IMPLEMENTING AGILE

eGuide

TECHWELL™

As customers and end-users increasingly demand rapid delivery of software that works well, development teams must find a way to release high-quality products fast and frequently. Traditional methodologies can't keep pace with today's marketplace, so many organizations are turning to agile practices to keep the work flowing.

Whether you are taking the first step down the road to agile or you want to make sure you're heading in the right direction, this eGuide provides resources to help guide you through your agile implementation.

In This Implementing Agile eGuide

Are You Agile? An Assessment Can Tell You

Plenty of companies want to be agile and go through the motions but are not really agile. An agile assessment allows you to evaluate how teams or even organizations are doing in their agile journey. But like any useful tool, there is no shortage of assessment options available. Here are the acceptance criteria to look for and a framework for using an agile assessment.

10 Things You Must Do to Become Truly Agile

Agile is not a state of doing; it's a state of being. Adopting business models on value and learning how to make teams autonomous are both necessary steps to reap the benefit of agility.

Growing Generalized Specialists on an Agile Team

A generalized specialist is not a jack of all trades. It is an individual with deep knowledge in a particular specialization who also has learned to be productive in other team roles. Here are some tips on how to grow generalized specialists on your team in order to maximize your team's productivity potential.

3 Common Collaboration Problems for Teams Transitioning to Agile

A shift toward working in smaller teams on tighter releases forces organizations adopting agile to rethink what successful delivery looks like. It can be a big change for those used to silos. Here are three key symptoms of agile teams that don't have close collaboration—and some solutions you can implement to fix them.

Don't Let Too Little Planning Tank Your Agile Adoption

Many organizations turning to agile believe it means you don't have to do any planning. This couldn't be further from the truth. A healthy agile team does just as much (if not more) planning than a team using a waterfall methodology. Preparing and setting goals sets up the team for a more successful agile adoption.

Insight from Around the Industry

Find out what experienced agile practitioners have to say about implementing agile.

Additional Agile Implementation Resources

Are You Agile? An Assessment Can Tell You

By Joel Bancroft-Connors

Tell me if you've heard this before: "Oh, we're agile. We do standups and have a product backlog." I've heard this exact statement from an engineering manager and dozens of similar statements from others, all claiming agility and tossing out a couple of buzzwords.

In this example, the reality was that their standups were an hour long, the team had more than twenty people, and the backlog was just a filter on enhancement requests in their bug tracking system. Few agilists would argue that what this team was doing could be considered agile, even if it may have been an improvement over what they were doing before.

So you know the organization isn't really agile. That's the easy part. The challenge comes when you talk to such an organization. Even if you're a highly regarded consultant, you'll be hard-pressed to change their minds based on your opinion alone.

Enter the agile assessment. This tool allows you to estimate, judge the value of, or evaluate how teams or even organizations are doing in their agile journey.

And like anything useful, there is no shortage of assessment options available. How do you choose the right one to use for your organization?

An Agile Assessment Tool Versus Model

The first thing to understand is the difference between a tool and a model or strategy. A Google search will uncover dozens of agile, Scrum, or lean assessment tools. While many of these tools are excellent, they aren't useful unless you know what you should be assessing. If you hand me a hammer and don't give me plans for building something, then all I can do is pound in nails all day with no real results.

When I first used agile assessments, all I had was a poorly documented tool. Without a strategy for how to use it, I fumbled about a lot and was pretty much ineffective with it—not unlike collecting metrics and not doing anything with them.

So while tools get all the fanfare, it's the end-to-end assessment model that is critical. You may end up using more than one tool in your model, so by opening up to looking at the overall approach, you don't get locked into a single-tool mindset.

> *If you hand me a hammer and don't give me plans for building something, then all I can do is pound in nails all day with no real results.*

Acceptance Criteria for a Good Agile Assessment

Let's start with what success looks like. If we think of an assessment like a research study or even a survey, we can draw on four well-understood criteria to measure against.

