CORE BTS

40 AGILE TRANSFORMATION PAIN POINTS

and how to avoid
or manage them

CONTENTS

CONTENTS (CONTINUED)

ABOUT THE AUTHOR

Rachael Wilterdink,
CBAP®, PMI-PBA®, CSM®, PSM

Started as a Front-end Web Publisher at Netflix (when it was a start-up) back in 1998

Transitioned to Business Analysis in 2005

Joined Core BTS in 2013

CBAP certified since 2015

PMI-PBA certified since 2017

CSM with Scrum Alliance

PSM with Scrum.org

A little bit about me. I started life back in the day as a Front-end Web Publisher at Netflix when it was a startup. It was a great experience and lots of fun. Then, over the years, I transitioned from that into business analysis through merger and acquisition.

I joined Core BTS in 2013. I have several certifications, which you'll see on the left. I also frequently speak at WI BADD and write about Agile and Scrum.

Yes, I do blame my husband for my last name, Wilterdink. His shtick is that if it's a "link, dink, sink or tink" then they're Dutch and they're from Oostburg or Hingham, Wisconsin. (It's actually pronounced "hing-um", but it's spelled hing-ham, so that's how I'm going to say it.)

OBJECTIVES

By the End of this eBook You Will Have Accomplished 3 Things:

1. You will understand the possible Pain Points you may face in transitioning to Agile.

2. You will be equipped with ways to avoid these problems when getting started with Agile.

3. You will know how to best lead Agile execution in your team and organization.

(Is This For Me?)

Who will benefit from this eBook? The intended audience is anyone who is about to embark on a transformation journey, or for those who have just started, to help you recognize and avoid potential pain points. The first set of 20 Pain Points focus on the beginning of the journey.

If you are already well-established with your Agile practices, you may have already learned many of these lessons the hard way - like I did.

The second set of 20 Pain Points focus on advanced agile transformation issues you will probably encounter once you have started on your Agile journey. They can all be read out of order, but #21–29 are Pain Points to be aware of when starting the journey, and #30–40 are Pain Points that will affect you more once you are engaged in the process and starting to execute.

What I have found is that no two organizations or teams are exactly alike. You may discover that your organization has some constraints that you must operate within that don't always seem very "Agile."

Whether you're about to embark on an Agile transformation journey or you've already begun, this eBook is for you.

For example, healthcare companies dealing with life-and-death may require more robust documentation. Another example might be consulting when you are working with external clients that still require artifacts like status reports.

The whole point is to be flexible and adaptable within your organizational boundaries.

And With That Said, Let's Get Started!

Agile Transformation Pain Point #1

UNABLE TO ESCAPE THE WATERFALL MINDSET

1

It's important when you're getting started with Agile to escape the traditional or waterfall mindset. This is crucial in being able to successfully transform.

One of the most important aspects of that is culture. If your culture is not ready to adopt Agile, then you're going to have to make some significant changes, and a lot of that is about the attitude of your company. Are they resistant to change? Are they welcome to change? If you are operating traditionally, this is typically one of the most challenging things to overcome.

Executive Buy-In

Besides that, you also need to get executive buy-in. Top down buy-in is important, but it's also important to have support from the bottom up. Most successful Agile transformations have that top-down support. It's not that you couldn't do it with a grassroots effort from the bottom up, but it really does help to have that executive leadership behind you.

Estimating Needs to Change

Another concept with Agile that can be difficult when first getting started is the way that you estimate. Estimating is no longer in hours - it's relative estimation. That's a difficult concept for a lot of people to get. Story points are meaningless to someone who doesn't have the context to understand what they are, and wrapping your head around that can be a little difficult to start.

There are a couple different ways you can estimate to help alleviate that initial pain, such as using t-shirt sizes versus a more complicated method like Fibonacci, but just understand that stories are sizes and not hours.

Metrics Must Change

Another thing that needs to change when you're adopting Agile is your metrics. You're not going to have your old-school status reports anymore. Yay! Bye-bye old traditional weekly status reports. But you are going to have new metrics to track like your team's velocity. You'll have other charts like burn-downs and burn-ups, and there are many other metrics that you might want to consider as part of your Agile transformation.

Estimates Are Not Commitments

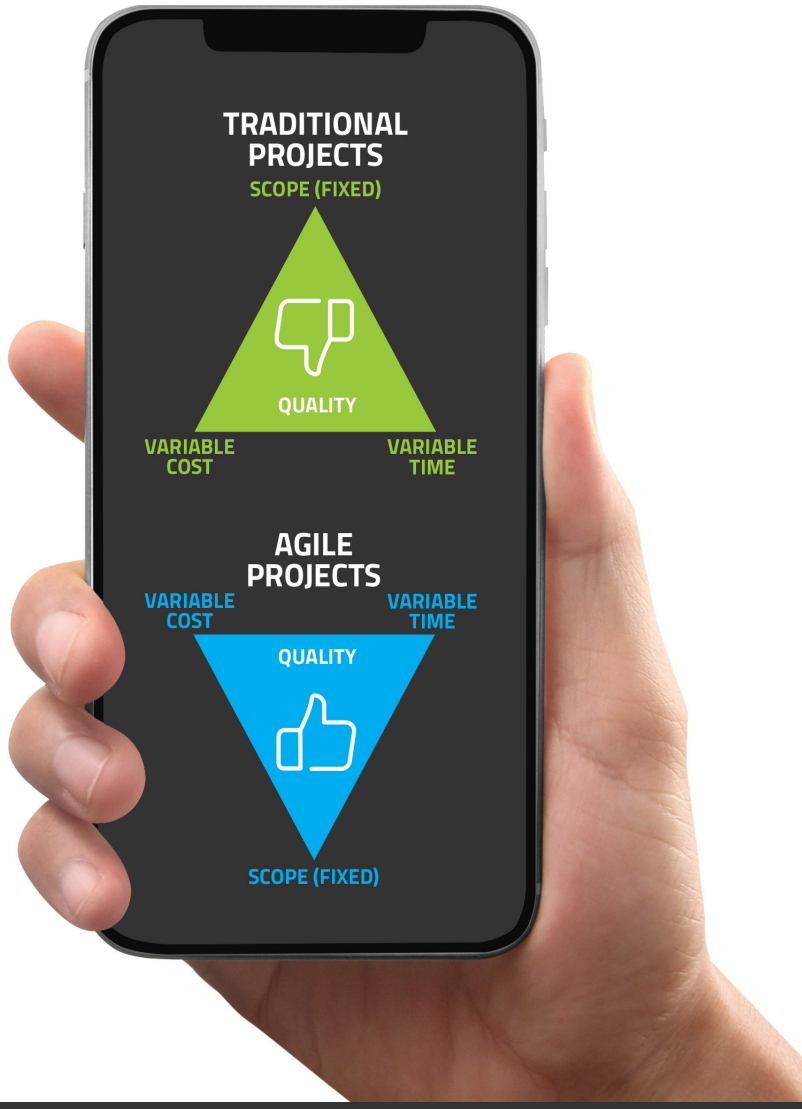
Also, you need to understand that estimates are not commitments. Going back to the point about relative estimation, estimates really are a forecast; they're not being used to make a commitment. You're not saying that it's going to take exactly this amount of time or this amount of effort. Basically, it just helps you predict what you can do in the future.

Triple Constraint is Flipped Upside Down

Another crucial point to get across is the triple constraint being flipped upside down. This is a concept that comes from traditional project management wherein the scope basically drives the cost or the budget of the project and the timeline, and so what's fixed is the scope.

Agile is completely flipped upside down. What's variable is now the scope. You pick your budget, you pick your time, and you get whatever you can get done within those constraints. The scope becomes negotiable, and that (to me) is the most important key point to get across, especially to executives who tend to hear Agile as a buzzword and think, "What a great idea!" But when they get down to it, they're not really understanding what it means. It's a significant mindset shift.

Finally, with Agile you need to be willing to make trade-offs. Agile does embrace change, but you also must be able to give things up. If your scope is variable, you would have to potentially trade out one item for another item of equal size to get something new. Or you could get to the end of your fixed time and budget and decide you want to keep going. There are a couple of options there.



Agile Transformation Pain Point #2

RESISTANCE TO CHANGE

Anyone who has been through any sort of change initiative, has suffered a major loss, or has moved from one place to another has gone through a significant change.

For many people, change is difficult, uncomfortable, and hard to deal with. There are many stages that people go through during change. I remember when I was going through the merger and acquisition that led me to becoming a Business Analyst. We were fortunate to have a pretty good change management plan where they taught us about the stages we would go through during this change, and it was very difficult.

Painful Transition from Waterfall to Agile

Even when I went through my original transition to Agile from Waterfall, it was painful. I was resistant at first (just like many other people), but over time you kind of get over that. You see the possibilities and opportunities, and finally, you accept it. But just recognizing that people are human, and that they do fear change, and that is something you're going to want to address and not ignore.

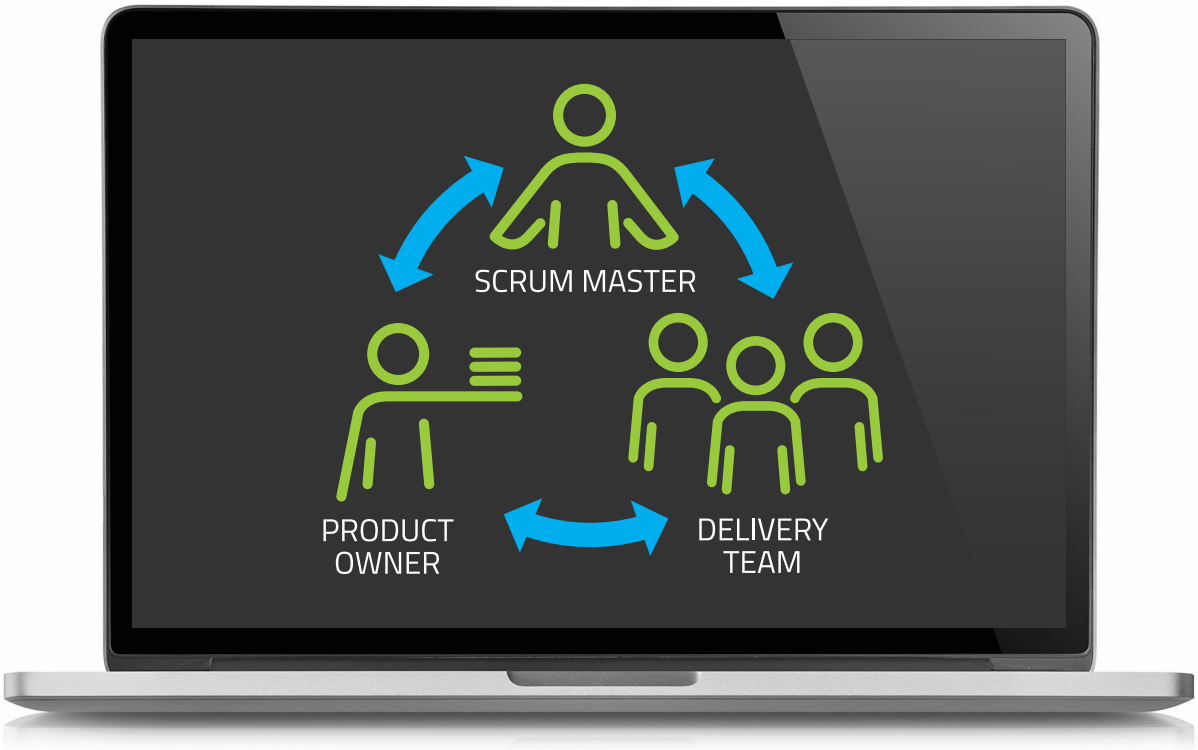
Change Threatens Management

Another thing that might be a problem, especially at the middle manager level, is that people with that role tend to be pretty threatened by this change because where is there a manager on an Agile team? There isn't one because the team is meant to be self-managing. Managers fear loss of control, and that's a legitimate fear because it's quite likely that they are losing some sort of control.

