software
development
academy

SDA Sp. z o.o.
Al. Zwycięstwa 96/98
81-451 Gdynia, Poland

+48 732 489 313
info@sdacademy.pl
sdacademy.pl

Table of contents:

1. Agile and Scrum

2. What is Waterfall

3. Agile

4. Agile Manifesto

5. Agile vs Waterfall

6. What is Scrum

7. House of Scrum

8. Scrum values

9. Accountabilities in Scrum

10. Artifacts in Scrum

11. Events in Scrum

12. Backlog Refinement and Estimation

1. Agile and Scrum

Hi, on this course we'll talk about the most common approaches when it comes to approaches are waterfall also known as the cascade model and agile. Next we'll talk about what is agile, what defines agile, what are the rules that define software development nowadays. After that we'll talk about Agile Manifesto which is one of the most important documents that defines the way of developing software. And lastly we'll talk about why most companies pick Agile and pick Scrum as the way of work. Okay so few words about me. I've been working in IT since 2013. I'm a professional scrum master third degree, professional scrum product owner and a professional agile leader. Previously I was a developer, scrum master, and semi-product owner. I'm a trainer at Software Development Academy and one of the authors of a blog for No Fluff Jobs and a freelancer.

2. What is Waterfall

Waterfall, also called the cascade model, was written around 1970, first publication can be found in managing large software systems. Waterfall assumes a cascading way of work - step by step, phase by phase and stage by stage. Here you can see the possible examples of steps phases that this model can contain, and those phases are:
analyze - we are analyzing what we want to develop, what we want to do regarding this project, this product
coding - this is the phase where we are performing the actual work starting to code, to actually develop our product
testing - in this phase we are figuring out if the things that we have done in the previous phase in the coding phase works as it should work
implementation - this is the phase where we actually are delivering our product to the market and lastly maintenance our product is already on the market and we are delivering new patches features for the product. Let's take a look at those phases one more time in a little bit more detail. As previously mentioned Waterfall assumes a cascading way of work step by step phase by phase stage by stage and so on. Let's assume that we are working on a really big project on a really big software project for a big bang. How much time do you think we need to analyze this product? It could take months. Next is coding - this can take even years. Testing - this can take several more months and during this phase, during testing we found out that the things that are being developed are not working as they should be or they are not the things that we wanted to develop. What do

we do? We go back to the analysis and we have to go through the old documentation, proper analysis. This can take several months and it's a really big pain in the back. So we corrected the things that were not properly implemented in the analyze phase and we are heading to the next phase to the coding phase and here we also have to make changes based on the previous changes made in the analyze step. Then,we are going to testing and what could happen, well, something else might not work as it should be. What do we do? Probably you have already guessed, big products can be developed for years and go back to the same old phases when we are analyzing the stuff again and again and the document made half. For a couple of years it has been a really really big problem and this is the biggest flaw. When it comes to Waterfall. Waterfall doesn't like changes. It's just not prepared for changes and what I want you to understand. Changes are inevitable. They are just going to happen no matter what methodology you would use. Waterfall comes from management methods, specific to the factory method of development. Let's consider that you are working in a factory and your factory is producing garden gnomes. When you have one garden gnome you can produce one million garden gnomes which will be the same as the first perfectly made garden gnome. There will be no changes in the upcoming garden gnomes that you will produce. When it comes to software development, this is definitely not the case and the changes will come. So what happened, that waterfall model became not sufficient for us? Well actually a lot has happened. You can see a bunch of photos on this slide. Cassette tape which was one of the mainly used data carriers both for software and audio. Next you can see floppy disks which you can probably remember and probably used because they were very popular in the 90s. You can see that on this picture, those discs contain one of the most popular operating systems. Next we have a home console, Famicom, known in the US as NES. On the right you can see a PC, which doesn't resemble our PCs right now. On the bottom left we see a kiddo playing Donkey Kong and lastly we can see a bunch of retro gaming stuff. So why am I talking about this old stuff? This old hardware and old video game? Well because the products that were developed a few years back were not that complicated, not that complex and changes haven't been made so often in them. Now let's take a look at a few images of how much our software has grown over the years and how much the way that we are developing software has changed. In the top left corner you can see MissMargaret Hamilton and she's standing in front of Apollo 11 code which was developed by her and her team from MIT. On the next two images you can see lines of code comparison and for example mentioned before, Apollo 11 software code has 145 000 lines of code where World of Warcraft, a modern PC game has over 5 million lines of code. Does this mean that World of Warcraft is more important to us than Apollo 11? Well actually no, it's just a comparison on how many lines of code we are developing to do a video game and how many lines of code were used over 50 years ago to fly to the moon and get back. What I want you to understand is that the software that we are producing nowadays has become more complicated and more complex. We need more time, people, and resources to produce our software. Also a very important thing that is really worth mentioning, the maintenance phase has become nowadays very important. We cannot just develop a product and go with it in the market and forget about it. Now we are not making the garden gnomes we are making software and this software needs to be maintained both using patches, new features and stuff that i mentioned before. Previously mentioned products like NES video games which were stored on cartridges or even operating systems stored on floppy disks. Those products were not maintained, there were no patches for them because this was before the internet. Once the product was developed, well, it went on the market and we couldn't add a patch for it. We couldn't make any add-ons to them or even if we could do that, that was really expensive and not always possible. We've stopped using Waterfall for two main reasons. First, low flexibility when it comes to changes and the second one long delivery time of the finished product. So what's the alternative? Let's go and talk about Agile.

3. Agile

Earlier, we talked about waterfall, one of the methods of developing a product, basically a software for us. Agile is an alternative to Waterfall when it comes to developing software. The model assumes software development in an agile way, in short iterations and frequent delivery of value. And what might seem surprising for you, is that agile consists of exactly the same production phases. So we still have coding, testing. The difference is the way of working in these phases and the method by which the software is produced. As mentioned before, Agile is more focused on short iterations and frequent delivery of value rather than one big bang release. As you can see right now, I wasn't lying, when I said that Agile development phases are the same as in Waterfall. We still have to analyze, then to code test , implement and maintain our software. However, we will do this in a different way in smaller portions. The difference to the Waterfall model is that all these phases are running simultaneously in a fixed time period called iteration in each iteration. Every phase is present in a smaller scale, so we are actually doing coding, testing, analyzing, implementation and maintenance in a smaller time scale. But why are we doing this in a smaller case? What are the benefits of this approach? Well, for example, we have bigger resistance to changes. Let's take an example, that we have made a mistake and we are able to correct this mistake and do it in a rather short time period. When we have made a mistake in a waterfall, we have to get back to the first phase to analyze phase. When we make a mistake in Scrum in Agile, we are referring to the last iteration. And just to point out the iterations in Agile in Scrum will last approximately starting from one week till four weeks or 30 days or one calendar month. Also as mentioned before, we are able to deliver fast because we are delivering products in small iterations in small parts and after each iteration we are able to release it to the market and this can benefit us in various ways. It can give us feedback from the clients so we will know if our approach to develop software is proper, if we are developing the things that our client needs. And it can benefit us with money because we are releasing value to the market and people might be interested to buy it. Those are only examples of some benefits that we might have if we are releasing our software in smaller parts. Trust me, there are a lot more. Also we are able to work on a few things simultaneously so we are able to analyze and do coding so no one gets bored. Testing the coding all the phases are engaging various peoples with various roles.