- **Retest reliability:** You test a team in January and test them again in March. You know this team has not made any real changes. Does the assessment show the same results? If it does, that's good retest reliability.
- **Inter-rate reliability:** You go in and review a team and give them an A+. Your colleague goes in and reviews the same team and gives them a D–. This would be poor inter-rate reliability. Using the same assessment for the same team, you should not get two such wildly different results based on the person administering the assessment.
- **Internal validity:** It's clear to you from just looking at the team that they got remarkably better. Does the assessment reflect this? That's internal validity—knowing that the program (the agile transformation, in this case) is having the effect you desired.
- **External validity:** Is your test so specific that it will only work with this company or department and would be totally inapplicable anywhere else? That would be poor external validity. If your assessment can apply to multiple places and applications, then it is likely a much better tool than something completely customized.

What to Measure with the Agile Assessment

Once you have the tools to ensure you are building a good model, you need to think about what you are going to measure. This is the hardest place to offer guidance, because what you need to measure can be very different depending on the organization, type of agile program, where the teams are in their journey, and so on. Generally, though, there are some key areas you want to make sure you are looking at.

Here is what I personally recommend.

- **Ability to define the product:** One of the larger threats to agile success is a lack of clarity in the backlog. If a team can't work together to define what gets built, they will never get to how to build it.
- **Ability to plan effectively:** Now that you know what you want to build, do you know how you will go about it? Do you communicate well in the scrum events? Are your plans clear? Are your information radiators visible?
- **Ability to deliver a working, tested product:** At its core, this point is about the ability to get to "done." As teams evolve it becomes more about technical practices like pairing, test-driven development, and continuous integration.
- **Ability to continually improve:** If a team is not able to learn and take actions from their past sprints, they will never truly improve.

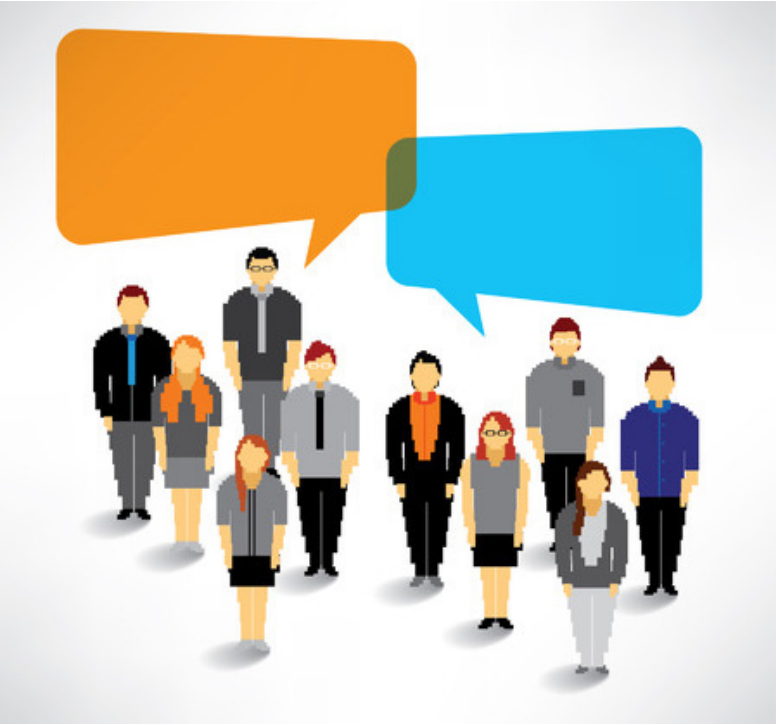
Choosing an Assessment Strategy Model

There is no silver bullet. Just keep reminding yourself of this, and you'll be so much more successful at any agile implementation that you do.

In the case of an agile assessment, there isn't a single tool or system that will get you to success. Instead, it is a series of interlocking events and tools that will give you the ability to best assess an organization or team. Just as you need a strategy for approaching an agile transformation, you also need a strategy for how you will approach an agile assessment.

