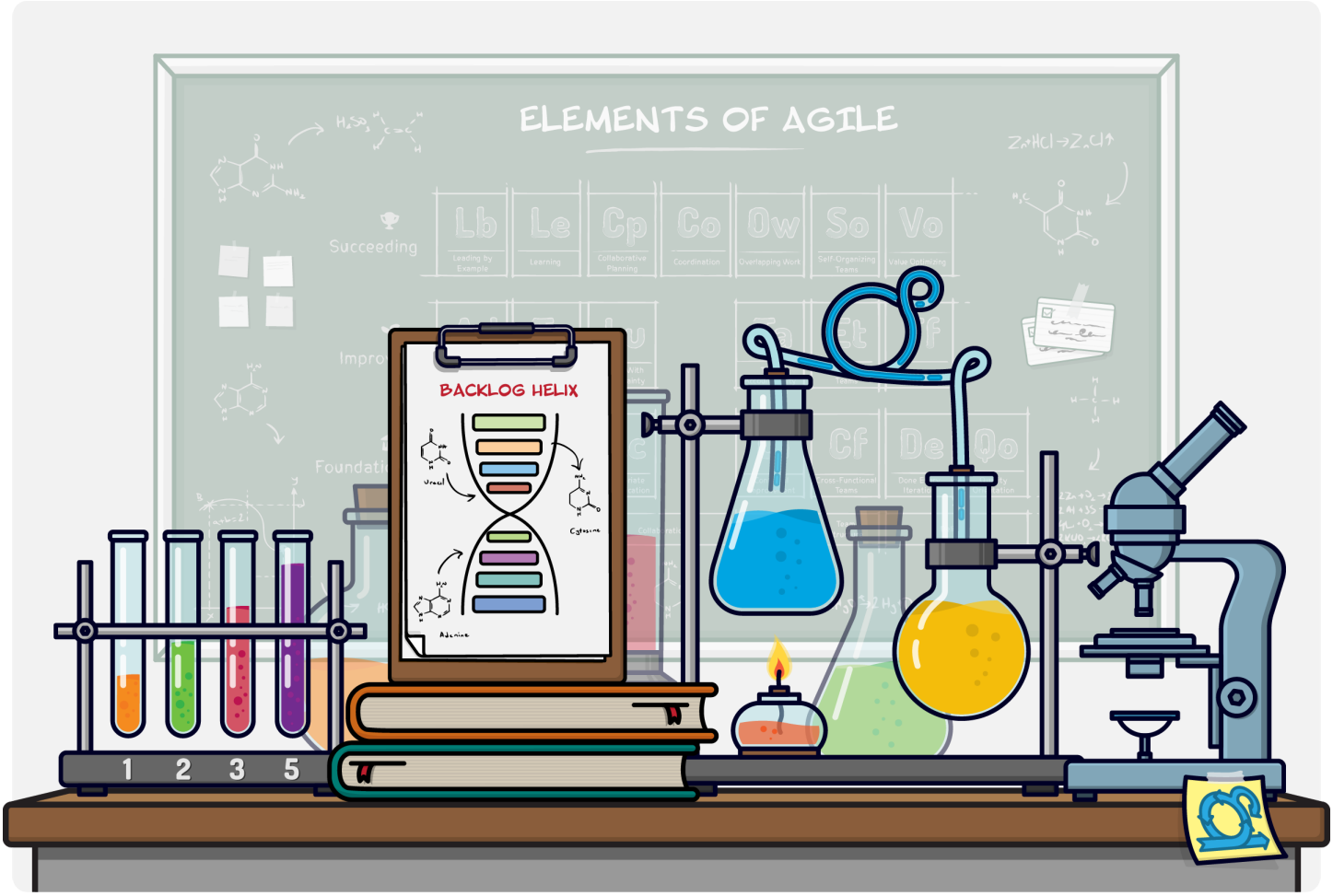


Your Personalized Report

Prepared for helena on October 3, 2022

A Message from Mike Cohn

Hi helena,

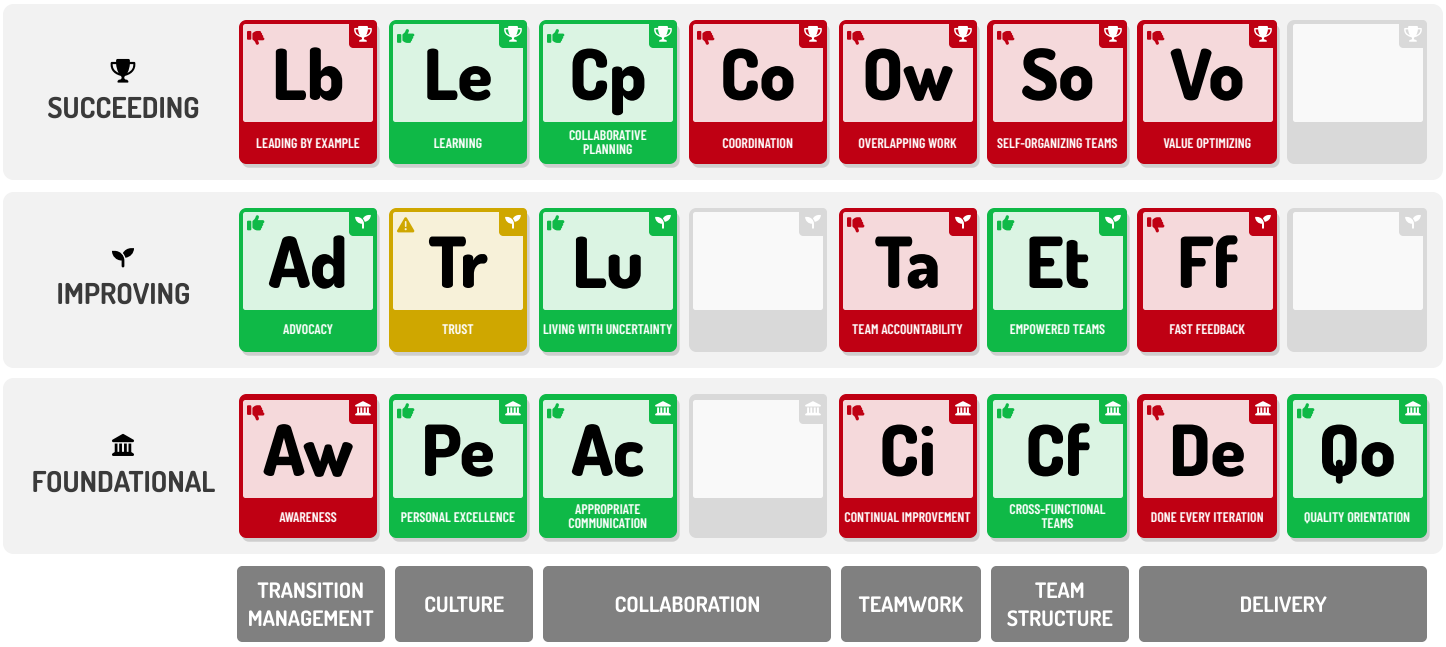
Congratulations! You've taken the first step toward succeeding with agile. We hope this playbook will help you emphasize your current strengths and identify some growth opportunities.

You now have a powerful tool in your kit to create a prioritized improvement backlog. As you review each element, remember that becoming agile is a continuous journey of improvement. You never reach some end point where you can spike the football and do a touchdown dance (though I hope you do take time to celebrate every small victory along the way).

What follows is a snapshot of how you are currently doing at being agile. It might seem like a lot to sift through, but you don't need to improve immediately at everything contained in this report. In fact, if you tried to improve everything immediately, you'd almost certainly fail. Follow the advice on the "How to Use This Report" page and start small, even with just one Element of Agile.

Whatever you do, no matter how small, you'll be learning and growing. And at the end of the day, that's the secret to succeeding with agile.

Elements of Agile



How to Use This Report

Every agile adoption is different. However, based on our involvement helping hundreds of companies successfully transition, we have identified twenty elements of successful agile adoptions.

Collectively referred to as the Elements of Agile, each Element represents an important achievement in becoming agile. The twenty Elements are organized into six categories:

- Transition Management
- Culture
- Collaboration
- Teamwork
- Team Structure
- Delivery

Within each category are three levels. The **Foundational level** 🏛 contains core Elements, which form the basis of any successful agile transformation. At this foundation are Elements such as cross-functional teams, a desire for continual improvement, and striving for personal excellence. Until these (and other Foundational Elements) are at least partially in place, you'll find it challenging to make progress at the next level: Improving.

The **Improving level** 🌱 is where agile has taken hold and is starting to deliver real benefits to the organization. Elements at this level—such as increased team accountability and learning to live with uncertainty—build on Foundational Elements.

They also support Elements at the highest level, **Succeeding** 🏆. Once an organization starts to work in accordance with the Succeeding Elements, agile has fully taken root in an organization. At this level an agile transition can be self-sustaining with each improvement encouraging further improvements.

Although the Elements are grouped in three levels (Foundational, Improving, and Succeeding), you do not need to work strictly from top to bottom. You should generally focus on Foundational Elements first, but if you think a higher level Element needs to be addressed earlier, you should absolutely seek improvement in that Element.

We recommend you use this report to create an Improvement Backlog, which is a list of improvements you'd like to see in your team or organization. To start, pick one or two Elements on which to focus. Review the pages in this report on those Elements. For each, select one or possibly two things from that Element's Things to Try section.

Those items now form the beginning of your Improvement Backlog. From that backlog, commit to doing one or two new things in the next iteration. Because sometimes a

change feels worse at first, consider repeating the change for a second iteration before evaluating its impact.

Add to your Improvement Backlog any time you want to try something else for an Element you've chosen to work on. You can also add to the Improvement Backlog any time you are ready to take on an additional Element.

If you'd like to take a deeper dive into any Element, you can use our on-demand or live online courses, extensive blog library, or free videos. You might also choose to ask us how we could customize an engagement that meets your specific needs.

If you've taken this assessment with a Mountain Goat consultant, they will sit with you to help sort through your results and help you make the necessary changes. If you'd prefer to go it alone, we've included many ways to get started.

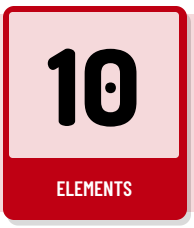
A Plan to Succeed

On the next pages, we'll take a look at your personalized report, and find places where you can begin to improve how you are succeeding with agile.

We found strength in 9 Elements. These will generally need maintenance to keep your good score.

We found 1 Element that needs improvement.

We found 10 Elements with significant growth opportunities. These need attention.



Take a Deep Breath

And remember, you only have to try one thing at a time, for two iterations at a time.

You can use this assessment as the beginning of an improvement backlog for yourself, your team, your division, or your entire enterprise. It all depends on your role and sphere of influence.

The report details each Element where you scored well and each Element where you have some growing to do. Pick any Element at which you'd like to improve. Consider starting with a Foundational Element in the bottom row of the table. But, that's up to you. The most important thing is to start.

As you grow more agile, your needs will change too. Feel free to take this assessment every year to see where you are making progress and identify new opportunities that arise.

And if at any time you get stuck or want help, send us an email at hello@mountaingoatsoftware.com. We're happy to create an engagement custom fit for you.

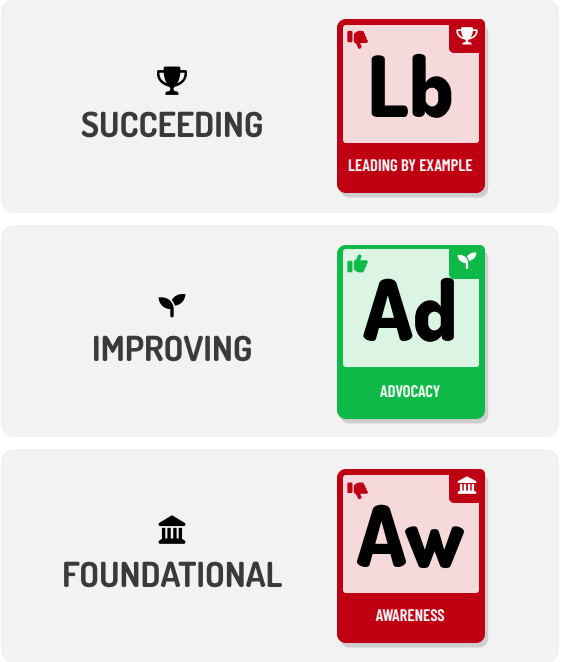
Let's Get Started

Transition Management

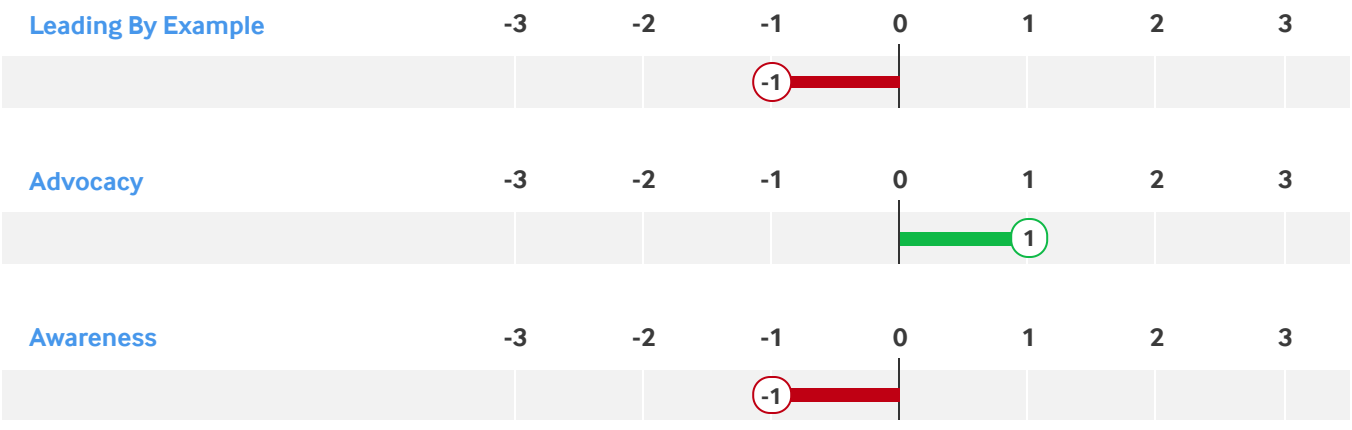
Today, every organization is striving to become more agile. And who can blame them? Successful agile teams are producing higher-quality products that better meet user needs, and they do it more quickly and at a lower cost. Being agile is no longer a competitive advantage; it's table stakes.

Yet transitioning to an agile approach is hard—harder than many companies anticipate. To reap all of the rewards of agility requires a great deal of change from not only the teams but the entire organization.

Changing practices is one thing; changing minds is quite another. To succeed, you'll need to make steady progress in all three Elements in this section: Awareness, Advocacy, and Leading by Example.

