[Rich communication](https://app.qa.com/course/agile-fundamentals-module-1/rich-communication/?context_id=983&context_resource=lp)

Have you ever been in a situation where you were asked to do something but the instructions felt kind of vague? Then you tried to do it but ended up failing? Well, one of the agile principles reminds us that the most efficient and effective method of conveying information to and within a development team is face to face conversation. We decided to put this to the test. We invited three people to try their very best to create an origami snapper. The catch was that we didn't show them what a snapper is and we limited our instructions to a simple written list. Let's see how they got on.

All three of them failed. It turns out that even clearly written instructions aren't always enough. Once we showed them what a snapper actually is, gave them instructions that included pictures and let them work together instead of alone, they created one quickly and easily. And this is really the point here; at work, it can be really easy to just fire off an e-mail or text, asking a colleague to do something, but this kind of communication isn't enough and will probably just end up with everyone being frustrated. Agile thinking tells us that rich communication and collaboration allows us to quickly and easily understand what we need to do, and how we need to do it.

[Why Agile?](https://app.qa.com/course/agile-fundamentals-module-1/why-agile/?context_id=983&context_resource=lp)

Making any kind of change to the way we work can be scary so we should always have good reasons for doing so. So, why do you want to work in an Agile way? In this video, we'll discuss some of the reasons organizations around the world have been adopting Agile to give you a benchmark to see how your reasons compare. [Stateofagile.com](https://www.stateofagile.com/#ufh-c-473508-state-of-agile-report) is a great place to start this conversation. They've been publishing annual reports for over a decade and gather feedback from all over the world on people's reasons for adopting Agile ways of working and other Agile rated stuff. Check out the website when you have a moment.

In the most recent report, the top four reasons for adopting Agile were accelerate software delivery with 74%, enhance ability to manage changing priorities with 62%, increase productivity with 51%, and improve business and IT alignment with 50%. There are a bunch of other reasons too and interestingly, these numbers are not stable. For instance, improving team morale went from 28 to 34% while increase productivity went from 55 to 51%. Also, the kind of reasons are pretty varied, from management, to risk, to quality and morale, people are turning to Agile to improve pretty much everything about their working environment. To understand more of this, we've called a resident Agile expert to get her opinion about why organizations are adopting Agile.

"Why Agile? Well, organizations are finding to be competitive in today's market, they need to innovate which means they need to bring out new products relatively more quickly than the competition and that invariably entails them covering new territory and covering new ground and it means that the customer will generally at the outset not know exactly what they want. Today's businesses don't have the luxury of time with technology reducing the barriers to entry. At a much quicker rate, industries across the board are finding that they are being disrupted by smaller companies who are coming to the market with technology that reduces the barriers to entry. So another key benefit of Agile and what they look to Agile to do is to be able to deliver value a lot more quickly. Because Agile is iterative and is incremental, it allows businesses to deliver value a lot more quickly than traditional methods of delivery."

That's it for this video. There are loads of reasons why people around the world are adopting Agile which should give you a lot of confidence in doing so yourself. Even if you adopt Agile ways of working for one specific reason, you'll probably end up improving other areas of the work environment as Agile touches in pretty much every part of it and in the end, we think this is probably the best reason to adopt Agile ways of working.

[When Agile?](https://app.qa.com/course/agile-fundamentals-module-1/when-agile/?context_id=983&context_resource=lp)

Hi, guys. In this video, I want to talk about when agile thinking becomes a powerful tool in the complex working environments many people face. To start off, I want to make a distinction between defined and empirical processes. Defined processes follow an exact set of stages and every part of the stage needs to be completed before you can move on. An example of this is waterfall project management. Empirical processes start with a hypothesis, tested and observed results, understand the results, and adapt the process to improve.

Agile processes are always empirical and this is especially well reflected in agile frameworks like Scrum that emphasize iterative and incremental work with retrospectives built in, so teams and stakeholders can reflect on the work being done. I'll come back to defined and empirical processes in a bit. But next up, I want to introduce you to the Stacey's Process Complexity Model. This model was developed by Ralph D. Stacey and helps to illustrate issues around decision making in organizations. The vertical axis represent agreement with the horizontal axis representing certainty. As you move up the vertical axis, there is less agreement. And as you move right along the horizontal axis, there is less certainty.

With me so far? The zone within the graph that is both close to agreement and certainty is the simple zone. In this space, everyone understands what the problems are and agree on how they need to go about overcoming them. This is where business as usual sits and there is no need to use an agile framework to start a project or something like that. Although, of course, you can still use an agile mindset here. If there isn't certainty on what the issues are, but there is an agreement on how to deal with it or vice versa, we enter into the complicated zone. This is where a lot of projects sit and any project management method, be it waterfall or agile, can be used here. As we push further out away from both certainty and agreement, we move into the complex space. Here, there is a lot of disagreement about what the issues are or how to deal with them. Agile frameworks like Scrum are great here as they use empirical process control to shorten the feedback loops and react more quickly to a changing situation.

Finally, at the extreme range of certainty and agreement there is only anarchy and chaos. Here, there is very little knowledge of what's going on with almost no agreement on how to deal with anything. In this zone, frameworks like Lean Startup can be helpful as they shorten feedback loops even more by embracing the idea of minimum viable product, MVP. This allows teams to do the smallest amount of work possible to get feedback and then adjust. So, let's summarize them. In an ideal world, everything would be simple. But it's safe to say that most of the issues organizations are facing are complicated, complex, or even chaotic. If you find yourself in the simple zone, both defined and empirical processes should work just fine. After all, everyone knows what the issues are and what needs to be done to address them. However, the further you push away from the simple zone the more defined processes start to struggle. Defined processes need every part of a stage to be finished and signed off before the work can continue. This limits teams who need to deal with complicated or complex decision making.