4. Agile Manifesto

Agile methodologies were created in the 90s. But in february 2001 there was a very important meeting in the history of software development. 17 people met in Snowbird, USA. And those people created a document called the Agile Manifesto. This is one of the most important documents when it comes to software development. What Does an agile manifesto looks like? Let's take a look at it. We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value individuals and iterations over processes and tools,working software over comprehensive documentation, customer collaboration over contract, negotiation responding to change rather than over following a plan. That is while there is value in the items on the right we value the items on the left more. And this is it this is the whole Agile Manifesto. Let's break this manifesto into those four sentences and talk about them in a little bit more detail: individuals and interactions over processes and tools - well, every organization consists of people. The people are the organization so the people in every organization are different based on culture, geographics and many many many factors. Each organization is different because the people who worked there are different and the process and tools are to serve us. Never the other way around we shouldn't be slaves for processing tools because if we are then we are not productive as we can be and as we should be. Don't get me wrong processes and tools are important because like I said they can help us to obtain our goals to obtain goals of the organization. But people and interactions between them are much more important to us.

The second sentence - working software over comprehensive documentation - the software is something that our users will use. They are paying us for the software. This is the way through which our clients gain funds or other benefits. They want to see the working product, not just beautiful documentation about things that are not implemented. In Agile and Scrum we are creating documentation at the right moment and to a degree that corresponds to reality. Remember when we were talking about Waterfall in the analysis phase, we created a lot of documentation. In Agile we are still producing some documentation but we are doing it at the right time when we are producing the value. We document that value. The thing that I want you to remember is that documentation is important. We cannot forget about it or stop doing it but the running software is far more important to everyone.
Third sentence - customer collaboration over contract negotiation - formal agreements such as contracts are important but we should always be looking for cooperation for partnership with our client. This is far more sufficient for everyone.
And lastly responding to change over following a plan. Every plan, even the best no matter how much time you spend on it, doesn't guarantee that you will be successful. Changes in the produced software are something natural and they will happen from various reasons not only because your client will change his mind but also because the market is changing. One of the biggest advantages of Agile is the ability to react quickly and effectively to the change. We are not spending unnecessary resources. We are not planning our work for the upcoming two or five years. Regarding our product now we are planning our work on a much smaller scale which helps us to be more prepared for the changes that might come. Also what I would like you to remember is that blindly following a plan can cause huge damage. If we continue to produce software-produced products that none of our clients would like to buy, then we will lose money continuously. It's better to take a look at our plan and change.

5. Agile vs Waterfall

It's time for a quick summary of what we were talking about regarding Agile and Waterfall. So, Agile can be used in small and large products. Waterfall? We use Waterfall rather for large products. For example when we are developing a product for a large company, for example a corporation. Agile is relatively new. Yeah, I know that it's almost 30 years on the market right now, but Waterfall has existed for several dozen years. Agile is ready for changes. The changes are welcome, when in Waterfall well it's reluctant to change, it doesn't like change at all. Agile is also focused on delivering product in smaller parts, small fragments and doing it, when in waterfall the product is released after a long time and we have one big bang release. There are several more differences and probably it will be a good exercise for you to find at least three more differences between those two methodologies and also, try to pick up three things that are common for both of these product development methodologies.

6. What is Scrum

While preparing materials that you will find in this course I was referring to using scrum guides. On the screen you can see the license and some very important information, related to scrum guide I encourage you to take a look at it. Scrum is not a methodology or a management method. Scrum is probably the most popular framework of agile. So Scrum takes the general concept of Agile and adds a few tweaks and tricks to it. Scrum is 100% compatible with what you can find in Agile. Worth mentioning is that the creators of Scrum Guide - Ken Schwaber and Jeff Sutterland are co-creators of Agile Manifesto. Scrum Guide is a document which describes the rules of the game, the basics of what Scrum is, when to use it and how to use it. This is a very short document; it has approximately 13 to 18 pages. Number of pages depends on the language version. I encourage you to read Scrum Guide, because those materials are based on the information that you will find in Scrum Guide. However, I won't dive deep into the details. This is just a super quick introduction to Scrum. On the other hand some parts of this training will also have a few additional informations that you won't find in this Scrum Guide. Remember this is only a very simple introduction, more details and a chance. To experience Scrum in practice will happen during your workshops. Okay, let's have a very simple and very short overview of Scrum. It will be really super fast and without diving deep into details because we'll cover the details later on. Product backlog will be an artifact in which you will find all the elements called product backlog items that we would like to develop and deliver for this product. Those elements will have their priority which will be set by the product owner. Product owner is the person which is responsible for the product backlog. He is the owner of this artifact during an event, known as sprint planning. We'll be planning our upcoming iteration in Scrum called sprint. Product backlog items, that we would like to deliver in the upcoming sprint, plan how to deliver those product backlog items and the sprint goal formulates the second artifact called sprint backlog. Sprint backlog is owned by developers and they are responsible for maintaining it having it transparent and understandable which will help us to track progress of work during sprint. Scrum distinguishes 5 events and those events are sprint, sprint planning, daily scrum, sprint review and sprint retrospective. Sprint is an event and it holds the four other events that we can find inside scrum, so sprint planning, daily scrum, sprint review and sprint retrospective. All the work inside scrum happens during the sprint sprint should last no longer than one calendar month so it's time box is equal to one month. Usually we work inside sprints that last entire weeks for example one week sprint, two-weeks sprint etc. Daily scrum is everyday event on which we are analyzing how we are progressing in this sprint. We might say that this is a plan for upcoming 24 hours. Usually only developers participate in daily scrum daily scrum will happen every day in the same place at the same time. Next event is called sprint review, during which we are presenting the work that was completed in this sprint to our stakeholders, gather the feedback from them and present the work that we would like to do in the upcoming sprint. And the last event sprint retrospective. This is an event during which we analyze our work, our performance in regards to the process, effectiveness of our work and how we can improve how we can become more efficient eliminate some problems that hinder us etc. Each sprint should have an increment of working product which is ready to be released. We can have more than one increment in sprint and we don't have to wait to the end of this sprint to release it. Increment is the third scrum artifact, work that has been completed so it meets the definition of done requirements, gets released is called an increment. That's it when it comes to some very quick and basic introduction to Scrum workflow and Scrum in general. I know, some of those things might sound a bit weird but we'll dive deep into details right now.