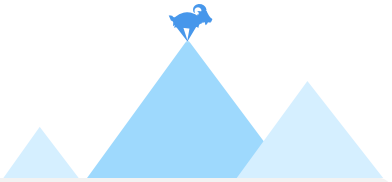


RESPONSES



Awareness

TRANSITION MANAGEMENT | FOUNDATIONAL



GROWTH OPPORTUNITY

Responses indicate that agile is not well understood. Until teams, leaders, and stakeholders share a foundational understanding of what it means to be agile, you will have a hard time making substantial gains in an agile transformation.

A successful transition to agile begins with an awareness of what it means to agile. Those who will be working on agile teams will need to acquire a solid understanding of agile and how it differs from their prior method of working.

People who work with the team but are not directly on the team can likely get by with less comprehensive knowledge about agile, but they need to be at least familiar with its implications on their own work.

Without clear, widespread awareness of agile, teams often adopt partially agile processes—which fail.

SYMPTOMS TO WATCH FOR

Without sufficient awareness of what it means to be agile, transition efforts often struggle due to misunderstandings. Here are some symptoms that might warrant closer inspection.

1. Disagreements arise as to the meaning of various process-related terms.
2. People don't know how to perform their role on the team or what is expected from them.
3. People in new or modified roles—such as Scrum Master, product owner, or agile coach—are often either tentative in their new roles or fall back into old habits detrimental to agility.

THINGS TO TRY

The following activities can help when, and if, you notice that teams are suffering from a lack of awareness about agile.

1. **Arrange Training for Everyone Working Directly on an Agile Team**
   Prior to agile, software developers were educated to understand a full problem, prevent changes to the scope of that problem, and then pursue a solution to the whole problem. Agile requires developers to shift that thinking and work on portions of a problem before the whole is entirely understood.

   This dichotomy is perhaps strongest among software developers but is not unique to them.

To create the needed shift in thinking and acquire the new skills to go with it, many team members will benefit from training.

This will be especially important for those in roles that are new with agile such as Scrum Master, product owner, and agile coach.

2 Provide or Conduct Simplified Training for People Who Work with the Teams

Training may also be beneficial for leaders, stakeholders, and others who work with agile teams. Many of these individuals will need to adjust some of their behavior as they work with teams adopting agile. For example, a salesperson may have traditionally chosen delivery dates singlehandedly and promised those dates to customers. In agile, the teams should be viewed as equal partners in determining delivery dates.

Training for those working with, but not on, an agile team can be much more targeted and shorter. The approximately one-hour course at www.ScrumFoundations.com could be sufficient for many stakeholders.

3 Use Terms Consistently

Agile uses its own vocabulary, which can be difficult for those new to working with an agile team. Worse, not all agile terms are used consistently by agile experts, authors, bloggers, and tools. It gets even worse when terms are used inconsistently within an organization transitioning to agile.

A good way to avoid these pitfalls is by creating a glossary of your own agile terms. People tend to use the vocabulary that appears onscreen in their agile software, so that might be the best set of definitions and words with which to start.

4 Establish Guidelines for What Can Be Called "Agile"

Half-hearted or incomplete adoptions of agile almost always fail. When that happens, more fervent agile adoption efforts must fight an uphill battle because people at the company feel like "agile" has already failed there once.

To avoid these false agile efforts from damaging your agile transformation, establish guidelines for what can be called agile. For example, one successful agile transformation instructed teams that could not call what they were doing by the name of Scrum unless:

- Everyone on the team had taken at least the half-day training developed by the company's agile coaches
- The team deployed at least something once a month onto production servers
- The team had a designated Scrum Master and separate product owner
- The team conducted daily standup meetings, iteration reviews, iteration planning meetings, and retrospectives

ADDITIONAL RESOURCES

Here are some resources that can help:

Free Scrum Foundations videos

The Innolution Scrum Glossary

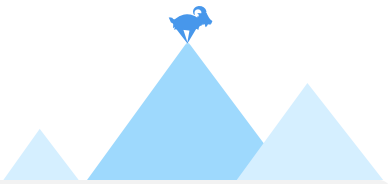
The official Scrum Guide

Mountain Goat's Public Training Schedule

Succeeding with Agile book by Mike Cohn

Advocacy

TRANSITION MANAGEMENT | IMPROVING



CONGRATULATIONS

Responses indicate that agile team members are supported by advocates among their stakeholders and leaders. Your agile advocates will help overcome future challenges or resistance.

A successful agile transformation must be supported by leaders within the organization. Agile may begin as a grassroots effort, but eventually agile team members will meet resistance. At that point they'll need support and guidance from leaders willing to advocate for the effort to become agile.

These agile champions are not necessarily on agile teams themselves. Most of the time, they are stakeholders to agile teams, departmental heads, or executives. Their role is to foster an environment in which agile can take root and thrive. To do so, they provide energy, support, resources, guidance, and occasionally direction.

SYMPTOMS TO WATCH FOR

Without advocates championing the agile effort, teams encounter obstacles they cannot overcome. Here are some symptoms that might warrant closer inspection.

1. After successfully introducing basic aspects of agility, teams are thwarted in attempts to broaden or deepen the use of agile.
2. Agile efforts among different teams are very haphazard without the coordinated approach to experimenting that would find the best ways to be agile.
3. Team members are becoming discouraged by the slow rate of change in the organization.
4. When teams encounter issues they cannot solve on their own, they have no clear escalation path.

THINGS TO TRY

The following activities can help when, and if, you notice that teams are suffering from a lack of advocacy from agile champions in the organization.

1. **Identify a Guiding Coalition of Agile Champions**
   In his groundbreaking book, Leading Change, John Kotter introduced the idea of a guiding coalition as part of successful organizational change. This applies to agile, too: form a guiding coalition of strong agile advocates. Some coalition members can come from agile teams; others should be leaders, ideally extending as high in the organization as possible.

A strong guiding coalition works to remove organizational impediments to agility. More importantly, though, it creates a culture in which people feel safe and encouraged to do that work. A guiding coalition will do the following:

- **Articulate the context for becoming agile.** Why transition to agile? and why now? are just two questions that should be answered to help people understand the context for the transition.
- **Stimulate conversation.** Adopting agile can have far-reaching effects on an organization. Those effects should be discussed.
- **Provide resources.** Some agile initiatives fail because people are expected to take on additional responsibilities but are not given time to do so.
- **Set appropriate goals.** Research has shown that change efforts with clearly defined and truly transformational goals are ten times more likely to succeed.

2 Form Targeted Improvement Communities

Accelerate the success of an agile transformation through the use of improvement communities. An improvement community (IC) is a group of individuals who work together to collaboratively improve some aspect of an organization's use of agile.

More than one IC may exist at the same time, with each IC focused on a different aspect of agile. For example, an IC may form to create a process for identifying and developing new Scrum Masters from within the organization. A separate IC may form to improve automated testing practices.

Members of improvement communities are drawn from actual agile teams. IC members remain on their agile teams, but often with slightly reduced responsibilities so they have time for the IC, and that could be as little as an hour a week. The time should be viewed as an investment that will pay back across an overall transformation effort.

Improvement communities are catalysts for improvement. Each works from its own *improvement backlog*, which is a list of desired changes or experiments. It is important that ICs do real work rather than contribute from an ivory tower. For example, that IC focused on developing new Scrum Masters might have these tasks on its improvement backlog:

- Develop training materials for new Scrum Masters
- Identify ten candidates for evaluation as Scrum Masters
- Meet at least twice a week with the new Scrum Master on the XYZ team to mentor her and assess progress
- Implement mentoring program for all new Scrum Masters based on lessons learned from mentoring the new XYZ team Scrum Master

Actionable items such as these are much better than vague or passive items like:

- Think about what makes a good Scrum Master
- Make a list of ways we could mentor the new XYZ team Scrum Master

Improvement communities should not be permanent. Once sufficient improvements have been made in an area, the IC should disband or refocus its attention to another area needing improvement.

3 Create Transition and Improvement Backlogs

With a guiding coalition and one or more improvement communities established, you should encourage those groups to create their own backlogs, which could be a transition backlog or individual improvement backlogs.

ADDITIONAL RESOURCES

Here are some resources that can help:

ADAPTing to Enterprise Agile

How Do You Get from Here to Agile? Iterate

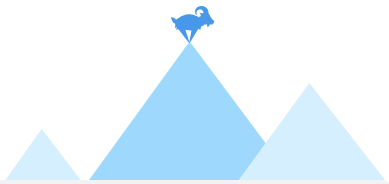
Salesforce.com's Three-Month Transition to Agile

There Is No End State When Transitioning to Agile

Succeeding with Agile

Leading by Example

TRANSITION MANAGEMENT | SUCCEEDING



GROWTH OPPORTUNITY

Responses indicate that leaders may be saying the right things about agile but they aren't backing that up with their actions. To achieve maximum benefits from agile, it is important for leaders to demonstrate that they have embraced agile principles in their work.

It's one thing for leaders to express support for agile. But leaders must also demonstrate their commitment through their actions. As an Element, Leading by Example goes beyond the support provided by leaders in the Advocacy Element. One can advocate for something–even strongly–without being an example of it. At the Succeeding level, teams need leaders who exhibit agile values, not just advocate for them.

Agile represents a significant departure for many people from their normal method of work, and when the going gets tough teams will be more likely to stay the agile course if they have witnessed their leaders act in alignment with agile values.

Teams receive an inconsistent message when leaders tell them they must respond to change but are then resistant to change themselves. This could happen, for example, when a product delivery date is no longer feasible (possibly due to changes to the product that the team agreed to) and the team is told they chose the deadline so they need to live with it.

SYMPTOMS TO WATCH FOR

Without leaders who show their commitment to agile through their actions, team members struggle to remain committed to the long-term work of truly succeeding with agile. Here are some symptoms that might warrant closer inspection.

1. Leaders say they support agile, but they work in ways that are contrary to it.
2. Pockets of agile exist, but agile rarely spreads to other teams or groups.
3. Leaders tell teams they want them to be agile but then continue to make demands from teams that prevent them from becoming more agile.

THINGS TO TRY

1. **Check Alignment Between What Leaders Say and Do**
   As a leader in the organization or in its transition to agile, your words and actions must be aligned with agile. Review both the values and principles of the Agile Manifesto to confirm that you are acting in accordance with it.
2. **Incorporate Stories of "Walking the Talk"**

To lead by example, it is important to "walk the talk." Leaders must demonstrate their commitment to agile through their actions. Stories of leaders demonstrating their own agility should be shared with agile coaches, Scrum Masters, trainers, and others who can weave them into training material and informal conversations with agile teams.

3 **Conduct a Retrospective Focused on Agile Values and Principles**

Consider conducting a retrospective focused on the values and principles of the Agile Manifesto. For each item (such as "responding to change over following a plan"), ask participants how well the organization has embraced that ideal.

A simple way to do that is ask participants to vote on each item. A scale from –3 to 3 works well, with –3 indicating the organization has not embraced the ideal, 3 indicating the organization has embraced the ideal and 0 neutral. After assessing each item, select a small number to act on first and discuss how to remedy the situation.

A retrospective such as this can be done with a broad audience, ideally everyone involved in the agile effort.

ADDITIONAL RESOURCES

Here are some resources that can help:

Leadership Agility by Bill Joiner and Stephen Josephs

The Role of Leaders on a Self-Organizing Team

Scrum Alliance's "How to be an Agile Leader - in any role"

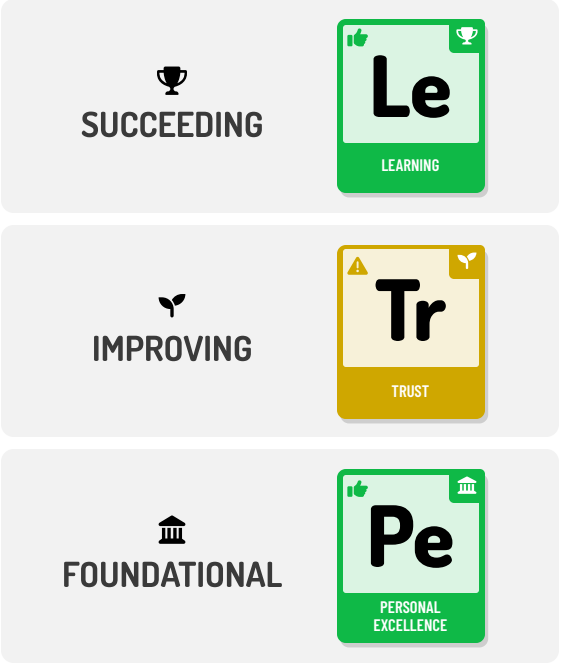
Forbes' "15 Qualities that define an Agile Leader"

Culture

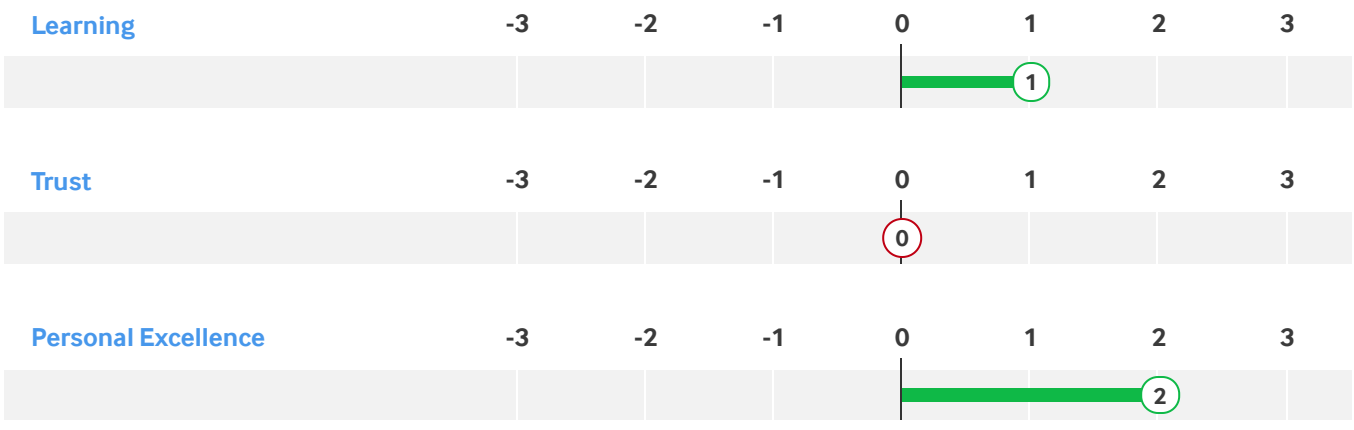
Organizations that realize the most benefits from being agile create a culture where learning and sharing knowledge is the norm.

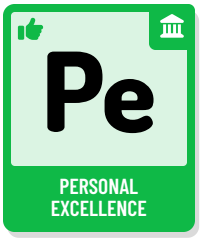
It's not passive. You'll need to deliberately design teams that have the resources, outlook, and goal of energetically pursuing and sharing better ways of working. The most effective teams and their leaders take a very active role in optimizing the rate and significance of learning, and of reinforcing its importance.

