Agile Testing

A Practical Guide for Testers and Agile Teams

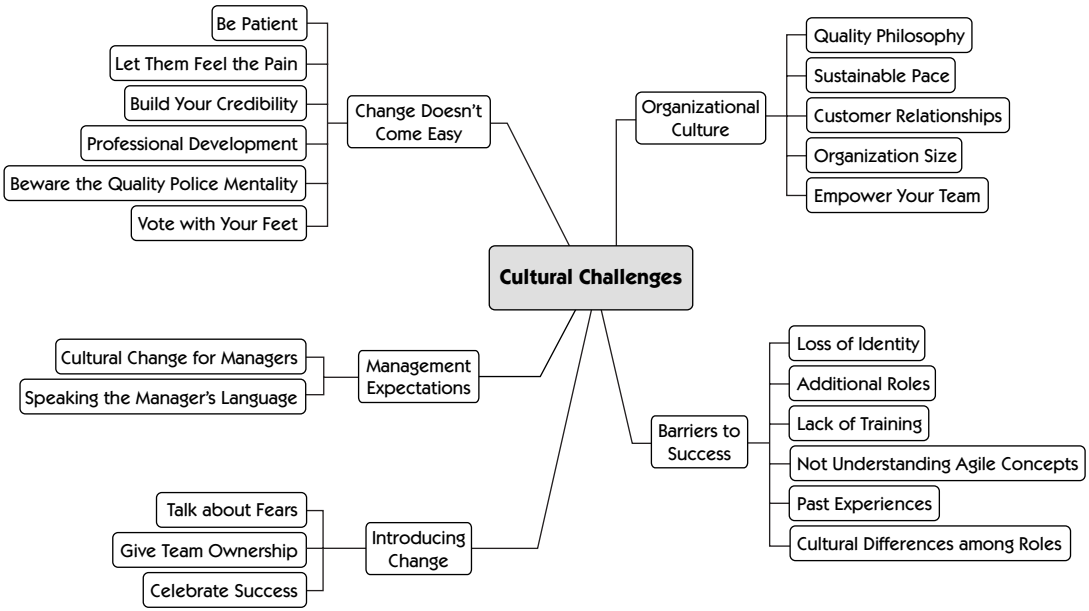
Lisa Crispin
Janet Gregory

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Chapter 3

Cultural Challenges



Many organizational influences can impact a project, whether it uses an agile or a traditional phased or gated approach. Organizational and team culture can block a smooth transition to an agile approach. In this chapter, we discuss factors that can directly affect a tester's role on an agile team.

Organizational Culture

An organizational culture is defined by its values, norms, and assumptions. An organization's culture governs how people communicate, interrelate, and make decisions, and it is easily seen by observing employee behavior.

The culture of an organization can impact the success of an agile team. Agile teams are best suited for organizations that allow independent thinking. For example, if a company has a hierarchical structure and encourages a directive management style for all its projects, agile teams will probably struggle. Past experiences of the organization will also affect the success of a new agile team. If a company tried agile and had poor results, people will be suspicious of trying it again, citing examples of why it didn't work. They might even actively campaign against it.

Organizational culture is too frequently not considered when attempts are made to implement an agile process, leaving people wondering why it didn't work as promised. It's hard to change established processes, especially if individuals feel they have a stake in the status quo. Each functional group develops a subculture and processes that meet their needs. They're comfortable with the way they work. Fear is a powerful emotion, and if it is not addressed, it can jeopardize the transition to agile. If team members feel that a new agile process threatens their jobs, they'll resist the change.

We'll talk specifically about how organizational culture affects testers working in an agile environment. The bibliography contains resources that deal with other cultural aspects that may affect teams.

Quality Philosophy

Consider an organization's quality philosophy in terms of how it determines the acceptable level of software quality. Does it tolerate poor quality? Does it take customers' quality requirements into account, or is it just concerned with getting the product into the customers' hands as fast as it can?

When an organization lacks an overall quality philosophy and pressures teams to get the product out without regard to quality, testers feel the pinch. A team that tries to use agile development in such an environment faces an uphill battle.

Some organizations have strong, independent test teams that wield a lot of power. These teams, and their managers, might perceive that agile development will take that power away. They might fear that agile runs contrary to their quality philosophy. Evaluate your organization's quality philosophy and the philosophy of the teams that enforce it.

Peril: Quality Police Mentality

If an existing QA team has assumed the role of "Quality Police," its members usually enforce quality by making sure code reviews are completed and bugs are religiously entered into the defect-tracking systems. They keep metrics about the number of bugs found, and then are charged with making the final decision as to whether to release the product.

We've talked to testers who brag about accomplishments such as going over a development manager's head to force a programmer to follow coding standards. We've even heard of testers who spend their time writing bugs about requirements that aren't up to their standards. This kind of attitude won't fly on a collaborative agile team. It fosters antagonistic behavior.

