SCRUM!

THE ULTIMATE BEGINNERS GUIDE TO MASTERING SCRUM TO BOOST PRODUCTIVITY & BEAT DEADLINES

ADAM VARDY

SCRUM! 1st Edition

The Ultimate Beginners Guide To Mastering Scrum To Boost Productivity & Beat Deadlines

TABLE OF CONTENTS

Introduction

Thank you for purchasing “Scrum! The Ultimate Beginners Guide To Mastering Scrum To Boost Productivity & Beat Deadlines.”

This book contains proven steps and strategies on how to learn Scrum fast and how to use this framework in order to conserve time and budget while hitting targets in a timely manner. This book will also show you what kind of products are worth developing and how you can use Scrum principles within a large group to create projects that are measurable, efficient, and reliable when it comes to delivering customer satisfaction and good return of investment.

Managers, employees, and entrepreneurs would be able to benefit from the knowledge on how they can be better involved in tasks within their organizations and how they can be more efficient in hitting their targets. By using the Scrum framework, you would realize that there are better ways to complete projects and have them launched on the right time without compromising time for development.

With this book, you will learn how to use Scrum to cut the time you need to plan and organize in order to submit deliverables way before the deadline. By learning Scrum, you would be able to generate more products that meet customer satisfaction and have efficient use of your resources to meet organization goals.

Chapter 1: Why Scrum is Probably the Most Efficient System Out There

A lot of organizations in the world operate this way – they fulfill requirements that people from the higher ups tell them to meet, start working on a plan that they have discussed, and then meet up with a client or a boss to check if they like what they have done.

Now, you can see that there is a likelihood of failure in this type of management, and that failure is more likely to be discovered right on that project's deadline. When that happens, the organization and its client have already spent a lot of time and resources working on a product that they both won't like in the end.

There's a project management system that was designed to address this problem. Called Scrum, this system is designed to cut the time and resources that any organization out there needs to finish a project.

What is Wrong with the System, Anyway?

The traditional method of getting a project done is designed for people to make sure that they have all the business requirements that they need in order to start designing something. After meeting all these requirements, they will start working on a plan or a design. Right after that, they will start making that plan work and test the outcome. Then, the client or the boss walks in and then says that it is okay to get that project launched.

Of course, the plan is to make sure that everyone does what they are tasked to do – they have to make sure that all things written on the Gantt chart is done before the next step is taken. They also need to finish all the agreed tasks on a specific deadline so that they can start testing.

When the teams are done working on what they are supposed to do, then that is the time that they will know if what they did is satisfactory or not. This method of following one plan after another is called the Waterfall Method.

The Waterfall method makes it a point that project developers go through these steps:

1. Analyze Requirements

If you are into software development or any type of project creation team, you would want to know the business context of what you are trying to create – you want to define what kind of problems you are trying to resolve and how people would react to your finished product. After you define all these “requirements”, you have the input that you need to move on to the next step.

2. Designing

This step is made up of all the steps that you need to satisfy all the requirements that you have determined earlier. In software development, this is the part where you define all the software and hardware architecture, programming language, data storage, etc. This is also the part wherein you determine how the project would be useful to its end user.

3. Plan Implementation

In this step, you begin to construct what you have designed in your plan. This part of the Waterfall method is dedicated to meeting the standards that you have made in the previous steps. This is the part where people from the development team come in and make all the things discussed in the previous steps happen.

4. Testing

This is the part of the method where quality assurance people enter to ensure that the development team did not make any mistakes. This is also most likely the part where people realize what is working or not working in their plan.

5. Client Approval

When all things are satisfied by the project implementers, the client or the end user comes in and makes the final call if the project is ready to be launched.

The Waterfall method makes it a point that when something goes wrong in a particular stage, people can go back to the previous one to see what went wrong. For example, if there is a problem in the Plan Implementation and people know that they simply followed the blueprint that has been handed over to them, then managers look at their plan and make their revisions from there.