7. House of Scrum

Scrum is based on three pillars and those pillars are transparency, inspection and adaptation. Okay, so transparency. It means visibility and understanding, so what should be visible and what should be understandable. Let's take for example the product backlog which is the list of elements that needs to be developed. Those elements should be understandable by everyone and should be visible at all times. Inspection - so, we need to inspect our way of work and our progress to be sure that we are on the right course. And lastly, adaptation. So let's think that based on our inspection, we've found out that our product backlog is not as transparent as we would like it to be. So we need to adapt. We need to be able to change something so that we will adapt to the proper state of the situation that we would like to have.

8. Scrum values

Scrum has five values, and those values are commitment, courage, focus, open it and respect.
Commitment - we should be committed to work that we are working on. For example, we'll be committing to our sprint goal.
Courage - for example, we need to be brave enough to challenge the problems that we might have and to resolve them.
Focus - for example, we should be focused on work that needs to be done on our commitments on our goals.
Openness - for example, we need to be open for changes, acknowledge the changes, and acknowledge the fact that change is inevitable and be prepared for it.
Respect - for example, pretty obvious we should respect each other inside the Scrum team, but also we should respect people outside the Scrum team or even outside our organization. We need to understand and respect other people's point of view, their thoughts, feelings etc.

9. Accountabilities in Scrum

Scrum distinguishes three accountabilities and those are: scrum master, product owner and developers. Together those three account abilities are called the scrum team. So whenever I'm referring to the scrum team, I'm referring to those three roles combined, so scrum master product owner and developers. Scrum team will be a small group of specialists usually consisting of 10 or less people with one product owner, one scrum master and developers. Adequate structure enabling effective work on product development and delivering increment, cross-functional and self-managing and there is no division into sub-groups or sub-teams inside the scrum team. Let's talk about this adequate structure, cross-functional and self-managing. Adequate structure and cross-functional will mean that this scrum team has all the necessary knowledge, experience and everything that they need to work on, the product that they are developing so they are not dependent on some other team from people outside the scrum team. Self-managing will refer to the high level of independence of the scrum team. They know what needs to be done and they will make the decision how the work will be done. They do not need to be guided. They do not need to have someone who will show them how to do their work. They are professionalists and they need to figure out how the work can be done and what will be the most effective. Worth mentioning in the previous version of scrum guide the scrum team consists of the scrum master, product owner and the development team and those were the roles in scrum in the current version of scrum guide we are referring to accountabilities. The development team inside the scrum team made the false impression that we have a team inside the team. Development team inside the scrum team was just confusing so this is why now we have a scrum team which consists of scrum master, product owner and developers. And the second change,accountabilities instead of roles. Well, in our life we have some rules but being accountable, responsible for something or for someone is more reliable is more understandable and well, it's just more important. Okay, let's talk about developers. Developers are responsible for developing usable increments of working product each sprint. So they are the folks that will actually do the necessary work to implement some solutions, to develop a product. They are highly independent when it comes to choosing how the increment will be developed. They are also cross-functional, so developers inside the scrum team have all the skills, all the knowledge necessary to develop a working product increment. And self-managed, so as also mentioned in the scrum part of this course, they do not need to be guided by someone. They made the decision about how the work will be done so they have high autonomy and all the necessary skills. Developers are responsible for estimating the work that needs to be done. We'll talk about estimation a little bit later on in this course but just for now you need to know that developers are responsible for estimating the work and we do not estimate only in time and only time to develop some solution. We are estimating the effort that we need to put in, we need to invest to develop something, to accomplish some work. There are no divisions into team roles or there is no team inside the developers. Inside scrum developers are cross-functional and not functional so we do not have a team of frontend developers and team of back-end developers. The developers inside the scrum team have to have all the necessary skills, knowledge, etc to develop a product. So if we need to have frontend developers and backend developers, they should be a part of the developers inside the scrum team. Every developer has the same title - developer, no matter how experienced this person is or in which technologies he or she is working. Developers are responsible for creating the sprint backlog continuously, adjusting it to achieve the sprint goal and ensuring quality to the implementation of the definition of done. Probably at the moment you know nothing about sprint backlog definition of done or sprint goal. We'll talk about it later on in this course in more detail. Just for now, the sprint backlog will be a plan of what we would like to achieve in this sprint and a plan of how we would like to achieve this. As mentioned in the scrum team part of this course in the previous version of scrum guide we're referring to a role known as a development team. In the current version of scrum guide we are referring to accountability called developers. Still, they are the folks that will develop the product and they are professionalists that know how to do it best. Product owner is responsible for maximizing the product value. So he or she knows what will be the best for the product, what will bring the most value out of the product that we are developing. Another responsibility of a product owner is to create and maintain for example through prioritization clearly defining product backlog. We'll talk in detail about the product backlog later on in this course. As for now, you need to know that the product owner is responsible for the product backlog, for creating it and maintaining it. And the product backlog will be a list of elements that we would like to do, that we would like to develop in the product that we are currently working on. Product owners are responsible for contacting stakeholders and understanding their needs. So if he must maximize the product value then it's pretty obvious that he needs to know what will be the most important, what will bring the most value to the people that will be interested in this product. So to the stakeholders, the product owner will provide some answers regarding the product that we are developing, what will have the most value, what we should be focused on, what functionalities the product will have, what is the business value, what is our goal, etc. And this is why also the product owner is responsible for crafting a product goal and we'll cover the product goal later on in this course. Product owners should have a strong respected position in the organization. Product owner decisions should not be questioned because he should have the most knowledge about the product that is being developed. He knows how to maximize the value of the product. This is why also this accountability should be held only by one person, never ever by a committee. The rule is that one product should have only one product owner. Having more than one product owner will let the confusion, who makes the decisions who knows best what will be the most valuable for the product. This is very important. Remember, that one product should have only one product owner. As mentioned before the sole owner of the product backlog is the product owner but the product owner can delegate part of his responsibilities, his tasks to for example developers. A very common example, product owner can ask developer - hey, please add some technical details to one of the elements of product backlog. Nevertheless the product owner remains accountable. So in the given example the product owner will still be accountable for product backlog, for its transparency, prioritization, etc. Scrum master will be responsible for creating an appropriate work environment, which will enable effective work and implementation of scrum. He is a true leader who serves the organization and the scrum team. In the previous version of scrum guide, scrum master was referred to as servant leader. Now he is a true leader but the general concept remains the same. Scrum master should not focus only on his or her scrum team. Scrum master mentioned before is responsible for implementing scrum but not only inside the scrum team but inside the whole organization. He should serve the entire organization. One of the responsibilities of the scrum master is to help to find effective ways how the scrum team can work things that can be improved in this scrum team. So for example, scrum master will be responsible for implementing some ways of work which will boost up the effective work of the scrum team. Scrum master will try to find out what are the weak points of the scrum team and how we can manage them, what are the strong points of our scrum team and how we can use them and so on. Scrum master facilitates which will mean simplifying, carrying out scrum events. In Scrum we will have some events like sprint planning or daily scrum. And scrum master is responsible for conducting those events and making them as efficient and as simple as fast as possible. Scrum master is also responsible for working closely with the product owner. For example scrum master can help product owners to find ways how the product backlog can be successfully managed, how to for example prioritize the items in the product backlog. Scrum master can also support cooperation with the stakeholders so with the people that are interested in the product that currently is being developed. Scrum master will help with the identification and removal of impediments which are limiting the progress of the scrum team. However scrum master is not responsible for removing all those impediments or he is not the only person that is responsible for doing it. Other people also can and should be involved in the identification and removal of impediments. Scrum master should spread good scrum practices in the entire organization. He can do that by, for example, conducting training. Scrum master increases the work efficiency of the scrum team. For example he will help them to become self-managed and cross-functional. He will be a guide for the scrum team on how to become more efficient, how to do things better, how to remove impediments and continuously improve. But I would like to point out one more time - scrum master is responsible also for working with the entire organization, so to enable efficient scrum implementation not only inside the scrum team but in the organization in general. Let's head up to a quick summary about the scrum accountabilities. Okay, let's sum up everything that we know about scrum accountabilities in a few words. Inside scrum will have a product owner, developers and a scrum master. Those three accounts all together are called scrum teams. The Scrum team will consist of 10 people or less. They are cross-functional and self-managing. Self-managing and cross-functional will mean that they have all the necessary skills to develop a product without being dependent on someone outside the scrum team and they do not need to be controlled by an external manager. They do not need to be told how the work should be done. They are professionalists and they know how the work should be done. Product owner will be the accountability, which is closest to the business and he's also called very often a value maximizer. The accountability which is responsible for maximizing the product value. Product owner provides the answer to the question, what will bring the most value, what will be the most beneficial for the product that we are developing, what will be the product goal, what will be the best for the product that we are developing. Developers will be responsible for developing the product. This is the group of specialists that have all the necessary knowledge needed to develop the working product they are responsible for producing, for developing a working product increment each sprint. Developers answer the questions: how the work will be done, how the work will be performed. Everyone has the same title - developers, no matter how much experience they have, on which technologies they are working. If those are developers or QAs, testers, etc, everyone has the same title. And the last accountability scrum master. Scrum master will be responsible for preparing a working environment which will help to boost up the efficiency of the scrum team and produce a working environment which will help everyone to grow. Scrum master is responsible for the process for the implementation of scrum inside the entire organization not only within the scrum team. Those are all the mandatory accountabilities inside scrum. It doesn't mean that you cannot have some additional accountabilities or roles. Scrum is only a simple framework and those accountabilities are mandatory. You can have additional accountabilities. If you think that you will need them, just remember that new accountabilities should be compatible with those that already exist. They should not interfere with the existing responsibilities.