However, empirical processes like Scrum framework really come into their own here. Teams can react quickly to the challenging situations they find themselves in. As they have the trust of the organizations to self-organize, they can complete work in a way that the environment demands. Agile thinking is especially powerful in the chaotic sector. Teams can focus on creating an MVP and failing fast by building, measuring, and learning. Defined processes are only really good at dealing with work in the simple zone, and to a lesser degree in the complicated zones. They become a blocker as decision making becomes complex and are very difficult to use in the chaotic zone. Agile thinking on the other hand, can be used regardless of how far from agreement and certainty teams are because they are empirical by their very nature.

**SCRUM**

[Scrum Roles](https://app.qa.com/course/agile-fundamentals-module-2/scrum-roles/?context_id=983&context_resource=lp&training_plan=155304)

So scrum is a framework to work in an agile way. Scrum teams are self-organising and cross-functional. They decide who will do the work and how it will be done. But how do they organise themselves? It's actually pretty simple. The key thing is that work in a scrum environment is divided between three roles and each role has its own responsibilities. The three roles are: product owner, scrum master and the development team. In this video I want to discuss each of these roles and talk a little bit about their responsibilities. I need to talk about some scrum artifacts and events to do this properly. So don't worry if that sounds like jargon right now, I'll explain everything as we go along.

So let's start with the product owner. This is a single person, not a committee or some kind of group. They are in charge of managing the product backlog. The product backlog is a scrum artifact. And it's basically a list of work that needs to be done. Now it's the product owner's job to understand and engage with the stakeholders to understand what needs to be done and create that backlog, which is then used and pulled into the sprint planning. They also need to prioritise that backlog. This is just making sure that the most important work is at the top of the backlog with the work that's least important at the bottom. And maybe the least important stuff will get done at some point but because it is the least important, it will sit at the bottom of the backlog until the team can around to it.

During the sprint planning the scrum team will pull as much of the product backlog as they can get done into the sprint backlog, which is just the list of work they're looking to get done by the end of that sprint. So the product backlog has to be well organised and managed by the product owner. To do this, the product owner has to be empowered by the business. It's really important that there is a commitment within the business or the organisation that empowers the product owner to take control of the product backlog and make sure that their team is able to self-organise in the way that they need to.

Next up I want to talk about the scrum master. So this is a servant-leader who is there to help the entire team achieve the scrum goals and work within scrum. Being a servant-leader means that they lead the team by putting the team first and helping them to adopt the values of scrum. Being a servant-leader means that the main goal of the scrum master is to serve the scrum team and help guide them to be the most effective and efficient team possible. Because of this, it's really important that the scrum master really understand scrum and can help guide the team. The scrum master will also support the product owner with their responsibilities in terms of managing the backlog as well as, supporting the development team. You can think of them as a kind of support for the entire team. And they'll often do this by facilitating scrum events like the daily scrum or the sprint review and retrospective.

Last up we've got the development team. Their responsibilities may seem pretty obvious, I mean, it's in the name. It's worth noting though, that people in the development team aren't necessarily coders or technical people. They could be designers or writers or any role that contributes to the development of the product or service. Development team is just the term we use to describe the role of the people who actually make the things within scrum. The development team are the people who are creating the product or service and delivering done increments at the end of each sprint. Now what done looks like is really up to the scrum team as a whole to decide, but it should essential be a usable piece of software or product or service.

So how do they create this increment? There are two main ways. The first is that they're self-organising, so they choose how they want to work. It's up to them to decide how they'll deliver the increments. So they'll need to have discussions and come to a consensus as a team. So when they take an item from the backlog, they have to look at it, understand it and then be able to really quickly and easily turn that into some kind of workable increment. This brings me to the second way they're able to deliver the increment and that is, the development team needs to be a cross-functional team. Basically this means that within the team there are all the skills that are required to deliver that increment. Now this is a really tricky thing because teams which are too large might have loads of different skills but then they'll really struggle to work in a scrum environment because communication channels will become too complex. But teams which are really, really small won't have all those cross-functional skills.

So the key thing really, is to have smaller teams with people who can do multiple different things. Essentially what you really want to aim for is people who are specialists in a number of areas, often referred to as generalists, rather than owing a single area, because otherwise you're going to have to hire more people and the team will grow too large. Scrum teams are ideally made of six, plus or minus three people, and any more or less, they become ineffective. So just to summarise the three roles in scrum then. Product owners manage the work to be done. They engage with the stakeholders and they make sure that the product backlog is really well organised. The scrum master supports the entire scrum team and working within the scrum framework and acts as a coach and servant-leader to the entire scrum team. The development team self-organises in a cross-functional way to create and deliver increments for each sprint.

[Scrum Artifacts](https://app.qa.com/course/agile-fundamentals-module-2/scrum-artifacts/?context_id=983&context_resource=lp&training_plan=155304)

Scrum artifacts are a way of thinking about the work that is being done or needs to be done by a scrum team. The main shared characteristic of scrum artifacts is that they are really, really visible. So all the stakeholders involved in a project can see how work is being done. What has priority and what the outcome of any given sprint will be. There are three artifacts. The product backlog, the sprint backlog and the increment.