The following steps compose a framework I've found to be successful.

1. **Conduct observation interviews:** By asking the same questions to all your stakeholders, you create a view of how the organization is currently operating and where the clear areas of improvement are. Doing these observation interviews has helped me to better tailor the next stages of the assessment model to fit with the organizations I am helping.



2. **Get a quick read on agility:** I started doing a quick test last year to gauge teams' agility, and it made a huge difference in the effectiveness of my assessment strategy. While having a facilitator ask the questions is ideal, the quick test can be done by teams on their own.

   Come up with roughly ten questions that touch on the basic practices for agile, such as how big the team is, whether team members are collocated, and how often they release. The answers should be able to be either yes or no or strict data (e.g., How many people are on your team?). The output will be a score that allows a basic assessment of the agility of the team or organization.

   By starting with the quick test, you can quickly prioritize teams to work with in more detail. A team that scored well will probably be farther down the list for a facilitated assessment.

3. **Assess the team:** The majority of assessment tools are some kind of facilitated questionnaire for individuals or the team. It's important to remember three key points here.

   First, take some time to observe the team in action. You can be effective only if you first create a relationship of trust. If you just walk in on day one and start asking questions, you're likely to get the same accuracy as throwing darts to pick your lottery numbers.

   Observing the team first allows you to come up with better questions, get better answers, and temper those answers with your firsthand knowledge.

   Next, ensure every voice is heard. While teams could use many of these tools by themselves, having a facilitator is vital to ensuring the best results. It's no different from any other potentially stressful meeting or event. A good ScrumMaster knows that if the team is really struggling, it is good to bring in an outsider to facilitate. Don't just hand over a tool to a team and

> *Organizations and people both are biased toward positive reviews, and without a facilitator, your end results can show a false view of the state of agile.*

expect good results. Large risk factors here are the organizational system and the team's own emotional blinders. Organizations and people both are biased toward positive reviews, and without a facilitator, your end results can show a false view of the state of agile.

The last point is to make sure your tool follows the acceptance criteria for a good agile assessment tool.

4. **Share the results:** After completing the assessment comes the tricky part: sharing it with the team or organization. If an organization is early in its agile journey, the results will likely be bleak. You have to decide just how much to share by using the classic retrospective style of focusing on future improvement. Instead of showing a detailed assessment filled with low scores, share a summary report with suggestions for improvements that will reduce time while improving quality and team happiness.

5. **Keep observing:** It's important to once again conduct an observation, this time mapping it to the assessment questionnaire and observing how the team performs firsthand. This is the bridge between passive and active involvement. You've assessed the team, you've briefed them, and you've identified areas of improvement. You now get to see the team in action for an extended period of time. And unlike during your earlier observations, the team now has a road map and near-term backlog of things to work on.

6. **Rinse and repeat:** Incorporate that agile standby, the inspect and adapt loop. Every three to six months you should check on the team's progress and aim for continuous improvement.

Choosing an Assessment Tool

There are dozens of assessment tools available. Some are free, and some cost money. Some don't come even close to meeting the acceptance criteria for a good test, and some have Ph.D. theses backing up their data.

Rather than skew opinions with my own personal preferences, I want to refer you to Ben Linder's excellent compilation of tools. [1] While his own self-assessment is on the list, Ben has done a great job listing, without bias, all the tools he's run across. My last client used his list as a starting point for deciding what tools we wanted to incorporate into our assessment model.

Let Your Data Be Your Guide

When a sprint is over, you know if it was successful. You can look at the backlog and see what is done and what's not done. The team can look at their velocity to determine the next sprint's capacity, and there are several reliable ways to forecast with the sprint data.

An agile assessment model gives you the same level of awareness for your agile transformation. You just have to make sure you're working from good data.