It starts with a commitment to Personal Excellence, continues through Trust, and is reinforced through the active pursuit of Continual Learning, from the individual through executive levels.



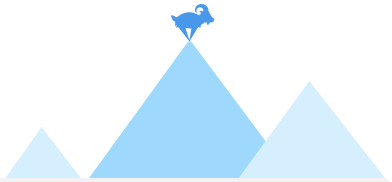
RESPONSES





Personal Excellence

CULTURE | FOUNDATIONAL



CONGRATULATIONS

Responses indicate that team members understand the importance of pursuing personal excellence in their work. They seek to deepen their current skills while learning new ones.

Excellent teams are composed of individuals who take pride in a job well done and who continually seek to improve. To see why, imagine a jazz quartet and some problems it could encounter:

- The trumpet player wants to be heard over all other instruments and so plays louder and at all times
- None of the musicians has ever played their instrument before
- The musicians don't listen or watch one another and each plays a different song
- Some of the members refuse to learn new songs

Personal Excellence is a foundation of agility. Individuals must care about the quality of their own work. To create a high quality product or service, the work of each contributor must be high quality.

Members of a good agile team seek to deepen current skills and learn new ones. They willingly embrace new ideas and new ways of working.

Leaders of agile teams understand that time spent learning a new skill or deepening an existing one is an investment in future productivity.

SYMPTOMS TO WATCH FOR

When team members neglect to pursue excellence in their own work, the team as a whole will struggle to deliver great results in the time they could have. Here are some symptoms that might warrant closer inspection.

1. Individuals are content with their current skills, neither seeking to deepen nor expand them.
2. New technologies or practices are rarely introduced by team members.
3. Team members see their work as "just a job" and have little motivation to create better products or solutions.

THINGS TO TRY

The following activities can help when and if you notice that team members are struggling to pursue personal excellence.

1. **Praise the Right Behavior**
   Seek opportunities to praise behaviors you would like to propagate. Try praising team members for any of the following:

- Beginning to learn a new skill
- Demonstrating progress with a new skill
- Surpassing expectations in some way such as delivering early or with higher quality
- Admitting a mistake and learning from it

2 **Provide Access to Training and Personal Development Materials**

If you want team members to improve at their jobs, it's imperative they are provided with the resources they need. Minimally, that means time. People need time to read blog posts or books, watch educational videos, participate in training courses, and practice new skills.

Ideally, set a budget to purchase books, subscriptions, videos, training, conference registrations, and more.

3 **Encourage Cross-Training Within Teams**

The preference in agile for cross-functional teams leads to the misconception that everyone should be a generalist. (See the Cross-Functional Teams Element for more on this.) Agile stops short of forcing people to become generalists, but team members do benefit from learning new skills from others on the team, gaining:

- Greater understanding of and empathy for the work done by others. In some cases, how one person does their work affects the ease with which the next person can do theirs. The more the first person knows about the work of the second, the more they'll consider that when making decisions.
- Ability to help when needed. At some point, any given skill will be in short supply during an iteration. Programmers may bottleneck this iteration. Next iteration it may be testers or designers. Cross-trained team members should not be expected to possess the full skills of another discipline. But they can often help with necessary but simpler tasks.

4 **Set Appropriate Expectations Around Professionalism**

A professional always does everything necessary to complete a job. An amateur sometimes chooses only the fun parts. Consider the amateur golfer who hates to putt (and is probably not very good at it) who picks up the ball when it's a foot from the hole. An amateur can do this. A pro cannot.

A good agile team is made up of individuals who do all parts of their job, even the boring or unpleasant. Software engineers are known to disdain writing documentation. Yet on some projects some amount of documentation may be necessary. A professional may not enjoy documenting but does so knowing it's necessary. An amateur may instead put the work off, hoping it gets forgotten.

Teams who embrace personal excellence act as professionals, doing everything required.

ADDITIONAL RESOURCES

Here are some resources that can help:

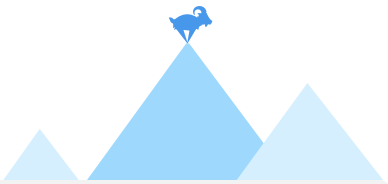
[The Difference Between a Professional and an Amateur]

[Eight Rules For Personal Excellence]

[Managing Yourself: The Paradox of Excellence]

Trust

CULTURE | IMPROVING



KEEP GOING

Responses indicate some initial success with Trust. Take a moment to appreciate that, but keep moving forward and improving at this Element.

On agile projects, team members work more closely with one another than on traditionally managed projects. This makes trust vital. Team members must be able to rely on one another to do what they say, do it well, and do it on time. When crises arise, team members need to trust one another to share bad news and be open to discussing possible solutions.

Beyond trusting one another, members of an agile team need to trust their leaders and stakeholders, who must demonstrate their trustworthiness.

When trust is absent, people are reluctant to share information and then teams make uninformed decisions. Morale isn't optimal without trust. People are justifying or defending themselves when they could be making products.

SYMPTOMS TO WATCH FOR

Without trust, teams move more slowly and are more prone to bad decisions and mistakes as information is withheld, or honest opinions are not shared. Here are some symptoms of distrust that might warrant closer inspection.

1. Individuals show a lack of concern when they fail to meet expectations or a commitment.
2. Team members frequently blame one another for problems.
3. Team members spend an inordinate amount of time trying to prevent future criticism or responsibility (that is, CYA behavior).
4. Teams want excessive detail or exact instructions so they can avoid blame in the future.
5. Team members tend not to change their minds because doing so might be considered a sign of weakness. Experimentation is stifled.

THINGS TO TRY

The following activities can help when, and if, you notice that teams are suffering from a lack of trust.

1. **Discuss Missed Commitment in Retrospectives**

If team members are frequently missing commitments they make, consider discussing it during a retrospective. Don't do this for every missed commitment–commitments are not generally meant to be guarantees. Rather, do it when the problem is happening repeatedly and predictably.

2 **Encourage Giving Everyone the Benefit of the Doubt**

Individuals need to act in ways that make them trustworthy. But everyone should strive to start from a position of trust rather than distrust.

Norm Kerth, often considered the father of project retrospectives, asserted what he called the prime directive (a reference to Star Trek). His prime directive asserts that "we must understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand."

3 **Exhibit Openness to Dissenting Views**

Trust is harmed when someone is criticized for sharing a dissenting opinion. In meetings or team discussions, be sure team members are respectful.

4 **Model Trust**

Consider your own behavior. Are you doing anything that encourages distrust? Are you doing anything that could be misinterpreted to indicate distrust?

5 **Focus on Collaboration Even In the Absence of Trust**

Professor Per Hugander points out that while trust is important, it is not a prerequisite for collaboration. He recommends focusing on "getting the conversations right" as a way of building trust. Do this through the techniques above.

ADDITIONAL RESOURCES

Here are some resources that can help:

The Three Elements of Trust

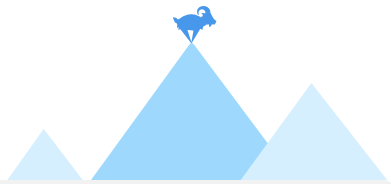
Build Trust Between Teams with Ambassadors

Distributed Teams: Build Trust Through Early Progress

Trust--The Key for Successful Delivery Using Agile Methods

Learning

CULTURE | SUCCEEDING



CONGRATULATIONS

Responses indicate team members work in a culture that encourages learning, knowledge-sharing, and experimentation. Team members can admit mistakes and grow by learning from them.

Truly high-performing agile teams proactively seek new ways to learn and share knowledge. Some learning occurs naturally—a user tells the product owner that she likes how a feature behaves or a programmer discovers that scalability needs cannot be met using a particular technology.

Other learning is sought deliberately. Rather than passively waiting for learning to occur, the most effective teams and their leaders take a very active role in optimizing the rate and significance of learning.

When teams seek to learn, some of their efforts will be ineffectual. Those efforts should be seen as necessary steps rather than as mistakes. Let's consider a songwriter, for example: a three-minute song isn't written in three minutes. The songwriter will experiment with different chord progressions, tempos, lyrics, never thinking of an abandoned chord as a mistake. It's viewed instead as a step toward the ultimate goal of a new song.

SYMPTOMS TO WATCH FOR

In organizations with a low emphasis on learning, knowledge is gained and shared much more slowly. Here are some symptoms that may warrant closer inspection, perhaps indicating that learning is not yet embraced as a cultural value.

1. Team members are afraid to show weakness or be perceived as weak.
2. An attitude of "not invented here" or "that's how we've always done it" pervades the culture.
3. New methods or practices are rarely, if ever, proposed or adopted.
4. The team rarely examines their process.

THINGS TO TRY

1. **Design Teams for Learning**

   Managers in agile organizations often determine who will be on each team. They should use this responsibility to combine individuals who will make a team that's more than the sum of its parts. These individuals should be diverse enough that new, creative ideas are generated, but not have so many differences that the team fails to jell.

The best thing a manager can do for a team is to allow it to remain together for as long as possible; they need time to learn how to work well together. Frequently altering who is on the team forces the team to start this process over each time a new member is added or leaves. Research from Harvard University professor Richard Hackman has shown that changing as little as one person on a team every three or four years is enough to "maintain creativity and freshness."

2 Exhibit Behavior that Reinforces Learning

Team members will interact in ways they see modeled by leaders, including product owners, Scrum Masters, functional managers to whom they report, and other executives and managers in the organization. To foster the right kind of behavior, then, team and organizational leaders should demonstrate the type of learning behaviors they would like to see on their teams.

Model flexible, curious, humble behavior that encourages agile teams to act similarly and learn. They can do this by:

- Asking questions and genuinely listening to the answers
- Being accessible to team members
- Asking team members for opinions on issues above the scope of the product they're working on

3 Ensure Teams Have a Motivating Challenge

Here two Elements combine, because self-organizing teams are a Foundational Element of Agile. Teams are best able to organize themselves around a problem that is:

- Important
- Urgent
- Challenging
- Motiviating

When teams are given challenges like this, learning will be amplified out of necessity. A team cannot typically achieve an important, urgent, and challenging goal using only what they have learned already.

4 Create a Supportive Learning Environment

Organizations with a supportive learning environment foster the following characteristics:

Psychological Safety

Creating psychological safety is particularly important when transitioning to agile because of the expertise shift that occurs. It's likely that certain individuals are accustomed to being viewed as experts, perhaps in the technology, the code base, or the domain. Transitioning to agile disrupts existing expertise and introduces the need for new expertise.

One of the best ways to learn is to try something, make a mistake, and then do it a better way. Other ways to learn include asking questions and engaging in debate. If someone doesn't feel safe doing these things, they won't.

Product owners, Scrum Masters, functional managers, and others must find ways to create a feeling of safety for these activities; otherwise, team members will not risk trying new things for fear of failing, looking stupid, or suffering similar repercussions.

Appreciation of Differences
Individuals on a team need to appreciate rather than attack differences. When everyone has the same background, the same skills, and approaches problems with the same style, the result can be a lack of creative thinking. Ensure team members appreciate rather than suppress differences.

Openness to New Ideas
Agile teams are often asked to meet difficult challenges: develop this faster than we've done similar projects before, do that project with fewer resources, and so on. To rise to meet these challenges, team members often have to look beyond the tried-and-true. An openness to new ideas—and occasionally to the temporary failures and setbacks this creates—is vital.

Time for Reflection
Teams need time apart from the fast pace of iterative development in which to reflect upon what they are doing and how they are doing it. Agile teams get this through end-of-iteration retrospectives.

Some teams dispense with retrospectives as a "waste of time." Ensure each team is doing a retrospective at least once a month and members are earnestly participating.

ADDITIONAL RESOURCES

Here are some resources that can help:

Continuous Learning Culture

A Culture of Improvement & Experimentation

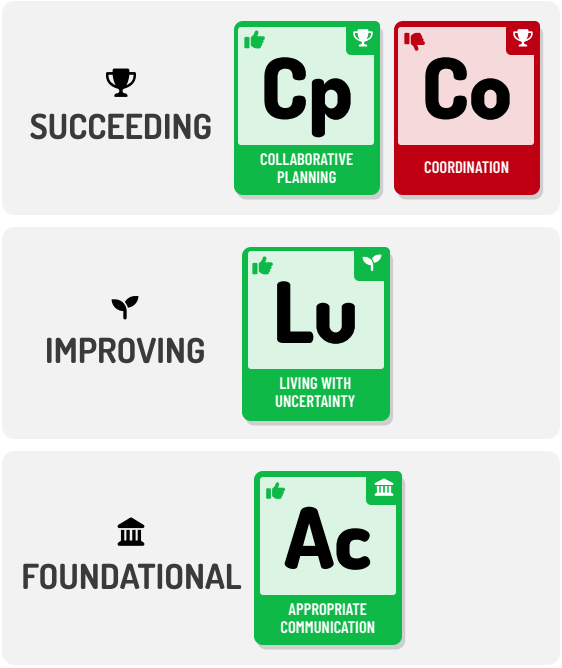
The Beatles: Get Back

Collaboration

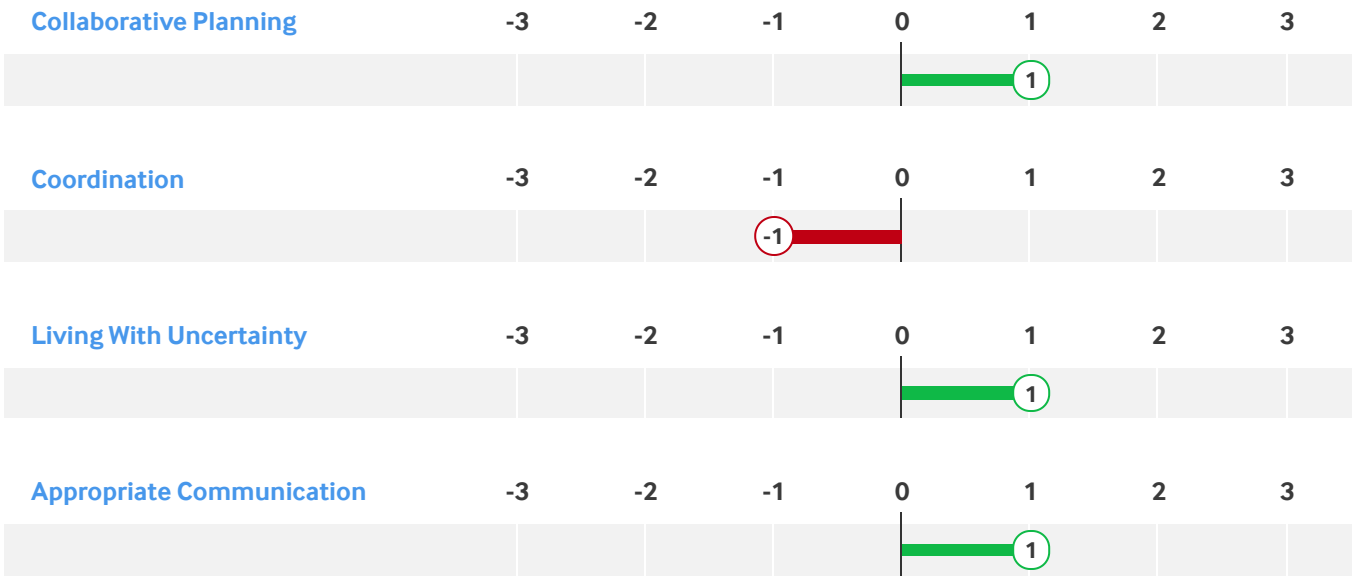
The most successful products happen whenthe technical skills of the creators combine with the customer/user knowledge of the stakeholders. And the best teams know that the way to make excellent products quickly and well is to accept two important truths: not everything can be known upfront, and the details of every product will emerge over time.