So lets go through each of them, one at a time. The product backlog is owned by the product owner, and is generally made up of things like epics, user stories, tasks, bugs and other things like that, but essentially these are all things that represent work to be done.

The ones I'll focus on in this video, are epics and user stories. So epics are large amounts of work that can be broken down into lots of smaller user stories, tasks, et cetera. User stories on the other hand are that smaller piece of work that stands on its own, and in fact the best way to think of these, are as simple stories that tell the development team more about why they need to create a specific product or service. So it really gives them the context they need in order to understand the requirements, but also the flexibility to choose how they go about achieving that story. User stories should help a team understand the need for a certain product or service, but exactly what that product or service needs to look like, is still up to the development team.

Development teams are self organising units. So they need to decide how they will approach any user story and the best way they can solve that need. Next up the sprint backlog which is owned and forecast by the development team during the sprint planning session. They look at the product backlog and say; "All right, we believe we can achieve X, Y and Z, by the end of this sprint. So we'll pull in all those epics or user stories or whatever into the sprint backlog." This can obviously be quite difficult and teams tend to get better estimating what work can be pulled into a sprint backlog over time. The important thing is that teams have a way to decide together if a product backlog item is ready to pulled into the sprint backlog.

The last scrum artifact that I'd like to talk about, is the increment, and this is really quite simple actually. An increment is just all of the work that has been completed during a sprint plus all of the related work that was done in a previous sprint. Work can only be included in an increment though, if it is done, but when is work done? Well work is done when the organisation says it's done, but if this doesn't come from the organisation, then the team agrees that it's done. So as a team, you need to create a common understanding of what done means for you, because teams are self organising and everyone is doing different kinds of work, it is really important that teams have that discussion and understand what they believe done looks like. And this is really important for two reasons: Firstly, because a loathsome to deliver the increment at the end of the sprint, and secondly so that they can properly estimate the effort for future work. There are a few general ideas a lot of teams use, to help them define that. Like the acceptance criteria being met, a p.o being signed off or, the audit being signed off. In the end it's up to the scrum team to decide what done looks like.

That's it for this video. Basically there are three scrum artifacts. The product backlog, sprint backlog and the increment. Each of these has quite a few elements and to be an effective scrum team, you need to master all of them.

[Product Backlog Items](https://app.qa.com/course/agile-fundamentals-module-2/product-backlog-items/?context_id=983&context_resource=lp&training_plan=155304)

So you want to work in a scrum team and you know that scrum teams use product backlogs to capture all of the requirements. But what does an individual product backlog item actually look like? There isn't really a simple answer here, as every team needs to figure out their own product backlog, but a common way of creating product backlog items is to create user stories. User stories are transparent and easy to understand, and they're created by the product owner. In this video, we'll discuss a quick and easy method you can use to create user stories using two simple models together. First up we have the three Cs, which is cards, conversation, and confirmation. And then to determine if those cards are actually ready to be used, we can use the INVEST model.

Let us talk about the first of the three Cs, the card. It doesn't matter if cards are made physically on sticky notes, or written up on a board, or on a digital tool. The important thing is that they are simple and easy for anyone who reads them to understand. Cards should always ask and answer three core questions: Who is the user? What do they want? And why do they want it? For example, as an employee, I want a system to book leave so that I can spend more time with my family. This story gives us a simple explanation of the requirement, but as you can clearly see, it doesn't provide specifics of exactly what the solution should look like. In other words, the user story tells us generally what the solution needs to do, but not exactly how it should do it.

Okay, now that we've got a card, it's time to move onto the second C and have a conversation. The conversation should be ongoing, and happens between all stakeholders from when a development team first sees a user story during the development process through to the sprint review and demo, everyone involved needs to be talking. This ongoing conversation helps avoid misunderstanding and lets everyone collaborate more easily. Last up, the development team have realised the user story and [delivered the increment](https://app.qa.com/course/agile-fundamentals-module-2/incremental-delivery/), and it's time for the final C to be used, confirmation. It's up to the product owner to confirm that yes, the increment meets the requirement laid out in the user story, or no it doesn't. If it hasn't, the story may need to be worked on, and if it has, the team will be able to move onto new work.

Everyone in the scrum team should know what done looks like, so getting the product owner's confirmation shouldn't be difficult. So now you know how to create user stories and how to use them in an Agile environment using the three Cs. But how do you know if a user story is actually ready to be used? A simple way to figure out if a user story is ready to be pulled into the sprint backlog is to use the INVEST model. INVEST is a simple acronym, so let's run through it quickly.

So the first one, independent. Stories need to be independent of each other. They cannot all be related to one another, as we don't want to fall into a scenario where we're blocking each other in a team. All of the user stories need to be able to be worked on potentially even simultaneously. User stories need to be negotiable. This means they are not demands or contracts from the product owner to the development team. Instead, they should capture the imagination of the person who's to work on the user story, and allow them to interpret and discuss how they want to deal with it.

Obviously they need to be valuable. We don't want to be creating any kind of product or service that's not valuable for any of the stakeholders involved. They need to be estimable, so at the point that you're going into a sprint, teams can estimate how much effort each user story will take to action. It's important to have some way of understanding the capabilities of your team, and how much they can achieve within a sprint so you can estimate how much effort any given user story will take to action.