References

1. *https://www.benlinders.com/tools/agile-self-assessments/*

10 Things You Must Do to Become Truly Agile

By James Schiel

As an agile coach and consultant, I have one question I ask my clients when I start an engagement: "Is your organization truly agile, or are you just using the right words?" You would think, by this time, that there would be a significant number of truly agile organizations positively changing their environments.

For many software development organizations, being agile is all about doing Scrum, kanban, or Extreme Programming (XP). But even though agile frameworks model agile principles, using these frameworks doesn't automatically make the organization agile. Agile is not a state of *doing*; it's a state of *being*.

Being truly agile is clearly stated in the Agile Manifesto as being focused on individuals and interactions, working software, customer collaboration, and responding to change. These values must be built into the organization and consistently put into practice on projects.

So, how do you know if your organization is truly agile or just using the right words? Based on my years of experience, I have created a list of ten things you must do to become truly agile.

1. Focus business models on value

Business models of agile organizations focus less on the efficient delivery of products and services and more on listening to customers, earning their loyalty, anticipating their needs, and determining how to create new customers. Instead of producing more, the agile organization learns to emphasize value and continuously adapting plans.

In 1999, Whirlpool was desperate to improve customer loyalty and implemented a program of innovation from everyone, everywhere. They required a certain amount of innovation in every product development plan.

Whirlpool's customer loyalty index has risen by 68 percent and their stock has nearly tripled. [1] Whirlpool may not be a software provider, but the example shows how the right business model can positively influence outcomes. Organizations that still create detailed, linear development plans, and then force the team to follow the plan, are missing the boat. The truly agile organization employs agility to

focus their business on delivering value with frequent opportunities for feedback and reassessment of their plans.

2. Realize that agility is a mindset, not a framework

Looking for easy answers and promised gains from agility, management is frequently convinced to invest in sophisticated frameworks to support enterprise-scale agile projects. Instead of teaching how to be agile, these frameworks frequently create structures other than agile that the organization must learn.

In an agile organization, scaling is organic and driven by an internalization of agile principles across the entire company. In many ways, agility becomes an instinctive mindset. Proper scaling occurs when agility is incorporated across the organization and when management processes and practices are reinvented. Development and management teams alike must adopt and use agile principles to get work done.

3. Make teams autonomous

A significant degree of productivity in agility comes from teams that have been given the mandate to get a job done and have been given sufficient autonomy to do so. Unfortunately, many teams are surrounded by non-agile management.

This creates friction that can often be seen when teams are told they are autonomous but find management overriding their decisions and plans. They frequently must wait for those decisions to be made based on traditional metrics (e.g., productivity, efficiency, and estimate-to-actual).

The problems created between autonomous teams and non-agile management can be solved by doing two things. First, you must ensure that the management team understands and uses agile.

Second, and much more importantly, you must look at effective external team leadership as a completely different skill set. This requires training management to support the team by building relationships, articulating clear goals, and helping the team make decisions.

4. Work in small batches

In all work, complexity and risk go hand in hand and are directly proportional. Usually, the more complexity, the greater the risk. Complex work requires significant effort, and less complex work requires less effort. If complexity results in the team spending more time fixing mistakes, we've gained nothing. Productivity can be improved by breaking down complex work into smaller, less complex tasks.

While doing testing on small batches of work may seem like it adds a lot of overhead, small batches allow organizations to get better by doing something repeatedly and frequently. It has been my experience that tasks should be sliced into small units not to exceed a few days' work. Of course, completed work should still be inspected to ensure it meets the needs of the customer.

> *A significant degree of productivity in agility comes from teams that have been given the mandate to get a job done and have been given sufficient autonomy to do so.*

5. Aim for small teams

Small teams consistently outperform individuals and large teams, especially when a variety of skills and perspectives are needed to complete a project. I recommend no more than eight people on a team. The bigger the team, the harder it is to get everyone together and to reach agreement on decisions. Smaller teams simply work more efficiently.

When creating a small team, the agile organization should keep some key practices in mind.