However, that does not mean they don't have a role to play in Agile. Managers would be suitable to become Scrum Masters if they can get away from the command-and-control philosophy. If they're able to become servant leaders, then they could successfully transition to Scrum, or they could become a resource manager or something like that.

Ego and Agile Don't Mix

Another thing that is important in Agile transformations is that your title doesn't matter anymore. The only official three titles on a Scrum team are the Scrum Master, the Product Owner, and Delivery Team members. If you had a lofty title before like "senior architect" then that really doesn't matter any more. You must lose your ego when you join an Agile team. It's no longer about the individual and their personal performance, it's about the team's performance. You win as a team, or you lose as a team.



Agile Transformation Pain Point #3

HANGING ON TO OLD ROLES

This is an important Pain Point because many people have a hard time letting go. Again, your title no longer matters when you transition to Agile. It's only the roles that are important and the skills that you bring to the team.

Your team needs to be flexible and should have all the cross-functional skills needed to get to a done increment within your sprints. That is something that a lot of organizations may struggle with if they have component teams today.

Combining Team Expertise

Let's say you've got a group of project managers on one team, and a group of business analysts on another team, and a group of developers on another team, and a group of QAs on another team…you need to have all those skills within your team to get to that done increment. You're going to have to break up the old teams and form new teams.

That can be challenging, but it can be done, and people can learn new things. It's not like you have certain skills that you start with and that is all you ever bring to the table. It's a great opportunity for people to cross-train and learn new things on an Agile team.

Focus on Generalization, Not Specialization

Another important aspect of letting go of old roles is to focus more on generalization versus specialization. Many organizations tend to go the specialized route, but as they transition into Agile they find this can be quite a bottleneck. They've probably already experienced the bottleneck in their Waterfall world, and it will only become more apparent in an Agile world. The more generalized your team members can be (and the more flexible and adaptable they are), the better off you're going to be.

Clarify Roles and Responsibilities

One thing I like to do at the beginning of a new Agile team is to ensure that all the roles and responsibilities of the team members are clear. I do an activity where I gather everyone into a room, get a bunch of sticky notes, and have people write down what they think they're responsible to do on the team. Then we look at the results and see if anything is missing or if there's any overlap. That really helps everyone come to a clear idea of what they're supposed to do on the team. One of those things could be to help other team members and pitch in wherever they can.

Get the Right Team Together

Another point with letting go of the old roles is to get the right team together. You want to pick the right people: people who are excited about this change, who are going to be advocates of it, and who aren't afraid. Then you need to pick the right team. The size of the team is important; you don't want it to be too big, and you don't want it to be too small. I like the Jeff Bezos rule about two pizzas: if you can't feed the team with two large pizzas, then the team's too big. I would say somewhere between three and nine people would be the optimal team size (not including the Product Owner and Scrum Master). Any bigger than that and you're just too big to be Agile.

New Role: Product Owner

As you move away from old roles, you encounter a new one in an Agile team: the Product Owner. It could be someone who used to be a product manager or it could be a subject matter expert. Whoever that person is, they need to be engaged, authorized and truly empowered (not just in name) to be able to make the decisions and have those decisions be respected throughout the organization all the way from the team members up to the CEO.

New Role: Scrum Master

Another new role is the Scrum Master. It's a servant leader role, not a command-and-control role. Some people argue that they don't like the word "master" because it might have some negative connotations. But the Scrum Master is a servant leader; they're there to help get workers out of the way of the team, they're there to coach, they're there to teach, they're there to ensure that the team is living up to Scrum values and principles, and they are not to get in your way.

Self-Managing is Key

A successful Agile team needs to be allowed to self-manage. Rather than dictating to the team what they should be doing, let the team figure out how they want to get things done. That's their job; they're solely responsible for it, and no one should be telling them the order they do things in, the way they do things, and how they're going to get the work done.

Avoid Conflicting Roles

Another key point is to try to avoid people playing multiple roles. I know that's not always possible because people usually have to wear a lot of hats in smaller companies. But if possible, try to split the different roles. For example, you wouldn't want to have, let's say, a developer also play the Product Owner.

That could be difficult, and it adds inherent conflicts of interest. Likewise, avoid including a direct manager with their reports. That could also lead to some unfortunate things happening.

Finally, the Product Owner should prioritize what work is going to be done, but they're not going to tell the development team how to do it. The Product Owner is looking at it from the view of the customer or the business, and the team is looking at it from the view of how they can get it done.

Agile Transformation Pain Point #4

POOR (OR LACK OF A) PRODUCT OWNER

Product owner is the most crucial role on an Agile Scrum Team. Choosing the right person to do this role is absolutely critical. The product owner is responsible for the vision of the product.

The Product Owner should be the one evangelizing what their product is, what value it provides, and how it's helping the company. The Product Owner needs to be trained in what it means to be a Product Owner. It's not enough to just say, "Susie over here is an expert; we're going to slap that title on her and off she goes." She needs to understand what Agile is, her role in it, and how important it is.

It Can Be Learned

Being a Product Owner can be learned, but the person needs to have a particular set of hard and soft skills. This person needs more than technical or business knowledge: he or she also needs to have a wide sphere of influence and be able to work in an organization from the highest level down to the lowest.

Product Owners Must Be Available

Product Owners also need to be available to the team to answer questions. If they're not available, they're going to quickly become an impediment. I once worked in a team where the Product Owner was completely MIA, and I (as the business analyst) was tasked with stepping in and being the Product Owner on her behalf – or what is called the Product Owner proxy. That is to be avoided if at all possible because I was not truly empowered to make the decisions that I was making, and ultimately that usually leads to things being wrong or redone.

Product Owners Must Participate in Scrum Events

Product Owners also need to regularly participate in all the required Scrum events – especially Sprint reviews. If the Product Owner isn't there, who is reviewing the team's work? Who is giving you feedback? The Product Owner is really the key participant in that meeting, and must be at all the meetings that they are required to be at.

Agile Transformation Pain Point #5

PART-TIME TEAM MEMBERS/ SPLIT RESPONSIBILITIES

5

When transitioning to Agile, avoid having part-time team members or members with split responsibilities. In a perfect world, each person on the Scrum Team would be fully dedicated to that team and nothing else.

I know that's not always reality. Every time you're switching from one task to another, you are losing efficiency. That context switching is very costly. The more things you're on, the more time you're going to lose, and time is money.

Singular Focus is Key

Team members also need to be allowed to focus on the task at hand. It's easy to get distracted by other things that are going on at an organization. The framework for Scrum and Agile is lightweight: it has just enough events so you can get what you need and then focus on your work. If you have more than one job, say you're split 50-50 between two different projects (or, in my case, I'm often split between two different clients), it's very hard to get the same amount of work done when you have to ramp up and down between the two different things you're trying to tackle.

Avoid Conflict of Interests

Dual roles create a conflict of interests. If you had a manager and their direct report on a Scrum and Agile team, that probably would not work out well. If you can avoid that, you're going to be better off.

Plan For the Unexpected

If you are supporting a production system that can take 25% of your time, you can plan accordingly. I wouldn't recommend doing it that way, but if you had to you could just plan that work in as you're planning your Sprints so it's accounted for. It's not ideal, and it would be preferable that this be handled by a separate team, but (in the real world) sometimes you have to do what you have to do.

Agile Transformation Pain Point #6

TRYING TO GO IT ALONE

Some companies have been successful in doing it themselves, but it's not recommended.

When companies go it alone, it usually goes like this: they get some education off the Internet, maybe watch some videos, read a few books, and decide to go do it. That is a great idea in theory, but a lot can go wrong. My advice would be to hire a professional coach with lots of experience who can come in and train you, get you ramped up, get you rolling, and give you advice until you become truly self-managed and operating smoothly.

Embed Agile Experts

Another option would be to hire some people who are already experts and embed them in those teams. If they already have previous experience, they're going to be able to leverage that, and the team will come together and get to that productive stage much more quickly.

Shadow Successful Agile Teams

Finally, if you already have some Agile going on in your organization, you could shadow another team that's already doing it and model your team after theirs. You could probably also ask for advice from people in your organization if they're already doing it. If that fails, there are meetups and public groups you can join to get advice and help from other peers.

Agile Transformation Pain Point #7

GETTING TRAINING, BUT WITHOUT FOLLOW-UP SUPPORT

7

When transforming to Agile, if you've had training, that's wonderful, but that alone is not a guarantee of success. You still want to engage a coach or someone who can help you through this transition. Your company also needs to provide organizational support through the transition.

I once joined a company after practicing Agile at previous organizations, but this company was Waterfall and had a project management office (PMO) that wanted to bring Agile to the organization. They spent millions to bring in a professional training company, gave everyone in IT a week-long boot camp on Agile, and then turned them loose.

However, they didn't adjust anything in their PMO to account for Agile, and they didn't support anyone who wanted to do something with their week-long training. It was a significant failure. Only a couple groups in the organization thought it was worth doing and that was only because they were in a technology space that's very fast-moving (like mobile). That was the team I happened to be on so I was a little bit lucky, but the rest of the organization wasn't.

Another option would be to get experienced Agilists if there isn't someone inside. Get a Scrum coach, get a meetup together, have lean coffee, go to conferences, and continue your learning online. There's a huge community out there of people like you who have gone through this before and have a ton of advice like:

1. Get Training

2. Get Support After Training

3. Don't Go Too Long After Training Before Starting

Research shows that if you don't directly apply what you learned within two weeks of training, then you're going to have to start all over again because you're going to forget everything you learned. Follow up quickly with a coach.

