiration towards the "essence" of each goal.

ORGANIZATIONAL INTEGRATION

It is the norm in most non-Agile organizations to construct your strategies and goals around functional silos. Sure, there might be some cross-cutting goals at a very high level, for example: meet some esoteric cost savings targets. But in general, goals were defined by the test teams' leadership for the testing teams, and similarly for the development teams, etc. It is purely a functional activity.

Within an agile organization, this needs to fundamentally change. That's what I've alluded to in the third point above. Yes, the testing organization should define, refine, and execute on their strategies. But they need to be integrated cross-functionally so that everyone is working together on the same essential goals.

For example, the testers cannot implement user stories as a new requirement medium without it being a whole team initiative. The training, driving forces, ultimate intentions, and goals surrounding this initiative needs to cross all teams. Sure, the

test organization may be the clear leader of the initiative as an inspirational change agent, but it needs to be reinforced more holistically. It needs to become everyone's goal with very specific definition and responsibilities to boot.

So minimally your strategies need to touch across:

- Project Management – Scrum Masters
- Architecture and Business Analysis
- Development
- Testing
- UX Design
- Product Team – Product Owners
- DevOps or Technical Operational Team(s)
- Documentation
- Customer Facing Operational Team(s)

I know this might sound daunting and risk becoming a significant time-sink, but this sort of integration is required. Normally it's the job of the organizational agile champion to "connect the dots", as in the example where Mary would connect her strategy to my overall vision.

Whether I'm an internal leader and agile change agent or an external coach, I try to inspire my teams with a vision that knows no silos or functional boundaries.

WHERE DO YOUR "STRATEGIC PLANS" RESIDE?

I hate to tell you this, well, it might be great news for most of you, but you no longer have to write independent plans, at least not at a strategic level. Sure you might pull together a high-level set of thoughts on a PowerPoint deck, but the details – the tactical steps, the tasks, the work plans, the sequencing, the milestones, and even reporting out results is sort of done for you.

That is if you do two simple things. Write user stories expressing your strategic plans and then executing them in an iterative, transparent, agile fashion. Sort of eating our own dog food if you will.

By capturing your strategy efforts in user story format (or whatever container you're using to craft work on your Backlogs) and working with your Product Owners to "feed" them to your teams, you'll be injecting:

- High-level work into your Product Roadmaps;
- As the stories are broken down, they'll surface in your Release Plans;

- And as they're broken down into executable chunks, they'll be broken down for sprint execution;
- And you'll see the results illustrated and discussed at each sprint review.

This natural process of story decomposition not only happens for "features" but it also happens for "initiative based work" such as your strategic plans. Essentially, the organizational leadership team becomes the Product Owner for agile transformation based strategic initiatives.

QUARTERLY PLANNING & PROGRESS

Agile tempo is a wonderful thing. I hope you've come to appreciate the value of truly time-boxed iterative development that delivers narrow slices of functionality on a regular tempo. One of the advantages of the approach is the "heartbeat" it creates for the entire agile organization. Usually there are two tempos that seep into your strategic planning:

- Sprint Tempo – the periodic delivery of "working software" that meets your Definition-of-Done and represents your agile teams' velocity for value delivery, and;
- Release Tempo – the periodic delivery of customer software. Software that is production environment based and in-use by your customers. Often this is represented by the term Release Train[5].

It's incredibly common to wrap your tactical steps towards your goals around your release tempo. That is, you plan for your strategy increments that overlap your release increments. Then progress is made towards these goals, on a sprint-by-sprint and release-by-release basis.

I often look for the teams themselves, to demonstrate strategic accomplishments in sprint reviews at the same level of import that they do their feature work. This level of tempo based progress transparency is another side-effect of taking this approach. Then all you have to do is understand your strategy, attend the sprint reviews, and pay attention to progress – of course asking relevant questions as opportunities present themselves.

MEASURES

As I mentioned in the last section, results should be transparent along sprint and release tempo boundaries. Now you'll want to set a tone of "whole team" metrics rather than measuring individual functional silos. In fact, measuring functional results is an anti-pattern in agile transformation because it leads to functional group sub-optimization rather than cross-team continuous improvement and results.

As I discussed at the end of the Pillar Two chapter, the metrics should generally surround four crucial areas: predictability, value delivered, quality, and team health.

---

5. This term originated with Dean Leffingwell and his agile scaling work. Now it's mostly referenced as part of the Scaled Agile Framework or SAFe. We also explored its origin's in the Pillar Two chapter.