Another risk of the "Quality Police" role is that the team doesn't buy into the concept of building quality in, and the programmers start using testers as a safety net. The team starts communicating through the bug-tracking system, which isn't a very effective means of communicating, so the team never "jells."

Read on for ways to help avoid this peril.

Companies in which everyone values quality will have an easier time transitioning to agile. If any one group has assumed ownership of quality, they'll have to learn to share that with everyone else on the team in order to succeed.

Whole-Team Ownership of Quality

In Chapter 1, "What Is Agile Testing, Anyway?," we talked about the whole-team approach to quality. For many testers and QA teams, this means a mind shift from owning quality to having a participatory role in defining and maintaining quality. Such a drastic shift in attitude is difficult for many testers and QA teams.

Testers who have been working in a traditional setting might have a hard time adjusting to their new roles and activities. If they've come from an organization where development and QA have an adversarial relationship, it may be difficult to change from being an afterthought (if thought of at all) to being an integral part of the team. It can be difficult for both programmers and testers to learn to trust each other.

Skills and Adaptability

Much has been observed about programmers who can't adapt to agile practices—but what about testers who are used to building test scripts according to a requirements document? Can they learn to ask the questions as the code

is being built? Testers who don't change their approach to testing have a hard time working closely with the rest of the development team.

Testers who are used to doing only manual testing through the user interface might not understand the automated approach that is intrinsic to agile development. These testers need a lot of courage in order to face their changing roles, because changing means developing new skill sets outside their comfort zones.

Factors that Help

Although there are many cultural issues to consider, most QA teams have a focus on process improvement, and agile projects encourage continuous improvements and adaptability through the use of tools like retrospectives. Most quality assurance professionals are eager to take what they've learned and make it better. These people are adaptable enough to not only survive, but to thrive in an agile project.

If your organization focuses on learning, it will encourage continual process improvement. It will likely adopt agile much more quickly than organizations that put more value on how they react to crises than on improving their processes.

If you are a tester in an organization that has no effective quality philosophy, you probably struggle to get quality practices accepted. The agile approach will provide you with a mechanism for introducing good quality-oriented practices.

Testers need time and training, just like everyone else who is learning to work on an agile project. If you're managing a team that includes testers, be sure to give them plenty of support. Testers are often not brought in at the beginning of a greenfield project and are then expected to just fit into a team that has been working together for months. To help testers adjust, you may need to bring in an experienced agile testing coach. Hiring someone who has previously worked on an agile team and can serve as a mentor and teacher will help testers integrate with the new agile culture, whether they're transitioning to agile along with an existing team or joining a new agile development team.

Sustainable Pace

Traditional test teams are accustomed to fast and furious testing at the end of a project, which translates into working weekends and evenings. During this end-of-project testing phase, some organizations regularly ask their teams to

put in 50, 60, or more hours each week to try to meet a deadline. Organizations often look at overtime as a measure of an individual's commitment. This conflicts with agile values that revolve around enabling people to do their best work all the time.

In agile projects, you are encouraged to work at a sustainable pace. This means that teams work at a consistent pace that sustains a constant velocity that permits maintaining a high-quality standard. New agile teams tend be overly optimistic about what they can accomplish and sign up for too much work. After an iteration or two, they learn to sign up for just enough work so no overtime is needed to complete their tasks. A 40-hour week is the normal sustainable pace for XP teams; it is the amount of effort that, if put in week in and week out, allows people to accomplish the most work over the long haul while delivering good value.

Teams might need to work for short bursts of unsustainable pace now and then, but it should be the exception, not the norm. If overtime is required for short periods, the whole team should be working extra hours. If it's the last day of the sprint and some stories aren't tested, the whole team should stay late to finish the testing, not just the testers. Use the practices and techniques recommended throughout this book to learn how to plan testing along with development and allow testing to "keep up" with coding. Until your team gets better at managing its workload and velocity, budget in extra time to help even out the pace.

Customer Relationships

In traditional software development, the relationship between the development teams and their customers is more like a vendor-supplier relationship. Even if the customer is internal, it can feel more like two separate companies than two teams working on a common goal of producing business value.

Agile development depends on close involvement from customers or, at the very least, their proxies. Agile teams have invited customers to collaborate, work in the same locations if possible, and be intimately involved with the development process. Both sides learn each other's strengths and weaknesses.

This change in the relationships needs to be recognized by both sides, and it doesn't matter whether the customer is internal or external. An open relationship is critical to the success of an agile project, where the relationship between the customer team and the development team is more like a partnership than a vendor-supplier relationship.

Janet's Story

In a large project I was on recently, the customer was actually a consortium of five companies, with one of them being the software company creating the software. Each of the companies supplied three of their best domain experts to represent their needs. There was regular communication between these on-site users and their own organizations, and they were also an integral part of the team they worked with on a daily basis.

A steering committee with representatives from all five companies was kept in the loop on progress and was brought in when decisions needed to be made at a higher level.