Agile Transformation Pain Point #8

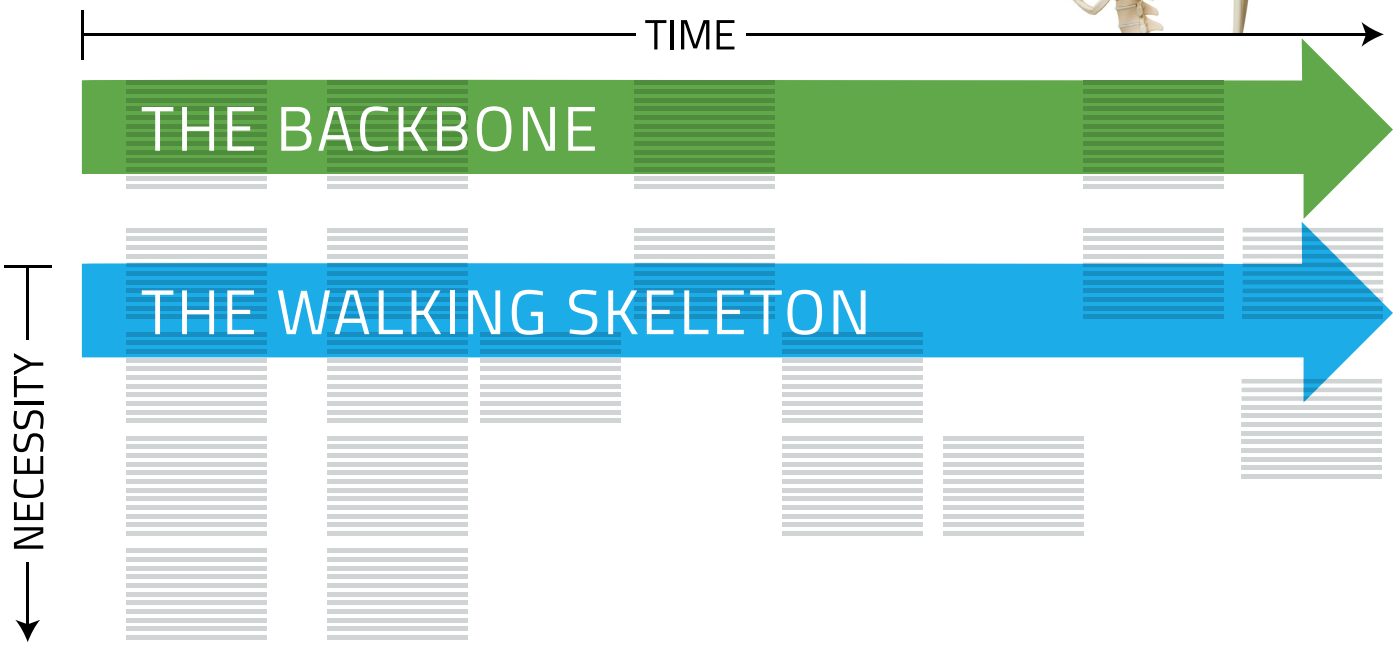
NO PLANNING / TOO MUCH PLANNING

8

There's a common misconception in Agile that there's no planning. That's not true, there's definitely planning, but it's just at different levels. There's also the concept of over-planning - which you don't want to do either. You've got to find that right balance.

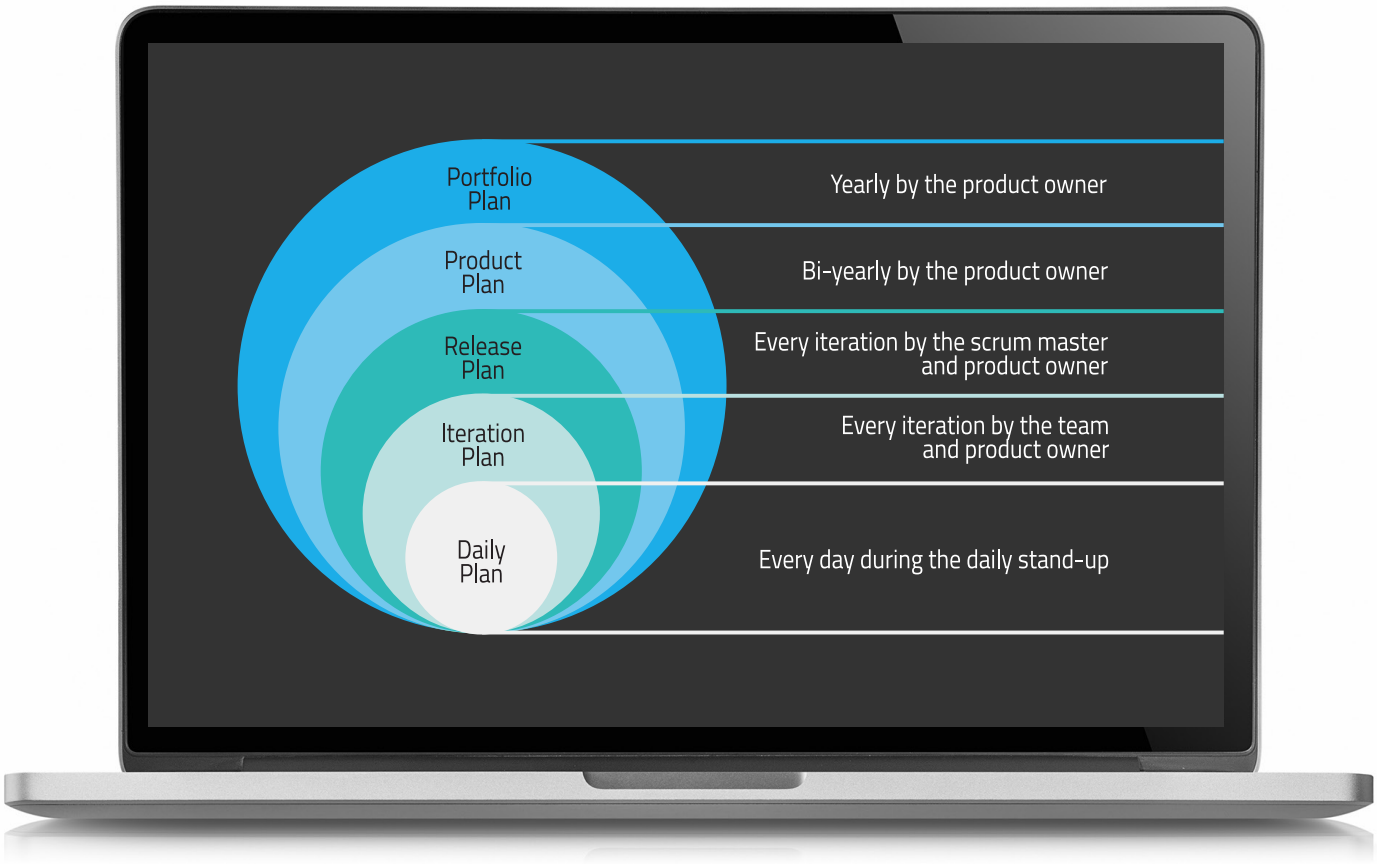
Fleshing Out the Barebones Story Map

On the image below you have a story map (a technique originally developed by Jeff Patton) with a backbone, and the basic stuff you need for Minimum Viable Product (MVP). Then underneath, you can prioritize your stories and "flesh it out". This is a nice visual tool to help you plan out your product.



The Daily Scrum Onion

Another option is the onion below. Every day, your daily Scrum standup is an opportunity to plan for that day. You're also going to plan every Sprint. Then the Product Owner and Scrum Master will work on the release plan. Above that, the Product Owner is responsible for the larger view of the overall product, and potentially the portfolio of the company as well. You can see the bottom three items on the onion diagram are where you're going to spend most of your time doing planning in Scrum and Agile.



Agile Transformation Pain Point #9

RELYING TOO MUCH ON TOOLS (AND TECHNOLOGY)

9

There are some wonderful tools out there, but you could literally just start with sticky notes on a wall - especially if you're co-located.

If you aren't co-located and are dispersed, that makes it quite a bit more challenging. But there are some tools you can leverage to make that a bit easier. Once you've got the board down using the concept of Kanban columns for your product backlog and sprint backlog, you can move the sticky notes across the board. It's fun, and there's something very cathartic about moving a sticky note to that done column (and then you can jump up and down and have a celebration).

Recommended Digital Tools

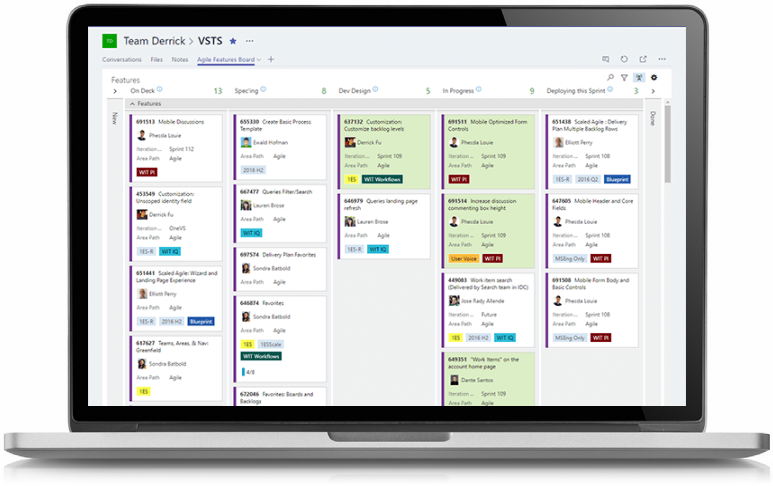
There are many great digital tools out there like Kanbanflow, Leankit, and Trello. There are also some other tools out there like Fun Retrospectives that can give you good ideas to keep your retrospectives fresh and avoid stagnating. There are also tons of requirements management tools. My favorite is Microsoft VSTS (pictured), but I've also used Jira.

Avoid Excel for Large Project Backlogs

I advise you to not do your backlog in Excel if you have a large project because it's going to get unwieldy pretty darn quick. I would look at something more robust. But if you have a small project, there's nothing that says you can't just have your backlog on a spreadsheet. Also, there's a fun tool for doing your planning poker with dispersed teams that lets you flip all your cards at the same time.

Get a Collaboration Tool

Ideally, you should also get a collaboration tool like SharePoint, Slack or Microsoft Teams. That's another key piece of technology that can help you if you're going to be Agile.



Agile Transformation Pain Point #10

STRUGGLES WITH STORY WRITING

10

Story writing struggles are so common and impactful that consultants often offer training workshops on it.