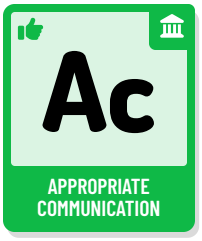
These truths point to the reason collaboration is such a central tenet to succeeding with agile. Since new information flows all through product development, discovery and communication must drive every activity inside an iteration, from daily meetings to end-of-iteration reviews to backlog refinement to planning.

As you grow more comfortable with working collaboratively, you'll experience several Elements: Appropriate Communication, Living with Uncertainty, Collaborative Planning, and Coordination.



RESPONSES





Appropriate Communication

COLLABORATION | FOUNDATIONAL

CONGRATULATIONS

Responses indicate that you have established appropriate communication. Teams are routinely conveying information in whatever means is most appropriate to everyone who might benefit from it. Appropriate communication makes it possible to reach living with uncertainty, collaborative planning, and coordination, the higher-level Collaboration Elements.

Teams that succeed with agile set expectations and protocols for appropriate communication, and find the right balance between documentation and conversation.

In different circumstances this could entail face-to-face conversation, email, telephone, video chat, message, written document, telepresence, and—we hope—even newer technologies.

SYMPTOMS TO WATCH FOR

Whether appropriate communication is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. **Misunderstandings are common.**
When teams are communicating poorly, the likely symptoms include mistakes, rework, and items falling through the cracks. Many times, this is because teams are relying on written forms of communication to share ideas, needs, learnings, and decisions.

   The problem is that written words (whether they're emails or product backlog items or anything in between) are quite easy to misstate, misinterpret, or misunderstand. Some people aren't great writers and some aren't great readers. That's why it's key to supplement documents with conversations, so that people can ask questions and clarify key points.

2. **Team members rarely talk to one another.**
A further indicator of poor communication is when team members go extended periods without talking to one another. If team members can go a week or more without talking to one another, it's unlikely that they feel the shared purpose that enables them to behave like a team.

   In general, team members need to communicate daily. This can happen at a formal daily meeting, such as the daily scrum, but let's hope it happens more organically throughout the day as well.

3. **The team creates almost no documents.**

On the flip side of teams who rarely talk: teams who talk often but rarely document. Some teams have a disdain for documentation. This attitude likely stems from a misreading of the Agile Manifesto, which says that we are to value working software over comprehensive documentation. Some teams misinterpret this to mean, "We value working products so highly that we don't write anything down."

Working software is the most valuable thing, and should be prioritized, but teams need some documentation. If teams fail to document important decisions, they can later end up with faulty implementations. People come and go in an organization, and even team members who stay forever won't have memories accurate enough to do away with all documentation.

Agile teams do not value working software to the exclusion of any documentation. Documentation should be viewed as part of an overall communication strategy. Some things should be talked about, some things should be written, and some should be both.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1. **Formalize Communication Agreements**
   Clearly identify expectations on which forms of communication to use in each situation. In many cases, more than one approach would work well. For example, a short video call or telephone call may be equally suited for coming to agreement on the last few details of a decision. Email or messaging would be better, though, for simply confirming agreement.

   A full written document is sometimes the best choice if the details of that agreement need to be communicated beyond the decision-makers or saved for later reference. This could be the case, for example, with an important design decision.

   Formally agree that when documents do exist they are supplemented by conversation, and then specify how and when that conversation might take place. It is too easy to misinterpret, misstate, or misunderstand the words in a document.

2. **Seek to Communicate at the Right Time at the Right Level of Detail**
   Paying attention to timing and detail will result in the most effective communication. If friends tell you they'll visit one year from today and arrive at 4 p.m. that is too much detail at the wrong time. With that much advance notice, you'll likely either forget what time they're arriving or feel the need to confirm the time a few days in advance.

   Timing matters. Be deliberate about both how your team communicates, and when.

3. **Encourage Conversations, Documenting Only as Necessary**
   Most teams write too much and talk too little. Good agile teams shift from documents toward discussions and other lighter mediums of communication, without eliminating documents completely. Here's a useful exercise to determine which documents will continue to be written and which could be replaced with other methods of communicating.

Grab some sticky notes, index cards, or an account on white-boarding software such as Lucid Spark. On each card (whether physical or virtual) write the name of one document the team produced on its last project. You'll have cards such as:

- Meeting minutes
- Software requirements specification
- Design decisions
- Test plans
- Test results
- UX mockups

and many more.

Then sort the cards into four groups:

- **Critical**–We could not have delivered the product without this document.
- **Useful**—This document was nice to have, probably helping us avoid miscommunications that would have delayed delivery.
- **Possibly Useful**—This document did not help this time, but it seems like it could have helped and may help in the future.
- **Unnecessary**—We could have delivered this type of product without this document.

Instead of sorting, you could annotate each card with four stars if critical, three stars if useful, and so on. This has worked for some teams who prefer to organize the cards sequentially, with the earliest document on the left and the latest on the right.

With cards grouped or annotated with stars, agree to stop producing any unnecessary documents.

Strongly consider eliminating those documents that were deemed "possibly useful." Consider whether the cost the team will incur producing those documents is worth the benefit they may provide. More often than not, producing those documents is tantamount to buying insurance against paper cuts.

For documents the team labeled useful, discuss each and see if that type of document could perhaps be shortened or simplified.

ADDITIONAL RESOURCES

Here are some resources that can help:

Writing the Product Backlog Just in Time and Just Enough

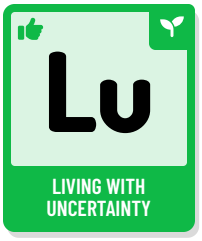
Upfront Thinking Is LIke Insurance

Miscommunicating with the Written Word

Nine Questions to Assess Team Structure

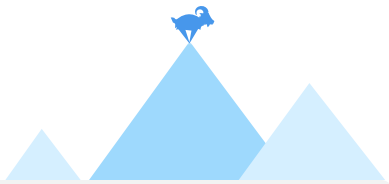
Distributed Teams: Build Trust through Early Progress

Agile Team Building: 25 Questions to Help Get to Know Your Teammates



Living with Uncertainty

COLLABORATION | IMPROVING



CONGRATULATIONS

Responses indicate that teams are accustomed to Living with Uncertainty. They know that on any project, not everything will, or can, be known upfront. The details about a product emerge over time. Living with Uncertainty makes it possible to reach Collaborative Planning and Coordination, the highest level Collaboration Elements.

With so much uncertainty everywhere around us, it seems like we'd be comfortable with it. And you'd think that teams working with agile frameworks would be used to not knowing. But many are not.

Uncertainty can be uncomfortable. Getting a team to accept uncertainty rather than attempt to eliminate it is necessary to succeed with agile.

SYMPTOMS TO WATCH FOR

Whether Living with Uncertainty is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. Team members expend significant effort in story writing workshops, refinement, or iteration planning meetings in an attempt to capture all details prior to starting work.
2. Team members want to lock down requirements rather than embrace the inevitable changes that result from showing a product to users. Teams in this situation often bring an excessive number of spikes and research stories into an iteration.
3. When a team is uncomfortable with uncertainty, iteration planning and product backlog refinement meetings will often take excessively long, as the team attempts to drive out all uncertainty. Those agile planning meetings are meant to confirm that enough is known to proceed, not that everything is known.
4. Team members who are uncomfortable with uncertainty don't just struggle when planning. They also may be reluctant to incorporate feedback from iteration reviews. This might show up as being slow to pivot from a solution that isn't working. These teams might also push back against requested changes, or even refuse to make them at all.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this

element.

1 Understand Estimates Aren't Guarantees

Start by ensuring there's an appropriate organization-wide understanding that estimates are not guarantees.

Remind people that every estimate is associated with an often-unstated probability of that estimate being correct. If a team says they can write a credible competitor to Microsoft Word in a week, there is a roughly 0% chance that they can meet that estimate. If instead they estimate they can do so in a 100 person-years, that estimate comes with a much higher probability of team success.

It is important that stakeholders, project managers, and other recipients of a team's estimates understand that the more pressure a team feels to be correct, the more team members will pad their estimates.

A team can use an estimate as the basis of a commitment (or, within reason, a guarantee). But a commitment implies a much higher probability of being met. And for that, the team will likely estimate a longer amount of time. A team might, for example, estimate that the most likely duration to complete something will be two months. But if pressed to commit, they'd say three months—to be safe—when based on a two-month estimate.

Once this is broadly understood by a team's stakeholders, team members can let go of some of their desire to answer all open issues before starting work on an item. But if team members fear they'll be blamed when things take longer than expected, they will push to know as much as possible up front.

2 Validate the Need for Answers, But Defer Resolving Some Issues

Point out examples, when they occur, of a team pushing for unnecessary information.

One way to do that is to tell a team, "Yes, I can see that you need an answer to that question. But do you need the answer before you start work on this product backlog item or do you merely need the answer before you finish?"

This response validates a team member's request for more information, but helps show that work can begin without some answers.

As a practical example, consider a team building the log-in capability for a new system. Work could begin on that without knowing things such as

- How many failed attempts before a user is locked out of their account?
- Exactly what criteria constitute a strong password
- What mechanism will be used for resetting a forgotten password?

Answers to questions such as those are absolutely needed before the log-in capability can be considered complete. But a reasonable team member will acknowledge that work can begin in advance of having those answers.

3 Share the Purpose of Meetings

Remind the team of the purpose of any meetings in which you see team members pushing unduly to remove uncertainty. The goal in iteration planning, for example, is to select the approximate or right amount of work into the iteration. Doing that does not require an answer to every open issue.

ADDITIONAL RESOURCES

Here are some resources that can help:

Iteration Planning Won't Deliver Perfect Plans

Introduction to User Stories

Estimates are Best Thought of as Ranges

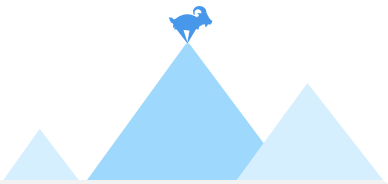
Rolling Look-Ahead Planning

Video Training: Iteration Planning

Video Training: Advanced Topics in Agile Planning

Collaborative Planning

COLLABORATION | SUCCEEDING



CONGRATULATIONS

Responses indicate that teams value Planning Collaboratively, enabling you to produce plans that both teams and their stakeholders willingly agree on.

This is in contrast to organizations in which either developers or stakeholders dictate dates and functionality and the other group is expected to acquiesce. Those situations typically result in plans that are inaccurate or simply not feasible.

Planning Collaboratively goes hand in hand with another high-level Collaboration Element: Coordination.

Collaborative Planning isn't easy. Teams need to learn to be accurate, to negotiate, and to keep plans up to date. The plan can be initiated and driven by the development team or by the business stakeholders, but it isn't complete until the other side's input is considered and incorporated.

Teams that succeed with agile combine the wisdom of those who will do the work with stakeholders' knowledge of where the project has wiggle room. Plans created collaboratively are more likely to be embraced by everyone. And plans that are accurate, with room to evolve as the team learns, are far more likely to be achieved.

SYMPTOMS TO WATCH FOR

Whether Collaborative Planning is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. **Project needs and project plans are in conflict.** This sometimes happens when a forecasted completion date is based solely on business needs. For example, the business imposes a deadline on the team that the team feels is impossible and the business feels is unmovable.

   It can also happen when a team dictates dates without consideration for what the business or client needs. The business might have very real deadlines. Sometimes these deadlines are so critical that the project itself makes no sense if it cannot be delivered on time.

2. **Plans are based on hope rather than data.** For example, the data shows that a team's historical velocity has been 20–30 points per iteration, but stakeholders insist that a plan be based on a velocity of 40.

3. **Plans are too precise to be accurate.** Teams and stakeholders who aren't used to ranges typically promise to deliver a fixed scope by a fixed date believing that precision makes their estimates more accurate. What they fail to understand is that the most accurate agile estimations use ranges.

4. **Teams burn out trying to adhere to a plan, even when things change.** Too often, a plan is made at project launch, and never seen again. Unless a team updates its plan when they learn something new, the business has no visibility into new information. When a plan is left to stagnate, the team is left trying to accomplish something that may no longer be feasible or advisable.

   Maybe an initial estimate of velocity has turned out wrong. Or perhaps a new team member was added (or removed). Maybe the team learns that certain types of work were customarily over- or under-estimated. That sort of information needs to find its way back to the plan, so that the estimates can be revised accordingly. Until that happens, the plan is at best outdated and at worst wrong.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1. **Review and Negotiate Before Sharing**
   Ensure that no plan is ever communicated and committed to before both the team and its stakeholders agree. Both sides of the development equation need to understand the importance of creating and reviewing plans together, and then negotiating a solution when needs and plans conflict.

   This does not mean stakeholders can reject a plan and force the developers to deliver more, deliver sooner, or both. It means that both parties seek a better alternative than the one in the initial plan. That could mean:

   - a later date with more features
   - an earlier date with fewer features
   - additional team members
   - relaxing a particular requirement that had an outsized impact on the schedule

   These same options should be considered when a team tells stakeholders that what they've asked for is impossible.

2. **Base Plans on Agile Estimates**
   Establish a precedent that plans will be based on agile estimates, meaning estimates provided by those who will do the work.

   No one likes to be told how long it will take them to do something. In fact, when someone outside my team tells me how long something should take or they provide a deadline, my first instinct is often to reject their estimate, even if the estimate is higher than my own would have been.

3. **Forecast Accurately and Use Ranges**
   Change the vocabulary from talking about deadlines to discussing forecasts. Accept that forecasts will often be expressed in ranges but will also be based on data, not

guesses.

Speak with stakeholders about the importance of plans being accurate, even at the expense of precision. It seems human nature to favor precision.

For example, I recently scheduled a doctor appointment for 1:25 p.m. My doctor has apparently decided his appointments should all be 25 minutes long, yet he's never once been on time for an appointment. Similarly, my $29 scale is precise to the tenth of a pound, yet it often differs by half a pound if I weigh myself twice.