Teams rely far more on the right mix of skills than the right mix of personalities. Attempts to align compatible personalities using assessment tools often provides the same result as having management decide on the membership. Teams form around common goals and frequently need little direction. Once you've focused your team members on a problem and made sure they have what they need to get it done, they'll usually deliver great results.

Teams work better when everything they need is already within their sphere of influence. Scrum, in particular, defines a team as a set of developers that has all the skills necessary to do what is asked of them.

6. Perform work in a single-piece flow

Championed by Toyota, the concept of single-piece flow processing has revolutionized how the factory floor is run. In single-piece flow, the team completes a single unit, then starts work on the next unit. As a team completes a unit of work (including coding, testing, and documentation), the item should be immediately demonstrable. This gives the team frequent feedback rather than waiting until every-thing is built before testing anything.

Single-piece flow yields completed, demonstrable software every few days. Mistakes are quickly found and fixed, and costs are lowered dramatically.

To accomplish single-piece flow, teams must abandon traditional approaches where work is done in steps and handed off from one specialist to another. Instead, team members bring their individual skills to bear on engineering a task or solving a key problem. The team works together to simultaneously collaborate on the execution of the work.

7. Work in short cycles

While shrinking team size and complexity of work offers true benefits, don't forget to reduce the length of your iterations, too.

I recommend shortening iterations to no more than two weeks. In agile development, the end of an iteration signals an opportunity to evaluate what's been built against customer needs. Shortened iterations improve quality, reduce risk, and reduce project cost. As a result, time to respond to the inevitable defect or incorrect decision is greatly improved.

I frequently ask coders and testers, "After you write some code or some tests, what do you do next?" Invariably, the coders run the application and the testers run their tests. "If the latest code doesn't work, the sooner we find out, the easier it is to fix," they say.

I couldn't agree more. The more often you test your assumptions, the easier it is to fix them. Longer iterations accumulate risk that your customer won't like what you've built. Shorter iterations help you build what your customer really needs.

8. Value soft skills

While technical skills are important, soft skills can be more valuable to sustaining high team performance. These skills include the ability to:

- Argue passionately while not attacking
- Share information freely; full transparency is necessary

- Disagree with a decision while fully supporting it
- Make abstract concepts visible and easily understandable

The best team member I've found is someone who has great communication skills, is innovative, and is adaptable to most any situation at hand. While the ability to code in Java or build effective functional specification is important, I'd recommend giving some attention to those soft skills, too.

9. Monitor debt closely

While building products or services, organizations tend to incur a significant amount of debt. Debt is a result of the difference between doing something right and doing something poorly.

Debt forces the organization to compromise because available choices are limited due to outstanding debt. While organizations are beginning to understand the importance of dealing quickly with defects in order to limit technical debt, many still don't recognize the need to pay attention to learning opportunities or research opportunities provided during the typical iteration.

To keep your debt under control, keep in mind the following:

Assume that any debt not being actively addressed is probably getting bigger. Take steps to reduce it by planning daily to improve skill sets, decision-making capability, innovation, and motivation.

When problems do occur, take time to identify the root cause and correct the appropriate issue to keep debt from growing. If an organization fails to deliver on time because of product performance issues, the root debt-related cause might be that the organization lacks the skills to do effective performance testing.

10. Be brave enough to experiment and fail

The world of business is replete with the remains of companies that got comfortable. A business that doesn't move forward is moving backward, and a business or organization that views experimentation as a gamble will probably lose out in the long run.

Sometimes that means you're going to get it wrong. Consider Thomas Edison's experiments with the light bulb. Edison's assistant logged more than 2,700 individual experiments. When asked by a reporter about his many failures, Edison responded, "I now know several thousand things that won't work!" [2]

To innovate, one must experiment and fail. Failure creates amazing possibilities for learning and growth, and must be made part of the organizational culture. A study was performed with Upworthy (not a software development organization) regarding their emphasis on fostering experimentation and divergent thinking. Their culture assumes that a 95 percent failure rate is a sign of a team doing a phenomenal job of experimenting and learning how to get to the right answer. [3] That's an unknown attitude in many organizations, where teams are encouraged to think until they determine the correct answer.