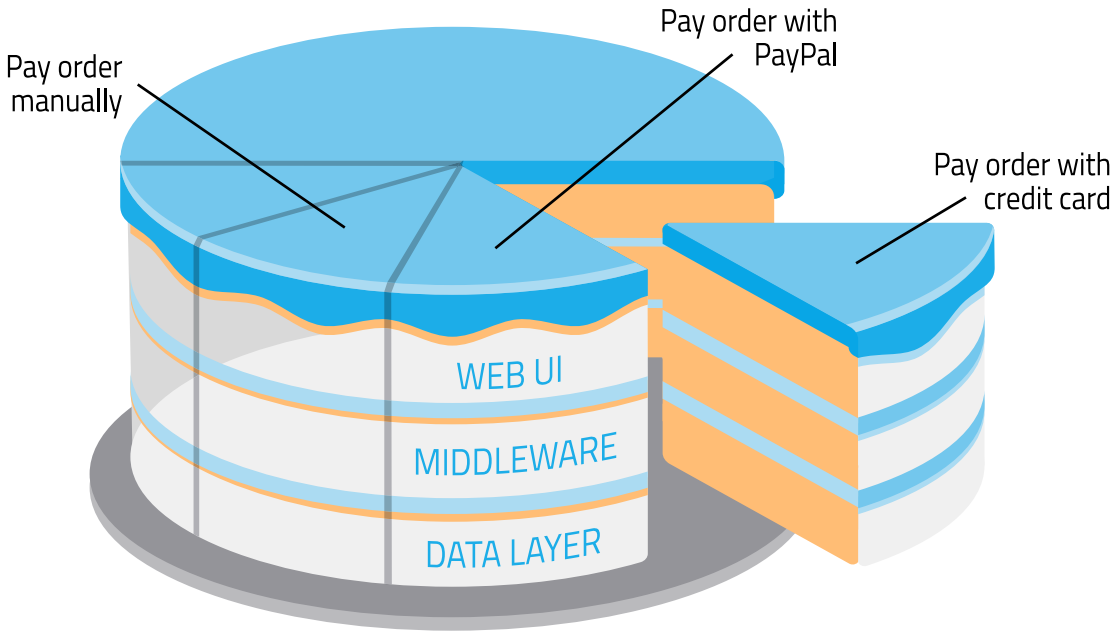
Avoid Technical Stories

Many Product Owners don't have a lot of experience writing requirements, and if you're a BA like I am coming from a Waterfall world, you tend to like to write a lot because that's what you used to have to do. But now user stories are meant to be from the user or the business perspective. I've seen some teams that tend to write stories from a technical perspective, and I would caution against that. I am not a fan of a "technical story". Unless a story has value that is delivered, and that is measurable and usable by the end user customer, then you're not delivering anything of value. Avoid those types of stories. Always try to look through the lens of your customer or your user.

Slicing the Cake

Another thing many organizations struggle with is slicing the cake (see below). In the old days of software development, you'd have to do these horizontal layers one at a time until they were all done, and you wouldn't get any value delivered until the very end.

In Agile, you're going to take a thin vertical slice of just what you need to do that action – such as pay with PayPal – and you're going to go through all the horizontal layers to get something that completely works. It just does that one very narrow function, but you have everything you need to get value out of that right away.



Don't Forget to Include the "Why"

If you've ever heard the user story format, "(as a [role], I want [something], so that [I can get some benefit])", the why is the "so that" part of the story. Many people tend to get lazy and forget to do the last part because it's not convenient. They actually have to think about why. But I caution against not including that. You really need to have the "why" for the development team. It gives them the context that helps them understand what value is provided by doing the story. Without that, they're not going to know why they're doing it, and they're probably going to ask you anyway. You might as well put it in there.

Write Stories at Just the Right Size

Finally, writing stories at the right size is also very difficult. Think about Goldilocks and her experience with the "just right" porridge. In this case, you want to have stories that can be completed within a Sprint, and it's great if you can right-size your stories so they're all comparably sized. It's not always possible, but you also don't want to have stories that are too big. Those are epics.

Epics are stories that would need to be decomposed into smaller stories to be delivered within a Sprint. An epic could never fully be delivered within a Sprint. On the flip side, you could also have stories that are too granular. In that case, you might want to consider combining some to make it a larger story.

Agile Transformation Pain Point #11

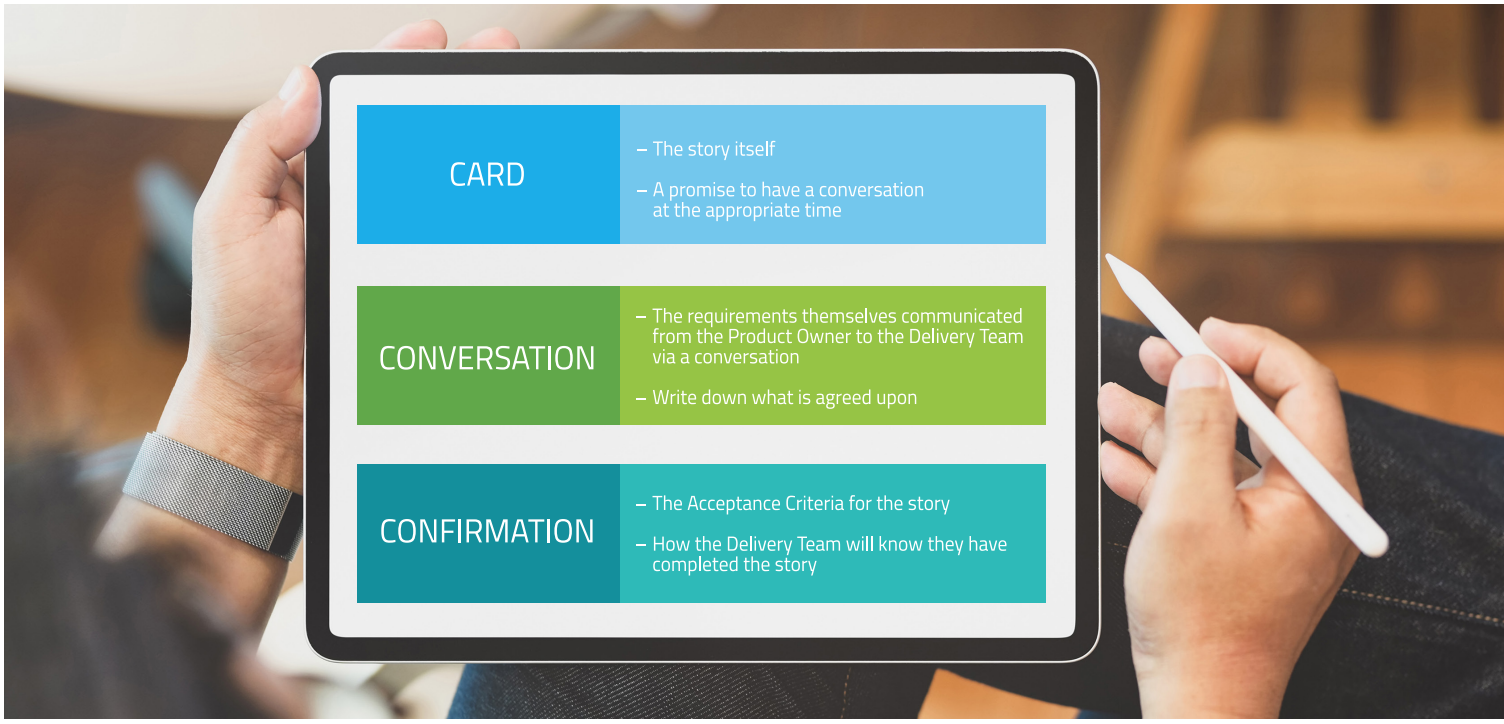
FORGETTING TO HAVE THE "CONVERSATION"

11

You may have heard the three C's of Agile: the Card, the Conversation, and the Confirmation. Many people will have the Card: they'll write the story, but they'll forget about the other parts. They'll just write it in a silo and not get any feedback. They won't actually talk to anyone about it and assume things. It's a very bad practice.

Get Actual Approval and Consult Your Team

You want to make sure you have that actual conversation, get the actual approval from your Product Owner, and make sure the team is consulted. This is meant to be collaborative, not done in a silo. Make sure you're also including acceptance criteria. That's part of your Confirmation. No user story is complete or suitable for development until you have that acceptance criteria; it provides the boundaries and how the team knows when a story is done.



Agile Transformation Pain Point #12

NOT HAVING A DEFINITION OF "DONE" OR "READY"

12

Most people who are familiar with Agile are familiar with the concept of definition of Done. This is something that the team should create when they are first formed, and they should continually be inspecting and adapting that at each of their retrospectives. They can update that definition of Done if they're finding it doesn't work for them, but they need to have one because everybody has a different idea of what Done means.

Definition of "Ready"

Ready is probably unfamiliar to many, but it is about your user stories' acceptance criteria. Are they complete? Do they have all the necessary information so developers can take that story and acceptance criteria and get to Done? Ready is really about the stories themselves; Done is about the actual delivery and development of the work.

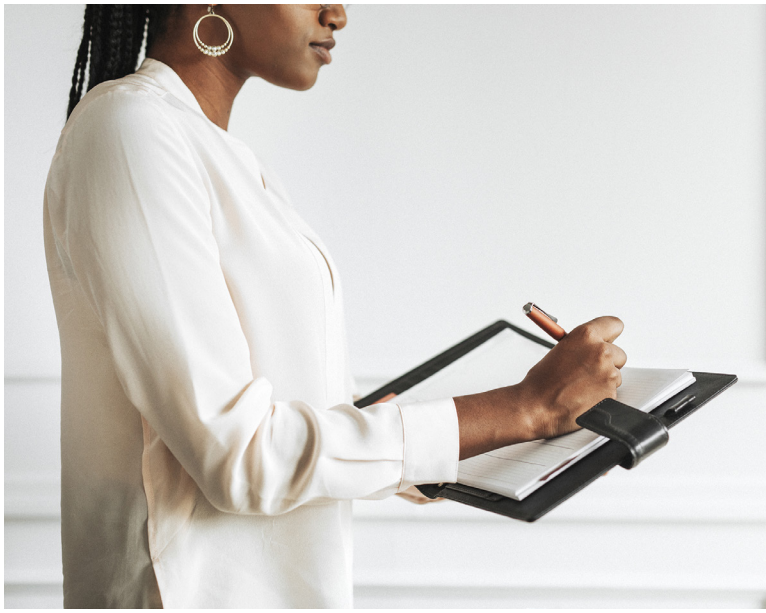
Definition of "Done"

When I talk about having a definition of Done, you want to make sure your stories have sufficient detail. They need to be able to be developed. Without all the detail, you're going to have questions coming back to you, and you want to get those questions answered up front so you're not slowing down your development team.

Also, if you are working on a team and you have not truly completed the stories to the group's shared understanding of the definition of Done, don't let that story be demonstrated or shown. It's not really done until it's met all the criteria.

Checklist Before Take-Off

Think of Ready and Done as a checklist you would go through and check off to make sure everything is complete for it to be done – much like what an airline pilot does before they can take off. There's no such thing as "kind of done", "sort of done", "almost done", or "done done". It's either Done or it's Not Done.



Agile Transformation Pain Point #13

TAKING ON TOO MUCH WORK IN A SPRINT

13

Some teams are super ambitious, and they think they can climb a whole mountain in a single Sprint, but the whole goal with Agile is to maintain a sustainable pace.