Agile teams and their stakeholders also instinctively love precision. Statements like "in seven iterations we will deliver 161 story points" sounds gloriously precise. A team that can so precisely know how much it will deliver must be well-informed and highly attuned to its capabilities.

Or team members multiplied a velocity of 23 by 7 iterations, got 161, and shared that as their plan. Precise, yes. But very likely precisely wrong. What if the team delivers only 160 points in seven iterations? Do stakeholders in that case have the right to be disappointed by the missing one point? Perhaps they do, since the team conveyed 161 as a certainty.

Everyone, stakeholders and team members alike, would have been far better served if the team had conveyed its estimate as a range. A more accurate plan might have stated that the team would deliver between 140 and 180.

4 **Accept That Plans Will Evolve**

Create plans knowing they are flexible from the outset and will be adjusted as more is learned.

ADDITIONAL RESOURCES

Here are some resources that can help:

Are We Really Bad at Estimating?

Why the Whole Team Should Participate When Estimating

Paying the Cost for More Precise Estimates

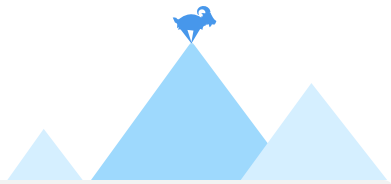
Estimates Are Best Thought of as Ranges

Free Video Training: Advanced Topics in Agile Planning

On-Demand Video Course: Agile Estimating and Planning

Coordination

COLLABORATION | SUCCEEDING



GROWTH OPPORTUNITY

Responses indicate that teams could do better at coordinating work. A lack of coordination on multi-team projects can create problems with dependencies, duplicated effort, a lack of trust, and more.

Becoming agile within a team is a start, but in many organizations multiple agile teams need to coordinate their efforts to develop a single larger product or service. Coordinating the work of multiple needs creates extra challenges for teams, including:

- Managing dependencies between teams
- Agreeing on deliverables between teams
- Preventing duplicated work
- Ensuring no work "falls through the cracks"

If these challenges are not addressed, they will impair the agility of all teams involved. But coordinating needs to be done without creating an excessive new communication burden on the teams.

SYMPTOMS TO WATCH FOR

Without properly coordinating their work, multiple teams will struggle to be agile. Here are some symptoms that might warrant closer inspection.

1. Work is duplicated because teams don't know what other teams are doing.
2. Dependencies between teams aren't detected until those teams attempt to integrate their work.
3. Team members don't feel appropriate ownership of the overall product or service they are building—they might act as if some work is someone else's problem.
4. Teams are distrustful of each other and defensive.
5. There is very little communication between teams. What does exist is often formal and defensive such as documents that must be signed or explicitly agreed to.
6. At the end of iterations, teams do not deliver a single, integrated product or service. Instead, each team delivers its own separate, discrete part.

THINGS TO TRY

The following activities can help when, and if, you notice that teams are not effectively coordinating their effort.

1. **Synchronize Iteration End Dates**

When multiple teams work together on a product or service that is too big or complex for a single team, the teams should consider synchronizing their iteration end dates. So if, for example, a product is being developed by two teams, those teams might agree that two-week iterations end every other Thursday.

Synchronizing iterations is good because iteration planning and review meetings then happen on the same day. For many multi-team products, it is particularly helpful to conduct joint iteration reviews.

Note that it is most important to synchronize iteration end dates. Teams can do that without necessarily synchronizing iteration start dates or lengths. For example, one team may do a two-week iteration while another team does two one-week iterations, with the last iteration timed to end the same day as the first team's two-week iteration.

It's generally better to synchronize both start and end dates, but synchronizing only end dates allows teams to select their own iteration lengths (within reason), which is more empowering.

2 Plan Iterations Together

When teams synchronize iteration end dates, all teams will conduct planning on the same day. On those days, teams should plan together unless timezone considerations make that impractical.

If teams are all colocated, this can be done in an appropriately large conference room. More likely, it should be done using video conferencing software (such as Zoom, Teams, or the like) and whatever supporting online tools are needed for interacting with the product backlog.

When teams that are planning concurrently discover dependencies, they can immediately discuss and resolve them. This is much better than sending an email, waiting for a reply, following up with a question, and so on, as would happen if teams were planning independently.

3 Plan Ahead for Interteam Dependencies

During typical iteration planning, each team focuses on what it intends to accomplish in the coming few weeks. Dependencies on other teams are often discovered at this time. But when dependencies are discovered during planning it is often too late to do anything other than tell the other team about the dependency and wait an iteration or two for them to provide what's needed.

The solution is rolling lookahead planning. This involves team members taking perhaps five or ten minutes at the end of a planning meeting to look ahead at what they think they'll work on in the next one or two iterations. It's just a guess, not a commitment. Having guessed at the work of the next two iterations, team members discuss those items, looking for dependencies on other teams that may exist.

To see how this might work, consider a team that has just planned its first iteration. Before they adjourn, they make a rough guess at what they'll do in iterations two and three. Suppose they identify something they'll need from another team in order to do their anticipated work in iteration three. They immediately request that work from the other team. With luck, the other team hasn't finished planning their own first iteration and can add it immediately. More likely, the other team has finished planning or is close

to done. But they can offer to develop the needed functionality in iteration two, providing it by the start of iteration three, when it will be needed.

4 **Insist on an Integrated Deliverable Each Iteration**

One of the most important things to do on a multi-team effort is to encourage teams to produce a single, integrated deliverable each iteration. Many teams, especially those still adjusting to agile, will prefer to produce separate deliverables. This is a mistake because it hides integration issues that may appear later and it also allows teams to avoid the deeper collaboration that would be required when creating an integrated deliverable.

5 **Encourage Cross-Team Communication**

On a multi-team project, one danger is that individuals become isolated, speaking mostly to others on their individual teams. The result? Good ideas are slow to propagate across the organization. Similar functionality is implemented differently by different teams. Alleviate these are related problems with communities of practices.

A community of practice is a group of like-minded or like-skilled individuals who span multiple teams and who share information and collaborate to help one another. For example, testers from five teams may form a community of practice. They would use their community to share good testing practices. Individual members would also solicit advice from fellow community members.

A set of good communities of practice can be one of the best things an organization can do to spread practices, consistency, and knowledge across a large project or even a department or organization.

ADDITIONAL RESOURCES

Here are some resources that can help:

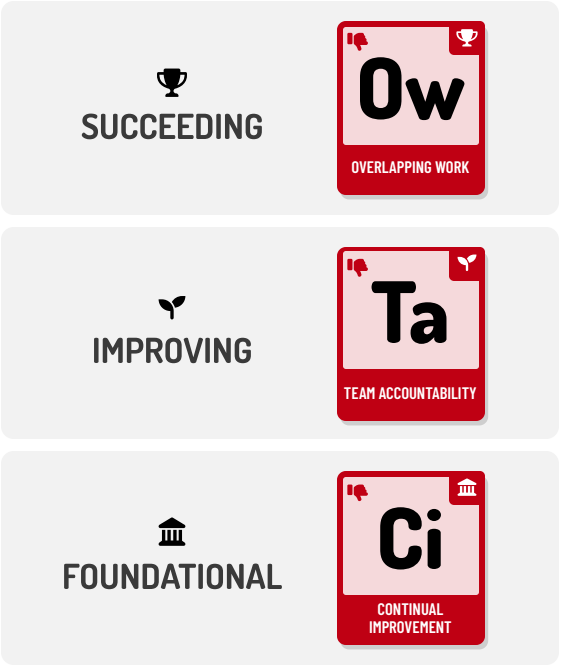
Cultivate Communities of Practice

Teamwork

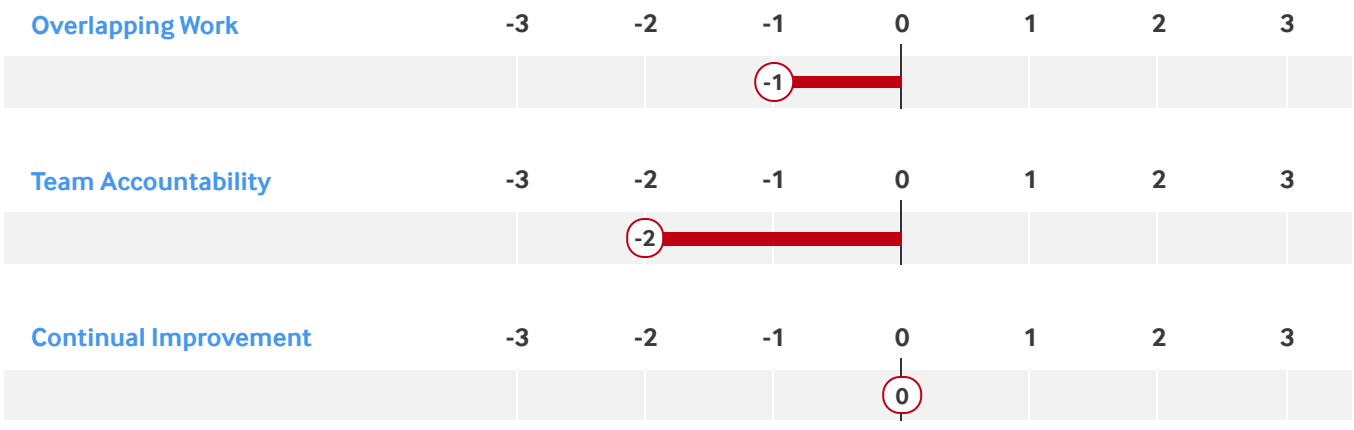
Teamwork is at the heart of every agile process. Agile teams succeed together and fail together. There is no "my work" and "your work" on an agile team; there is only "our work."

This is a radically different way of working for most people, especially those from specialty silo backgrounds or those who have developed a habit of doing only what they're asked to do. Teams that break free of this mindset rightly feel a sense of satisfaction and accomplishment.

Too many teams, however, stop improving at the point where they begin functioning as a unit, missing out on many of the advantages agile can bring. To become a truly high-performing agile team requires an ongoing concerted effort in the Elements of Continuous Improvement, Team Accountability, and Overlapping Work.

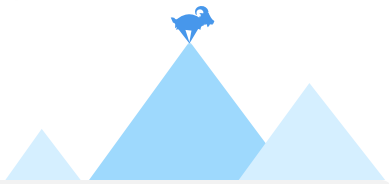


RESPONSES



Continual Improvement

TEAMWORK | FOUNDATIONAL



GROWTH OPPORTUNITY

Responses indicate that teams are not yet adept at Continual Improvement. You could use some help with making improvements a standard part of every iteration.

While it might be unrealistic to expect clear-cut improvement each and every iteration, it is reasonable to ask for an experiment each iteration. Try something new, or implement something that could potentially result in improved productivity, fewer errors, streamlined communication, or some other benefit.

When a team first forms or first adopts agile, the opportunities to improve are obvious and the improvements make a dramatic difference. It's normal for improvements to grow smaller and less obvious over time as the team matures—but a good agile team will pursue them nonetheless.

Teams that succeed with agile have a willingness to try new things. Encourage this mindset and maintain forward momentum by prioritizing process inspection and experimenting with solutions each and every iteration.

SYMPTOMS TO WATCH FOR

Whether Continual Improvement is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. **Team members feel stuck in a rut.** You can tell when a team is just phoning it in. It's often most apparent when team members come to daily scrums or retrospectives but their participation feels perfunctory rather than collaborative.
2. **Recurring issues are difficult to resolve.** Sometimes team members bring up the same issues over and over, but no progress is ever made. Maybe the problem feels too big for the team to solve. Maybe management won't be interested in solving it. Maybe it feels like there isn't the time or appetite to implement a solution.
3. **Team members are resistant to changing the process.** People are creatures of habit, so it's easy to fall into the mindset that says: If it isn't broken, why try to fix it? Making changes with no guarantee of the outcome can feel like a waste of time or a step in the wrong direction.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this

element.

1. **Measure Improvements over the Long Term**

   Remind teams that the goal is to become better when measured over a longer horizon, not every single iteration. A team focusing on continuous improvements will sometimes try something that makes them temporarily worse. That's OK. It's how the team improves over the long term that matters.

   Let's say a team decides to change its iteration length. Team members list a number of reasons to make the change, and eagerly agree to try the new length.

   But not every change is an improvement. After trying the new iteration length a couple of times, team members unanimously agree to revert to the prior length.

   For the length of this experiment this team can be said to have performed worse than it did before. They did not improve every iteration.

   Yet team members are smarter for the experiment. That learning makes up for any minor drop in performance while conducting a worthy experiment.

2. **Hold Regular Retrospectives**

   Encourage team members to reassert their commitment to holding a retrospective at the end of each iteration or at least once a month.

   Some teams forgo retrospectives with the attitude that whenever a team member notices an opportunity to improve, they'll share it. This rarely works because there are always more pressing needs and the improvement idea gets put aside. A dedicated time to discuss improvements each iteration works far better than discussing ideas as they're conceptualized.

3. **Focus the Team on Improvements They Can Make**

   Don't perpetually ask the team to fix the impossible or even the improbable. Team members will become frustrated (and understandably so!) when they raise an issue again and again in retrospectives but the situation remains unchanged.

   For example, if a new team member is needed but you've been told there's no budget for one, don't bring it up every retrospective. Instead, put a reminder in your calendar to ask about it again in perhaps a few months.

4. **Attempt Only a Few Improvements at a Time**

   Don't try to improve too many things at once. Retrospectives can generate lots of improvement ideas. A team might be tempted to take on all of them simultaneously. Don't. Instead, have team members agree on one to three things that they will make serious efforts to improve in the coming iteration.

5. **Choose Improvements People Are Ready to Make**

   Encourage team members to select only those items they sincerely wish to fix. I once worked with a team that thought they should do code inspections. Every retrospective, they agreed they'd start. They never did.

   I encouraged them to admit it was not something they were really ready to change and to focus instead on changes they were deeply willing to make.

6. **Try Process Changes for Two Iteration Before Judging Them**

Encourage teams to try new things for two iterations before deciding whether the change is good and worth continuing or not. After all, sometimes the change is so jarring that people react to the discomfort of the change, rather than the effect of the change itself.

So next time, make a rule that the team cannot discuss how the change went in the first retrospective following the change. Instead, save that discussion for after the second iteration. At that point, team members can assess the results, and decide what to do next. You may find your team embracing after two iterations a change they would have rejected after just one.

ADDITIONAL RESOURCES

Here are some resources that can help:

Overcoming Four Common Problems with Sprint Retrospectives

The Empirical Retrospective Approach

How Do You Get from Here to Agile? Iterate

There Is No End State When Transitioning to Agile

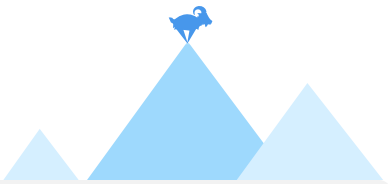
Best Practices Are Dangerous When Adopting Agile

Selecting the Right Iteration Length

Excelling with Agile

Team Accountability

TEAMWORK | IMPROVING



GROWTH OPPORTUNITY

Responses suggest that your team members could use some help establishing Team Accountability. This is normal.