If you can, you should stop measuring all testing team or testing based activity. Here are a few guidelines for your team-based measurements:

- Define quarterly goals, which align with your Release Train model;
- Measure progress to goal at a sprint level; use cross-team (organizational) information radiators to clearly expose progress;
- Don't overreact to sprint-by-sprint fluctuations, as with velocity, average your progress over 2-3 Sprints. What's even more interesting is to consider trending of your measures!
- Talk about progress to goal within each teams' sprint retrospective AND at a Scrum of Scrums level – adjusting accordingly;
- Upon release, measure quarterly progress and baseline it;
- Reflect on your goals, metrics, progress and results; then make adjustments as required;
- Then rinse & repeat for your next release…

MARY'S CORNER

Here is a Three Pillars scorecard that I used at a previous company. Just because I've evaluated the "Horizon" team at an "is doing" level in many areas, don't think they have little improvement work to do. In fact, many of the is-doing areas need quite a lot of work. It's just that they're practicing them relatively consistently.

The real benefit of scorecards such as this are the discussions they drive in retrospectives and amongst the team. It serves as a baseline for evaluating and planning for continuous improvement.

Horizon (Scrum)		
✓ Pyramid - based Strategy: (Unit + Cucumber + Selenium) ✓ Continuous Integration ✓ Attack technical infrastructure in the Backlog ✓ Visual Feedback - Dashboards ✓ Actively practice ATDD and BDD	× Risk - based testing: Functional & Non - Functional ✓ Testing planning @ Release & Sprint levels × Exploratory Testing ± Standards - checklists, templates, repositories ± Balanced across manual, exploratory & automation ✓ Agile - centric metrics	✓ Team - based Pairing × Stop - the - Line ✓ Code Reviews & Standards ✓ Active Done - Ness ✓ Aggressive Refactoring ✓ User Stories, "3 Amigo" based Conversations ± Building the 'Right' solutions
✓ Is doing, ± Working on, × Not working		

Figure 3, Example of Three Pillars Scorcard for a specific Scrum team

REFLECTION!

I can't emphasize enough the importance of discussing your strategic initiatives within your team-based sprint and release level retrospectives. These are crucial ceremonies to keep your goals and progress front & center for the entire organization.

You might even want to ask your Scrum Masters and teams to carve out a special section of each retrospective to discuss the goals and progress for your agile transformations. This will give you solid information as to what are working, what's not, and ideas for strategy improvement.
It's probably a good practice to keep an information radiator or two that focused on just these areas, with data being generated in part from the retrospectives.

MARY'S CORNER

At iContact we has a quarterly retrospective with the entire company. We would send out an anonymous survey, collate the results and then discuss in retrospective style the results. Many of our process improvement initiatives came from this. It was extremely valuable in that it made everyone feel like they had a voice.

2.3 AGILE STRATEGIES – IN THE BEGINNING

Next I want to cover some "strategy snippets" or general-purpose guidelines for establishing effective quality and testing focused agile adoption strategies. Some of them will be self-evident and others counterintuitive. Nonetheless, I've found these areas to be important considerations during my agile journey.

TRAINING

It's a very typical action for organizations to bring in general or overview agile training for teams moving towards agility. This usually includes an overview of various methods, agile requirements, and specific for roles such as Product Owner and/or Scrum Master. Often two groups, Business Analysis and Testers, get left behind in the training.

A major reason for this is most agile coaching & training firms have little experience with these niche roles. However, it's incredibly important to bring in training that couples agile testing, agile requirements, and agile quality dynamics—then running your whole team through them. Usually you want to plan for this before you actually start sprinting.

DEFINITION-OF-DONE

One of the first questions Mary and I ask of new agile teams relates to their Definition-of-Done or DoD. You'd be amazed how often the definition is something as shallow as: "the Product Owner signs-off on the Story"; and that's all they have. While I consider that a part of done, I normally coach teams to have a very broad and deep set of Done-Ness Criteria that they develop for their work, stories, sprints, and releases. This 4-level approach to DoD drives the collaboration and results that I feel are the hallmark of a solid agile team are represented in Figure 3.

Level of Definition-of-Done	Activities / Tactics
Work Level	These are activities that surround individual team members getting their tasks completed. For example, a developer needing to do a code review, write unit tests to a specific goal, or checking in and running code & tests in a specific environment. For the testers it might imply doing targeted SBET, writing test cases and reviewing them with their team, and running all tests and compiling results.
Story Level	These activities are usually x-team in nature and focused towards each User Story. Clearly, running the Acceptance Tests and achieving Product Owner sign-off would be part of this. But it could also include things like: 3 Amigo review steps, check-in rules, automation coverage, and acceptable defect levels.
Sprint Level	The usual measure here is meeting the Definition-of-Done for all committed work in the sprint and delivering to the Sprint Goal. Often sprint demo coverage and quality responsibilities and integration & dependency dynamics are a part of it.
Release Level	This level can be associated with very traditional Release Criteria that are usually understood from Waterfall dynamics. These constraints are usually related to full-production level release. For example, in regulated environments, then test traceability and full coverage of System, Integration, and Regression testing would be part of it. As might be documentation readiness and support training. As would be acceptable levels of defects that can and cannot be released.