Take on only what you really think you can get done within your Sprint. Don't overestimate your velocity, and don't look at your history and rest on your laurels. Look at what you have available for your capacity, your skills as a team, and be realistic. Just keep it real.

Carrying Over Work is Demoralizing

What really helps a lot of teams is to focus on meeting their Sprint goals. I can't tell you how demoralizing it is for a team to take on a pile of work, and then to not be able to complete more than 50% and to have to carry that over into the next Sprint…and then carry it over again and again. You don't want the team to experience this because they're going to feel like failures.

Be Realistic

You want to make sure you pick enough things you think you can realistically get done, and then get them done. If you have time left over, there's nothing to stop you from looking ahead and asking the Product Owner if there's something else that's ready that you can pull in to your current Sprint (so long as you can get it done within that Sprint). Keep it real and keep it sustainable. You don't want people working crazy hours to try to get stuff done. That is really something you want to avoid, if possible.

Agile Transformation Pain Point #14

PRE-ASSIGNING WORK TO TEAM MEMBERS

14

This Pain Point is something you definitely don't want to do. Again, Agile and Scrum teams are meant to be self-managing. They should volunteer to pick up assignments, not be assigned.

Development team members are volunteering to do stories, and they might want to learn something new – let them do that. You want to let that team to manage itself. Volunteering also lets people have opportunities to grow by learning to do something new.

Avoid Expertise Bottlenecks

Otherwise, if you pre-assign stories to someone based on their expertise, you're robbing another team member of the chance to learn. You're also doing yourself a disservice by creating a bottleneck where only one person has certain knowledge and expertise. You want to spread that knowledge across the team as much as you possibly can.

Again, just because one person has that expertise doesn't mean they shouldn't be actively trying to train someone else. Pair programming is a good way to go: team up a senior person with a junior person, or someone with database skills with a developer, or someone with one development language with another development language. Spread it out and let people learn.

Agile Transformation Pain Point #15

MISUSING SCRUM CEREMONIES

15

Avoid having runaway meetings, meetings that go past their time boxes, and meetings that are not for their original purpose. Keep it focused.

Create and follow an agenda for the more formal Scrum events – depending on your company and culture. I've worked at loose organizations and formal organizations. At both, the Scrum events worked because they followed the lead of the company and their culture.

Keep Out Saboteurs

Make sure you don't invite the wrong people. If there are problem people that shouldn't be involved, don't invite them. Maybe politics are going on, or maybe somebody's nosy and wants to see what's happening. You do want your stakeholders there, but you want to make sure they're the right people. Otherwise, you can get some saboteurs who try to create blockers for your team and set up your team for failure. You don't want those people in there if you can avoid it.

Stick To Your Time Boxes

Don't use meetings for purposes other than the original intention. Be prepared and make sure you're ready when you go into your review meeting. Make sure you know who's going to be showing what and what they're going to be showing.

I always have an extra meeting that I hold on my teams, which is the day before or right before the actual Sprint review meeting. I'll gather the team to get the logistics out of the way. We go through what we got done, who's going to show what and in what order. Showing the work with a dispersed remote team is more challenging, especially on something like a mobile application where you can only sort of simulate it. Ideally, you want to be there in person if it's possible and show them the real deal.

Preparedness is Key

Be prepared, especially for planning. You don't want to go into a Sprint review and say, "Oh, sorry, I only have a couple of stories that are ready." You could start sprinting and have the person continue to work on fleshing out and getting approval of the stories that are still to come. Ideally, however, you'd have done that and are ready for the team to pull from the backlog when they're doing their planning.

No Pointing Fingers

Also, don't use retrospectives as a blame game. That should be a safe place where everyone can be honest with each other and not take it personally. It's about getting better together. Things don't always go smoothly, so take those as learning opportunities.

Follow Through on Action Items

Finally, follow through by acting on action items from your retrospective. Typically, you want to pick at least one thing from your retrospective that you can improve upon that was part of the empirical process of inspecting and adapting. Either add it to a board that says "action items", or add it to your actual Sprint backlog with tasks and follow through on it.

Going back to the Waterfall world, many teams will hold a lessons-learned meeting at the end of a project, but then the team disbands and no one takes any action on the items learned. They're pretty much doomed to repeat the same mistakes they made before. I think Agile makes this more visible and more actionable because you're doing this throughout the process, not just at the end.

Agile Transformation Pain Point #16

SKIPPING OVER QUALITY

16

You should not skip over quality because it's inherently meant to be baked into Agile. There are many ways you can do that.

You could consider doing acceptance test-driven development, behavior-driven development, or you can write automated tests wherever possible. Again, I really like the idea of pair programming, peer checking, doing code reviews for each other, etc. It all helps to improve the quality of your product.

Release and Test Frequently

Another thing you should avoid is waiting until the end of the Sprint to check your quality, leaving no time for fixing things. This is a Waterfall problem. Agile is meant to address many of these issues. Release frequently and keep that cycle time short. As you're finishing things, they can be released to your QA environment so you can test them right away, enabling you to get to your definition of done.

No Credit for Partially Done

If you wait until the end of your Sprint (Waterfall style), then you're not going to have enough time to get to Done. Nobody likes to carry things over because there was a critical bug that wasn't found until the last day and there wasn't time to fix it. When the Sprint is over, the Sprint is over; you're not going to get credit for something that is not completely Done.

Misconception: Agile Doesn't Have Testers

Though Agile doesn't have an official tester role, Agile includes a lot of testing (or it should). Testing and quality is a responsibility of the whole team, not necessarily one person. Even if you had a quality analyst on your development team and their primary role would be for QA, that does not mean that the rest of the team couldn't pick up and help if needed. The idea behind a cross-functional Agile team is that each member can bring their skills and talents to the team and work together to get to Done.

Agile Transformation Pain Point #17

FOCUSING ON PROJECTS RATHER THAN PRODUCTS

17

This is a difficult transition for Waterfall companies to wrap their heads around. It's a big mindset shift that needs to happen to be truly successful.

I'm not saying that you can't have an Agile Scrum team that works on projects with a distinct beginning and end. That might be a good way to start with Scrum and Agile, but ultimately you want to move from project towards product.

Projects vs. Products

Projects are those that have a beginning and an end – they're temporary endeavors. Products are more than a project. Products do not necessarily have a specified beginning and end. They start as an idea, and they grow from there. They should continue to grow and mature and evolve throughout their lifecycle until they're eventually sunsetted.

Products need to be continually cared for, they need to be growing, they need to be evolving, and they should be ultimately growing based on the needs of the customers.

Product Mindset Keeps Your Momentum Going

This is a very difficult concept for companies who are used to budgeting certain amounts for specific projects. In this case, they need to allocate enough funds or resources to support a product over the entire life of the product. That's a tough mindset shift to make.

When you're first transitioning, you might still be focusing on projects, and it's a good way to start. But then, if you want to keep the momentum going, you need to keep the team together. Otherwise, you lose that momentum when the team breaks up after the project ends. All the knowledge you learned and the efficiencies you gained are lost.

Keeping your team together and focused on a product is the ultimate best idea.

Agile Transformation Pain Point #18

ACCRUING TECHNICAL DEBT

18

Everyone who works in technology probably knows that you already have a big pile of technical debt. The problem is that you are going to have to eventually pay that bill, and some of it is risky.

Dig Out a Little Every Sprint

You really need to be careful about technical debt. Every time you hack something to make it go faster and every time you slap a band-aid on something, it's technical debt.

The key is to dig out of that a little bit in every Sprint. Like picking an action item to tackle in every Sprint, you could allocate a percentage of your available capacity to tackling technical debt.

If You Don't...You'll Never Get Out

If you're digging out of that debt hole, then eventually you can get out. But if you don't dig, and you just let it pile up and up, then something is going to explode.

You're going to pay the piper at some point, usually when you can least afford to. Try to avoid that if possible.

Agile Transformation Pain Point #19

DISTRIBUTED GLOBAL CHALLENGES

19

So far, I've been primarily discussing co-located teams, which is ideally better. However, this is the real world, and it's a global economy – most people don't have the luxury of being co-located.

If you have a team with dispersed team members, set up your working rules so you have some time that overlaps with one another. It could be an open conference line, audio line, or video conferencing. Just make sure you have a time where you're all able to be together in some way.

Make the Most of Your Tools

If you have technology tools, make the most of them. Most companies nowadays have some sort of video conferencing. Audio is great, but everyone knows that video allows you to see body language, and that's going to feel a lot more like you're together. You can read between the lines. Not everything always comes through if you're just on the phone. If it's at all possible to use video conferencing, that would be even better.

Also, use collaboration tools. Slack is a great one. I really like Microsoft Teams. There's even SharePoint, Webex, and Zoom. Use anything that lets you smoothly communicate with your team.

Encourage Collaboration

If you do have the privilege of being co-located, don't just sit in your cubicle facing the corner. Turn around, talk to your team, build a collaboration space, get rid of the walls, and put a table in the middle so you can just turn around and have impromptu meetings. That will allow you to be more creative and innovative.

The fewer walls you have between you, the more knowledge transfer is going to happen in a very natural way. You're not going to feel like you're cross-training, but you're going to pick things up just by being near people as other conversations are going on.

Agile Transformation Pain Point #20

SKIPPING INSPECTION AND ADAPTATION

20

Inspecting and adapting is one of the key parts of Agile and Scrum. But over time, teams start to feel bored if every session is the same.