—Janet

Having a few representative domain experts, while keeping all stakeholders continually informed, is one approach to successful developer-customer collaboration. We'll talk about others in Part V. Customers are critical to the success of your agile project. They prioritize what will be built and have the final say in the quality of the product. Testers work closely with customers to learn requirements and define acceptance tests that will prove that conditions of satisfaction are met. Testing activities are key to the development team-customer team relationship. That's why testing expertise is so essential to agile teams.

Organization Size

The size of an organization can have great impact on how projects are run and how the structure of a company matures. The larger the organization, the more hierarchical the structure tends to be. As top-down communication channels are developed, the reporting structures become directive and less compatible with collaboration between technology and business.

Communication Challenges

Some agile processes provide ways to facilitate inter-team communication. For example, Scrum has the "Scrum of Scrums," where representatives from multiple teams coordinate on a daily basis.

If you work in a large organization where the test teams or other specialized resources are separate from the programming teams, work to find ways to keep in constant touch. For example, if your database team is completely separate, you need to find a way to work closely with the database specialists in order to get what you need in a timely manner.

Chapter 16, "Hit the Ground Running," describes how one large organization uses functional analysts to mitigate problems due to remote customers.

Another problem that tends to be more common in large companies is that customers might not be as accessible as they are in smaller companies. This is a big obstacle when you try to gather requirements and examples and seek to get customer involvement throughout the development cycle. One solution is to have testers or analysts with domain expertise act as customer proxies. Communication tools can help deal with such situations as well. Look for creative ways to overcome the problems inherent in big companies.

Conflicting Cultures within the Organization

With large software development shops, agile development is often first implemented in one team or just a few teams. If your agile team has to coordinate with other teams using other approaches such as phased or gated development, you have an extra set of challenges. If some of the external teams tend to be dysfunctional, it's even harder. Even when an entire company adopts agile, some teams make the transition more successfully than others.

Your team might also run into resistance from specialist teams that are feeling protective of their particular silos. Lisa talked to a team whose members could not get any help from their company's configuration management team, which was obviously a major obstacle. Some development teams are barred from talking directly to customers.

If third parties are working on the same system your team is working on, their cultures can also cause conflicts. Perhaps your team is the third party, and you're developing software for a client. You will need to think about how to mitigate culture-based differences. Part V goes into more detail about working with other teams and third parties, but here are a few ideas to get you started.

Chapter 15, "Tester Activities in Release or Theme Planning," and Chapter 16, "Hit the Ground Running," talk about what testers can do to help with planning and coordinating with other teams.

Advanced Planning If you have to coordinate with other teams, you will need to spend time during release planning, or before the start of an iteration, to work with them. You need time to adapt your own processes to work with others' processes, and they might need to change their processes to accommodate your requests. Consider arranging access to shared resources such as performance test specialists or load test environments, and plan your own work around others' schedules. Your stakeholders might expect certain deliverables, such as formal test plans, that your own agile process doesn't include. Some extra planning will help you to work through these cultural differences.

Act Now, Apologize Later We hesitate to make suggestions that might cause trouble, but often in a large organization, the bureaucratic wheels turn so

slowly that your team might have to figure out and implement its own solutions. For example, the team that couldn't get cooperation from the configuration management team simply implemented its own internal build process and kept working on getting it integrated with the officially sanctioned process.

If there aren't official channels to get what you need, it's time to get creative. Maybe testers have never talked directly to customers before. Try to arrange a meeting yourself, or find someone who can act as a customer proxy or go-between.

Empower Your Team

Chapter 4, "Team Logistics," talks more about separate functional teams and how they affect the agile tester.

In an agile project, it is important for each development team to feel empowered to make decisions. If you're a manager and you want your agile teams to succeed, set them free to act and react creatively. The culture of an organization must adapt to this change for an agile project to be successful.

BARRIERS TO SUCCESSFUL AGILE ADOPTION BY TEST/QA TEAMS

Any change faces barriers to success. Organizational culture, as we discussed in the previous section, might be the largest obstacle to overcome. Once organizational culture has become well established, it's very hard to change. It took time for it to form, and once in place, employees become committed to the culture, which makes it extremely resistant to alteration.

This section discusses specific barriers to adoption of agile development methods that can be encountered by your testers and QA teams.

Loss of Identity

Testers cling to the concept of an independent QA team for many reasons, but the main reason is fear, specifically:

- Fear that they will lose their QA identity
- Fear that if they report to a development manager, they will lose support and programmers will get priority
- Fear that they lack the skills to work in an agile team and will lose their jobs
- Fear that when they're dispersed into development teams they won't get the support they need
- Fear that they, and their managers, will get lost in the new organization

Chapter 4, "Team Logistics," covers ideas that can be used to help people adapt.

We often hear of QA managers asking questions such as, "My company is implementing agile development. How does my role fit in?" This is directly related to the "loss of identity" fears.