Teams must be permitted to experiment and fail. Where innovation is called for, so is experimentation. Throwing away a thousand lines of good code for the hundred lines that work is a cost-saver in the long run—consider all of those defects the team won't have to fix.

Becoming Truly Agile

Is your organization transforming into an agile one or one that simply uses agile terminology? If you want to be successful in getting the most day-to-day value out of your teams, you've got to be prepared to create new ways of working. To make real agility in your organization, you're going to need to make some real changes.

References

1. *Hamel, Gary. "The Why, What, and How of Management Innovation."* Harvard Business Review. *February 2006.*
2. *Rutgers School of Arts and Sciences. "Thomas A. Edison Papers." The Edisonian, Volume 9, Fall 2012.*
3. *Critchfield, Sara. "How to Push Your Team to Take Risks and Experiment."* Harvard Business Review. *March 9, 2017.*

Growing Generalized Specialists on an Agile Team

by Jeffery Payne

It's difficult for an agile team comprised solely of specialists to be fully productive. If team members are unable to help others with their tasks, the team will not maximize its potential.

The agile community calls team members who are capable of working in a variety of roles *generalized specialists*. A generalized specialist is an individual with deep knowledge in a particular specialization who also has learned to be productive in other team roles.

While we cannot expect our staff to become experts in all areas, it's possible to teach motivated individuals how to be productive in other roles, particularly those most related to their specialization.

Teach Developers How to Automate Tests

Test automation is essential in agile, if for no other reason than to support fearless refactoring. We already expect developers to be able to write effective unit tests, so why not leverage their skills to support other necessary test automation activities?

The test automation needs that developers can most easily learn to support include automation for story acceptance tests and application smoke tests. I would hesitate to throw a developer into designing these types of tests without some testing techniques training, but having them automate tests that others have created is a good use of their programming skills. Plus, their software engineering background should reinforce a "treat tests as code" philosophy that will result in more maintainable automation scripts for the team.

Teach Business Analysts and UI Designers How to do Exploratory Testing

Anyone with strong critical thinking skills, some training, and practice can learn how to become an effective exploratory tester. As testing time is often squeezed at the end of sprints, getting everyone on the team capable of participating in testing efforts can make a huge productivity difference.

Those with strong product domain knowledge, such as business analysts or UI designers, are great candidates. Because much of their analysis and design work is often performed during the first half of each sprint, they may have some spare cycles to help test later on.

Teach Testers How to Improve User Stories and Acceptance Criteria

While your testers may not necessarily be experts in your product domain, their role demands they understand your software product end to end. This understanding, coupled with their critical eye toward testability, can help make your user stories more understandable, less ambiguous, and easier to implement.

User story acceptance criteria also often goes hand in hand with tests that validate stories, meaning testers will be good at writing appropriate user story acceptance criteria, too.

Teach ScrumMasters How to Plug Holes

ScrumMasters come in all shapes and sizes, so it is difficult to give specific guidance for how they can broaden their skills, but a good ScrumMaster will seek out roles they can participate in to increase productivity. Spend some time understanding your ScrumMaster's background to see which role they can best support.

3 Common Collaboration Problems for Teams Transitioning to Agile

by Kevin Dunne

As teams move toward agile, there is pressure to deliver software more quickly and through collaboration among smaller cross-functional teams. This can be a big leap for those used to longer release patterns and siloed teams focused on specific functions such as testing, development, or operations.

A shift toward working in smaller teams on tighter releases forces organizations adopting agile to rethink what successful delivery looks like. While teams could work independently on their own functions in a waterfall environment, teams that work without close communication in an agile environment are destined to fail.

Here are three of the key symptoms of agile teams that don't have close collaboration—and some solutions you can implement to fix them.