Members of a high-performing agile team understand that they succeed or fail together. This level of collective ownership takes some time to achieve. Many of us are accustomed to individual assignments, with being personally accountable for stuff we've personally done.

Fostering a sense of shared, team accountability is critical for a successful agile transition. As with a sports team or a military unit, agile teams do more (and better) work when there's the mutual aid and cohesion that comes with group accountability.

Teams that succeed with agile learn how to hold one another accountable to their commitments. They celebrate each other's successes, rush to fill any gaps that come up, and have the honest conversations necessary to uncover solutions when the team is falling behind.

SYMPTOMS TO WATCH FOR

Whether Team Accountability is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. **Team members view their commitment as being to a personal set of tasks.** There will be a feeling that I have my tasks, you have yours. A programmer on a software team will be very willing to accept responsibility for his or her own code. The same programmer will likely give a confused look when told they are also responsible for someone else's code.
2. **Team members hew closely to what they know.** They may work only on areas of the product they know from the past. Team members may also be reluctant to work outside their primary skill or role.
3. **Tasks often remain undone at the end of the iteration (AKA, the team isn't meeting its iteration goal).** When team members each focus on their own goals they often fail to collaborate to ensure overall work is finished.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1. **Ensure All Team Members Share a Feeling of Personal Responsibility**

Start by making sure that every team member feels a sense of personal responsibility. If someone doesn't feel responsible for completing work that is clearly theirs, they assuredly won't feel responsible for the work of others. Scrum meetings are a good venue for increasing a sense of personal responsibility—especially the daily scrum.

During the daily scrum, ensure people state whether they completed the work they said they would on the prior day. Remind teams that members are accountable to each other and are expected to openly address issues that might prevent them from being successful.

2 **And Then Shift the Focus from Personal Responsibility**

Once personal accountability is established, remind team members that agile teams work in a collaborative manner to achieve their team goals, which requires more from individuals than personal responsibility toward a list of tasks.

Team accountability means that the team is accountable to each other and accountable as a whole. So, once you are sure that team members are owning their own work, begin to shift the focus of meetings like the daily scrum from individual status reports to group discussions of the team's progress and priorities. Remind the team that when one team member is behind, it's up to the other team members to help out.

3 **Stress Team Commitment During Planning**

Another opportunity for increasing the feeling of team commitment? Iteration planning meetings. At the end of each planning meeting, ask team members if they can collectively meet the goal and deliver the items that have been selected.

Stress that what is planned into the iteration becomes a team commitment. If someone will be overworked during the iteration, others will need to take on the excess work for that person. Because of that interdependency, if someone is about to accept too much work into the iteration, others are motivated to point it out. The team can then discuss ways to lighten the load on anyone who would be overextended. Alternatively, the team may decide to reduce the overall amount of work brought into the iteration.

Team accountability will always be bounded by the skills possessed by each member. A copywriter on a marketing team, for example, will be unlikely to do the job of the team's award-winning graphic designer. But are there things, like researching images, that the copywriter could do to save the graphic designer some time? If so, a copywriter who is ahead on her own work should do those.

4 **Encourage (and Allow Time for) Cross-Training and Skill-Building**

Beyond these attitude adjustments, the best practical way to increase shared accountability is for team members to broaden their skills. Support them in that endeavor by pointing out opportunities where team members might collaborate on work, or train one another.

Once you have identified opportunities for a team member to broaden their skills, ensure sufficient time is available for doing so. Team members will be justifiably frustrated if told to broaden their skills without being given time.

5 **Leave Some Tasks Free for the Taking**

Another way to increase team accountability is to refrain from allocating all tasks during iteration planning meetings. Instead, encourage team members to select items from the iteration backlog each day. This creates a more collaborative feel to the team's work.

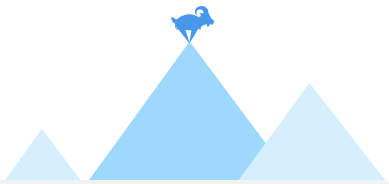
ADDITIONAL RESOURCES

Here are some resources that can help:

- Not Signing Up for Tasks in Sprint Planning
- What Is a High-performing Agile Team?
- Agile Teamwork
- What Does Scrum Mean by Cross-Functional Teams?
- How Team Members Collaborate on User Stories
- Video Training: The Scrum Repair Guide

Overlapping Work

TEAMWORK | SUCCEEDING



GROWTH OPPORTUNITY

Responses indicate that teams are not yet comfortable Overlapping Work. Teams that succeed with agile find ways to overlap work. When one type of task looks like it's dependent on another type of task, successful agile teams find ways to begin the second task before the first is completed. Doing this requires greater collaboration and teamwork.

Teams that overlap work finish more work, finish it more quickly, and achieve higher quality work.

Agile came about in part as a response to overly prescriptive sequential processes that had become common in the 1980s and 90s. Those processes were often described as a series of non-overlapping phases—analysis followed by design followed by coding followed by testing, for example.

Agile frameworks, on the other hand, encourage team members to become comfortable overlapping work; testing can begin before coding is finished. This practice is also known as concurrent engineering.

SYMPTOMS TO WATCH FOR

Whether Overlapping Work is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. **Work Is Owned by Individuals, Not by the Team as a Whole**

   When teams do not understand how to overlap work or its importance, you may notice individuals rather than the team take control of work. Then team members work on their own throughout the iteration, with very little collaboration with others.

   During daily standups, you'll know this is happening because team members talk about doing and completing tasks with very little mention of collaborating on them. You'll hear a lot more, "Yesterday, I worked on such-and-such..." rather than "Yesterday, I gave Ringo a preliminary version so he could get started" or "Yesterday, Ringo and I met about which part of this I should do first so he can get started."

2. **Iterations Are Divided into Sequential Phases**

   When teams are doing mini-waterfalls inside an iteration, they tend to create tasks for each state or stage—"coding", "testing", integrating". At the beginning of the iteration, almost all of the tasks are in one state: coding.

   Often this stems from the desire to optimize individual efficiency, which is the wrong goal.

   Let's say a tester, wanting to be as efficient as possible, would like to begin testing only after coding is complete. To test any earlier risks repeating work by re-running, or even

re-designing, tests.

This can happen with the interaction of any two roles—designers don't want to design until analysis is done and programmers don't want to code until design is done.

The intentions are good: all for the sake of efficiency. However, they come at the cost of throughput. Designing a system around the efficiency of any one role prolongs the amount of time it takes to complete each new feature.

Sequential work means fewer features will be completed over a given amount of time, such as an iteration.

3 **Feedback is Delayed and Mistakes Are Found Late in the Iteration**
Individual, phased work has a longer feedback loop, which means that mistakes made during one phase are not found until a later phase. Then the same type of mistake can get made repeatedly until those mistakes are found later and their cause fixed. Not overlapping work can also put the iteration goal at risk and cause bottlenecks at the end of the iteration.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1 **Work on Smaller Items**
Encourage team members to divide tasks. Working on small items creates overlapping work by shrinking the size of the handoffs between team members.

For example, consider the search results page on a typical apparel eCommerce site. The page allows users to filter results by product attributes such as size, color, and more.

Results can also be sorted by price, popularity, rating, and so on. If a programmer develops all of that before handing it over to a tester then no work has overlapped.

But if the programmer handed it to the tester in pieces, testing could overlap with programming. The programmer could, for example, provide the tester with a version of the page without filtering or sorting. While the tester validates the basic page functionality, the programmer develops the ability to filter by size. When that is handed over to the tester, the programmer continues overlapping work by perhaps adding the ability to filter by color.

The smaller the units of work, the easier it is for team members to overlap their work.

2 **Limit Work In Progress Plus Develop Swarming Skills and Mindset**
A second remedy is to set a very low limit on work-in-process to encourage team members to find new ways to collaborate on small pieces of work. This mindset shift may be very difficult for team members; they may have spent their careers owning some piece of work until they considered it perfect and ready for handoff.

Swarming is a useful technique that forces team members to over-collaborate on work. When swarming, team members agree to severely limit the number of product backlog items that are allowed to be in process at a time. Swarming teams will work on only one

item and finish it before moving onto the next. Some larger teams may set a limit of two items in process.

Swarming forces new ways of thinking. For example, three programmers who are working on something normally owned by just one of them would need to figure out how to collaborate without being in each other's way.

Limiting one or two items in process is an artificial constraint—it's not necessarily the optimal number of things to have in process at all times. The goal is to remove this limit later, and then team members continue to apply the lessons they learned when they were forced to over-collaborate.

ADDITIONAL RESOURCES

Here are some resources that can help:

Swarming

Agile Teamwork

Scrum Team

Working on Small Tasks (smaller than user stories)

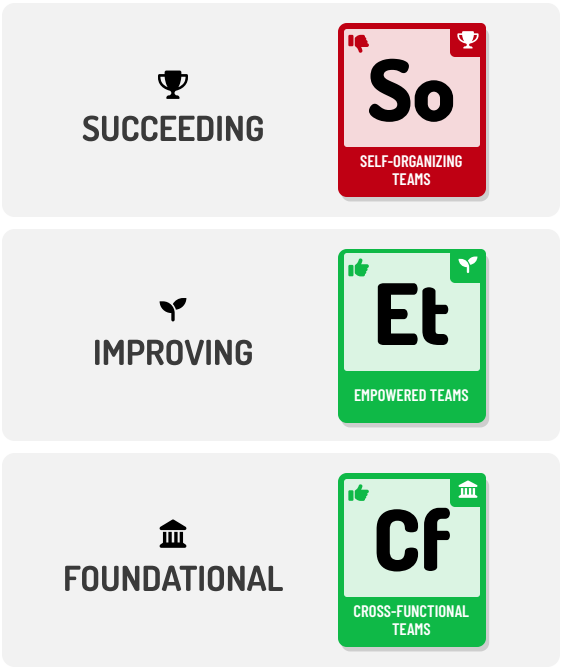
VIDEO COURSE: Scrum Repair Guide

Team Structure

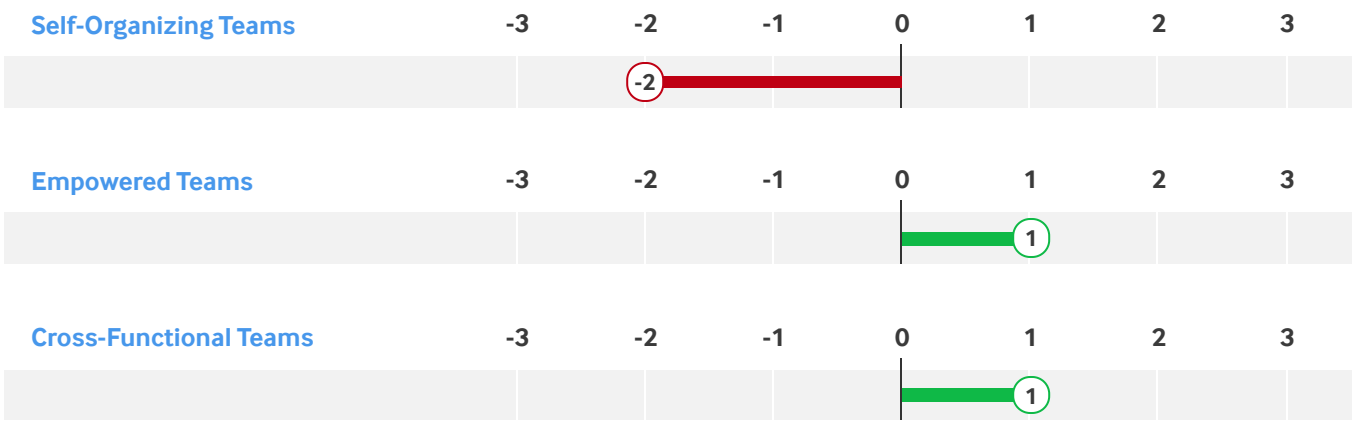
Becoming agile starts with organizing individuals into teams. Succeeding with agile requires more than just keeping teams small and orienting each team around the delivery of end-to-end user-visible functionality. How can you imbue your teams with diversity, morale, and agency?

On agile teams, everyone works together to complete the set of work they have collectively committed to complete within an iteration, regardless of their official title or preferred job tasks.Because of this, agile teams develop a deep form of camaraderie and a feeling that "we're all in this together."

The kind of high-performing teams you need to succeed with agile must be built and nurtured carefully and deliberately. Over time, they will grow from being merely Cross-functional to Empowered and, eventually, Self Organizing.

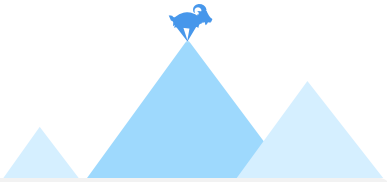


RESPONSES



Cross-Functional Teams

TEAM STRUCTURE | FOUNDATIONAL



CONGRATULATIONS

Responses indicate that cross-functional teams are in place and working well. Cross-functional teams decrease dependencies and handoffs between teams and reduce the time to value for new features or functionality., Having well established cross-functional teams makes it possible to create empowered and self-managing teams, two higher-level Elements of Agile.

A team is cross functional when it encompasses all skills needed to fully implement an idea from its product backlog. A rock band is cross functional, for example, when it includes members who can play the guitar, bass, drums and sing. In some cases, a rock band may need a second guitar player, keyboard player, background singers, and more.

Cross functional does not mean each person must have each skill. The band's guitar player does not also need to play the bass or drums. A team, just like a band, can benefit from multi-skilled individuals—the guitar player who also sings or the tester who can also design a user interface. But, a successful agile transformation requires cross-functional teams. It does not necessarily require cross-functional individuals.

SYMPTOMS TO WATCH FOR

Whether cross-functionality is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. Rather than finishing features or functionality, teams turn their partially implemented features or functionality over to other teams to finish.
2. Dependencies between teams delay time to value. Without cross-functional teams there are often dependencies between, for example, a design team and a coding team and then between a coding team and a testing team. These dependencies stretch the time it takes to deliver new capabilities.
3. Teams composed of individuals all with similar skills provide little opportunity for learning new skills. Non-cross-functional teams are formed around common skills or technologies. When everyone on a team has the same skill (programming, for example) team members do not have the chance to learn different but related skills.

THINGS TO TRY

1. **Get Functional Managers to Buy Into Cross-Functionality**

If teams are currently organized around skills (such as a coding team, testing team, and so on), you should strive to change that as quickly as possible.

A common first step in doing this is convincing functional managers to assign their people to agile teams. Functional managers are leaders such as QA, Development, Design managers and directors. Individuals with those skills report to them. They are sometimes reluctant to give up control by placing their individuals on agile teams.

To understand why, consider a QA manager, who supervises those who will validate the quality of a product. In a pre-agile organization, this manager would assign people to products but would retain the opportunity to move people as needed. That means one tester may be working on a product this week but could be replaced by a different tester next week.

