Help Us Translate

0:00 In this video we will discuss an overview of Agile. Much of what we discussed lightly here will be covered in more detail later in the course. We will start by describing Agile. Agile is an approach to managing and working on projects. This means that it relates to both planning and doing the actual work. In other words, it includes aspects of project management and product development. Agile is a simple approach to managing complexity. This is an alternative to addressing complexity with complex project management and heavy upfront planning which often increases the overall complexity and risk of the project. Agile approaches can be applied to any type of project. The classic use of Agile is in software development. The use of the word Agile for this approach was coined by some leaders in the software industry in 2001. We will discuss this more when we discuss the Agile manifesto. Most modern software projects run using Agile approaches. If you name any company that produces software, there's a very good chance that some or all of the company is agile. Companies that are using old-fashioned approaches are usually at a disadvantage to their competitors. Agile techniques can also be used to manage service tickets. These can be tickets related to software issues or any other type of service ticket such as a restaurant order. Inside of a company, Agile approaches can be used in any department. For example, the human resources or HR department might use Agile techniques to manage the flow related to the new hire process. As another example, the marketing team might use them for managing a marketing initiative. Agile approaches do not have to be used in corporate situations. For example the tasks that need to be done at a picnic can be managed this way. If you are writing a book, Agile techniques can help you focus and write a better book. Agile can apply to team projects and to personal projects. Finally, even if you are building a rocket, you can use Agile techniques. SpaceX uses Agile approaches to continuously improve their products. You can see that Agile projects apply to working on physical products as well as on knowledge work like managing an HR process. In this course, we will generically use the word product to refer to both physical and less tangible deliverables of a project. Each Agile project is unique but they usually have some common characteristics. We will discuss them briefly now, then discuss each of these characteristics more throughout the course. Agile products are built incrementally. This means that instead of planning, building and releasing the product all at once, small valuable increments of the product are planned and released in a succession. The definition of release may vary by project. For example, an increment may actually go to the customer or the increment may be releasable subject to a business decision. Either way, each increment should be actually usable and valuable to the customer. Agile projects are iterative. This means that you continuously obtain feedback, learn and improve the product and the process of building the product. Each increment provides tight feedback from both the users and from inside the team allowing continuous improvement. Agile projects relentlessly focus on value. The team is always working on what it currently considers to be the highest value parts of the project. After the current work, the team again decides what is the next highest value thing to work on. This is based on continuous feedback from the team and from users of the product. Agile projects have an empowered team. This means that the team decides how to organize and how to accomplish its work. This is not a command and control system. You can think of the team as containing a distributed brain where each member continuously helps make decisions. In many cases, the members of the team have the most current knowledge and information and are in the best position to make a good quick decision. An Agile approach to managing projects is often chosen because the results are better than more traditional approaches. This is especially true when the project involves a lot of uncertainty. Let's start by looking from the customer's perspective at the benefits of products created by Agile teams. The customer is receiving a desirable product. This is because the customer consistently provides feedback in the development of the product. This feedback comes directly from customers or indirectly by evaluating the data related to product usage. The team is always focused on delivering the highest priority features which means customers are getting the most important features earlier than they would using traditional project management techniques. Agile teams are consistently matching what they build to what the customer is asking for. They embrace and respond quickly to change so even recent changes in the market can be accommodated as the product is built. Agile projects also lead to higher quality. The assumption is that team members are knowledgeable and will build with high-quality at all times. However, mistakes and unanticipated issues will arise. In Agile projects, feedback loops are short and mistakes are expected to be fixed when they are discovered. Agile projects also have benefits for the team. Agile teams often have higher job satisfaction. This is due to the culture of empowering team members to define the best way to work together. Leveraging the decision-making ability and creativity of all team members is not only good business, but creates a stronger sense of purpose for the team members. It also results in the use of a variety of skills and provides continuous learning opportunities. Agile projects lead to better innovation. The Agile approach is experimental in nature which allows innovative ideas to be quickly built and tested with real users. Because the team and the product are adaptable, the new ideas may or may not survive in the long term. Either way, the Agile team embraces the learning process involved. Companies that adopt Agile approaches have an entrepreneurial spirit to them and are much less likely to be disrupted by other companies. Agile products have lower costs than traditional projects. This is because you are consistently receiving feedback and focusing on value. You are not performing wasteful activities. Also any problems are resolved when they are discovered leading to lower rework