Additional Roles

We know from experience that new teams are often missing specialists or expertise that might be key to their success. Lisa's team has run into obstacles so large that the only thing to do was sit back and ask, "What role are we missing on our team that is holding us back? What do we need? Another developer, another tester, a database designer?" We all know that testing is a vast field. Maybe you need someone experienced in testing on an agile team. Or maybe you need a performance testing specialist. It's critical that you take the time to analyze what roles your product needs to be successful, and if you need to fill them from outside the team, do it.

It's critical that everyone already on the product team understand their role or figure out what their role is now that they're part of a new agile team. Doing this requires time and training.

Lack of Training

We hosted a session in the "Conference within a Conference" at Agile 2007 that asked people what testing-related problems they were having on their agile teams. One of the attendees told us that they split up their test organization as advocated by the agile literature. However, they put the testers into development units without any training; within three months, all of the testers had quit because they didn't understand their new roles. Problems like these can be prevented with the right training and coaching.

When we started working with our first agile teams, there weren't many resources available to help us learn what agile testers should do or how we should work together with our teams. Today, you can find many practitioners who can help train testers to adapt to an agile environment and help test teams make the agile transition. Local user groups, conferences, seminars, online instruction, and mailing lists all provide valuable resources to testers and managers wanting to learn. Don't be afraid to seek help when you need it. Good coaching gives a good return on your investment.

Not Understanding Agile Concepts

Not all agile teams are the same. There are lots of different approaches to agile development, such as XP, Scrum, Crystal, FDD, DSDM, OpenUP, and various

mixes of those. Some self-titled "agile" teams are not, in our opinion, really practicing agile. Plenty of teams simply adopt practices that work for them regardless of the original source, or they invent their own. That's fine, but if they don't follow any of the core agile values and principles, we question giving them an agile label. Releasing every month and dispensing with documentation does not equate to agile development!

If different team members have opposing notions of what constitutes "agile," which practices they should use, or how those practices are supposed to be practiced, there's going to be trouble. For example, if you're a tester who is pushing for the team to implement continuous integration, but the programmers simply refuse to try, you're in a bad spot. If you're a programmer who is unsuccessful at getting involved in some practices, such as driving development with business-facing tests, you're also in for conflict.

The team must reach consensus on how to proceed in order to make a successful transition to agile. Many of the agile development practices are synergistic, so if they are used in isolation, they might not provide the benefits that teams are looking for. Perhaps the team can agree to experiment with certain practices for a given number of iterations and evaluate the results. It could decide to seek external input to help them understand the practices and how they fit together. Diverse viewpoints are good for a team, but everyone needs to be headed in the same direction.

Several people we've talked to described the "mini-waterfall" phenomenon that often occurs when a traditional software development organization implements an agile development process. The organization replaces a six-month or year-long development cycle with a two- or four-week one, and just tries to squeeze all of the traditional SDLC phases into that short period. Naturally, they keep having the same problems as they had before. Figure 3-1 shows an "ideal" version of the mini-waterfall where there is a code-and-fix phase and then testing—the testing comes after coding is completed but before the next iteration starts. However, what really happens is that testing gets squeezed into the end of the iteration and usually drags over into the next iteration. The programmers don't have much to fix yet, so they start working on the next iteration. Before long, some teams are always an iteration "behind" with their testing, and release dates get postponed just as they always did.

Everyone involved with delivering the product needs time and training to understand the concepts behind agile as well as the core practices. Experienced coaches can be used to give hands-on training in practices new to the team, such as test-driven development. In larger organizations, functional

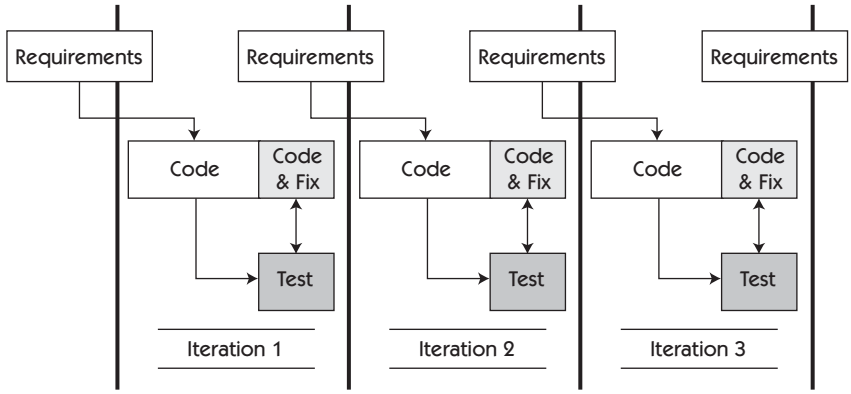


Figure 3-1 A mini-waterfall process

test managers can become practice leads and can provide support and resources so that testers learn how to communicate and collaborate with their new teams. Programmers and other team members need similar help from their functional managers. Strong leadership will help teams find ways to migrate away from "mini-waterfall" to true collaborative development, where coding and testing are integrated into one process.

See the bibliography for a link to more information about XP Radar charts.