10. Artifacts in Scrum

Scrum distinguishes three artifacts and those artifacts are: product backlog and its commitment - product goal, sprint backlog and its commitment - sprint goal and increment and its commitment, definition of them. Those artifacts will help us with improving our transparency with the inspection and adaptation. Also scrum artifacts represent the levels of product. So for example product backlog will show us the general level of product and sprint backlog will visualize the sprint, our iteration level. We already scratched some surface when it comes to the product backlog and sprint backlog. Now we will dive deep into the details and let's start with the product backlog. This artifact is evolving. List of items that we want to develop for the product that we are working on and this artifact exists as long as the product exists itself. Product backlog is managed by the product owner and he is responsible for the product backlog. Although, the product owner may pass some of his responsibilities. He may transfer those responsibilities to developers. For example, the product owner might ask the developers to add some technical details into the product backlog item, so to the work that needs to be done inside the product. Product backlog is the sole source of knowledge about the developed product. This is very, very important, because one product equals only one product backlog. We cannot have multiple product backlogs because this will harm our transparency. This will lead to confusion because if we have more than one product backlog, which one is the current product backlog. Which should be used for the sprint planning or which is the most important one. This will lead to many, many, many questions and well just can harm our transparency, inspection and adaptation. So this is why the same as we have one product owner for one product we also should have only one product backlog for one product. This is an evolving artifact and it is used to manage the product that we are developing. Inside the product backlog we'll have a hierarchy and the product owner will pick up the most important elements that need to be done inside the product and put it at the top of the product backlog. Remember, the product owner is responsible for maximizing the value of the product, so the product owner will pick up the most efficient way of managing the product backlog. Some items in the product backlog will be redefined. Some of them might be deleted, some will be added, edited and so on this is an evolving artifact. Usually and this is not part of scrum guide this is not an original part of scrum but usually product backlog will consist of two elements, two main elements - epics and user stories. Those are only two examples. The probably most known types of product backlog items. You can find many others as for example tasks, subtasks, test tasks, features, bugs, defects, etc, etc, but epic and user story are the most common. Epic is a product backlog item which is really big. It's really easy to spot it because, well, it probably will take more than one iteration for the developers inside the scrum team to develop to complete this product backlog item. So this is a really, really big product backlog item. User stories are very often created by refining by splitting epics into smaller parts. User stories are written in a very specific way - as a role I would like to some benefit some action that will help us to understand the user perspective what the user should have, what is the benefit, that user will have. Once this user story is completed, it's done. For example, as a website visitor, I want to be able to create an account on this website. Product backlog is necessary for planning our upcoming sprint so it's necessary for the sprint planning event. It's hard to imagine even starting planning without having any product backlog at all. Product backlog is used to create another scrum artifact, called sprint backlog. Sprint backlog will be a part of the product backlog that we would like to develop in the upcoming sprint. We will talk about the sprint backlog in the upcoming lesson. Product goal is a commitment of product backlog. Product goals will enable us to focus on what our product aims to achieve in the long term. Typically to achieve a product goal we need to spend several sprints on it. Let's imagine that we have a working online shop and we would like to introduce some new big feature to it, like validating tin numbers from all around the world. This is a big feature and probably it will take us several sprints to complete it. This is an example of a product goal. Product goal is defined by the product owner. He is responsible for creating a product goal. There can be only one product goal at the same time and a new product goal may be set when the previous one has been fulfilled or there was a decision that we would like to cancel this product goal. This is everything when it comes to the product goal. Let's talk about the sprint backlog. Sprint backlog will be mentioned before selected product backlog items. Sprint goal and plan how to deliver those sprint backlog will be the result of sprint planning. Developers plan the work that needs to be done and the way how this work will be done. It doesn't mean that the developers are choosing the work that needs or will be done in a sprint. No, they are planning, figuring out how the work can be done. The Sprint backlog will contain elements that we want to develop but only for the upcoming sprint. Elements that we would like to deliver, the items that are present inside the sprint backlog can change in some way, like we can change the way that we would like to complete some part of our work. For example we thought that some specific product backlog item to make it easier. Some part of our work will be done using Angular which is a Javascript framework. However, during the sprint we discovered that it will be much easier to use React. The main goal stays the same but the way that we would like to handle this specific work changes. Maybe it's easier for us to use React or we have experience using it. Either way, like said before we are changing only the way that we would like to handle this specific problem, that we would like to deliver the item, product backlog item updated on a regular basis with the progress of work and time usually during daily scrum. Developers are responsible for keeping the sprint backlog updated. The Sprint backlog makes it easier to observe. The progress of the work performed during the sprint. This is why it's so important to keep it transparent, inspected just to be sure that it resembles the actual progress of work during the sprint and adapt it if we are not on tracks, something is going wrong. This is really crucial. To better understand what sprint backlog and how it can look like, let's take a look at some very simple example sprint backlog that I prepared for you. Very often the sprint backlog looks like a Kanban board so the board that visualizes items. To make it simple for you let's say tasks that are placed in some specific states, statuses. On the top of the slide you can see the sprint goal. Under it there are three columns to do the elements that we would like to complete in this iteration. However, we didn't start working on them. In progress, the elements that developers are currently working on and the elements that are done so they are completed, tested and ready to be released. Those columns visualize the workflow, the progress of our sprint and statuses of each item, each task. Reminder - this is just a very simple sprint backlog. Sprint backlog can look different. You can see some additional columns like column in review which will gather elements that are awaiting for some review for some testing. You can have column like code review which will have elements that are waiting for code review, etc, etc. So once more, those three columns that you see in this example to do, in progress and done are just some very basic columns representing the current state of work, that are part of every Kanban board and every sprint backlog. On the bottom you can see an arrow that indicates the flow of the work. Items, tasks in the sprint backlog should move from left to right from to do. Next, some developer will start to work on it, so it will go on in progress and when the work is completed everything is working and it has been tested then this item, this task will go to the done status column. As it was mentioned, the sprint backlog is being updated on a regular basis with the progress of work and time usually during the daily scrum event. Sprint backlog makes it easier to observe the progress of the work performed during