costs. Agile projects are safer in the sense that risk is lowered because you are continuously getting feedback on what you are doing. You don't want to work on a product for a long time only to realize much later that major parts of what you have built have low value. Agile projects also result in predictable deliveries. Even if an Agile project has a strict deadline, the commitment can be made because you always have a shippable product. What varies is the number of high value features that are in the product. This reduces the overall risk of the project. Next we will discuss one of the foundations of Agile, the scientific method. We've seen that managing and working on Agile projects includes many small iterative learning loops. This resembles the scientific method and in fact the scientific method can be thought of as the foundation of Agile projects. Since most of us are at least somewhat familiar with the scientific method, we can use that knowledge to help understand Agile projects. The basic idea of the scientific method is to start with an idea or hypothesis then build an experiment to test the idea, then observe the results of the experiment and learn from it. You can then repeat the process using what you have learned to improve in the next iteration. It is a series of continuously learning and improving iteration's which also could be called loops or cycles. This is called an empirical approach to problem solving where you are learning from the results of experiments. In Agile projects, we shift our thinking a little bit so that instead of using the scientific method only for discovery, we also use it to accomplish the work of a project. In addition to creating a hypothesis, we plan some work. We then do the work. We then learn from the feedback on both our work and the process that was used to do the work. We repeat the cycle in order to do more work leveraging the knowledge that we have gained in the previous iterations. The scientific method can be thought of as a formalized description of what the brain is doing when it is problem solving. Accomplishing the work of projects involves a lot of problem-solving. So, it's not surprising that agile approaches are based on the scientific method. The iterative learning loop of the scientific method is the foundation of many processes. These loops are used in many contexts, including for general business and for software development. They are used by individuals and by teams. The specific number and names of the parts of the loop may vary, but they all follow the same basic idea. The main idea is to continuously learn and improve using iterations. Let's go through some examples of concepts that are basically derivatives of the scientific method. Plan, do, check, act or plan, do, check, adjust is a concept used to improve business processes. These steps are basically the same steps as the scientific method. Plan, build, release can be thought of the steps involved in software development. Spotify has an approach with think, build, ship, and tweak as the steps. If you realize that the scientific method is the parent of all of these approaches, you can see that there are many seemingly different explanations of basically the same ideas. Next, we will compare agile projects to the traditional waterfall approach. The traditional way to manage projects is also known as the waterfall approach. The simplified explanation of waterfall is that instead of continuously planning and releasing increments of the product, you develop the entire product in distinct phases. A typical first phase is the analyze phase, where customer requirements are defined. The output of this phase is usually a requirements document. The project then moves on to the design phase, where the features related to the requirements are designed. The output of this phase is usually a design document. The features are then built. Once the product is built, it is passed to the quality assurance department for testing. When the product passes quality assurance, it is passed to manufacturing or operations for release. Each of these phases usually is owned by separate teams. So one team is passing their output to the next team in succession. You can see that this simplified view of the waterfall approach contains effectively only one pass through the scientific method. There is no looping. This approach to projects is somewhat similar to traditional mass production in manufacturing. The batch size of each phase is large, effectively the size of the entire product. This is partially because of the perceived cost benefit of economies at scale. It also may be a practical approach if certain steps take a lot of setup and time to execute. The waterfall approach still has value for some projects, but for many projects, it is an outdated approach that leaves you susceptible to being disrupted by your competitors. Even when a waterfall approach is necessary, elements of agile processes can be included in the project. The waterfall approach to managing and building projects has many downsides. The biggest issue is that your big, upfront plan will be wrong. This is because you can't predict the future, especially for complex, one-off projects. High value features aren't correctly identified. Until the customer actually tries a feature, they don't really know if they will use it. As a result, you build many unnecessary features. For example, it may turn out that the customer really only uses two of the 10 features that you built. There's waste, not only in building those features, but also in making the customer wait for those two useful features while you build all 10. Things are harder, more problematic, and take longer to build than you think. In general, people who build things are optimistic about how long it will take to build it. As a result, most complex waterfall projects fall behind schedule, sometimes by quite a lot. The market and/or your team will change while you build the product. In the waterfall approach, you'd create a detailed plan for one big release in the future. During that time, the market may change in ways that you should be adapting to, but you don't, because you continue to follow the original plan. Also, if you lose key team members during the development of the project, it will delay the entire project because the original plan assumed that you would have the original team for the entire time. In a waterfall approach, change is