XP has developed a radar chart to help teams determine their level of adaptation to key XP practices. They measure five different key practices: team, programming, planning, customer, and pairing, and they show the level of adaptation to practices by teams. Figure 3-2 shows two such charts. The chart

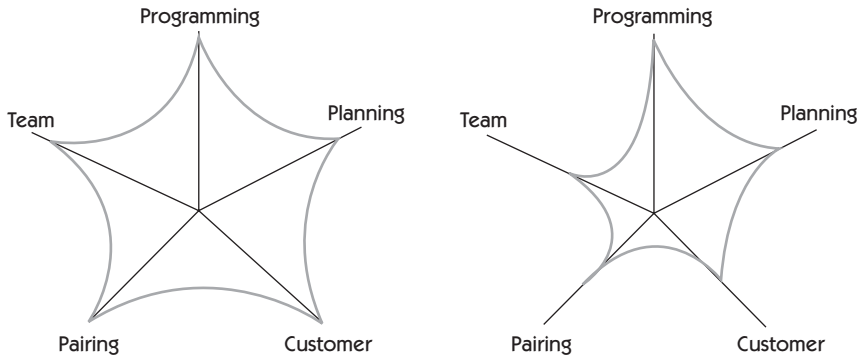


Figure 3-2 XP Radar charts

on the left shows successful adaptation, while the chart on the right shows that there are some problem areas.

Past Experience/Attitude

Lots of people have been through changes that didn't stick. Some development organizations have lived through a succession of the "methodology du jour." They throw up their hands and wonder, "Why should we do it again?" People get stuck in their old, unsuccessful patterns. Even when they try something new, they might revert to bad old habits when under stress. The following are just a few examples of people resisting change due to past experience and their perception of "the way things are":

- A tester sat in his cube and wouldn't talk with the programmers about problems he was having. He complained that programmers didn't understand what he wanted.
- A tester couldn't shake his existing attitude that programmers didn't know how to write good code, or how to test it. His condescending attitude was clear to all, and his credibility as a tester was challenged.
- A customer threw up his hands when the programmers did something he didn't like, because they "always" do what they want anyhow.

When faced with a transition to agile development, people like this often leave without giving the new process a chance. Agile development isn't for everyone, but training and time to experiment can help adjust attitudes. Ask everyone to be part of the solution, and work together to find out what processes and practices work best for their particular situations. The self-organizing team can be a powerful tool to use to reassure all members of the development team that they're in control of their own destiny.

Cultural Differences among Roles

Each new agile team member is making the transition from a different perspective. Programmers are often used to writing production code and getting it released as quickly as possible. System administrators and database experts might be accustomed to working in their own silo, performing requests on their own schedule. Customers may never have talked directly with development team members. Testers might be used to coming in at the end of the project and not interacting much at all with programmers.

It's no wonder a transition to agile can be scary. Teams can come up with rules and guidelines to help them communicate and work well together. For

example, Lisa joined a new agile team whose rule was that if someone asked you to pair with her, you had to agree. You might not be able to do it right that minute, but as soon as you could free yourself up, you had to go help your teammate.

Identify what people doing different activities need, and find ways to provide it. Customers need some way to know how development is progressing and whether their conditions of satisfaction are being met. Developers need to know business priorities and requirements. Testers need ways to capture examples and turn them into tests. All team members want to feel they are valued, first-class team members. Each team member also needs to feel safe and to feel free to raise issues and try new ideas. Understanding the viewpoint of each role helps teams through the transition.

INTRODUCING CHANGE

When implementing any change, be aware of the side effects. The first stage may be chaos; your team isn't sure what the new processes are, some groups are loyal to old ways, and some people are unsure and disruptive. People mistake this chaotic stage for the new status quo. To avoid this, explain the change model up front and set expectations. Expect and accept perceived chaos as you implement agile processes. Find the areas of the most pain, and determine what practices will solve the problem so that you can get some immediate progress out of the chaos.

Talk about Fears

When you start iterative development, use retrospectives to provide people with a place to talk about their fears and a place in which they can give feedback. Let people know that it's normal to be fearful. Be open; teach them it is acceptable to say they are fearful or uncomfortable. Discuss each source of fear, learn from the discussion, make decisions, and move on. Fear is a common response to change. Forcing people to do something they don't want is detrimental to positive change. Lead by example.

Lisa's Story

Janet and I each joined our first XP teams at a time when many XP practitioners didn't see any place for testers on an XP team. XP had a "Customer Bill of Rights" and a "Programmer Bill of Rights," but the "Tester Bill of Rights" was conspicuously absent. Tip House and I came up with our own "Tester Bill of Rights" in order to give testers the support and courage to succeed on agile teams. Over the years, many testers have told us how much this helped them and their teams learn how testers work together with other team members. I don't like too many rules, but they can

be a good thing when they help the team to overcome cultural barriers and to understand how to work in new ways. The following list presents a "Tester Bill of Rights." We encourage you to use it to help testers integrate into agile teams.