Figure 4, Example of 4-Level Definition-of-Done

And you might ask who establishes them? Of course it's an organizational and team focus, but I like to see the quality and testing folks serve as the inspiration for solid, broad, deep, and nuanced criteria. Criteria that supports the promises of agile quality!

ARTIFACTS

The most common mistake that teams in transformation make is either stopping documentation all together or continuing in documenting everything as they always have. Both of these lead to a very poor transformation, but I see much more of the former in practice. The best strategy for a new team is typically two-fold:

1. Leverage your existing artifacts: Test plans, test design and cases templates, defect templates, etc. The implication here is basically – always start from where you are. Don't throw everything out and don't reinvent everything.
2. But then, "agilify" everything: And I know that's not a word, but it does have meaning. I want you to slim everything down to its very essence. For each document, you need just enough of it to provide the inherent value. For example, a Test Plan is essentially a strategy document for the team. They are the audience and whole-team testing strategies developed in real-time are the goal. So keep it short and focused on driving discussion and collaboration.

Another key is to embrace the emergent nature of agile requirements; i.e. the fact that everything is not defined in advance of execution. Become expert in user story writing, if that's what you're using for requirements.

TOOLS

How many of you think that "leading with tools" is the right response in "going Agile"? Be honest. That's what I thought…about 90% of you think that tools are a critical early decision in your adoption efforts. And by tools in this case, I'm implying testing-centric tooling and more generic agile planning (ALM) tooling as well. An incredibly common strategic pattern I see in adoption is the following:

- Select a major vendor, identify and buy tool-sets;
- Have the vendor help install/setup the tool and do team training;
- Find a couple of Scrum Masters and Product Owners—overloading the roles is fine;
- Tell the teams they're AGILE and to start coding/iterating immediately;
- Sit back and wait to reap the benefits of agility…

In a word, "don't do that"! Remember the agile manifesto advice here of—Individuals and Interactions (People and Collaboration) over Processes and Tools. While tools are important, especially in larger scale or distributed team environments, an overemphasis early on can lead to dysfunctional collaboration. I've literally seen this pattern over and over again.

Your strategy should be to install or leverage a very minimal set of just enough tools in the beginning of your agile adoption. Only what's absolutely necessary. Then over time, engage your teams in surfacing tool needs that map to your ongoing growth and agile needs. Get the team involved in the effort and evolve your environments incrementally.

INITIAL METRICS

In the beginning, it's important to connect to a very simple set of metrics that align with agile principles and that will tell you how well your team is performing. Here's an example. We like the notion of measuring 'escapes' as a quality metric. At iContact we measured a couple of escapes:

- Sprint Escapes: These were violations of our agile principles. For example, if a team ignored a DoD then we would note that. If a feature (user story) was delivered, but it wasn't completely done (even subtly) then we would call that an escape. But they could be process-level escapes as well. For example, we agreed that every team would have a retrospective at the end of each sprint or that they would groom their backlogs twice a week. If a team broke one of these agreements/understandings, we would log it as a "process escape". There were no "magic numbers" in these cases, what we considered most important was to make the decisions transparent.

- Defect Escapes: these were bugs that escaped all of our testing efforts and landed in the lap of our customer. We would sample found escapes for a 2-week period after each of our production releases. If a customer found a new issue, we would log it as an 'escape'. This would drive team activity around: root cause analysis, retrospective, and actions to prevent this escape from reoccurring. That might include more defensive programming, improved DoD, additional automation or other improvements.
- Automated Test Cases: In addition to escapes, we also felt that automation progress was important to measure. In this case, we differentiated the measures along the lines of the automation pyramid – measuring Unit level, Middle Tier level, and UI level test automation as it evolved. We never drove with magic numbers or goals, instead measuring the trending sprint-over-sprint and release-over-release. What was important to us was that we were increasing our coverage and that it was driven by team decision-making.

These were literally our only iContact quality metrics at one time. And they drove all sorts of behavior and discussion and adaptation within our teams.

It's also important to engage your teams' when defining your metrics. So, follow the four critical areas for metrics that I shared in the Pillar 2 chapter, establish 1-2-3 key measures per area, and then iterate based on whether the measures are having a positive effect or not.