Problem #1: Tests don't align with requirements or often need to be rewritten

As requirements or user stories change within the course of a sprint, testers aren't in sync with the business analysts or product owners. They frequently have to update tests, or worse, they are executing the wrong tests.

Solution: Implement a process for daily standups to address requirements that are evolving. Building basic dashboards to display user stories needing review and test redesign is another common tactic.

Problem #2: Requirements can't be tested easily

When requirements are defined without close involvement from testers, they can be imprecise and difficult to test properly. Testers need to go back to business analysts or product owners for clarification, and often they end up writing the wrong tests.

Solution: Involve testers closely in the requirements definition process to make sure testability is considered from the beginning. A behavior-driven development approach can be used to bake in tester feedback and testability from the very start.

Problem #3: Testing timelines get squeezed or delivery dates get pushed

When testers do not get involved early on, they are beholden to the business analysts or product owners to finalize and deliver requirements in order to start planning their test approach. This leaves testers scrambling to put together a plan, so they frequently either encounter unexpected delays or need to cut out test coverage that may be critical.

Solution: Test-driven development moves test creation to the beginning of the process, ensuring that testing is never getting squeezed in as the final step in the delivery process. In this way, code cannot be written on a feature before a test is developed, so if anything does need to be squeezed, the development of feature code can be pushed into a later sprint or release.

All organizations feeling the pressure of agile software delivery processes should be eager to find ways to increase collaboration within their development teams. Without proper cross-functional communication, quality and delivery timelines will suffer. It's vital that teams have strategies to encourage close teamwork in their engineering organizations so they can drive better quality outcomes for their software.

Don't Let Too Little Planning Tank Your Agile Adoption

by Alan Crouch

One of the most prevalent misconceptions I hear from organizations struggling with agile adoption is about planning. Many organizations turning to agile believe it means avoiding planning—it's a common stance for those who have never truly experienced being on a highly functioning agile team.

This couldn't be further from the truth. A healthy agile team does just as much (if not more) planning than a team using a traditional, waterfall software development methodology.

This approach is derived from the Agile Manifesto: An agile team values "Responding to change over following a plan," but this doesn't mean an agile team does no planning. The expectation is that the team will deliver greater value from starting on the well-known requirements than it would from continuing to invest in additional planning.

The value of any learning that results from getting started will have a greater return on investment than the value from continued planning. An agile team expects that once it gets started, the plan that has been developed is likely to change before they are done based on what the team has learned.

Agile planning happens throughout the project. In the absence of this planning, projects are doomed to run over budget and over schedule. What's worse, the delivered product (if there ever is one) will not accurately meet the customer's needs or the original project objectives, resulting in project missteps or failure. Additionally, the code generated is likely to be of suspect quality (typically buggy and not easily maintained or extended).

Effective planning takes place before the project starts, and monthly sprint planning ensures the backlog is prioritized and requirements are well understood. When goals are presented to the development team early, it gives them a chance to understand the scope of the project, begin thinking about how they would deliver it, and get initial buy-in. It also gives the product owner an opportunity to refine the requirements before the story is deemed ready for development. This type of planning is important because it allows the team to understand the big picture so that they can think through a high-level application architecture.

It's important to spend enough time during your sprint kickoff to plan how you will deliver on the stories for each sprint. The team plans out the detailed tasks necessary to deliver on each story. These tasks are then estimated and the team commits to delivering the number of features they believe they can complete during the sprint.

During the sprint, these detailed requirements are used by the developers to design and build software, create end-user documentation, and successfully deploy the software. Sprint planning is also a key component for testers so that they can write well-designed tests, establish acceptance criteria, and, most importantly, outline a test strategy to reduce duplication of effort and ensure tests properly cover all the key functionality.

If you take the time to prepare and continually maintain a product backlog, your project will proceed more predictably, the end product will better address the customer's needs, and the code produced will be of higher quality—and you will enjoy a more successful agile adoption.