- You have the right to bring up issues related to testing, quality, and process at any time.
- You have the right to ask questions of customers, programmers, and other team members and receive timely answers.
- You have the right to ask for and receive help from anyone on the project teams, including programmers, managers, and customers.
- You have the right to estimate testing tasks and have these included in story estimates.
- You have the right to the tools you need to perform testing tasks in a timely manner.
- You have the right to expect your entire team, not just yourself, to be responsible for quality and testing.

—Lisa

Give Team Ownership

A critical success factor is whether the team takes ownership and has the ability to customize its approach. People can change their attitudes and their perceptions if they are given the right help. Lisa was able to observe Mike Cohn work with her team as a coach. As a self-organizing team, the team had to identify and solve its own problems. Mike made sure they had the time and resources to experiment and improve. He made sure that the business understood that quality was more important than quantity or speed. Every team, even a self-organizing team, needs a leader who can effectively interact with the organization's management team.

Celebrate Success

Implementing change takes time and can be frustrating, so be sure to celebrate all successes your team achieves. Pat yourselves on the back when you meet your goal to write high-level test cases for all stories by the fourth day of the iteration. Get the team together for a trivia game or lunch when you've just delivered an iteration's worth of work. Acknowledgment is important if you want a change to stick.

Chapter 18, "Coding and Testing," covers how testers and programmers work together throughout the development process.

Integrating testers into development teams while letting them continue to report to a supportive QA manager is one way to ease the transition to agile development. Testers can find ways to move from an adversarial relationship with programmers to a collaborative one. They can show how they can help

Overcoming Resistance to Agile

Mark Benander, a Quality Assurance Team Lead with Quickoffice, was on his fourth project on an agile team. The first was a major rewrite of their entire application, with a team of eight developers, one tester, and no test automation tool. He told us about his experiences in overcoming his concerns about agile development, especially about reporting to a development manager.

> We were in a matrix management type of system, where a tester reports to a development manager, but the test manager is still officially the supervisor. This comforted me somewhat, but the majority of the issues I expected to occur, such as being overruled whenever I found an issue, never did. My concern wasn't that I'd really end up thinking like a developer and just releasing anything, but that my manager, who was not a tester, wouldn't care as much, and might not back up my concerns with the application.
>
> Ultimately, I think I ended up thinking slightly more like a developer, being less concerned about some of the small bugs. My better understanding of the application's workings made me understand that the risk and cost of fixing it was potentially much more risky than the benefit. I believe that thinking like this isn't a bad thing as long as we are always mindful of the end customer impact, not just the internal cost.
>
> The corollary to my thinking more like a developer is that the developers began thinking more like testers. I'm actually a fan of the adversarial role of the tester, but in a relaxed way. I actually give the developers gold stars (the little sticker kind you used to get on your spelling test in second grade) when they implement an area of code that is especially solid and user friendly, and I give out pink stars when they "implement" a bug that is especially heinous. They groan when I come over, wondering what I've found now, and take great joy in "making my job boring" by testing their code themselves and giving me nothing to find. Needless to say, you need the right group to be able to work with this kind of faux-hostile attitude. I've never been in another company where this would have worked, but I've never worked in another company where spontaneous nerf gunfights broke out either.

Mark's experience matches our own and that of many other testers we've met who've moved from traditional to agile development. If you're a tester who just joined an agile team, keep an open mind and consider how your teammates might have different viewpoints.

the team understand the customers' needs and deliver appropriate business value. They can host enjoyable activities to build good team interactions. Having cookies or chocolate available for teammates is a good way to get them to walk over to your desk! Patience and a sense of fun are big advantages.

MANAGEMENT EXPECTATIONS

When we think of challenges involved with adopting agile, we generally think of the actual team and the issues it encounters. However, for successful agile adoption, management buy-in is critical. In a phased project, management gets regular updates and sign-off documents indicating the end of each phase. Upper-level managers might not understand how they'll be able to gauge agile project progress. They might fear a loss of control or lack of "process."

Cultural Changes for Managers

In an agile project, expectations change. In her previous life in waterfall projects, Janet remembers hearing comments like "this feature is 90% done" for weeks. Those types of metrics are meaningless in agile projects. There are no sign-offs to mark the end of a phase, and the "doneness" of a project isn't measured by gates.

Meaningful metrics are determined by each project team. In Scrum, sprint and release burndown charts track story completion and can give managers a measure of progress, but not any hard "dates" to use for billing customers. Test matrices can be used to track functionality test coverage but do not provide sign-off documentation.

The other change that is difficult for some managers to understand is letting the teams make their own technical decisions and manage their own workloads. It's no longer the manager who decides what is good enough. It is the team (which includes the customer) that defines the level of quality necessary to deliver a successful application.

Agile teams estimate and work in smaller chunks of time than traditional teams. Rather than building in contingency, teams need to plan enough time for good design and execution in order to ensure that technical debt does not increase. Rather than managing the team's activities at a low level, managers of agile teams focus on removing obstacles so that team members can do their best work.