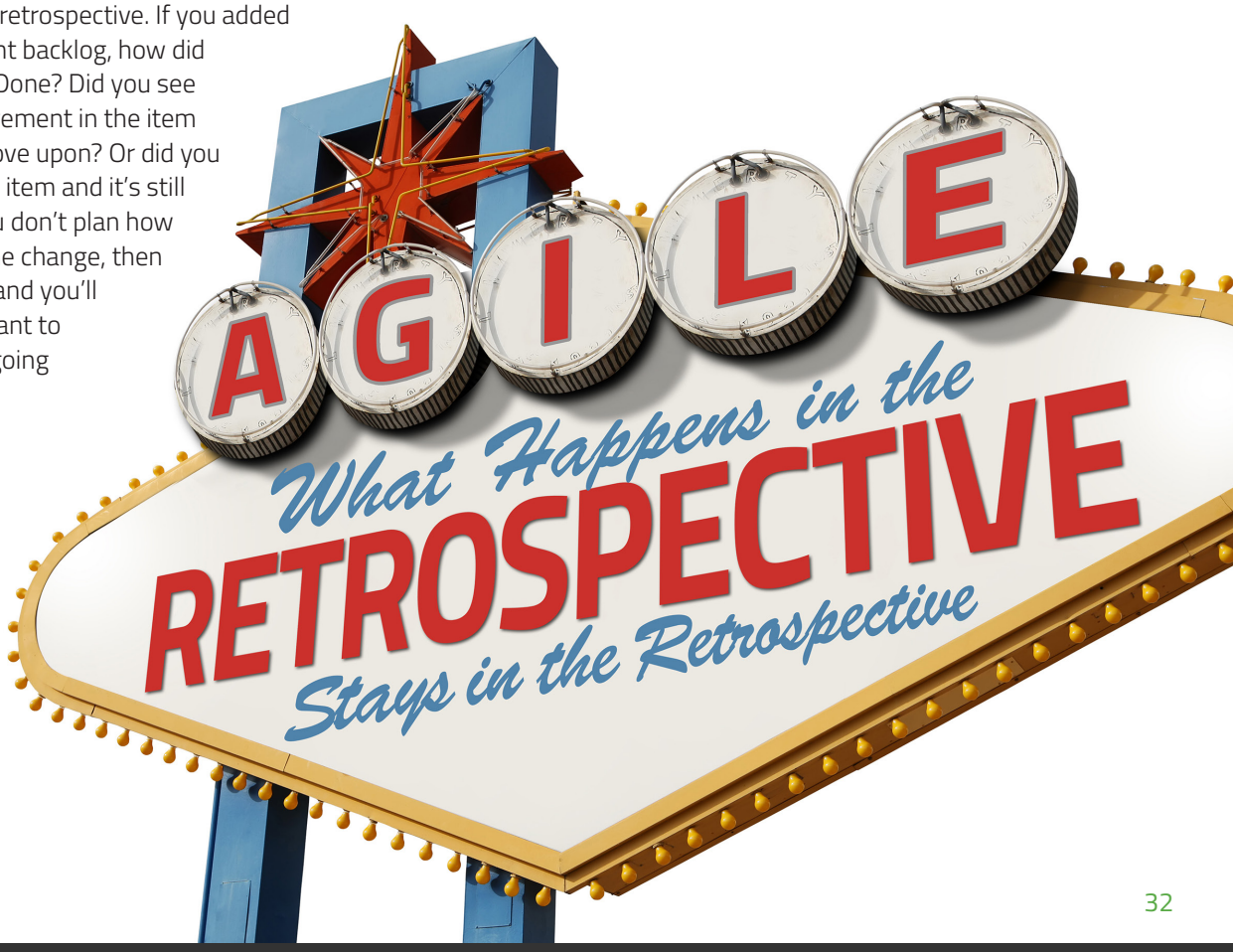
If it does become boring, make sure you switch it up to keep it fresh. The Fun Retrospectives website has lots of ideas. Don't skip this part because it gives everybody the opportunity to be honest and share.

Follow Through on Action Items

One thing you should probably do when you do your retrospective is to look at the action items that you picked in your previous retrospective. If you added something to your Sprint backlog, how did you do? Did you get to Done? Did you see any measurable improvement in the item that you chose to improve upon? Or did you forget about this action item and it's still a problem? Often, if you don't plan how you're going to make the change, then nothing ever happens, and you'll never get better. You want to make sure that you're going after those changes.

Retrospective = Vegas

In this session, have the courage to be honest. It's not going to help anyone if you just keep your mouth shut and don't say anything. If you had a problem, talk about it. What happens in the retrospective should stay in the retrospective. Call it Vegas. It should be a safe place for everyone, and people should be able to speak their mind and not be afraid.



Agile Transformation Pain Point #21

EXECUTIVES' LACK OF KNOWLEDGE AND UNDERSTANDING

21

Agile is no longer a buzzword or a fad – it's here to stay, and it's a fundamentally different way of working.

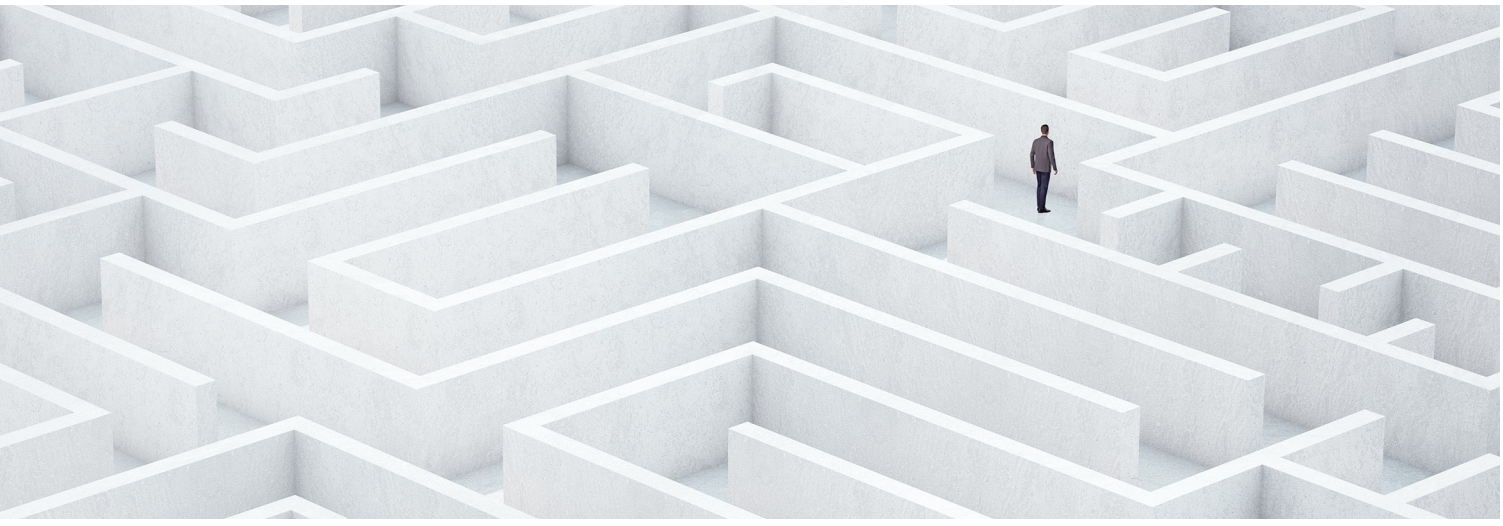
The teams doing the work need to be trained on Agile, and executives need training, too. They cannot successfully support an Agile transformation if they don't understand what they are directing.

Walking the Walk

The transition to Agile requires a significant mindset shift from leadership. The traditional methodology that was developed in the first industrial revolution was fine when applied to repeatable, mechanical types of projects, but it just doesn't work for the kinds of complex, complicated technology solutions we build today. Agile is a completely opposite way of thinking and operating. An academic understanding and acceptance of Agile is a good place to start, but executives need to truly understand what it means to be Agile and be able to model it. They need to walk the walk, not just talk the talk. This is difficult. If you don't get your executive leadership to support this effort, you will have a much harder time getting it to succeed.

Measure Success Differently

You also need to be careful on what you report as measurements or metrics when using Agile with executives. They are going to look at things like Agile velocity on a team and think they can compare one team to another. Most Agile practitioners know this is not something you can do. That number is merely a historical representation of the work that the team was able to do and can't really be used to measure one team against another. So, you will want to look for different metrics you can share with the executives so they can see how well the transformation is succeeding. Some options for that include customer satisfaction, employee happiness, business value delivered, and things like that.



Agile Transformation Pain Point #22

MANAGERS FEAR LOSS OF CONTROL

22

Managers are going to fear the loss of control. This is because, in many ways, they **are** losing control.

Agile teams are really meant to be self-managed, and in many respects, self-managing. Managers need to adjust to the change in the organization and maybe take on a different role. Many of you reading this are managers, and I know how hard it can be to let go of what's comfortable. Still, embracing Agile can have plenty of rewards, and you can transition into equally valuable roles focused on servant leadership.

Servant Leadership

Leaders must adopt servant leadership. This is a significant change if you're coming from an old-school command-and-control world where you are used to directing and controlling the work for your team. You really have to let go of that, and let the teams do that self-management.

Career Guide and Coach

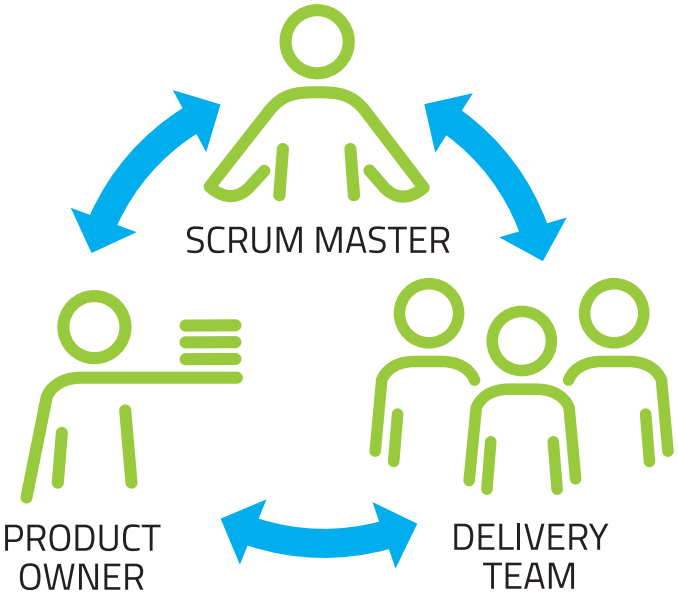
As you transition as an IT manager (or some other kind of manager) into an Agile environment, your role is going to change. Your role will become more of a career guide or a coach. You will be there to help with training, provide tools, and be a sounding board for your team members as they work through their role on an Agile team.

Leadership in Agile Roles

In many cases, I've seen Project Managers or IT Managers transition to become Scrum Masters, Subject Matter Experts, or even Tech Leads, depending on how they came up in the organization.

There are some cautions there: managers need to let go of the command and control and take on the role of servant leader, or they won't be able to successfully transition into one of those roles. But I have seen it successfully done many times.

Again, you just need to remember to let the team self-manage. Step back and let go of control.



Agile Transformation Pain Point #23

NOT HAVING A FULLY CROSS-FUNCTIONAL TEAM

23

It's not always possible to build that fully cross-functional team, but – wherever you can – try to remove dependencies and get all the skills on the team that you need.

At one client I worked with, they were midway into their Agile adoption, and they had formed several teams. They had done a decent job of getting most of the cross-functional skills needed on the team. However, many of their teams had some sort of external dependencies on specialists from other groups, so they didn't really have all the skills on the team.

You must include all those skills, or you will be slowed down on getting to "done" within a sprint. It's not ideal to have specialists.

The Shift from Specialist to Generalist

Another important aspect of letting go of old roles is to focus more on generalization vs specialization. Many organizations tend to go the specialized route, but as they transition into Agile they find this can be quite a bottleneck.

They've probably already experienced the bottleneck in their Waterfall world, and it will only become more apparent in an Agile world. The more generalized your team members can be (and the more flexible and adaptable they are), the better off you're going to be.

Agile Transformation Pain Point #24

UNEMPOWERED PRODUCT OWNERS

24

As most of you probably know, the Product Owner is the most important role on the Scrum team.

This person must have deep business knowledge and leadership skills. They are responsible for the product that is being delivered, and they must understand the market. They also need to have strong relationships and a big network of people they can turn to as a Product Owner.