However, it would be easy to point out what could possibly be wrong when you follow colorful, organized, and time-coded Gantt charts from the conception of an idea to delivering the finished product – there is no guarantee that all things that the organization has thought of would actually work. From here, you would realize that the Waterfall Method has the following issues:

1. People blindly follow plans.

In the traditional method, people pay more attention to how things will happen during the right moment without being mindful if things are really falling into place. While planning is important, it is also important that developers and quality checkers understand how things should happen, especially with the client or the end-user. It is also important that all people involved in the project can immediately say how a particular step in project fulfillment can fall apart without having to wait for the testing stage.

2. This method can take up too many resources.

Going back to a previous stage in the Waterfall method may mean that all the time and money spent is not going to turn up any product. That means that it is possible for management to just waste its resources when it finds out that something is not working out well in a stage, and then spend more resources going back to the previous stage to find out what went wrong.

3. End-users do not know what they want.

When it is time to hand over the finished product to a client, it is likely that they will not like how it turned out, despite deliberately saying otherwise during the initial stages. It is easy for clients and end-users to change what they want over time. The Waterfall system does not have a way to resolve that yet, without having to revise plans and redoing the entire project altogether.

4. Testing for quality may suffer.

It is impossible to accurately predict the outcomes of a project, and when the entire team is pressed for time, it is possible to cut the testing stage short in order to meet the deadline.

5. You’ll never know what stage you really are on.

Since the product that you are trying to create will not be produced until the very end, you are not really sure if you are still on planning or you are already on developing stage. That means that it is also likely for you to spend more time on a stage than what you have expected because of this poor visibility.

In the end, the Waterfall method can be too risky since it is too rigid. In order for you to make you produce a product that works and would be flexible enough to help you figure out what is working or not.

Enter Scrum

Scrum, an Agile method, aims to help you solve problems in product creation with extreme flexibility. This framework aims to make you address adaptive and complex problems with the highest productive value without compromising productivity in the process.

Scrum is based on the assumption that knowledge come from what you have already experienced, which means that you can only make decisions based on things that you already know. Since this framework works according to what is already out there, it implements an incremental and iterative approach to make sure that you have a handle over risk and also optimize predictability of your tasks. To do just that, this framework stands on three pillars:

1. Transparency

All important steps and aspects of all processes are visible to those who are responsible for outcomes. That means that everyone involved knows when a part of the process is already successful or completed, and everything understands what they steps need to be taken to ensure the success of a particular task.

2. Inspection

Those who are using Scrum are bound to always check artifacts and progress in Sprint Goals in order to see if there are any variances that they do not want. Of course, everyone involved knows that checking their progress should not get in the way of work.

3. Adaptation

If the person that took care of the inspection sees that there are aspects of the process that would make the end product unacceptable, then materials and processes are immediately adjusted to create the desired outcome. These adjustments are made as soon as possible in order to minimize any other resulting deviations.

Because Scrum stands on these pillars, people who use this framework does not have to rely on a single team to fulfill stages before they even have an idea of what they are supposed to create and to see if the product that they are making will actually work.

Since Scrum eliminates the waiting time between stages and immediately points out what process people concerned are taking care of, it is easier to see if the people involved are making progress or if they need to make corrections. This effectively cuts the time and budget people need to spend to complete projects.

Chapter 2: Will Scrum Work?

Scrum is a very lightweight and simple framework, and is the way of thinking that top companies such as Honda, Fuji-Xerox, and Canon has adopted to produce world-class and high quality results. With this framework, you would learn how you can create projects through an all-at-once product development using a team-based and scalable approach.

While it was popular as a software development process, Scrum's name actually came from the rugby sport – in a game of rugby, you restart the game when the ball went out of play or someone has committed an accidental infringement. Scrum as a framework serves a similar purpose – instead of making a game keep going despite a possible error, you fix the error right away and play the game right.

What You Do When You Scrum

Since Scrum is an agile approach, you would need to fulfill a product backlog, which consists of all capabilities and features that you need to create in order to come up with a successful product. This backlog will tell you the things that you need to do first. If you run out of any resource, any item on the backlog that has not been completed will have lower priority than the completed work.

Any project done using scrum is done within short iterations, known as sprints that are within time boxes, which would range from a week to a calendar month. During these iterations, a cross-functional team does all the work.