In agile, the QA manager would assign people to teams rather than products. The person would be expected to be on that team for as long as necessary. The QA manager would resist the urge to shift individuals around as often as they might have in a pre-agile organization.

Because of this loss of control over their people, some functional managers may resist a move toward cross functional teams. Talk to those managers about the overall benefits of agile to the organization and their people. Educate them that they can still move people between projects but should do so more strategically (that is, less frequently).

2 **Identify and Redress Missing Skills**

If cross-functional teams have been formed but teams are struggling to complete work or are delayed waiting for other teams, you should revisit how teams are structured. When an agile team struggles with cross functionality, it is often because one or more skills are missing from the team.

Identifying missing skills can be as simple as discussing it with the team. They should be aware of what skills are absent. Alternatively, look through the team's recent work and select five to ten product backlog items the team needed outside help with. Consider each item and make a list of the outside skills needed. In doing this, you'll likely discover that adding one or two skills to the team would solve most of the problems. If skills are missing from the team, work to get individuals added to the team with those skills.

3 **Assuage Concerns that Cross Functional Teams Make Everyone a Generalist**

Many specialists spend their entire careers developing the deep knowledge required in their chosen area of work. They think working on a cross-functional team will require them to forgo deepening their expertise and instead become generalists.

This is absolutely not true. If any of your team believe this, you'll need to speak to them and assuage those fears. There is plenty of room on an agile team for individuals to specialize. A cross-functional team means the team has all of the requisite skills, not that each person has all skills.

4 **Broaden or Shrink the Team's Responsibilities**

Some teams struggle to perform as well as they might because their responsibilities are either too narrow or too broad. For example, consider a software team that is

tasked with developing and deploying its product. In many cases, that is a great idea and can be accomplished even within the short iterations used by agile teams.

However, for extremely complicated, large deployments it may be too much for a small team. Such a team would benefit from having its scope reduced to just developing the product. A separate team would then deploy the system.

In other cases, the exact opposite could be true and a team works better together if its responsibilities are extended from only developing to developing and deploying.

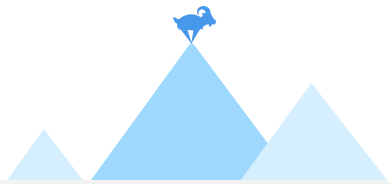
ADDITIONAL RESOURCES

Here are some resources that can help:

- Cross Functional Doesn't Mean Everyone Can Do Everything
- Nine Questions to Address Team Structure
- The Benefits of Feature Teams
- Rely on Specialists but Sparingly
- Market of Skills exercise
- Skills inventory sheet

Empowered Teams

TEAM STRUCTURE | IMPROVING



CONGRATULATIONS

Responses indicate that your team's members feel empowered to make decisions about how best to do their work to achieve the desired outcomes. Empowered teams generally move more quickly and people do best when they understand the importance and meaning of their work.

Empowering teams helps move the organization toward the highest-level Team Structure Element: Self-Organizing

Empowered teams can make decisions on their own. Managers, leaders, or other people in charge specifically transfer some of their authority to teams to make this happen, and in return the teams are responsible for what they do. This doesn't mean there are no boundaries on those decisions. For example, a leader gives the team members authority to purchase any tools they find helpful but sets a spending limit.

For a team to be empowered, authority must be granted to the team as a whole rather than to one person. From a team's perspective, nothing changes when a manager grants authority to one person on the team–there is still just one person making decisions for the team.

SYMPTOMS TO WATCH FOR

Whether Empowered Teams are part of the organizational culture or not, it helps to be on the lookout for signs that the organization might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. Teams frequently seek permission or approval before making decisions.
2. Decisions made by the team are occasionally reversed by others.
3. Teams spend a lot of time/energy in discussions or meetings before making a decision.
4. Team members are stalled while waiting for other people to make decisions.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1. **Clarify the Teams' Authority**
   Some empowerment issues arise from a lack of clarity in defining a teams' authority. If you suspect that may be the case, meet with team members and those leaders who

have delegated authority to the team. Discuss examples of decisions that should be made:

- By the team
- By the leader
- Together

Try to distill general rules from the examples agreed upon as representing an appropriate allocation of authority to the team.

2 **Further Empower Teams**

Many managers or leaders hold back from adequately empowering teams. Discuss this with team members: what types of decisions must they run by people who are outside the team? If you think the team might benefit from greater authority, discuss this situation with the leaders who may have been incomplete in empowering the team.

Try this useful argument: suggest that more authority be given to teams as an experiment. During the experiment the team is given greater authority but must inform leaders as soon as practical about any decisions based on that greater authority. If the experiment is successful, the need to quickly inform leaders can probably be removed.

3 **Be Sure It's Authority Rather than Problems Being Delegated**

In some instances, teams fail to embrace their new empowerment because leaders have pushed problems rather than authority onto the team.

Let's say a team has been told to work with two product owners. Team members raise this as a problem and are told they are empowered and should fix the situation themselves.

This is a case of a leader pushing a problem onto the team. The leader could not decide which of two people should be the product owner and so appoints both to the job. In most cases like this, the team has not been granted the full authority it would need to truly solve the problem—that is, to remove one of the two product owners from that role.

ADDITIONAL RESOURCES

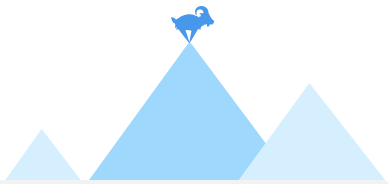
Here are some resources that can help:

Empowering Development Teams

Unleashing the Power of Small, Independent Teams

Self-Organizing Teams

TEAM STRUCTURE | SUCCEEDING



GROWTH OPPORTUNITY

Without self-organization, teams remained overly directed by managers. This leads to a team feeling less ownership of its goals. Self-organizing teams move more quickly due to a greater sense of investment in their work. Getting a team to embrace self-organizing frees managers for other work and benefits the organization and the job satisfaction of team members.

Self-organizing (sometimes referred to as self-managing) teams are a fundamental concept in agile. This means teams are responsible for:

- Executing their work
- Monitoring progress
- Defining how work is done

With these responsibilities, teams decide how to divide up work and do other management tasks that would often be done by an authority figure if it were a non-agile team.

Self-organizing teams do not necessarily have authority over who is on the team or the goals they pursue. Those are often decided by leaders outside of the team. A self-organizing team is given a goal and team members decide how best to achieve that goal.

Figuring out how to get to the goal requires the team to understand their own capabilities, gaps, capacity, and environment well enough to successfully direct themselves.

SYMPTOMS TO WATCH FOR

When teams have not been given (or are reluctant to embrace) self-organization, it will be hard to achieve the full goals of an agile adoption. Here are some symptoms that can indicate a lack of self-organization.

1. One team member has asserted him or herself as the leader of the team; others are resentful but do not speak up.
2. Team members hesitate before acting or deciding, as though they don't believe they truly possess the authority of a self-organizing team.
3. Leaders tell a team they should manage themselves, but then frequently reverse team decisions.

THINGS TO TRY

1. **Refuse to Step In and Make Decisions for the Team**
   Some teams are reluctant to embrace self organization. When told they have authority over how work gets done, these teams will often wait for someone to make decisions for them.

When you see this happen, resist the temptation to make the decision. Either wait silently for them to decide or reiterate that the decision is theirs and you'll support them in whatever they decide.

2 **Amplify or Dampen Differences Among Team Members**

Imagine a team of clones. How clones decide to organize around their work doesn't matter because each close is identical in all ways to all other clones.

Our team members are not clones; there are differences among them. These differences include technical expertise, domain knowledge, power, gender, race, education, connections to others in the company, problem-solving approach, and so on. The types and degrees of these differences influence how a team self-organizes.

When you suspect a team has not organized itself well, amplify or dampen differences among team members. For example, consider a team composed of individuals who each favor quick decision making, sometimes to the detriment of making good decisions. Look for an opportunity to add a team member with a different decision-making style.

3 **Praise Appropriate Performance**

To encourage self organization, you need to praise appropriate performance. If a team member makes a decision that would have been left to a manager in the pre-agile past, praise that team member. And, more importantly, praise the person regardless of whether the decision turned out to be the right one. Perhaps add a short discussion regarding how it might be possible to make a better decision next time. But first you'll want to commend the person for going beyond traditional responsibilities.

ADDITIONAL RESOURCES

Here are some resources that can help:

- The Role of Leaders on a Self-Organizing Team
- Two Types of Authority Leaders Must Give to Self-Organizing Teams
- Leading a Self-Organizing Team
- Placing Rules on Self-Organizing Teams
- Self-Organizing Teams Are Not Put Together Randomly

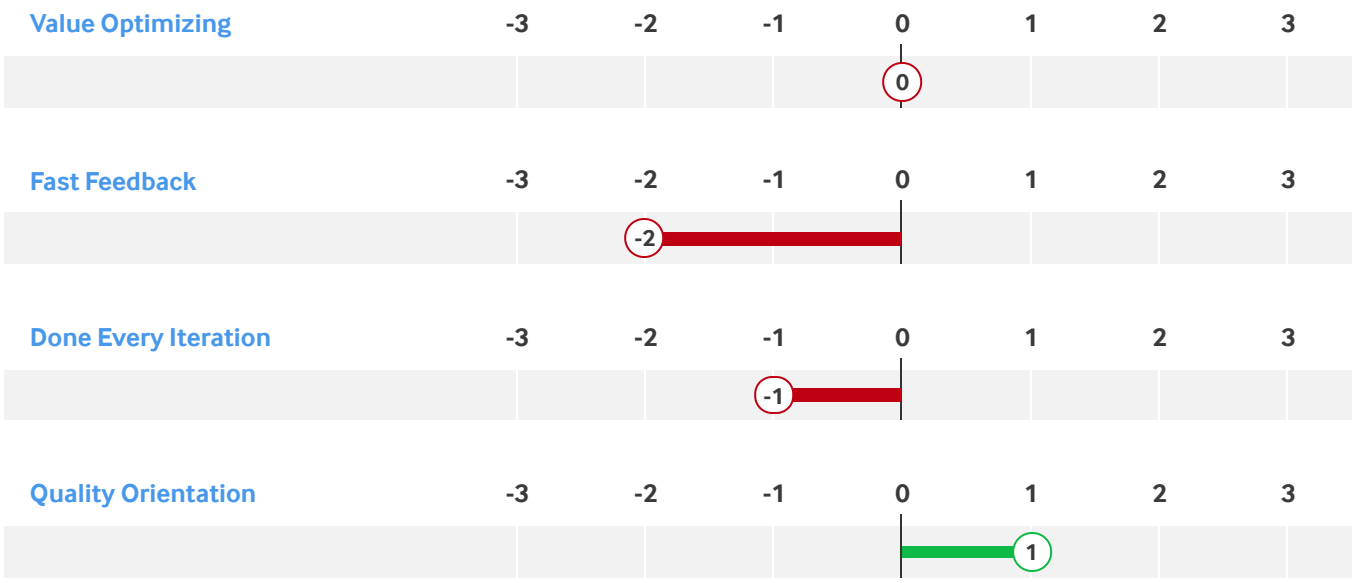
Delivery

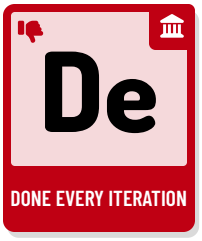
Agile teams deliver value more quickly for two reasons. First, agile teams are more productive and so produce functionality more quickly. Second, agile teams release incrementally. When stakeholders realize valuable functionality is being created every iteration, they often decide that they do not need to wait for one big-bang delivery of all functionality.

To make these small, frequent deliveries possible, teams have to get good at creating small chunks of a product each iteration that meet customer needs, deliver value, and are of high enough quality that they could be released to a customer.

That's a lot to ask, and a lot to learn. To succeed with agile, teams work to master several Elements, from Done Every Iteration to Quality Orientation and Fast Feedback, and at the highest level, Value Optimizing.

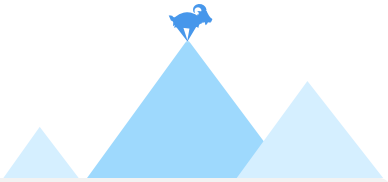
RESPONSES





Done Every Iteration

DELIVERY | FOUNDATIONAL



GROWTH OPPORTUNITY

Finishing some amount of work every iteration is important. When a team fails to do so, it is hard for them to get valuable feedback from stakeholders and to know how much progress they've made. A team that delivers every iteration also improves their morale.

Agile teams place a high value on fully implementing new ideas, features, or functionality each iteration. Rather than being partially done with a large number of things, agile teams seek to be really done with a smaller number. Why?

- It encourages teams to work collaboratively. Instead of starting something new, team members are expected to try first to contribute to finishing something already in process.
- It helps teams gauge progress. It is easier to distinguish the binary of when something has not been started or is done than to assess how far done they are.
- Stakeholders can more easily provide useful feedback on finished small things than they can on partially implemented large things.
- It improves morale as team members experience regular small wins.

SYMPTOMS TO WATCH FOR

When teams leave too much work in process or struggle to have work done every iteration, it shows. Here are some characteristics of teams that have not fully embraced working in short iterations.

1. Teams are leaving a lot of work unfinished at the end of iterations.
2. Teams show little concern for the amount of unfinished work at the end of iterations.
3. Teams try to look busy by having a lot of items in process.
4. Rather than helping teammates finish an item that is already in process, new work is started that is unlikely to be finished within the iteration.

THINGS TO TRY

The following activities can help when you notice that teams are struggling to be done every iteration.

1. **Encourage the Team to Commit to Less Work but Deliver What They Commit To**

   At the start of iterations, teams often commit to more work than they can possibly finish during the period. It's no different than a hungry person going through a buffet

and taking more food than they can eat. Our eyes are often bigger than our appetites, or the number of hours available during an iteration.

To counter this, first thank the team for being aggressive and dedicated in what they've committed to in the past. But next, inform them that they'll now pivot to more regularly finishing what they say they will, and educate them on the benefits of it.

With a team that habitually doesn't finish what they plan to, encourage committing to significantly less in the next iteration. The goal is to ensure they finish what they plan. Have them commit only to what feels extremely safe and let them add work during the iteration if all is going well.

2 **Establish Work-In-Process (WIP) Limits**

A limit on work-in-process, or WIP, defines how many items can be simultaneously worked on or in the same state. For example, a team may decide that no more than two product backlog items can be coded concurrently. In that case, a team would not be allowed to begin coding a third until at least one of the two has been coded.

3 **Emphasize that Only Finished Work Can Be Shown During Reviews**

Some teams fall into the habit of having a lot of partially finished work because they gain the psychic joy of showing off more work during iteration reviews. Discourage this by establishing a policy that only finished work can be presented/demonstrated at reviews.