sprint. Sprint goal is the commitment of sprint backlog. It helps us to define what we want to achieve in this sprint, what is the main goal that we have, what is the value that we would like to deliver in this sprint. Thanks to the sprint goal we gain focus on the ongoing sprint. Sprint goal is the commitment of developers. Developers decide how to achieve sprint goals. But not the sprint goal itself. This is formulated by the entire scrum team. Goal cannot be changed but the scope of work that we need to do to deliver this sprint goal or how we would like to achieve it may be refined, and may change during the sprint. For example, if our sprint goal is to enable visitors to register an account, we cannot change the goal, it has to remain the same. We should enable visitors to register an account but the way that we would achieve this goal can be changed, we can change the site to use some framework, libraries, change programming language, etc. Sounds like you had it earlier, right? Well, it's the same as changing the approach to deliver any item, tasks in the sprint backlog. As long as the goal is achieved you can take some other routes to achieve it. So one more time, the way to achieve the goal can change but the goal itself cannot. Those were the very basics of the sprint backlog. Let's talk about increments. Increment is the last artifact that can be found inside scrum guide. An increment is work that has been completed during the sprint and that complies with the definition of done. We'll talk about the definition of done, DoD in a moment. Each and every increment helps us to achieve product goals. The increments made in the current iteration are added to the increments from previous iterations. During the sprint, we can deliver more than one increment and we do not have to wait until the end of the sprint to deliver it. Scrum states that in the sprint we should have at least one increment which is ready to be released. All increments in the iteration are presented on the sprint review. However, which is very important, more than one increment may be produced during the iteration. The delivery of an increment may take place at any time during the sprint. Increments can be released in any moment of the sprint, you do not have to wait until the end of the sprint. This is important, because there was a myth or maybe just a misunderstanding that you have to wait until the end of the sprint to release an increment. No, increment can be released before the end of sprint as long as it's useful and there is a point of releasing it. For example, the product owner sees some value in releasing it right away or stakeholders, the market demands it. The best way to describe it, will be using an example. Let's imagine that you're working in a two-week sprint and after three days we were able to complete some functionality which will bring us for example two thousand dollars a day profit. Releasing such functionality that can start giving some profits right away seems to be a pretty good idea. But when functionality, item, work, task can be considered done, so, when can we say that it's ready to be released? Well, when it's done, in scrum we have the definition of done - commitment of the increment. Definition of done is kind of a guarantee that some work is really, truly done. Let's think about it as a health check, that helps us to check that the work is really done and it can be released. The work needs to be done, not almost done or partially done or done but something still needs to be done. No, there is no such thing as almost done. The work is done or it's not done. It's simple as that, the definition of done are the requirements that the work performed by us meet in order to be considered completed, done. After this work is being released, it becomes an increment. Definition of done often takes the form of a checklist of requirements. For example, quality requirements, but not only quality requirements that must be met as part of the work performed. Let's imagine that we are developing some software. To be sure that some functionality has been completed, so it's done, it should for example be tested. We need to be sure that it works. Maybe a good idea will be to write some tests, check if and how it will work on different devices like tablet, tv, how and if it will work on different hardware operating system browsers, etc. Definition of done can hold some requirements to check not only if it works but also the quality of our work. DoD will help us to boost transparency but also help us to inspect and adapt. It allows everyone in the scrum team to understand when the work can be considered as done. Thanks to the definition of done, we eliminate the phenomenon of almost done work. Everyone knows the requirements that need to be met to consider work to be marked as completed, as done, as being ready to be released, to become an increment. There may be an organization-wide common definition of done but if it does not exist, the scrum team should formulate its own definition of done. Let's try another example. Maybe a little bit more abstract but also easier to understand. Let's imagine that we are going on vacation. To be sure that we can leave the house and nothing bad will happen to it while we are gone we prepared some checklist just to be sure:

- we need to turn off water gas
- close all the doors, windows
- turn off any electronic devices that don't need to be plugged in and so on.

Thanks to that list, we can easily track what we need to do in order to minimize the risk of having some damages. That list is also pretty easy to track. Everything we need to check is on the one list and it's easy to just check every point from that list. Definition of done for software will help us to feel safer. We are checking that work has been really completed. The definition of done is evolving. Let's imagine that part of our definition of done was testing new functionalities on smartphones, tablets and laptops. After some time we discovered that users are using our software on TVs so we decided that we also need to add testing on TV to our definition of done. We should review DoD at least once in a sprint. Usually the perfect occasion to do that is during the sprint retrospective, an event during which we are trying to find a way how we can get more efficient, improve our work, etc. To summarize it in one sentence, work that meets the definition of done and gets released becomes an increment. Okay, let's do a very quick summary of scrum artifacts. Scrum distinguishes three artifacts and those are product backlog, sprint backlog and increment. Product backlog is a list of all the elements that we would like to develop for the product that we are currently working on. This is an evolving,prioritized list and that list is owned by the product owner. He decides how the product backlog will be managed, what will be the priorities and which elements will be in it and those elements are called product backlog items. Product owners can delegate some of their tasks to other people. Yet, still the product owner is accountable responsible for the product backlog so that the product backlog will remain transparent, understandable by everyone, visible, etc. Product background has a product goal, which is product backlog commitment. Product goal is, well, our goal, that we would like to achieve while developing this product to achieve the product goal. Usually, we need to invest in a couple of sprints. Product goal helps us to remain focused, understand what's our goal and it will influence the product backlog itself, the elements that will be inside the product backlog, its evolution, changes in it, etc. There can be only one product goal at the same time. If you want to have another product go, we need to achieve the previous one or decide to cancel it. Second artifact sprint backlog. Sprint backlog will be all the elements from product backlog that we would like to deliver in this sprint, .plan how to deliver it and a sprint goal. Very often the sprint backlog has a form of Kanban board which represents the work that's in this sprint, status of the work, flow of the work and a visible sprint goal. Sprint backlog is owned by developers and they are responsible for this artifact. Usually, the sprint backlog is updated on a daily basis during the daily scrum. Sprint goal is a commitment of sprint backlog, value that we would like to deliver at the end of the sprint. It will help us to focus on the goal of the sprint, to gather everyone in the scrum team at the same goal. And the last artifact - increment. This is the work that has been accomplished and released during the sprint. This work needs to be aligned with. Definition of done -