They should be small. It's generally a good idea to have really small user stories, because if they're nice and small and compact, they're probably gonna be estimable and achievable in a single sprint. The smaller the better.

Last up for this we should be able to know how we can test the deliverable, and that it does what we think it should do. And that's it for this video. If you want to capture your requirements for a product backlog, you can use the three Cs and the INVEST model to create simple and transparent user stories which give your development team the freedom to work in a self-organising and cross-functional way.

[Incremental Delivery](https://app.qa.com/course/agile-fundamentals-module-2/incremental-delivery/?context_id=983&context_resource=lp&training_plan=155304)

Traditional ways of working, like waterfall, tell us to complete all the work in a stage before we move on to the next one. At the end of this process, we finally deliver something to our customers, and we hope that it meets their needs. Agile thinking is changing this and asks us to deliver incrementally. An increment is a done product or service that we deliver to our customers at the end of an iteration. But how do we do this? And is incremental delivery really that much better than traditional methods? To find out, we invited three people to do the coin challenge. Let's see how they did.

So what did we just see? Our participants flipped each coin and created a stack of coins before they could pass it on. Each coin flip represented work being done, with the stacks being phases of our waterfall process. Working this way meant that the people down the line were blocked by the people who were flipping and creating stacks before them. They had to wait until a coin stack finally came their way and were only working once an entire phase had been completed. This also meant that it took quite a bit of time for coin stacks to make their way all the way to the very end of the process. Now, let's check out how this same process could be done incrementally.

As we just saw, incremental delivery was quite a bit faster than its waterfall counterpart because our participants were passing on each coin as soon as it was flipped. This meant that everyone was able to get involved in incremental delivery much more quickly. Crucially, the same amount of work was being done. The coins were still flipped the same number of times in both challenges. The big difference is that the coin stacks were only created at the end of the process when work is considered done instead of trying to get a done state for every phase of the project. Working incrementally helped our team to deliver the done increment more quickly by getting the whole team involved in the work sooner. Agile ways of working and thinking encourages us to work in this way because it means that we can get a working product or service into our customer's hands quickly. We can get their feedback and move into the next iteration to continually improve.

[Scrum Events](https://app.qa.com/course/agile-fundamentals-module-2/scrum-events/?context_id=983&context_resource=lp&training_plan=155304)

The scrum framework helps us to deliver work in an agile way, and it does this by giving us five events to organise our lives and create a cycle that allows us to work at a consistent rate. The five events are, sprint planning, the daily scrum, the sprint review, the sprint retrospective, and of course, the sprint itself. The scrum events are all facilitated by the scrum master, and this is one of their major responsibilities. In this video, we'll go in depth into each of them to help you take what you've learned here and move into a scrum environment.

So first up, we've got sprint planning. The sprint planning happens at the start of every sprint, and it should probably be about between four and eight hours. Four hours for about a two week's sprint, and a maximum of eight hours for a month long sprint, and of course, sprints really shouldn't be for any longer than that. During the sprint planning, the product owner will present the product backlog to the development team and the development team will pull product backlog items from the backlog that they think they can achieve in that sprint. Once they have done this, they'll estimate the effort they need to put into each item, and this effort is represented by things like story points or T-shirt sizes. I won't go into any detail on these right now, but they'll essentially figure out how much effort they'll spend on each item and once they've got all those items in, they've created their sprint backlog. Once each requirement has been pulled into that sprint backlog, there will also often be a task for individual members within the development team to divide or break up requirements into specific tasks that need to be achieved.

The scrum team as a whole will then set a sprint goal. This is basically the objective of the sprint. And that goal is really important to just keep everyone focused on what they're trying to achieve for that sprint. To keep momentum during the sprint, the scrum team use the second scrum event, the daily scrum. This is a very short time-boxed event. Usually only lasting no more than 15 minutes. The daily scrum is used to make sure that everyone in the development team is up to date with each other, and helps everyone understand the status of the sprint. Normally, every member of the development team will answer three simple questions. What did I do yesterday that helped the development team to meet the sprint goal? What will I do today I order to help us achieve the sprint goal? And, do I have any impediments or blockers that are stopping me from achieving what I need to achieve? This conversation allows everyone to quickly understand where the whole team is at. And if anyone needs help, people can jump in, collaborate and support each other. We don't need to use these exact questions. Any will do as long as they allow everyone in the team to be up to date with the sprint progress. It's also very important that the daily scrum isn't thought of as just a progress report. The focus and point of the daily scrum is to synchronise and recommit to the work in line with the sprint goal and to keep the momentum of the sprint.

Towards the end of the sprint, we go into the third scrum event, the sprint review. In it, the development team will present the increment they have been working on to stakeholders and collaborate on what needs to be done going forward. Like the other sprint events, this is a time-boxed event that should last no more than four hours in a month long sprint. So sprint reviews are really important collaborative events to demo what has been achieved and to help keep everyone who's involved working together.

At the end of the sprint, we move into the fourth scrum event, the sprint retrospect. During this, the team has the opportunity to really reflect on their past sprint and talk about what worked well and what didn't work well. It should be time-boxed to no more than three hours in a month long sprint. This allows the team to focus on their ways of working and collaborate to improve going forward. So all of the lessons learned can be captured at that retrospect potentially but is not necessary that this is written down again. Going back to the agile principles, we don't need comprehensive documentation necessarily, but as long as everyone understands those lessons and can take them forward in whatever way, this is probably enough. The scrum team also needs to make a concerted effort to take any lessons learned and discussed in that retro into the next sprint. The retrospective is about continuous process and improvement and we need to take what we've learned into the next sprint planning session.