Janet's Story

I asked the vice president in charge of a large agile project what he found to be the most difficult part in the new agile environment from a management perspective. He said that in a traditional waterfall type project, the reports all showed that everything was going according to plan until the very end, and then everything was in a panic state and "nothing worked."

In the agile project, there were problems every day that needed to be addressed. Agile projects were more work on a consistent basis, but at least he was getting realistic reports. There were no surprises at the end of the project.

—Janet

Business stakeholders don't like surprises. If they can be convinced to give the team enough time and resources to make the transition, they'll find that agile development lets them plan more accurately and achieve business goals in steady increments.

Sometimes it's actually management that drives the decision to start doing agile development. The business leaders at Lisa's company chose to try agile development in order to solve its software crisis. To be effective, they needed to have a different set of management expectations. They needed to be sensitive to the difficulty of making big changes, especially in an organization that wasn't functioning well.

In all cases, managers need lots of patience during what might be a long transition to a high-functioning agile team. It's their job to make sure they provide the necessary resources and that they enable every individual to learn how to do high-quality work.

A Testing Manager's Transition Tale

Tae Chang manages a team at DoubleClick that conducts end-to-end testing to ensure that all integration points, both up and downstream from the target of change, are covered. When they implemented Scrum, the development teams were reorganized into numerous application teams. Communication problems resulted in missed dependencies, so Tae's team stepped up to help make sure problems were detected early.

Tae told us, "I believe agile development effectively magnified the importance of cross-team communication and a coordinated end-to-end testing effort. It was not easy to work out a noninvasive (in terms of fitting into current sprint

structure) integration testing process; in fact, we are still tweaking it, but the overall benefit of such a testing effort is apparent." Their teams began to slide into the "mini-waterfall" trap. "In retrospect," explains Tae, "one of the reasons for this is because we started with the agile process before internalizing agile practices."

Knowing that test automation and continuous integration were key, the teams at DoubleClick came up with new ideas, such as a specialized build and automation team to help the development teams cope. They brought in expert training to help them learn TDD and pair programming. They started taking steps to address their legacy system's technical debt.

Tae's team attends all of the sprint planning and review sessions, using both formal and informal communication to facilitate cross-functional communication and coordinate testing and releases. He has found that it helps to keep meetings small, short, and relevant. He's also a proponent of everyone sitting together in an open work area, as opposed to sectioned-off cubes.

Tae offers the following advice to testers making the transition to agile:

"Agile development in general will initially frustrate testers in that they will not have access to full requirements documentation or defined stages of testing. In my view of agile development, at any given moment, the tester will be engaged in tasks from multiple stages of the traditional development process. A tester can be sitting in a design session with engineering and product management (she should be taking notes here and start thinking of areas of risk where proposed code change will most likely impact) and on the same day work on automating and running test cases for the proposed changes. It's a change in mind-set, and some people are quicker to adapt than others."

Tae's experience mirrors our own and that of many other teams we've talked to.

If you're a QA manager, be prepared to help your testers overcome their frustrations with moving from defined, sequential testing stages to fast-paced iterations where they perform widely varied tasks on any given day. Help them adapt to the idea that testing is no longer a separate activity that occurs after development but that testing and coding are integrated activities.

If you're a tester or other team member who isn't getting the support you need in your transition to agile development, think about the difficulties your managers might be having in understanding agile development. Help them to understand what kinds of support you need.

Speaking the Manager's Language

What do business managers understand best? It's the bottom line—the ROI (return on investment). To get the support you need from your management, frame your needs in a context that they can understand. Your team's velocity translates into new features to make the business more profitable. If you need time and funds to learn and implement an automated test tool, explain to management that over time, automated regression tests will let your team go faster and deliver more functionality in each iteration.

Lisa's Story

My team needs big blocks of time to do risky refactoring, such as trying to split the code base into multiple modules that can be built independently. We also need time to upgrade to the latest versions of our existing tools, or to try out new tools. All of these tasks are difficult to integrate into a two-week sprint when we're also trying to deliver stories for the business.

We explained to our management that if these "engineering" tasks were put off too long, our technical debt would accumulate and our velocity would slow. The number of story points delivered each iteration would decline, and new stories would take longer to code. It would take longer and longer for the business to get the new features it needed in order to attract customers.

It was hard for the business to agree to let us devote a two-week iteration every six months to do the internal work we needed to manage our technical debt, but over time they could see the results in our velocity. Recently, one of the managers actually asked if we might need to have "engineering sprints" more often. Both the product and the team are growing, and the business wants to make sure we grow our infrastructure and tools, too.

—Lisa

Like all members of an agile team, managers need to learn a lot of new concepts and figure out how they fit as team members. Use big visible charts (or their virtual equivalents, as needed) to make sure they can follow the progress of each iteration and release. Look for ways to maximize ROI. Often, the business will ask for a complex and expensive feature when there is a simpler and quicker solution that delivers similar value. Make sure you explain how your team's work affects the bottom line. Collaborate with them to find the best way for stakeholders to express the requirements for each new feature.