If they are not truly empowered, they will not be able to be successful. If they have to go five levels up within the organization to get approvals, they are going to struggle. They are not going to be able to make decisions quickly and will slow their team down – which is the opposite of agility.

They also need to be allowed to make decisions. So, another important aspect of Agile is that the decisions are pushed down to as low a level as possible; you can push decisions to the team during sprinting, while the Product Owner needs to be able to make decisions for the product.

They are the ones responsible for ordering the backlog and making sure that the highest-value, most important things are at the top of the Product Backlog.

Splitting Product Owner Responsibilities

Ideally, a Product Owner should only be one person. I have worked at clients where there were multiple people (but usually no more than two) involved – sort of like a duo that acts in the Product Owner role. They would swear up and down, "Yes, we know everything

that the other person knows – each of us knows everything, and we're on the same page, and there won't be any conflicts." But, lo and behold, way into the project if one of them is out, and the other one makes a decision, then invariably there is rework. It's not ideal to have multiple Product Owners.

Product Owners are responsible to communicate with their stakeholders. They should be able to communicate from the top of the organization all the way down to the members on the Scrum team. They could represent a product committee, but ideally you want one person to be where the "buck stops" – and that is the Product Owner.

Agile Transformation Pain Point #25

NOT STARTING RIGHT AFTER TRAINING

25

If your organization gets you training – that's awesome! The best way to retain what you learned is to immediately apply it to how you work.

I once worked at a company that spent a ton of money to give us all a week-long bootcamp in Agile, and then they didn't follow it up with any support. They gave us the training, they set us loose, and we didn't have coaches or guides; they didn't adjust their PMO in any way to account for Agile. It was a failure.

If you don't want that to happen to you, then make sure you get started right away after training. It's really the only way to get proficient.

You are going to make mistakes – it's not going to be perfect. But you learn from those mistakes, adapt and get better. This is what Agile is all about.

Make sure you apply that training right away. The longer you wait to apply the training, the more you are going to forget. This is a well-known concept in education and training. You need to apply that learning within two weeks, or you will forget a lot. And if there is a big time-lag in between, then you will have to be retrained. You might have to do a refresher, which might not be as deep as the original training, but you will need to revisit your materials from your original training.

You Can Start Before It's "Perfect"

Teams should also realize they don't have to have everything completely ready to start sprinting. The team I am coaching right now has a dedicated Product Owner and Scrum Master, and they are struggling with getting their initial product backlog ready; they feel like they have to get it perfect, and they don't yet realize that it's a living artifact. It will constantly be changing and evolving, and it will never be perfect or complete. And that's okay. **You don't need to have a fully fleshed-out backlog to be able to begin.**

Agile Transformation Pain Point #26

STARTING WITHOUT A PRODUCT BACKLOG

26

I caution you to have at least something ready when you start sprinting.

It's okay to start with nothing, but it's not preferred. If possible, you want to have at least enough for the team to pull in one or two sprints' worth of work. It doesn't have to be complete or perfect, but it helps the team get off to a good start if they have solid product backlog items to pull into their sprint. As they start sprinting, the Product Owner and their support team can continue to build out and refine the backlog.

The key with this is that you don't get too far ahead. Try to be no more than two or three sprints ahead. Your backlog might also depend on how long your sprint length is. If you get too far ahead, there's a chance that the work you do will be irrelevant by the time it comes time to look at it, and you'll have to do wasteful rework. You want to avoid that. The mantra is to get "just enough, just in time".

Another key point that a lot of teams I've worked with have missed is that they need to have a definition of "ready".

When I am coaching and forming new Agile teams, part of the team launch is to create that definition of ready so that everyone understands what a ready story looks like. Typically, it means they've had the three Cs **(Card, Conversation, Confirmation** – credit to Ron Jeffries), there is documented acceptance criteria, the Product Owner has approved it, and the story has been sized appropriately. That is a simple definition of ready that I think most people could agree with, but each team should come up with their own definition of ready.

Agile Transformation Pain Point #27

FEAR OF AMBIGUITY AND NOT KNOWING EVERYTHING UP FRONT

One of the hardest things for people who are transitioning from a Waterfall environment is getting comfortable with ambiguity.

When you are used to knowing it all – getting all the requirements in advance – it's hard to let go of that feeling of certainty that you captured everything. Instead, you need to realize things are going to change. You are not going to have everything up front – and that's okay. That's a very difficult mindset shift to make. I know because I spent years in a Waterfall environment. When I first transitioned to Agile, I was very resistant because I was uncomfortable with ambiguity. It took me a long time to be okay with it. Now I recognize that you just need to know enough to be able to start, and that's good enough. Everything will evolve and emerge as you go about your way.

Know there is change and welcome it. Instead of Waterfall where you "resist, resist, resist" change, you want to embrace it and look at it as an opportunity to build the right product for your users. Don't say "that's a bad idea" – instead, consider it; it might be a good idea.

Also, make sure you've got the "big picture" but work in smaller chunks. The Product Owner will have the overall vision, and that will guide the work that goes into the Product Backlog. You're not going to be building huge features all at once; you will be decomposing those into smaller chunks and building them bit by bit.

Agile Transformation Pain Point #28

SEPARATE ROLES FROM TITLES

28

In Agile, it's no longer about the individual – instead, it's about the team. The team needs to work together, collaboratively and in concert to deliver on their Sprint goal each iteration.

This one is hard for people to do, especially if they've worked their way up in an organization and have a title like "Senior Architect" or "Senior IT Manager" (or whatever it might be). Many people feel like this is their identity, and it's hard to leave this at the door. But when you join an Agile/Scrum team, you really need to leave your title and your ego at the door. If you don't do that, then there will probably be a lot of infighting and tension, and you don't want that on your team. You want everybody coming together to get to "done".

Win as a Team, Lose as a Team

I love the phrase, "win as a team, lose as a team". It's not about the individual anymore. You won't have any superheroes on your team – everyone should be doing everything they can to get to "done". As you probably all know already, there are only three "official" roles in Scrum: Product Owner, Development Team, and Scrum Master. There are many skills that are needed on the team. Your development team could be comprised of technical leaders, business analysts, quality assurance, database administrators, user experience, programmers, etc. You might have any or all these skills, or maybe others like security or designers. Your team needs to have all the skills to get to "done". If you have skills you can leverage that aren't your key competency, you can bring those to the team.

As a Business Analyst, I happen to have a background in web design. I can do a little bit of UX, I can use Photoshop, I can do testing, I can do research, I can do a lot of things – I could probably even do a little bit of programming (but don't tell anybody, I really don't want to do any programming – but I could do a little if I were tasked with it). So, bring all the skills you have to the team, and leverage those.

Agile Transformation Pain Point #29

FINDING THE RIGHT TEAM SIZE

29

When you're first starting with Agile, you try to put together a team; you think you've identified everyone you need, and invariably you're probably wrong.

I see this as a problem more with new teams, but it can be a challenge later in the agile transformation as well. Personally, I love Jeff Bezos' "two-pizza" rule. If you can feed your team on two large pizzas, then it's a good size. But if you can't feed the whole team, then it's probably too big.

The Scrum rule says that it's six people plus or minus three, so the ideal team size is between three to nine people. This does not include the Scrum Master or Product Owner. I've seen teams that went a little bigger than that, but it starts to get more chaotic. It can work, but you must be careful – I'd consider splitting it into two teams if it gets too large.

Likewise, I've seen teams as small as three have a few issues because everyone will have to wear multiple hats. The developer might be the Scrum Master, and wearing those different hats can be really hard. You take one hat off and put another one on and shift your mind as you're moving in and out of those different roles. If you go any smaller than three, you probably also wouldn't have all the necessary skills to get to "done".

Agile Transformation Pain Point #30

SPECIALISTS VS. PART-TIME TEAM MEMBERS

30

Another problem I've seen while coaching a company is that they formed their teams, but they didn't include all the cross-functional skills.

So now they need specialists. But they're having trouble defining what the difference is between a specialist and a part-time team member. We worked with them to understand the definition of each so they could treat them accordingly and set the proper expectations.

Specialists

Specialists might be brought in for a very short period – maybe one or two sprints – and they are just there to do that one specific skill they are an expert at. They wouldn't be asked to pick up other tasks and help the team get to done. They might not attend all the Scrum events. They are there just to do those discrete tasks and get out.

Part-Time Team Members

Part-time team members might be less allocated than a full-time team member, but they are still fully dedicated to the team. This is their #1 priority, and they will spend all the time that was allocated toward their part-time team membership in completing the sprint goal. That person might be expected to attend all the Scrum events and pick up any task they can do to help accomplish that sprint goal.

Agile Transformation Pain Point #31

EXTERNAL DEPENDENCIES

31

Dependencies are something you have very little control over when you're working with vendors or other contractors. Because of this, it can slow your teams down.

Outside regulations, filing deadlines, and even other teams inside your organization still operating in a Waterfall mode can cause delays as you work to coordinate. External vendors or partners, especially an offshore partner where there is a time difference or a language barrier, can also be tricky to coordinate. You will run into a few issues like this. Again, ideally (and I will just hammer this home) you want to have the skills on your team necessary to get to done.

Whenever possible, you want to reduce dependencies. You can do that by cross-training, changing your team by moving people in and out, getting additional training, and maybe doing some pair programming.

There are many ways to reduce dependencies. For example, instead of bringing in a piece of outside technology owned by a specialist, try bringing that into your system so you can control your own destiny. I've seen some teams do this with great success.

Agile Transformation Pain Point #32

LACK OF TRAINING FOR ANCILLARY PARTICIPANTS

32

It's one thing to train your core Scrum team (Dev team members, Scrum Masters, and Product Owners will all have special training). But what about everyone else?

Do you train all your stakeholders? What about the people you represent? What about your customers? You are not the only one who needs training.

If you just throw them into the mix without any sort of training – even a 10-minute 101 explaining the events and the terms – you will run into trouble. Those people will be clueless when you start talking about velocity, burndown, and story points. They will shake their heads and be very confused. You want to make sure that, at a minimum, they have an introduction to Scrum (or whatever Agile framework you are using) so they can understand what's going on.

Who Would Benefit From Training?

Down the line, you should invest in more training because, as you continue to adopt Agile over time, it's more and more likely that those participants will be involved in multiple Scrum teams.

This Could Include Anyone Like:

- Specialists
- Stakeholders
- Executive Sponsors*
- Vendors
- Managers
- And more!

Anyone and everyone who is going to be involved should at least have a rudimentary understanding of what Scrum or Agile is.

***Training for executive sponsors can be especially important to secure that top-down support we talked about in a previous Pain Point.**

Agile Transformation Pain Point #33

PRODUCTION SUPPORT ISSUES

33

I'm sure every team has experienced some sort of production support issues, unless you're lucky enough to have a dedicated support team - which most organizations that I have encountered do not.