definition of done will be a list of requirements. For example quality requirements that determine if the work is completed. Work completed means that it meets the requirements that are present in the definition of done. Product backlog items become an increment when it meet the requirements stated in the definition of done and it's released. During the sprint we can have multiple increments and those increments can be published at any time. We do not have to wait for the end of the sprint to do that.

11. Events in Scrum

Scrum distinguishes the following events, also very often called meetings: sprint, sprint planning, daily scrum, sprint review and sprint retrospective. A very common mistake, backlog refinement also known as backlog grooming is not an event. Backlog which should take place during the sprint. We'll talk about backlog refinement later on this course. But why do we have those events? What are the benefits? Well, for example we can inspect and adapt scrum artifacts inside scrum. We'll have three artifacts: product backlog, sprint backlog and increment. Let's take the product backlog as an example. Product backlog will be the list of everything we would like to develop inside the product that we are working on currently. So it will be a list, a prioritized list which will hold all the necessary information and all the things that will bring value to the product, that will make our stakeholders happier. Inside scrum we'll also have an event called sprint review during which we'll present the work that was done inside the sprint, gather feedback and present the potential work that we would like to work on in the upcoming sprint based on the discussion during the sprint review, based on the things that we were actually able to deliver, based on the product backlog. There might be a decision to change the priorities, maybe some features have more value or maybe there were some changes in the legal system and we are obligated by the law to implement some solutions. Based on the discussion we have the chance to inspect and adapt our product backlog, which will help us to maximize the product value. Events in scrum might also help us to maintain appropriate pace of work and reduce the complexity. For example we'll have a daily scrum meeting which will be held in the same place every day so we won't have to bother "oh where is the daily scrum meeting today and on which hour it is?". No! We'll have the same meeting every time every day in the same place, for the same amount of time, for the same time box. Time box is the maximum amount of time which an event should consume. So for example sprint planning for one calendar month should take not more than eight hours. If spring planning for one calendar month exceeds this time box so exceeds eight hours then this should indicate that we have some problem. Maybe our product backlog so the list of elements that we would like to develop for our product is not ready, it's not transparent enough, it's not understandable for everyone, it doesn't have necessary information or we even do not know what has the most priorities. Maybe our product owner is not available etc. This is why time boxes are crucial. They help us to understand if we are efficient with our time. Events should not exceed the designated time box. Of course our events can be shorter. Timebox for the meeting indicates only the maximum amount of time. It doesn't mean that you have to sit for the entire time because the time box didn't exceed. No, if you are able to achieve the goal for this event in a shorter time, that's great, that's awesome. The only meeting with fixed time books inside scrum is called daily scrum. No matter how long the sprint is, the timebox for daily scrum is always 15 minutes. Other events should be appropriately shorter for shorter sprints. So for example if we are working in a two weeks sprint, then the sprint planning time box should be equal to around four hours. All work in scrum takes place in an iteration which is repetitive activity, repetitive action called sprint. Sprint is a unit of time where all the mentioned earlier phases of software development. So for example analysis, coding, testing, etc, etc takes place. All of the events which I mentioned in the previous lesson, so sprint planning, daily scrum, sprint review and sprint retrospective takes place inside the sprint. Sprint lasts no more than one calendar month so the time box for this event is no more than one month. Why shouldn't we have sprints longer than one month? Well, first of all it would be really hard to plan and estimate our work. As mentioned in the previous lessons. Our work and our world in general is very complex. There are many unknowns. Many things can change. So planning something upfront for more than a month is, well, not the greatest idea things can change. Things can happen. So this is why the Scrum Guide refers, that sprint should take no more than one calendar month. Sprints also cannot be shortened or extended sprints should have a fixed length. We shouldn't shuffle their length, we shouldn't change the length of a sprint. There is no transition period between the sprints so once the sprint is over, the next one starts immediately. There is no time in between them. Sprint can be cancelled but this is very very rare, when the spring goal is no longer valid and the decision about canceling the sprint is made by a product owner. During the sprint we are developing a working increment of potentially releasable products. So each sprint should bring us some value and with something that is ready to be released. There is no sprint zero or release sprint. All the work needs to happen inside the sprint. Every sprint should have a clearly defined sprint goal. The goal that all the scrum team should be focused on. Inside the sprint, during the sprint we do not make changes that could endanger the achievement of the sprint goal. We are not changing our priorities. We are not pulling out people from the scrum team. We do not do anything that could endanger achievement of the sprint goal. Okay, we talked about the sprint, about characteristics of the sprint and I mentioned about the sprint goal, scope of work, etc. So let's talk about sprint planning, the event on which we will plan, our sprint, our iteration, the things that we would like to deliver, to complete during the sprint. As already mentioned and pretty obvious, judging by the name of this event. This is a planning meeting and we plan our upcoming iteration, upcoming sprint. This event takes place at the beginning of each spring and the entire scrum team participates in it. Product owner is responsible for clarifying which elements of the product backlog have the highest value. So we might say that, he answered what will be the most important, what will bring the most value on what we should be focused on, etc. Developers are responsible for answering how the work will be done. The role of a scrum master is to facilitate and help with efficiently conducting this meeting. If you are working in a monthly sprint, the time box is equal to 8 hours. Usually it's shorter for shorter sprints. So for example if we are working in a two weeks sprint then time books for such sprint planning should be around four hours. We can invite people outside of our scrum team for this meeting and those can be for example advisors that would help us to plan our upcoming sprint. We are planning only upcoming iteration, only upcoming sprint. We do not try to plan a few sprints up a hat or a few months of our work. As mentioned before, we are living in a complex world and things like to change. So planning for more than one sprint is not efficient. Scrum Guide distinguishes three parts of this meeting. Quoting: "why is the sprint valuable", "what can be done this sprint", "how will the choosing work get done". First part, why is the sprint valuable, product owner describes how value and utility can be added to this product. The Scrum team collaborates to define sprint goals. Second part, what can be done this sprint. Product owner together with developers select the elements of the product backlog, called product backlog items. The Scrum team adds the details about the work that needs to be done. The amount of work that can be done depends on many factors, for example team performance, availability, size, etc. And the last part, how will the chosen work get done. Developers and only them decide how the work will be done, so that it meets the definition of done. Usually they decide to divide the product backlog elements into smaller parts into smaller items. Sprint goal plus selected product backlog items and plan for delivering them is equal to sprint backlog. The second artifact that we can find inside scrum, we'll talk in details about sprint goal and sprint backlog a little bit later in this course. But just for the moment spring goal will be our commitment for the sprint will be the most important thing that we would like to achieve during this sprint. Sprint goal is usually a short sentence, simple and understandable for everyone and this is our commitment. This is the most important thing that we would like to achieve during the sprint. Sprint backlog will be as mentioned a few seconds ago - sprint goal, selected product backlog items so elements, part of product backlog and plan how to deliver them, the technical solution for delivering this. We'll talk about sprint goals and product backlog and sprint backlog a little bit later on in this course. I just wanted to give you a very simple description. I know, this might sound a little bit complicated or even overwhelming. So just to quickly sum up what I told about sprint planning. Sprint planning is an event on which we are planning our work for upcoming iteration. Timebox for one calendar month is equal to 8 hours, usually shorter for shorter sprints. The entire scrum team participates in this event and we can also invite other people from outside of the scrum team. For example, some experts in order to help us with our planning. We craft our spring goal during the spring planning meeting. Spring goal will be our commitment and selected product backlog items, sprint goal and plan how we want to deliver the sprint goal and those items are called sprint backlog. Daily scrum is an everyday 15 minutes long event. Time box for this meeting is fixed. It doesn't matter how long your sprint is. Time box for daily scrum is fixed so no matter if you have a two week sprint, three weeks or one calendar month sprint, the timebox for daily scrum will always be 15 minutes. Usually, developers and only them participate in this event. But there is one exception. If the scrum master and/or product owner have work