Budget limitations are a reality most teams face. When resources are limited, your team needs to be more creative. The whole-team approach helps. Perhaps, like Lisa's team, your team has a limited budget to buy software, and so

you tend to look at open-source test automation tools that usually don't have a large up-front purchase cost. A tool that uses the same language as the application won't help the non-programming testers unless the programmers collaborate with them to automate the tests. Leveraging all of the expertise on the team helps you work within the business limitations.

As with all challenges your team encounters, experiment with new ways that the development team and management can help each other to build a valuable product. At the same time, regardless of your development approach, you might have to make sure that some processes, such as conformance to audit requirements, receive the necessary attention.

CHANGE DOESN'T COME EASY

Agile development might seem fast-paced, but change can seem glacial. Teams that are new to agile will be slow to master some practices they've committed to using. We've met many testers who are frustrated that their "agile" development cycles are actually mini-waterfall cycles. These testers are still getting squeezed; it just happens more often. Iterations are over before stories can be tested. Programmers refuse or aren't able to adopt critical practices such as TDD or pairing. The team leaves responsibility for quality in the hands of the testers, who are powerless to make changes to the process.

There's no magic that you can use to get your team to make positive changes, but we have some tips for testers who want to get their teams to change in positive ways.

Be Patient

New skills such as TDD are hard. Find ways to help your team get time to master them. Find changes you can make independently while you wait. For example, while programmers learn to write unit tests, implement a GUI test tool that you can use with minimal help. Help the team make baby steps. Remember that when people panic, they go back to their old habits, even though those habits didn't work. Focus on tiny positive increments.

Let Them Feel Pain

Sometimes you just have to watch the train wreck. If your suggestions for improvement were rebuffed, and the team fails, bring your suggestion up again and ask the team to consider trying it for a few iterations. People are most willing to change in the areas where they feel the most pain.

Build Your Credibility

You might now be working with programmers who haven't worked closely with testers before. Show them how you can help. Go to them with issues you've found rather than opening bug reports. Ask them to review code with you before they check it in. When they realize you're contributing real value, they're more likely to listen to your ideas.

Work On Your Own Professional Development

Read books and articles, go to user group meetings and conferences, and learn a new tool or scripting language. Start learning the language your application is coded in, and ask the programmers on your team if you can pair with them or if they'll tutor you. Your coworkers will respect your desire to improve your skills. If your local user group is willing to listen to your presentation on agile testing, or a software newsletter publishes your automation article, your teammates might notice you have something worth hearing too.

Beware the Quality Police Mentality

Be a collaborator, not an enforcer. It might bug you if programmers don't follow coding standards, but it's not your job to make sure that they do so. Raise your issues with the team and ask for their help. If they ignore a critical problem that is really hurting the team, you might need to go to your coach or manager for help. But do that in a "please help me find a solution" vein rather than a "make these people behave" one. If you're seeing a problem, chances are high that others see it too.

Vote with Your Feet

See the bibliography for some good resources on being an effective change agent for your team.

You've been patient. You've tried every approach you can think of, but your management doesn't understand agile development. The programmers still throw buggy, untestable code "over the wall," and that code is released as is despite your best efforts, including working 14-hour days. Nobody cares about quality, and you feel invisible despite your best efforts. It might be time to look for a better team. Some teams are happy the way they are and simply don't feel enough pain to want to change. Lisa worked on a team that thrived on chaos, because there were frequent opportunities to figure out why the server crashed and be a hero. Despite a successful project using agile practices, they went back to their old habits, and Lisa finally gave up trying to change them.

SUMMARY

In this chapter, we talked about how cultural issues can affect whether testers and their teams can make a successful transition to doing agile development.

- Consider organizational culture before making any kind of change.
- Testers have an easier time integrating into agile teams when their whole organization values quality, but testers with a "quality police" mind-set will struggle.
- Some testers might have trouble adjusting to the "whole team" ownership of quality, but a team approach helps overcome cultural differences.
- Customer teams and developer teams must work closely together, and we showed how testers can be key in facilitating this relationship.
- Large organizations that tend to have more isolated specialist teams face particular cultural challenges in areas such as communication and collaboration.
- Major barriers to success for testers for agile adoption include fear, loss of identity, lack of training, previous negative experiences with new development processes, and cultural differences among roles.
- To help introduce change and promote communication, we suggest encouraging team members to discuss fears and celebrating every success, no matter how small.
- Guidelines such as a "Tester Bill of Rights" give testers confidence to raise issues and help them feel safe as they learn and try new ideas.
- Managers face their own cultural challenges, and they need to provide support and training to help testers succeed on agile teams.
- Testers can help teams accommodate manager expectations by providing the information managers need to track progress and determine ROI.
- Change doesn't come easy, so be patient, and work on improving your own skills so you can help your team.