If you do – lucky for you – but if you don't, the product team that built the product is usually responsible for dealing with any support issues that come up (especially if they're critical show-stoppers, and your business can't operate). That's going to trump everything else in your sprint backlog. And this does happen. This is a real-world scenario, and you need to be prepared for it.

How Do You Handle Support Needs?

My advice is to know that this is going to happen and allocate a percentage of your time each sprint for dealing with support.

If you have team members assigned to support as a rotating responsibility, you would just reduce their capacity.

If they happen to not have any support issues – fantastic! Then they go ahead and work on other items to help the team to get to the sprint goal.

But if you do have production support issues that end up compromising your sprint goal, and you are not able to accomplish everything you planned, then you have a legitimate reason why. You can then be transparent and explain that to your stakeholders. In most cases, they will be very understanding; they know that things happen, and they are not going to hold your feet to the fire.

Agile Transformation Pain Point #34

DECOMPOSING WORK INTO BITE-SIZE CHUNKS

34

It's hard to know when to break items into smaller pieces. The rule of thumb is, "Can it be done within a sprint"?

That is going to vary, depending on how long your sprints are. Each team might have a separate rule that helps them determine whether it's too big. I've worked on teams where a story size of eight was the rule. If we had a story that size, it was a clue that it was too big, and we needed to break it into smaller pieces. For another team, it might be a 13, or it could be other rules like complexity. The key is to make sure you can complete the work within the sprint.

How Do You Break Down a Larger Story?

There are many ways to split those larger stories. I recommend splitting them vertically instead of by horizontal system layers. This way, you are going to make a thin slice through all the layers to get a small, discrete function completed. Some examples include:

- Workflow steps
- Input options
- Test cases
- Business rules
- Data types / parameters
- Acceptance criteria
- Happy / unhappy path
- Operations (Create, Read, Update, Delete)
- Roles

(Source: Christiaan Verwijs)

This is the just the tip of the iceberg, though. There are at least 25 story-splitting techniques I know of, and we published blogs on many of them on the Core BTS website.

Agile Transformation Pain Point #35

MANAGING THE FLOW OF WORK

35

I've seen problems where there is way too much in the Product Backlog, and no one can figure out what to do – as well as the opposite problem where there is not enough to do, and it's hard to keep the team fed with high-quality, ready work.

The key is to get just a few sprints ahead, make sure you have a definition of ready, and ensure there is enough work for the team. Again, don't go too far ahead, or you'll have waste and re-work. If you don't have enough work to go around, many companies get concerned with team utilization because it's the metric they care about the most. But in Agile, you need to let go of that. There are a few things you can do if have spare time in a sprint.

Adjust the Team

If spare time is a regular thing, then you might need to consider adjusting the team size. If this is a consistent problem, and you never have enough work, maybe your team is too big (going back to the need for the right-size team I talked about in pain point #29). Maybe you need to make a few cuts to the team – that might be kind of hard, because it feels like throwing someone off the island. But you might have to do that sometimes.

Help Others

You should be helping someone else on the team if you run out of things to do. If you got everything you signed up for done, then go ask your team members if they need any help. Your goal there is to meet your sprint goal and get everything to done so you have that fully working increment.

Clean Up Technical Debt

This is also a great opportunity to clean up technical debt. I don't know of any company that didn't have any technical debt. If you have technical debt, make it a "go-to" activity if you run out of things to do.

Learn Something

This can be a great opportunity to learn something new. The more general knowledge the team members have, the more value they are going to deliver to the team. Take the opportunity to learn a new technology or skill.

Likewise, you could do some market research and help the Product Owner; see if they want to do some benchmarking or competitive analysis. You want to understand those things, and you can help the Product Owner with that.

Pull Something

Last, but not least, if you are in the situation where everything has been completed, you can always pull something in if it meets the definition of ready. But only pull it in if you think you can complete it within the current sprint. If you can't get it done, I wouldn't encourage pulling it and having it be not done. You could get a head start or do some research, but don't pull it ahead unless you feel like you can complete it. Or negotiate with your product owner on something lower on the product backlog that is of a size that could be completed.

Agile Transformation Pain Point #36

TOO MUCH WORK IN PROGRESS (WIP)

36

This is a recurring problem I've seen on different teams where they have a lot of items they pulled into their sprint backlog.

And it ties into the last pain point. The developers on the team will pull an item, start working on it, maybe get stuck (but not raise an impediment) and then start working on another thing. And then another thing, and then another thing until they've got six or seven items currently in progress. That's not the best way to operate where you're in Agile.

Focus on Priorities

The key is to focus on the one item at the top – the highest-value, highest-priority thing until it's done. If you have something that's in your way, raise the issue. Get the impediment removed so you can get to done.

Set WIP Limits

You should also consider setting a work-in-progress limit. This is more of a Kanban idea, but I think it works well in Scrum, too. Maybe have a rule for your team that they can only have two active work items in progress at a time – this might be manageable. We just want to limit it so they're not multi-tasking – which we all know is not very effective. That way, they'll be able to focus and get things to done.

If you continue to have the problem where there are too many things in progress, you're basically going to go back to where you were before you started Scrum / Agile. It's kind of a Waterfall problem where you're assigned to a dozen different projects, and you're working on them a little bit at a time, and nothing is ever going to get done. Put that away and just focus on one or two things and get them completely done. There are no real productivity gains or efficiency benefits in multi-tasking.

Agile Transformation Pain Point #37

VISIBILITY OF WORK TO MANAGERS AND STAKEHOLDERS

37

We all know that one of the tenants of Agile is transparency. We need to strive for transparency.

Being transparent is difficult if you're managing all your requirements inside a tool, but not everyone has access. You need to make sure your Product and Sprint backlogs are somehow visible to your stakeholders. Anybody at any time should be able to look at these – whether it's a wall or a digital tool – to see where you stand. You shouldn't be hiding anything. Strive to make that transparent and visible as much as possible.

Put the Right Metrics in Front of Stakeholders

For people higher up in the organization, they might not care to get down in the weeds, so they wouldn't have any interest in looking at things at a granular level. But they might care about the overall key metrics. Identify those KPIs, create some dashboards, and make those available so they can consume them at their leisure.

Also consider that different stakeholders might want different types and levels of information. So, ask them! Ask them what they want to know and what they care about. Tailor the information you share to your audience.

Make sure everyone understands what they're looking at, too. If you give everyone access to a tool, and they go in and poke around, but they've had no training on it, they might have no idea what they're looking at.

It's easy to misinterpret something. If you do give people access to something like that, make sure they at least know what they're looking at.

Agile Transformation Pain Point #38

TOO MANY SILOS AND HANDOFFS

38

Silos and handoffs are traditional Waterfall problems, but they still happen in Agile, especially if the organization's structure is siloed.

They don't like to talk to each other, and they don't like to cross over. This becomes especially concerning when you have multiple teams. If and when they have to coordinate, there are hand-offs and integrations, and it can get really complicated and sticky.

This is also a key problem for highly regulated organizations that have gated systems where they go through formal approvals (ex. government or other regulating bodies). Sometimes you cannot get away from this, but if you can find a way to work around it or streamline the process, try to do that.

Also, try to identify the choke points that are slowing things down. If you have visibility to your board, and it's in a visual state, you can see where the blockers are building up. That's a very easy thing for the Scrum Master to see and try to fix.

Always Ask Why

Finally, always ask the question, "Why?" Don't just take things at face value; always be challenging the status quo. That will help you come up with better solutions all-around.

Agile Transformation Pain Point #39

DECISION-MAKING AT TOO HIGH OF A LEVEL

39

The Product Owner should be the primary decision-maker, and not have to go up the chain to get approvals.

You don't want to have to go through a whole committee to make a small decision. Empower the Product Owner and let them make the call. If they make a mistake – guess what? It was only a mistake for two weeks, and you can quickly fix it.

Try to remove the layers of bureaucracy as much as possible. Ideally, you want to push the decisions down as close to the work as you can, so the team can make choices as well.

Decision-By-Committee Creates Delays

The more people that are involved in a decision, the longer it's going to take. If your organization has many different layers, it's going to be hard for them to let go of that process. If they're used to signing off on every single request – be it small like a new field on a screen, or a huge 3,000-hour project – you're going to have some issues.

People like to be in control, and they like to know what's going on. You still need to communicate what's going on, but because the decisions are smaller, they're much less risky. Even if someone makes a mistake, you can quickly discover it and pivot in the next sprint…not two years from now. If something goes wrong, don't view it as a failure, but as a learning opportunity.

Trust the Product Owner to make the final call. They are the CEO of their product, and they need to have their decisions respected throughout the organization.

Agile Transformation Pain Point #40

GETTING TO THE RIGHT LEVEL OF DOCUMENTATION

40

There is a myth in Agile that there is no documentation. I think anyone who is a practitioner of Agile knows this is not true. There is documentation, but it's just enough, and just in time.

Keep it as lightweight as possible, but documentation levels will depend on your company. Does your company have any organizational standards? Are you in a regulated industry? Could you possibly be audited? These are some of the considerations to keep in mind when figuring out what type of documentation you're going to need.

Choose the Right Models for Your Requirements

You will still need models for your requirements. You might need to have some diagrams, or a wireframe, or a data dictionary – something that accompanies your requirements to help them be complete enough to be suitable for development. The key is to pick the right tools out of your toolbox to create those models, and make sure you're only doing just enough for the development team to be able to proceed.

Sometimes Specific Documentation is Required

Depending on your organization, some companies might need documentation. For example, in healthcare where a quality product could mean the difference between life and death, you might have to do more documentation. If you are in insurance or finance and are regularly audited, then you may also need to have traceability of your requirements. The key is to not overdo it. If your organization has existing standards, you should keep them in mind but also know they may need to be adjusted for Agile.

The bottom line is that you should only do documentation that provides actual value for your organization.

CLOSING REMARKS

Agile Transformation Can Be Painful, But Much of the Pain Can Be Avoided or Managed

Thank you taking the time to read my eBook. I hope this information will help you in your Agile transformation journey. As I mentioned in the beginning of the book, I learned these lessons the hard way. The advice I share is from my own real-world experiences. My intention in sharing these experiences is to help you either manage or avoid the same painful experiences I went through.

Now that you've heard about my experiences and advice, are there are any other Pain Points that you are currently going through that you have learned from or need help with? If so, I would love to hear from you.

How Can Core BTS Help You?

If you have been struggling with one or more of the Pain Points I mentioned in this book, and you are looking for an experienced partner you can trust to guide you in your Agile journey, Core BTS is an ideal fit.

We are staffed with numerous experienced and certified Agile practitioners, trainers, and coaches. We have a track record of success with our clients, and our main goal is to help you achieve self-sufficiency in your Agile practices.

You may reach me personally at:

Email: Rachael.Wilterdink@corebts.com
LinkedIn: linkedin.com/in/rachaelwilterdink

Or contact Core BTS to discuss how we can help you:

Website: corebts.com
Email: info@corebts.com

CORE BTS