The final scrum event we need to talk about is the sprint itself. Sprint contain and consist of the sprint planning, daily scrums, the development work, the sprint review, and the sprint retrospective. The sprint is really the beating heart of scrum and all the scrum events take place in it. In it, the development team will create whatever deliverables they're looking to create or whatever services they're trying to deliver or craft whatever they're trying to do. A sprint is a time-box of one month or less. And while it's typically two to four weeks long, it can be as little as one week.

So yes, that's our five sprint events which help us to work in an agile way within scrum. We start with sprint planning where we pull product backlog items into the sprint backlog and estimate the effort needed to complete requirements. We use the daily scrum to keep our momentum going everyday and support each other when we face challenges. At the end of the sprint, we present our work to all our stakeholders and collaborate with them in the sprint review. We then use sprint retro to reflect on the past sprint and to see how we can improve our ways of working going forward. All of this happens in an overall time-boxed event, the sprint itself.

[What is Value](https://app.qa.com/course/agile-fundamentals-module-3/what-is-value/?context_id=983&context_resource=lp)

In our working environments, we're always looking to create value, but what is value, why is it important, and how is it created in the first place? In this video, we're going to take a short look at the idea of value and try to answer some of those questions.

"What's valuable to me is time. It's the one thing I don't have enough of. I never have enough time to do all the things I wanna do every day."

"Honesty is valuable to me. I want to be able to trust those I work with and spend time with on a day-to-day basis."

"Sleep is valuable to me so that when I wake up, I'm refreshed, rejuvenated, and can get on with my day."

As you can tell, the problem with value is that it's subjective. Any number of things could provide the basis for what value is. Perhaps it's differentiating yourself from your competition or eliminating your competitors' advantages. Maybe it's reducing costs or reaching new markets. Maybe it's just meeting basic requirements like achieving compliance. The important thing, though, is that whatever value you're aiming for must be clear and obvious to the agile team. In other words, everyone who's involved in creating a product or service needs to understand exactly what value they hope to achieve with it.

There are a few reasons why value can be really, really important. So the first is the prioritisation of work. During sprint planning, the development team are looking to pull items into the sprint backlog from the product backlog. Work needs to be prioritised in the product backlog by the value it'll provide to the stakeholders and, of course, to the organisation.

Next up, understanding the value we are trying to create provides us with a focus. This is why we're doing what we're doing. It gives us a reason, something to work towards. Understanding what value we're trying to create also helps us align work we are doing with other work we have done and with the rest of the organisation. Let's say we're trying to create something that must be compliant, but your organisation generally tries to create products and services that differentiate them from the competition. By aligning to your organisation's vision for value, you can know that you need to create something that doesn't just meet compliance. It also needs to be unique and interesting to consumers.

Understanding value also removes waste. We're not going to do work that isn't valuable because we understand what value we're trying to achieve. This means we won't waste resources on things that don't add the value we need them to. Last up, if you understand what value you're trying to create, you can get a clear idea of what the return on the investment should look like. When an organisation invests resources into a [project](https://app.qa.com/course/agile-fundamentals-module-3/what-is-a-project/), they expect a return of some sort. Often this will be in the form of revenue, but it could be something else. Whatever it is, as long as the agile team understands what the goals are, they can align themselves to help achieve it.

Well, the link between agile and value is that agile builds in an understanding of value from the get-go, from the requirements and the user stories. User stories are crafted around our understanding what the customer wants and what the perceived value of that particular product or feature is. There are other areas in which value is clarified. There are other opportunities such as that, demos.

During sprints, the customer is invited to see and confirm that value has been created from the work that the development team have been involved in that sprint. The feedback that they get from the client confirms that value has been created. Overall, agile ways of working are one of the reasons why they are the best ways of generating value is because agile is structured in such a way that work is delivered in an incremental and [iterative way](https://app.qa.com/course/agile-fundamentals-module-3/iterative-development/) that requires value to be created every sprint and can be validated by the customer as being produced every sprint. So agile ways of working, therefore, give value faster than sort of big bang waterfall approaches to delivery that results in value being created at the end of the project.

-So those are some of the reasons why understanding value is really important, but there is one last thing we need to discuss before the end of this video, and that is the idea of the co-creation of value.

Previously, the idea of value was pretty linear. We as an organisation create something we provide to our stakeholders, and that's value. But today we talk about the co-creation of value. This is the idea that we as an organisation create value by collaborating with our stakeholders. We work with them, take on their feedback, and constantly look to deliver work in line with their expectations. They, in turn, are always looking to be in conversation with us and will tell us what they want. We take on what they say they want, and we work together to create that product or service.

That's it for this video. Although co-created value is subjective, it still needs to be well-defined. Having a well-defined idea of what value is for the organisation and for each product or service has many benefits impacting ways of working, all the way through to strategic thinking.

### Lectures

* [How Do I estimate?](https://app.qa.com/course/useful-agile-tools/how-do-i-estimate/)
* [Measuring Progress and Kanban](https://app.qa.com/course/useful-agile-tools/measuring-progress-and-kanban/)
* [What is a Project?](https://app.qa.com/course/useful-agile-tools/what-is-a-project/)

[Measuring Progress and Kanban](https://app.qa.com/course/agile-fundamentals-module-3/measuring-progress-and-kanban/?context_id=983&context_resource=lp)

So, now you're looking to work in an agile way, but how do you actually monitor the progress you're making in an agile environment? Well, there are two really common ways to monitor progress and in this video we're going to discuss both.