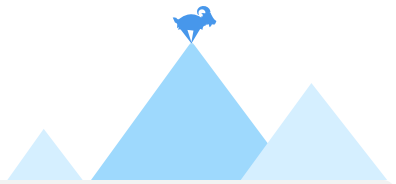
4 **Change the Iteration Length**

Some teams struggle getting something done every iteration because of their chosen iteration length. If you suspect iteration length is a problem with your team, consider changing it.

The natural inclination is to lengthen the iteration—if a team can't get things finished in the current amount of time, giving them more time is often the right response. However, sometimes reducing the length of the iteration forces team members to collaborate more intensely.

Quality Orientation

DELIVERY | FOUNDATIONAL



CONGRATULATIONS

Responses indicate that team members value quality and are pursuing practices that can create a high quality product right from the start. Vigorously enacting practices to achieve high quality from the start of a project serves agile teams well as they strive for improvements in other Elements.

In 1979, author Philip Crosby proclaimed, "Quality is free." By doing high-quality work, he reasoned, companies would save time fixing problems later or having to "test quality into a product."

Agile teams embrace the idea that the best way to go fast is to go well. They do this by setting high expectations for quality in each iteration. The product is tested each iteration to validate that it performs as desired and meets user expectations.

Quality is never an afterthought to an agile team. It is rarely, if ever, sacrificed to meet a deadline. Team members exhibit pride in the quality of their work, even in parts of their work that will only be visible to other team members.

SYMPTOMS TO WATCH FOR

When a team fails to set and meet high standards for quality, they create challenges for themselves. The low quality will hinder their ability to be agile. Here are some symptoms that might warrant closer inspection.

1. Quality is viewed as an afterthought that can be added into a product later.
2. Quality assurance or test personnel are added to teams after the start of projects.
3. A large number of defects are reported by users.
4. Teams spend a large portion of their time reworking items they had previously claimed were finished.
5. Customers become frustrated with the product's lack of quality, possibly even stopping use in favor of a different product or solution.

THINGS TO TRY

The following activities can help when, and if, you notice that teams are insufficiently focused on building quality into their products or solutions early on.

1. **Incorporate New Quality Practices or Recommit to Current Ones**
   Many teams have practices in place that are designed to improve quality, but teams perform them halfheartedly. For software development teams these could include

automated testing, test-driven development, pair programming, acceptance-test-driven development, and more. If you work with a software team, consider adding one of these practices. (You can add more later, but do so one at a time.) If the team is already doing these practices, discuss with them whether they could be doing them more frequently or completely.

Non-software teams can have similarly effective practices. Discuss with the team what could be done to find problems earlier. Consider devoting an entire retrospective to the topic.

2 Pay Down Technical Debt

To meet deadlines, teams often choose easy but limited solutions or take other shortcuts to solving a problem. The cost of remedying these shortcuts is known as technical debt, an idea first identified by Ward Cunningham.

As an example, a software system may be designed initially to support only non-negative values. All of the initial data is non-negative so the system can be put into use this way. But limiting data to non-negative values is a short cut. It will need to be addressed at some point, ideally before the first negative value enters the system.

Taking on technical debt can be wise, if the debt is paid off quickly. Until then, the debt can potentially cause problems and the problems associated with it can compound, which is the beauty of the debt metaphor.

If technical debt (or, more generally, product debt) is a problem in your organization, consider dedicating time to paying it down or off entirely.

Team members have an easier time committing to doing quality work themselves when working with a product they consider high quality.

3 Slow Down to Speed Up

Deliberately slow down. Extra time spent initially to ensure higher quality ultimately costs less than speeding through and having to fix issues later.

It can take time to prove this to a team (and its stakeholders). After all, it may take a few months (or longer, depending on the product) to see the downstream effect of doing higher quality work.

But do it anyway: commit to slowing down, really focusing on doing quality work for a couple of iterations. After that, you should have a good feeling for whether it's helping to improve quality.

ADDITIONAL RESOURCES

Here are some resources that can help:

Blog: Reduce Manual Test Technical Debt

Information on Automated Testing, Test-Driven Development, and Pair Programming

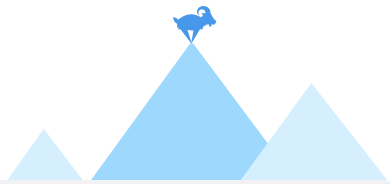
Martin Fowler's post on Technical Debt

The WyCash Portfolio Management System (Technical Debt)

McKinsey & Company's Slow Down to Speed Up

Fast Feedback

DELIVERY | IMPROVING



GROWTH OPPORTUNITY

Responses indicate that teams are not yet getting regular feedback and acting on it. Without Fast Feedback, teams can waste time and resources working on the wrong things.To create the product users need, stakeholders need frequent opportunities to evaluate the product and the team needs time to act on that feedback to improve the product. That's why it matters that the feedback be fast.

Fast feedback is key in agile product development. Without timely feedback you can't know you're building the right product. The short feedback loops in Scrum give teams multiple chances to inspect and adapt, ensuring that products are headed in the right direction.

Feedback loops validate the product and process. The goal is to receive both positive and negative feedback as rapidly as possible so that it can be acted on quickly. Fast feedback speeds up and improves the overall development process as well as preventing the frustration of futilely refining a product that was never going to meet organizational or user goals.

SYMPTOMS TO WATCH FOR

Whether Fast Feedback is part of the organizational culture or teams haven't quite achieved this yet, it helps to be on the lookout for signs that teams might be stuck or slipping into old habits. Here are some symptoms that might warrant closer inspection.

1. Reviews aren't being held at the end of every iteration. Some teams think they are saving time by conducting iteration reviews less frequently. But a small misunderstanding in one iteration can compound into a much bigger issue a few iterations later.
2. The product fails to meet the expectations and needs of the customer. The only way a team can be sure it is delivering valuable capabilities is including stakeholders—and stakeholders must be honest and available to the team.
3. Competitors release features important to our customers that we had no idea they needed. To anticipate needs, teams need to drill down and ask detailed questions. Go beyond asking whether a stakeholder likes a particular feature (or not) so you can uncover hidden customer needs.
4. Customers begin to feel as if the company doesn't care about their opinions or desires for the product. Some teams ask for feedback but don't act on it. Sometimes, teams and their product owners get so attached to a solution that they cannot hear negative

feedback from users or stakeholders. Instead, they dismiss it and push live anyway, invalidating customer concerns.

THINGS TO TRY

The following activities can help when and if you notice that teams are struggling with this element.

1. **Create Screen Recordings**

   Today's globally distributed workforces benefit from a recording to share with those who cannot participate live in the review meeting. In some cases you can record the actual review meeting. In most cases, though, you'll be better served by creating a narrated screen recording that demonstrates the new functionality.

   Creating a screen recording like this can be very quick if you do it after each iteration. After all, there isn't usually that much functionally added. A link to the screen recording can be shared with everyone or just those who could not participate live. You can request people send you feedback directly or allow people to leave feedback on the page with the video.

2. **Add One-at-a-time Feature Reviews Throughout Each Iteration**

   In addition to an iteration review meeting, you may want to consider a series of one-at-a-time feature reviews. As each feature is implemented, show it to the small number of stakeholders or users who requested it and solicit their feedback right then.

   Not everyone who participates in an end-of-iteration review needs to offer feedback on every feature. Doing one-at-a-time reviews with select audiences can often generate higher quality feedback earlier.

3. **Get Better Feedback by Asking Better Questions**

   When soliciting feedback, ask a variety of questions. In many reviews the only question asked is "What do you think?" That often generates comments such as "I like it" that are hard to act on. I like a lot of things—but a lot of things could be a little bit better. I like the tacos from my nearby taco truck, but they'd be a bit better if they were spicier.

   Consider asking more targeted questions instead, such as: Is there anything we overlooked that users might want to do here? Would any of our user types find this confusing? Do you see anything here we could leave out?

   Drill down further: When asking questions such as these, consider asking them of specific individuals rather than the full group. Be open, of course, to anyone's comment—but directing questions to a particular participant reduces awkward silences and can sometimes lead to better conversations.

4. **Ask for Feedback on Partial Solutions**

   Timing of reviews is also critical. Try to demonstrate and get feedback on features well before they are 100% done. Suppose you are developing a new capability that can only be delivered after ten product backlog items have been completed. Don't wait until the tenth and final backlog item is done before soliciting feedback.

5. **Solicit Feedback Directly from Users**

Whenever possible, strive to get feedback directly from users rather than from stakeholders commenting on their behalf. For internally used products, this can be as simple as inviting users to participate in reviews. For commercial products you might need to create a public forum on which users can share feedback.

6 **Give Teams Time to Act on Feedback**

Be sure the team includes time in their plans to act on the feedback. Nothing is more disheartening to stakeholders and users than to provide feedback only to see it ignored.

ADDITIONAL RESOURCES

Here are some resources that can help:

The Product Owner's Second Team

Getting Stakeholder Engagement Right

Four Questions to Fix Low Stakeholder Attendance

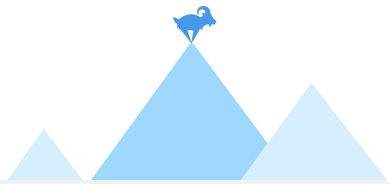
Seven Ways to Get Stakeholders to Attend

An Agenda for the Sprint Review

Only Show Finished Work at A Review-Maybe

Value Optimizing

DELIVERY | SUCCEEDING



GROWTH OPPORTUNITY

Responses indicate that some changes are needed to ensure teams are optimizing the value they deliver. As agile teams begin to deliver more quickly, it's important to ensure they are always working on the most important things.

As teams become more agile, they produce features, functions, or benefits more quickly. That's great—as long as they are delivering the most valuable features, functions, and benefits. If a team is building the wrong things, it doesn't matter how fast they go. This places Value Optimizing at the top level (Succeeding) of the Elements of Agile table.

To optimize value, teams need to work on the most important items. That does not necessarily mean the items that bring in the most revenue or have the highest return on investment (ROI). Those are great, and often appropriate goals. But other goals may be more important.

For example, broadening a platform, deepening a strategic partnership, staking out space in a new market, or generating positive publicity may all at times be more important than ROI.

SYMPTOMS TO WATCH FOR

Without an intentional approach to prioritizing work, teams are unlikely to deliver as much value to the organization as possible. Here are some symptoms that may indicate an agile team is not always delivering the most value it could.

1. Stakeholders don't understand how prioritization decisions are made.
2. Stakeholders disagree with prioritization decisions.
3. Suboptimal prioritization decisions are sometimes made after pressure from a stakeholder.
4. It is not clear how organizational objectives contribute to prioritization.

THINGS TO TRY

1. **Ensure Organizational Goals Are Understood**

   To deliver the most value, a team or product's goals must be aligned with the higher-level goals of the organization. In some cases, leaders have never clearly defined organizational goals. Once organizational goals have been established or communicated, a team's product owner or key stakeholder can identify incremental goals for the team that support them.

2 Prioritize over Multiple Time Horizons

Agile teams place a lot of emphasis on the short horizon of a single iteration. That's to be encouraged, but can lead to suboptimal value delivery.

When a team's sole planning and prioritization horizon is no more than a few weeks, items that matter but take longer can look less attractive—and get shoved to the back of the line.

To see why, consider a feature that will take five iterations to deliver. No value will accrue to the organization until some time after the fifth iteration. If the team, and especially its product owner, prioritize only one iteration at a time, delivering one-fifth of something grand will probably not look as appealing as delivering all of something that provides value after one iteration.

To get around this, successful agile teams will augment planning an iteration with planning that looks out over a longer horizon, typically a quarter or two. To do this, a long-term objective is established that will take perhaps a quarter to achieve. Each successive iteration is then planned in support of that long-term objective.

3 Deliver as Often as Possible

The more frequently a team can deliver its product or service, the more immediately it begins earning back against the investment put into it.

Additionally, frequent delivery generates more and faster feedback, which can be profitably incorporated into future plans.

4 Encourage a Formal Approach to Prioritization

A product owner's intuition or gut feeling can play a part in prioritization. But no stakeholder wants to be told their feature will be excluded because the product owner's gut said so. More important and useful than intuition is prioritizing through the use of a formal approach.

A formal approach is one that is explainable or repeatable. When a formal approach is used, it can be easier to gain agreement among stakeholders with differing needs. Agreement becomes easier because the discussions move from "which feature is most important" to "which factors determine priorities and which features best contribute to those factors."

You can create your own formal prioritization framework or choose among many existing ones. Some popular ones include:

- RICE
- Relative Weighting
- Weighted, Shortest Job First
- Kano Analysis

5 Communicate with Stakeholders About How Priorities Will Be Established

Formal approaches to prioritization succeed when you communicate to stakeholders about the approach and about how they can contribute.

Since formal approaches take more work than just trusting one's intuition, they are most frequently used for quarterly (or similar) prioritization sessions. To get the most out of those sessions, invite stakeholders well in advance. And at the start of the

session, explain how the approach works. Plan on additional time in the first meeting or two to cover that.

You should also explain to stakeholders the importance of either participating in this meeting or sending a representative. Progress is stalled if you prioritize only to have a stakeholder who missed the meeting insist that you revisit priorities using their new input.

6 **Consider the Team Members as Stakeholders**

When optimizing for value, remember that team members are stakeholders, too. It's likely their needs are not as pressing as those who pay the bills. But team members' opinions should be considered when prioritizing. This often leads to better decisions and increases team agreement with the decisions.

ADDITIONAL RESOURCES

Here are some resources that can help:

Prioritize and Optimize over a Slightly Longer Horizon

Four Steps to Persuade a Product Owner to Prioritize Refactoring

How to Ensure You're Working on the Most Important Items Each Iteration

Prioritizing Your Product Backlog

Next Steps

The Elements in this report are essential for success with agile. We know from experience that even small improvements to these Elements will bring you more of the benefits you expect from your transition to agile.

Remember the Elements are organized into three tiers: Foundational, Improving, and Succeeding. Although you can seek to improve in any Element at any time, it is generally best to focus first on Foundational Elements before moving to Elements at the higher levels.

Need Improvements Now?

There is a lot of pressure to see success from agile initiatives. People can be skeptical and resistant to making the changes that are needed to really succeed.

If you need help making any of these changes, or just don't want to do it alone, you might be interested in:

Elements of Agile - Guided Transition

We have experts on staff who have spent years helping companies just like yours. With a Guided Transition, you'll work on the impediments that are currently preventing you from truly succeeding with agile.

The Guided Transition is customized to your needs and a team of experts will help you:

- **Refine and prioritize your Improvement Backlog** - so you can make high-value changes first
- **Implement long-lasting changes** - by building improvement communities—groups within your organization that will lead your transition over time.
- **Support your teams with coaching and mentoring** - with check-ins, or focused workshops to improve specific skills like story-writing.

We don't embed trainers or coaches inside your organization on long-term engagements, but instead focus on delivering your teams the skills they need to become self-sufficient.

If you want to see how we can help you improve your agile results email us at hello@mountaingoatsoftware.com