to do and this work is related to the items in the sprint backlog, so the work that needs to be done inside the sprint, they participate in this meeting same as developers. A very common situation, other people might be present at daily scrum but only to listen about what the people at the daily scrum are talking about so they do not participate, they do not actively take part in this meeting but they only listen to what others have to say. The main purpose of this meeting is to establish a plan for the next 24 hours, to check the progress towards the sprint goal and possibly react to obstacles that hinder effective work. So if we have any impediment, if we have any problem with achieving our sprint goal, we see any complication - this is the perfect meeting to, well, just raise your hand or say, that okay I see some problem, our sprint goal is in danger or I have some problems and I'm stuck, I do not know what to do. So this is the perfect situation, this is the perfect event to raise any problems that you might have. This is not a status meeting. It's a synchronization meeting. This is a synchronization event during which we are analyzing the progress of work in this sprint. We are not reporting. We are just talking about the progress. We are trying to figure out what we can do to achieve our goals. Are we on plan, are we on track, do we see any problems, etc. Developers make the decision regarding how this meeting will be organized so they can pick whatever technique they. want as long as it's effective. In the previous version of scrum guide there were three sample questions that might have been used during the daily scrum: what did i do yesterday that helped the development team meet the sprint goal, what will I do today to help the development team meet the sprint goal, do I see an impediment that prevents me or the development team from meeting the sprint goal. Those were the sample questions from the previous version of scrum guide. But like I said, the developers will make the decision regarding how this meeting will be organized. They can use those questions but they can also decide to use some other questions as long as we have the opportunity to synchronize, to have a plan for the upcoming 24 hours. That's perfectly fine to ask any other questions, to reduce complexity. Daily scrums should be held in the same place and at the same time every day during the sprint. Like said before this is only to reduce complexity, so to reduce the situations that "oh on which hour today the daily scrum is held today", "where do we need to meet", etc, etc. This is not the only opportunity for developers to synchronize themselves. It's just to be sure that they have at least one such meeting, one such event, that everyone should participate and everyone should have time to be there. If there is a need to have another meeting, another event or there is a need to talk a little bit longer after the meeting, that's perfectly fine. And a very common mistake is that you do not have to stand during this meeting. This is a common mistake. This is not a daily stand up. This is a daily scrum meeting. Very often scrum teams decide that they want to stand during this meeting. This is only to, well, keep in mind that this meeting should be quick and effective because if you will be standing for too long, your back and your legs will just hurt you so, uh, this is another indicator that, oh, this meeting is taking maybe a little bit too long. Okay, just to quickly sum up things about the daily scrum, a synchronization meeting that happens every day with the fixed time box no matter how long your sprint is. Usually the developers participate in this meeting but there can also be a scrum master or a product owner present there if they have some work regarding this sprint that we are working on currently. There can also be other people listening to this meeting but not participating, not taking an active part. This is not the only opportunity for developers to synchronize themselves. This is only to be sure that we have at least one such opportunity during our day. Sprint review is a meeting during which We summarize and present what we have managed to accomplish during the sprint. The entire scrum team and interested persons, for example stakeholders, clients, CEO of our company are present at this meeting. Timebox for this meeting is no longer than four hours for a monthly sprint, usually shorter for shorter sprints. So for example for a 2 week sprint it should take approximately up to 2 hours. There is no standardized agenda for this meeting. It all depends on internal arrangements. Although, we are presenting what we have managed to do. It should not be equated only with the product demo. This is not about only showing the product. The main goal of this event is to obtain feedback and present plans for a coming sprint. As mentioned earlier, we are living in a complex world and things might change. Things such as our business goals,priorities, law. We saw what happened during the COVID outbreak. Many companies have to change their priorities and this is why during this sprint review presenting our plans for the upcoming sprint is very crucial because client stakeholders can say that, no we do not want you to work on this, work on something else because there is some change and this feature which we thought that we would like to work on it during next sprint is not as valuable as we thought there are some more valuable. This is why during sprint review very often our product backlog will change. Remember, product backlog holds all of the elements of all of the work that we think should be done inside the product that we are developing based on the discussions during the sprint review. Product backlog might change very often. After the sprint review is over there are some further discussions between the stakeholders, the product owner, the developers, etc. Okay, let's head up to sprint retrospective, the last event in scrum. Sprint retrospective will be the last meeting in the sprint. This is a meeting where we will summarize our work but we will focus on how we worked this sprint and what we can do to work more effectively. Sprint retrospective is an internal meeting and we are focusing on the scrum team. Timebox for this meeting is no longer than three hours for a monthly sprint and as usual shorter for shorter sprints. For example, if you are working in a two-week sprint, the timebox for this meeting should be around one and a half hour. There is no standardized agenda for this meeting. It's up to the scrum team's internal decision how they will handle this event. It can be based on just a discussion, on some retrospective techniques like starfish technique. We can also use some flip chart, white boards, sticky notes, anything that will help us to achieve our goal finding the elements that are our impediments, that are interfering with our work and trying to resolve those impediments we are trying to improve our work, to be more efficient, to be maybe more happy with the work that we are doing. Each retrospective should end with conclusions and improvements that we want to implement. So with actions we do not want to only talk about what's not going well, what's not working inside our team but we would like to change things. We would like to improve our work. Usually only the scrum team participates in this meeting. This should be a meeting for the scrum team. We would like to be open, we would like to be honest during this meeting and inviting for example our client or our manager might hinder our openness, our honesty. Okay just to quickly sum up everything about sprint retrospective. During this event we are focusing internally on the scrum team and it is performance, how we can improve our work, what we can do to be better, and what we are doing, how to resolve our problems, how to eliminate the impediments that we have and this is a very internal meeting. We usually do not invite anyone outside of the scrum team. The goal of the meeting is not to grumble about what's not working but to have an action to have a plan on how we can improve our work. We can even decide to pick up the most important action that we found during the retrospective meeting and put it in our sprint backlog for the upcoming sprint so treat it as normal work that needs to be done during the sprint. Okay, let's head up to the event summary. Sprint is a container for other events inside scrum. Time box for this event is up to one calendar month and it helps us to effectively work to support inspecting, adaptation. Sprint planning - timebox for this event is up to 8 hours for a monthly sprint shorter for shorter sprint. Usually, the entire scrum team participates in this event but also we can invite other people's for example specialists, advisors, that would help us to plan the upcoming sprint. The sprint planning helps us to understand the value, what we would like to deliver and how we would like to deliver and set up our sprint goal. Daily scrum is every day 15 minutes long, so the time box is equal to 15 minutes, regardless of the length of the sprint meeting of developers, usually developers. But scrum master and product owner might be present at this meeting if scrum master and product owner are doing some work which is connected to the work that is being done during the sprint, then they participate in this meeting same as developers. The goal for this meeting is to have a plan for the next 24 hours. Sprint review time box 4 hours for a monthly sprint, shorter for shorter sprints. Participants scrum team and the stakeholders, people that are interested in the product that we are developing. The goal is to present the product increment, obtain the feedback and present the expected scope of work for the next sprint so the elements of the product backlog that we think we will work on in the upcoming sprint. And finally sprint retrospective - timebox up to three hours shorter for shorter sprints. Participants - only the scrum team. The goal is to answer the question how we can improve our work, to be more efficient, to be maybe even more happy with the work that we are doing we would like to have. An honest, open discussion so this is why usually we do not invite people outside of the scrum team for this meeting. All of those events serve the same purpose - to implement the three pillars on which scrum is based on - transparency, inspection and adaptation. Events are very often referred to as meetings. There is nothing wrong with using events and or meetings. Probably I've made the same mistake at least once in this course but I will encourage you to use the word event. This is it when it comes to a very short summary of the events in scrum.