The first one I'd like to mention quite briefly is the burn down chart. This is simply a chart that plots the estimation of effort to get work done against the number of days in a sprint to help us visualise our progress. On the horizontal axis you have time, or the number of days in the sprint, and on the vertical axis you have the number of effort points you have pulled into the sprint. In a perfect world there would be a straight line down from the total estimation to the last moment of the sprint. Of course this isn't how things actually work. Sometimes requirements take a couple of days to complete and other times work gets done in a flurry. This means that the burn down chart line you created will have horizontal and vertical lines; horizontal while work is ongoing and vertical when it's completed, creating a graph that looks more like this.

A second way that we can monitor and visualise our progress is by using a Kanban board. So Kanban comes from the Toyota Production System, TPS, and translates to signal card. It's a just-in-time flow-based system to help deal with the buildup of inventory, but ultimately it works well in an agile environment too. So Kanban boards are quite simple actually. You can think of them as a system of flow. So imagine we've got a few headings, something like to do, in progress, in test and done. Now all we have to do is create our requirements and pull them across the Kanban board as we go.

While this is a really easy way to monitor progress, using Kanban properly also means that we need to embrace the four principles and five practices of Kanban. Let's go through each of these quickly. The first principle is start with what you do now. Don't try to change anything at first. Start with where you are and map your tasks into the Kanban board. Next up, agree to pursue incremental, evolutionary change. Make small changes over time and monitor them. Don't look to make sweeping changes as soon as you start using Kanban. Our third principle is respect the current process, roles, responsibilities and titles. Current business process, roles, responsibilities and titles exist for a reason. Respect them and use them going into Kanban. Look to make incremental changes to these over time instead of sweeping up front changes. The final principle is encourage acts of leadership at all levels. Simply put, everyone in the team can show leadership through their day-to-day work. Everyone should look to take the lead when they're best placed to do it and feel empowered to do so. Doing this is really a part of continued improvement mindset and helps individuals and teams get the most out of Kanban.

Great, now that we've covered the Kanban principles, let's take a look at the Kanban practices. The first is that you have to visualise the workflow. You can do this in a Kanban board and by using the requirements to populate it. These could be accounts payable, recruitment, expenses, anything really. The next is limit work in progress. They key thing here then is that you should only be working on whatever you can achieve at any one point. Don't look to have lots of work in progress, instead have less work in progress and get that done as quickly as possible. This allows us to limit context switching or task switching, helping us to keep focused and limiting any wasted effort. Kanban is a flow-based system, so as a team you need to manage the flow of requirements through it to make sure that no value stream is being blocked. To do this constantly measure and manage the flow through the system, make incremental changes to see if you can get more value and iterate. Make process policies explicit. It's very important that everyone understands exactly how things are going to be done, why they're done that way and to keep this at the forefront of everyone's minds. These could be in the form of working agreements, which must be visible to everyone. Lastly, improve collaboratively. It doesn't help to improve only yourself. As a team you need to be looking to improve.

So those are two really fantastic ways you can measure progress in agile environments. Burn down charts are a quick graphical representation of the progress that has or hasn't been made. Kanban boards are a visual representation of all the work that is being done and represents a workflow.

[AgilePM/DSDM](https://app.qa.com/course/agile-fundamentals-module-3/agilepm-dsdm/?context_id=983&context_resource=lp)

When working in complex environments we may need an agile framework that deals with the full project lifecycle and agile PM DSDM is great for this. In this video we're going to take an introductory look at the framework elements which include the philosophy, principles, pillars and foundations. While you learn about agile PM DSDM it's important to keep in mind the core philosophy of the framework that any project must be aligned to clearly defined strategic goals and focus upon early delivery of real benefits to the business.

Great, so DSDM has eight principles that underpin the philosophy and create the mindset you'll need when working with the framework. Let's take a look at all of them quickly.

First up, focus on the business need. The project should always satisfy a business need first and foremost and this will keep it aligned with the business strategy and deliver true business value.

Next, deliver on time. Delivering a solution on time is a very desirable outcome and sometimes underpins the entire rationale for delivering the project. DSDM identifies that the safest place to trade on this type of project is to lower priority features so we prioritise features and time box our work. This way, if we have a delay, we can trade features to keep on time.

The third principle is collaborate. Working together with your team and stakeholders as much as possible will help you to work through the project in the most efficient way.

Fourth, never compromise quality. The quality of any service or product that you deliver must always be a top priority. Agile PM DSDM always delivers a fit-for-purpose solution as a reduction in scope should never be a reduction in quality.

Our fifth principle is build incrementally from firm foundations. Incremental delivery is a key way in which most agile frameworks deliver solutions and agile PM DSDM is no different. We try and deliver something usable into the business so we can get early business benefits.

Sixth, develop iteratively. Iterate development is about feedback loops. This means we have a conversation with stakeholders about the work we're going to do, then take action. Once we finish the work, we'll review it together to make informed decisions about what to do next.

The seventh principle is communicate continuously and clearly. Communication should never suffer in an agile environment and communication is also a big part of collaborating.

Last up, demonstrate control. The people working in the project need to demonstrate control, both to their colleagues and their clients. This builds faith in the project and keeps momentum.