Ensure that you are listening well to your team along the way. In fact, each metric should be clearly understood by your whole organization – so keep things simple.

2.4 TESTING STRATEGIES

There's a tremendous amount of bravado in the agile testing community that focuses on automated, unit, and story-based functional testing as the only testing that goes on within agile teams. And the allusion is usually to 90% automation and to 10% "of that other pesky testing stuff". The other allusion is that it's easy to achieve this balance.

However, these positions are unfortunately wrong in 90% of real-world contexts.

As I mentioned in the Pillar 2 chapter, agile testing is at its core a risk-based testing model. There are essentially three types of testing that you should carefully focus towards—

Manual – functional and non-functional testing (scripted)
Automated – functional and non-functional testing (using a multi-tiered model)
Exploratory – testing (non-scripted)

And around all of these is a test design process that usually creates or mines for new tests from the manual and exploratory activity. And in this process, the non-functional testing gets as much focus as the functional, with coverage of all major testing activities. For example: regression, system, integration, and UAT testing phases.

In my experience, there is typically a healthy balance between manual testing (25-40%), automated testing (50-60%) and exploratory testing (15-20%) techniques in mature agile organizations. It might take quite a bit of time to achieve it, but your longer-term strategy needs to be one of balance.

There is absolutely nothing wrong with focusing more on manual and exploratory testing in your early stages of agile transformation—if you lack an automation framework and automated tests.

IMPLICATIONS OF YOUR AGILE RELEASE TRAIN

I also want to remind you that there is also a strong relationship between agile testing and agile release tempo. Initially you'll simply want to pick a release train tempo and explore the testing in it at various points:

- **Pre-release train** – Is your planning opportunity at a release level. This is where you'll look for testing "risks" to mitigate incrementally. It's where you'll define the amount of Technical Test Debt[6] you might incur and plan for mitigating it. It's where you'll define the high-level strategy for your hardening sprint(s) with the Release Trains. A key focus is for breadth of testing.
- **Then…during each Sprint** – Clearly the focus is on the sprint committed features meeting the organizational and team DoD in all aspects. But at the same time, previous work needs to be assured that it's still working. So quite often notions of partial-Integration and partial-Regressions testing are executed on a team-by-team basis.
- **Then…during Hardening Sprint(s)** – Hardening sprints are a place to "catch up" on your Technical Test Debt with respect to coverage. If you've not run a full regression for a few sprints and are concerned about regressions, then you might plan for running one. It's a place for increased non-functional testing work as well, although it should not preclude you from running non-functional tests during sprints.
- **During Release** – You'll want to agree on final testing activity and checklists that you'll run through. At iContact, we had a release night game-plan for how we would be deploying and testing our product as we moved it fully into our production environment. There were checkpoints as it moved from environment to environment and as each were configured. Production level security and performance were also a key part of our testing and verification.

As part of our release planning, we would plan the overall testing strategy from a release tempo point of view. Of course things would change based on each sprint's discovery and progress, but the overall strategy usually hung together.

As they say, the plan wasn't that important, but the planning was priceless.

---

6. Bob has written an article about the notion of Technical Debt as it applies to Quality and Testing activity. It broadens the definition beyond the code. You can find it here http://rgalen.com/s/testingexperience18_06_18_Galen.pdf

2.5 WRAPPING UP

This chapter is focused towards helping QA and Test Leaders and Managers map your way forward into agile quality and testing leveraging the Three Pillars framework. It's not intended to do it for you. Instead, I provided some thoughts around common and useful tactics.

Unfortunately (or fortunately) the "hard bits" will remain for you and your teams. But I do hope it helps.

In the next and final chapter in the first edition, Mary Thorn will share on the QA Manager Role in Agile Organizations. I feel it nicely compliments and overlaps with this chapter. Between the two of them, I feel we've armed you with sufficient ideas to get you moving forward.

3. References

1. Salesforce was kind enough to share their Definition-of-Done at the 2010 Agile Conference. I've captures it in a slide deck I've used to share on various aspects of agile-centric release criteria. You can clearly Google for the Salesforce presentation. You can find mine here: http://rgalen.com/s/Agile-Release-Criteria-v5.pdf
2. Dean Leffingwell first mentions the Agile Release Train in his book Scaling Software Agility: Best Practices for Large Enterprises that he published in 2007. He later brought these ideas forward in his 2011 book on Agile Software Requirements and in the Scaled Agile Framework – SAFe. The Pillar 2 chapter discusses more details on the Agile Release Train.

www.eurostarconferences.com

Join us online at the links below.

EuroSTAR
Software Testing Conference