12. Backlog Refinement and Estimation

Worth mentioning because this is a very common mistake. Backlog refinement is not an event, it's not a meeting in fact this is an activity to manage the product backlog. Managing the product backlog will refer to doing some activities such as granulating the elements inside the product backlog called product backlog items. So dividing them into smaller parts removes unnecessary product backlog items, adding a description or acceptance criteria to product backlog items may be clarifying some details. Adding some business justification or some clarification etc. Like I said before, just remember, this is not an event, this is not a meeting, this is an activity. Product backlog will change and will emerge in time. In the previous version of Scrum Guide, product backlog was referred to as a living artifact so product backlog will evolve in time, will change. About the sprint planning I said that, the scrum team makes the decision about how many items can be completed, how much work can be done during this sprint that we are currently planning during this iteration that we are planning. Let's dive a little bit deeper into details and this is not part of Scrum Guide. The estimation is done usually in abstract units. So we are not estimating in days or hours and we are not estimating days or hours to complete some work. We are estimating the effort that we need to put to complete some of the work and this estimation is done by developers so the developers will estimate the product backlog items. There are various techniques and methods on how you can estimate your work. The most popular technique is Planning Poker and the two most popular ways of estimating are the Fibonacci sequence and t-shirt sizing. We won't go into details regarding the Planning Poker or the estimation method because those will be shown during your workshop and I do believe that you will also have an experience to try estimation on your own during those workshops. I know that we only scratch the surface when it comes to the estimation so if you're interested in this topic. You can find information about Planning Poker, Fibonacci sequence method of estimating and t-shirt sizing estimation on your own. There are plenty of articles regarding this topic. I just don't want to confuse you about this. The main thing that I would like you to remember from this short lesson is that we are not estimating only time. We are estimating the general effort that needs to be put to complete some work and the estimations are done by the developers. Scrum Guide doesn't insist on any methods or techniques of estimation.

13. Why the most of the companies work in Scrum?

Surely you are able to answer this question yourself already. However let's go and do a brief summary of what you have learned so far regarding Scrum and Agile itself. What is the profit for the company for the organization to implement Agile, Scrum or other well-known Agile implementations? Well, we can deliver our product faster to the market. Like I said we are working in sprints and every sprint should deliver a potentially releasable product increment so we are able to deliver the value to the market really fast, get the feedback and, well, gain some profits. Also we are ready for changes and as we spoke before the changes will come and we are flexible and ready for them. Our product is ready for them, our organization, our developers, everyone inside the company is ready for the changes. Our product takes a lower amount of money to be produced. Most of the time because we are able to react in the proper time to upcoming changes, we are more flexible . We do not have to like waterfall, go back to the first phase and, well, start our work from the beginning. We do have a quick feedback and decision on further product development or stopping the further work if we see that, okay this product is too complex for us or the market competitors have most of the market share and we are not able to compete with them. We can just stop developing our product and save some money. This will help us to reduce investment failures and save the money for the company, so we will be able to produce, to develop the software which can provide us the actual profits and also people inside the organization have the real impact on product implementation. They can talk about it, they can say what needs to be done and how it needs to be done. For example, the development team can say that okay we need to use this tool and this language and this is also very profitable when it comes to morale. People that can be hurt, people that will be listened to, they have higher morale.

14. Why is it worth coming to training with Scrum?

This is it when it comes to introduction to Scrum. I highly recommend you to take part in the classes, in the workshops that you will have during your course. This will be a unique chance for you to learn how the IT industry works before you even have your first job in the IT industry. You will also simulate all the Scrum events and you will take a look at more advanced topics in the Scrum because this course is just an introduction to Scrum. During the workshops you will find new elements like estimation and why it's done in the abstract way, what is story point, what is velocity and so on. You will get a ton of new information like how to increase your chance to get the job and what are other paths in the IT industry. Because, well, it isn't only about developer, tester, devops, it's also about product owner, scrum master and many many many other roles so I highly recommend you to take part in the workshops that you will have during your course. Thank you very much and have a good day.