hard and expensive. This is because you've created a big upfront plan and any changes can have major ripple effects for the rest of the project. A change later in the project life is especially difficult and expensive to make because a lot of work has already been put into the original design and the product isn't designed to handle changes like that. This is why in any waterfall projects, there's often a separate change management process for dealing with changes to the original plan. In a waterfall project, you tend to create a lot of obsolete documents. Because the waterfall approach assumes that the project will go through distinct phases while the product is being built, the output of one phase tends to include a document or documents that the next phase picks up and works with. This is problematic in many ways. The document may not be written in a way that the people in the next phase truly understand, resulting in waste. Also, as the next phase discovers things that need to change, they often don't update the documents from the previous phase causing the documents of the project to become out of sync and eventually obsolete. Also towards the end of waterfall projects, changes are often made because of discovery of last minute problems. These changes are often made without being properly documented causing the documents from previous phases to become further obsolete. In waterfall projects, the feedback on what you are doing often is drastically delayed. For example, let's say that you identified a high value feature during the analyze phase. That feature eventually is released when the product is released. This can be months or years after the high value feature was identified. That is a long time to wait to learn that your high valued feature is actually of low value. As another example, let's say there the software feature is built in the build phase. If testing is a separate phase, that test may be months after the feature was built. If the testing fails, the engineer who built it will barely remember building it or may no longer be available to help fix it. Delayed feedback like this is a source of waste in the project. A question that you may have is when is waterfall used? Well, if the setup cost for a phase is high, you tend to plan very carefully and work in big batches. In other words, if it takes a lot of time or expense to setup what is needed to send a single feature through a single phase, you tend to batch the features in order to achieve economies of scale. Let's look at software development as an example. In the not-too-distant past, companies had to buy computers used for every phase in the process. They would have to predict how many they would need, order them, wait for them to arrive and set them up. They would then have to install the software on the computers depending on how they used it. It made sense to batch all of the testing in this way because all that is needed simply to set up and run a single test. There were separate people whose job it was only to setup computers and others who only set up and ran tests. As a result, the developers and testers were different people doing work at different phases of the project. The team members didn't necessarily want the steps to be so difficult and would have preferred a more agile approach, but the state of the art at the time, made that very difficult to achieve in reality. Another reason to use waterfall is when the work is relatively predictable. You can do the work in separate steps with large batch sizes. However, this is often not the case with one-off projects. Another question that you may have is, why is waterfall outdated in many cases? Well, the setup costs of some phases is trending to zero or at least no longer a major factor. This is true in terms of both time and expense and means that the batch size can be reduced to much smaller numbers. You no longer have to think in terms of a project going through each phase successively and instead can think of each feature or part of a feature going through each phase very quickly. As the setup cost of the phases of the project become negligible, the entire justification for managing projects using the waterfall approach begins to fall apart. Agile approaches become superior. Now hopefully, you are beginning to see the clear difference between agile and waterfall approaches. In waterfall, you are effectively performing one giant iteration of the scientific method with the entire product being built phase by phase. In an agile approach, there are many small iterations where small batch sizes go through each of the phases relatively quickly. The end of each iteration results in a product increment that is usable by the customer. The agile approach provides the benefits of rapid feedback and because planning is continuously happening, the product can adapt quickly. All projects are unique and many projects will contain elements from both agile and waterfall approaches. Even if you need to use waterfall techniques for some projects, it is a good idea to have a solid understanding of agile principles and practices and use them where they make sense. If you are using waterfall techniques on a project that doesn't really require them, there's a good chance that your competitors using an agile approach will be at a strong advantage. We have seen that a waterfall process is more sequential than an agile process. Agile processes are more concurrent with each feature going through phases quite quickly. If you have multiple developers on an agile project, each of them may be working on different phases at any one time. Because of the fundamental differences between waterfall and agile processes, project management for the two approaches is different. Waterfall is more of a predictive, command and control process with big upfront planning, and agile is more of an adaptive, distributed process with just enough and continuous planning. In some ways, agile processes mean that everyone on the team is somewhat involved in planning and project management work. We will discuss this more as we move through the course. Here is a review of what we discussed in this video. Agile is an approach to managing and working on projects. Agile uses many small increments and iterations. The waterfall approach uses one giant iteration and provides little continuous feedback. Agile project management is fundamentally different than traditional project management.