Insight from Around the Industry

"Before you attempt your migration from traditional methods to agile and Scrum, you should evaluate your organization for its willingness and ability to adapt to the inevitable organizational changes. Otherwise, your agile deployment may be impossible from the outset."
» John Holmes and David Nielsen

"The days of big planning phases are over, as are long-implementation phases and rolling results out to everyone at the same time. Small changes and improvements help you fail faster, inevitably helping you improve faster."
» Sven Peters

"That's really the message of agile. It's not about doing a daily stand-up or about pairing or any of the other practices. Those are all good things that I believe in, but what's most important for a team is to constantly be learning about what works."
» Linda Rising

"Companies are being created, compete for a little while, and become unable to compete because these are changing so rapidly. We live in an age of extreme flux. It's really up to all of us to be able to respond to change much more quickly."
» Sanjiv Augustine

"By "agile," I mean agile Scrum or Scrum. Agile Scrum is a good place to get to and stabilize on if your team is not there yet. If you have multiple dispersed teams, Scrum is a good method to align and coordinate between them."
» Michael Nauman

We live in an age of extreme flux. It's really up to all of us to be able to respond to change much more quickly.

"With agile, we see a lot of folks that are coming in and saying, 'We've been trying agile and it's not working for us.' What it generally turns out to be is, you come in and you lay a framework down or a process down, and you follow it blindly. Everything is context dependent, so you need to use what works in your context, don't just come and apply these rules and think it's just magically going to work."
» Brandon Carlson

"You don't want to 'do' agile, as it's often viewed. Instead, you want to cultivate and grow your skills, and your team's skills, so that you can 'be' agile—so that you can face a sea of unexpected changes with the elegance and finesse of a Swiss Army knife and a roll of duct tape.
» Andy Hunt

"The biggest hurdle is actually a mental one. Agile is not a switch that you just throw, that just turns on agile automatically for you. It's a process that you transition into. Getting to the state of agile testing, of being able to test in an agile way—is a journey."
» Samir Shah

"First, the environment needs to be set up so that part of the agile mindset is to be OK with making mistakes, with exploring those, learning from them, and gaining new perspective. I think the more that people see there's a new perspective, or a new idea that comes out of the mistakes, the more they're willing to sort of put themselves out there and not just do status quo because they're afraid that somebody's going to get mad if they do something wrong."
» Jessie Shternshus

Additional Resources

MORE INFORMATION FOR SOFTWARE PROFESSIONALS

AGILECONNECTION™
A TECHWELL COMMUNITY

AgileConnection brings together the latest agile ideas and practices from experienced software professionals and thought leaders. AgileConnection offers how-to advice on the latest agile development principles, practices, and technologies through Q&A discussions, articles, interviews, presentations, and more. Join the community to gain access to member exclusive content such as conference presentations, weekly newsletter updates, and more.

CLICK HERE

NARROW YOUR SEARCH TO A SPECIFIC TYPE OF RESOURCE:

Better Software Magazine Articles

Better Software magazine is a digital quarterly filled with expert analysis, how-to articles, and real-world case studies covering all aspects of software development. **Click here** to join AgileConnection and access agile articles from *Better Software* magazine.

Interviews

Each year, TechWell interviews dozens of software professionals including well-known thought leaders, seasoned practitioners, and respected conference speakers. **Click here** to read, listen to, and watch interviews with agile experts.

Conference Presentations

Couldn't make it to a Better Software or Agile Dev conference? TechWell conference presentations are available to AgileConnection members soon after a conference ends. **Click here** to join AgileConnection and access conference presentations related to implementing agile.

TechWell Insights

Find stories on DevOps, agile, testing, development, and more, written by software industry professionals. New stories are added each day, so **click here** to read the latest or sign up to receive the weekly newsletter.

Agile Conferences

The Agile Dev conferences are full of keynotes, tutorials, and classes covering everything from agile project management and agile test automation, to the agile development life cycle. Learn from experts in the field and network with your peers to get the most immersive agile conference experiences possible. **Click here** to learn more.