Fantastic stuff. The next thing you need to know about are the four pillars of DSDM. The four pillars are there to uphold the principles. They are process, people, products and practices. The process gives us a full end-to-end project lifecycle. The people element identifies the roles and responsibilities of the project management team. The products area brings into focus the management control of project information and last up, practices help us to understand techniques we can use while working on an agile project. Always remember that DSDM is built on a foundation of common sense and pragmatism, which means don't get bogged down with the theory, practically apply the framework.

And that's it for this video. By using the DSDM philosophy, principles, pillars and foundations, organisations can manage projects in an agile way and tackle complex issues with a systematic and pragmatic approach.

[What is a Project?](https://app.qa.com/course/agile-fundamentals-module-3/what-is-a-project/?context_id=983&context_resource=lp)

Projects are a universal way of doing work, and you'll find them in every industry. But what is a project? Well, a project is a temporary endeavor, if you'd like, set to deliver a unique change. The key point here is that it has a clear start and end point. At the beginning of the project, we all need to understand what is most important. There are four key factors within any project to keep in mind when you're trying to understand what is most important to that project. First, the time. How long can we spend on this project? Next, how much money can we spend on delivering the project? After this, we have quality, and of course, the quality can vary. Finally, scope, and by scope, I mean specifically what the deliverables of the project will do or in other words, how many features it will have. Now, in Agile ways of working, we tend to believe that a large percentage of features that we deliver won't actually be used. So often, the safest variables to focus on is the scope. Now, there's a fantastic way to figure out just exactly how the scope might change, and it's a process called MOSCOW Prioritisation. To help explain this in a little bit more detail, we've brought in one of our resident experts who has a lot of experience using MOSCOW to help Agile teams deliver projects successfully.

- MOSCOW is a tool that is used to prioritise a list of requirements or a list of tasks in order of value to the customer or the end user. So M within the MOSCOW acronym stands for must haves which are features or tasks that are critical, ie, without those tasks or features, that product or that service would not be viable. S stands for should have, which stands for a feature that is, if it were possible, it would be nice to have it included and will add some value to the business. C stands for could have, which is a feature or an aspect or a task that is not critical, but should there be enough time and budget, it would be good to have included so that they are nice to have. And then W stands for won't have. It's typically a requirement that was requested that is not deemed a high business value and neither is it deemed critically urgent, so it could easily be dropped off as in not done as part of the list of things to do. An example of where you could apply, a good of that example might be the sort of prioritisation an event planner would undertake to plan a conference or plan an event. The event planner might list several tasks such as venue, procuring a venue for the event, might want that venue to be within half a mile radius of an underground station. Might also want to have doggy bags to give out branded materials to delegates who attend the event, might want to put on refreshments and part of those refreshments could include sandwiches, buffet, or alcohol, and so if you assume that the list that I have spoken about is ranked in the order that the event planner has in mind, then the venue would be a must have, for sure. You can't have an event without a venue, and ideally, most events I've been to are in close proximity to an underground station so those two criteria would be must haves, perhaps goody bags to be the should haves, because definitely they would be a very nice to have accompaniment to any event, and alcohol would be a won't have. Food would definitely have a higher priority, maybe sandwiches, at least, would be a higher priority than alcohol. So that would kind of be an example of how a MOSCOW prioritisation approach might be applied by an event planner.

- So, projects are a universal idea used to deliver work. They tend to have four variable qualities, time, money, quality, and scope. In Agile projects, we use MOSCOW to ensure on time delivery. When a project becomes delayed or over budget, we reduce the scope instead of adding time or cost, and that allow us to guarantee on time delivery.

[Iterative Development](https://app.qa.com/course/agile-fundamentals-module-3/iterative-development/?context_id=983&context_resource=lp)

In Agile environments, teams need to work iteratively. This means that they have a set amount of time to deliver an increment. To do this, they need to plan, design, develop, test, and deliver all within that same set iteration. Once they've delivered, they can look to improve on the increment in the next iteration, but working in this way doesn't come naturally to most of us, and has to be learned. We invited three people to take on the marshmallow tower challenge, and see if they could work in an iterative way.

Our participants, like most people, didn't manage to create a freestanding structure with a marshmallow at the highest point in the time they had. There are a few reasons for this. The first is that they didn't have a bias to action. Instead, they spent a lot of time coming up with a plan of action instead of building. The second is that they were too ambitious, trying to create a really tall tower from the very beginning. The last reason is that they didn't fail fast and iterate. If they had been iterating, they would have started with the simplest possible version of the tower. Then, iteration by iteration, they could have created a taller and taller tower.

Agile thinking is all about failing fast and delivering a minimum viable product. In your workplace, start with the very basics, then iterate to continually deliver better and better increments to your customers.

[How do I Estimate](https://app.qa.com/course/agile-fundamentals-module-3/how-do-i-estimate/?context_id=983&context_resource=lp)

To work in an agile way, we have to be able to estimate the amount of effort required to complete any requirement. But how do we do this? Well, the first thing to remember is that an estimation is always at its best, really just a guess. If we look at Steve McConnell's cone of uncertainty, which plots time against the error of estimation, we can see that initially, we will probably have a very large variation between the highest estimation and the lowest estimation. Over time as we deliver increments, our estimations will become much more accurate. So, the key thing then is, whenever you work with stakeholders or at the start of a project, is that you let them know that you won't be able to give them accurate dates, times, costs, etc. For exactly when the project will be delivered until the end of at least the inception stage. Instead, you need to say, we're going to work iteratively on incremental way to deliver you some of the project as we do that, we'll be able to really get a stronger idea of exactly how much longer will take us to complete the project.

When you're estimating, it's extremely important that you do so in a relative way rather than an absolute way. Have a look at these three cubes for instance. I can't tell you anything about them absolutely, other than, they're cubes. But I can tell you that the cube on the furthest right seems to be the largest one, and the one on the farthest left seems to be the smallest. I can even say that the biggest cube seems to be about four times as large as the smallest one. It's easy for me to estimate the cube size relative to each other, but if you ask me how much the smallest cube weighed, absolutely in kilos or pounds, I don't have the information to tell you this. All I can say is that it looks about four times less than the biggest cube. We can easily apply this relative estimation to our requirements. We can discuss a project, compared with the work we've done before and estimate how large it is based on this.

Okay, so the question is about relative estimation versus absolute estimation. Well, if I was to brought before you 10 people of different sizes, different body matters, and asked you to estimate their body mass, wouldn't be able to do that with any degree of precision and there's a reason for that. Humans aren't designed to be able to, tap in with any degree of precision, people's body mass, and that is because you have to take into consideration the height, and it's difficult to estimate that, you know, visually their weight and then complete that into a body mass estimate. But if I were to ask you to, sort of grade, every single person of those 10 people in terms of their average body size in terms of small, medium, large, you would be able to do that, and that is because people can, you know, are able to make those kinds of estimates, because it doesn't require precision.

The next thing we need to do, is attach numbers to our estimations. This is really helpful when it comes to figuring out our capacity and velocity as a team. One of the best ways to give our requirements and estimation in numeric form is to use the Fibonacci sequence. This is because the sequence gets exponentially bigger the further up it goes, helping us to represent larger projects versus smaller ones more easily. Every number in the Fibonacci sequence is the sum of the two previous numbers, so, one, two, three, five, eight, 13, 21, etc. It's worth noting that planning poker games sometimes use an alternate sequence, that's pretty much the same, except that it goes to 20, not 21. And from 20, it goes up to 40, and then 100. It still roughly follows the same sequence and serves the same purpose. So, maybe we agreed that our smallest cube is one and our largest is an eight. Going forward, we now have a benchmark to estimate all cubes we encounter or in work terms, we have a way to put a number that represents the effort it will take for us to complete that work. Great! So let's talk about velocity. Velocity is simply the number of points we get done in a specific time. Let's say two weeks, maybe our velocity is 200. If we go into sprint planning and come out with a sprint backlog that has 250 points in it, we should know that we have overestimated how much we will be able to do, which means some of that work probably won't get done. We should only ever plan to work on as much as our current velocity allows for. Once our velocity is set and steady, we can also start to have a better idea of exactly when a project is likely to end.

A great way to go about your estimation is to play a bit of planning poker. So this is really quite simple actually, everyone in the team gets a set of cards with a Fibonacci sequence on it. The stakeholder or product owner will present the requirements to us and each person will use their cards to vote for how much effort they think is involved to complete it. There will probably be some difference between the team on this, so we need to discuss it amongst ourselves, and then vote again until we can come to as close to a unanimous vote as possible. This method allows the whole team to understand what the work entails and to collaborate on understanding and estimating, rather than this sitting with one or two members of the team. So yes, that's that for this video. Estimation is just about a relative number you can use to understand the effort required in doing something. Relative estimation allows us to be as accurate as we can be, but still gives us the flexibility to change over time. It also helps us to build a velocity, which helps us to plan our work going forward. Planning poker is a great way to go about your estimations as it promotes collaboration and joint responsibility for the work we can do as an agile team.

[Your Agile journey](https://app.qa.com/course/agile-fundamentals-module-3/your-agile-journey/?context_id=983&context_resource=lp)

Whenever we start something new, or want to do things in a new way, it can be pretty daunting. We have to learn new rules, change our mindset, adopt new principles, and take on different roles. Of course, this applies to agile, too. If you're feeling this way, there's a cool technique you can use to visualise for yourself and your team where you are, and where you're going. It's called Shu Ha Ri, and the idea comes from Japanese martial arts.

During the first stage, Shu, we need to really immerse ourselves in the discipline. Imagine we want to become musicians. To start with, we need to take lessons from a master, practice, and do exactly what they say if we want to improve. We mustn't change anything to suit ourselves during this stage because we're still learning the ropes and basically, we're not qualified to know what's best yet. After a bit of time, we'll be really familiar with our new way of working. At this point, we can start to question some of the things we've been doing, to make sure they're actually working for us. If we continue with the musician analogy, at this point, we would be able to start experimenting with the way we play. We'll still probably stick to the music, but maybe we'll start to play a little more arrowy and bring our own flair. We won't change anything too much, but we'll start to make a few small innovations and test how well they work for us.

Finally, we reach Ri, and we've become masters of our new way of working. Now, we can fully depart from the laws and rules we followed so closely in Shu, and create our own rules. As musicians, we are now fully in control of our music, and can write songs and perform as experts. We are masters of this new way of working, so we know what works for us, and what doesn't.

Now, this may not seem too complex, and it's not, but it's a great way of thinking about how your agile journey will probably progress. Expect a few hiccups along the way, especially during Shu, when you look to change the way you work. Agile might seem difficult, but as you keep at it, moving through Ha, and eventually Ri, you'll be in a new perspective, and become that much better at what you do.