The jobs would include building, designing and planning, and testing. After they are done, they will connect with the stakeholders to get their feedback. Based on what the stakeholders will say, the owner of the product and the team will make alterations on the product and then plan on how they are going to do the next iteration.

If the stakeholders see a feature that has already been accomplished by the team and then realized that they want a new feature that has not been considered earlier, the product owner can just make a new item in the backlog so that it can be done in a future iteration.

By the end of each sprint or iteration, the team produces a product that can already be launched, or an addition to an existing product. If everyone has decided that they cannot release a product yet after a completed iteration, then everyone can decide to release a bunch of features from multiple iterations instead.

What Makes Scrum Great

While the previous chapter has already discussed the Scrum pillars, you need to know what makes Scrum great when you develop products or create a flow for different types of jobs. Here are the things that you expect from a Scrum project

1. Customer satisfaction

Scrum eliminates the danger of not meeting customer needs – instead of simply focusing on features, Scrum identifies what kind of product would be useful to the end user.

Because of this foresight, Scrum minimizes the risk of producing a product that would be declined by client.

2. Less resources needed

Scrum makes it a point that all teams would start working as soon as they have a list of what they need to accomplish. Instead of working on a definitive plan that should not be broken at any cost, Scrum requires teams to focus on finishing backlogs that are already created ahead of time. Since the plans are always flexible according to what works, teams do not need to spend more time planning and instead spend resources on building and maintaining the product.

3. Faster turnaround time

Since people are more focused on doing a working product than creating its project architecture, this framework is more likely to produce a quality product with less time.

4. Better ROI

Clients and organizations that want to create projects are likely to enjoy improved return on their investments since products created through Scrum are more likely to be tested for quality and launched earlier in the calendar. Since the team releases smaller and more frequent improvements or additions to an existing product, stakeholders see that they enjoy better returns for what they have invested.

5. Better employee satisfaction

Since employees are able to measure their success and know that they are contributing to the success of a project, they are found to be more motivated in engaging in projects that they are involved in. Of course, more motivation for employees means better productivity during iterations.

Can Scrum Work All the Time?

While Scrum may be an effective framework for many kinds of projects, it is important for you to realize that it is not the panacea for all kinds of projects – it can be an excellent solution for a lot of situations in your organization but it may not be the proper answer in some scenarios.

A framework known as Cynefin allows you to make sense of situations that you would encounter in project scenarios. Take a look at these possible disorder scenario domains in your projects to find out whether Scrum would work for you or not:

1. Simple

Simple scenarios are situations wherein everyone can see the cause and effects – people know what the right answer is. This type of domain exists is best for reproducing products over and over again since there are steps that you can repeat to solve existing problems. While Scrum can be a way to address this type of scenario domain, you can achieve faster and more certain results with the assembly line frame work since you have legitimate best practices anywhere.

2. Complicated

Complicated scenarios are often dominated by experts in order to ensure good practices in a project. While there is a possibility of having multiple answers, having an expert to diagnose problems would be more efficient in addressing scenarios that have multiple possible outcomes. Scrum may be used to resolve scenarios like this, but of course, it may not always be the best framework to use.

For example, if you need to adjust parameters in optimizing performance, you may know the best course of action when you assemble experts to take a look at the situation, check out available options, and generate a response based on their experience of what a good practice should be. Complicated scenarios are best resolved through quantitative and tactical approaches like Six Sigma.

3. Chaotic

These kinds of scenarios require a solution right away – when your project goes to the chaotic domain, it means that you are experiencing a crisis that your need to put out before you experience any additional harm and to go back to the state of order. This is the type of scenario that you experience when someone files a lawsuit or the only expert that you know is nowhere to be found.

Scrum is not suitable in addressing these kinds of scenarios since you do not have the luxury to prioritize backlogs and what you need to do during the next situation. When chaos hits, you need to have someone with full authority to take charge and act ASAP.

4. Disorder

Disorder domains happen when you become uncertain of what stage you are in in your

project and you do not know if any plans are working or not. This is a very dangerous situation that you need to get out of immediately.

This requires you to break the situation apart and look at its components to know if they fall into any other categories to see what kind of approach you should use to solve the situation and go back to making progress. You can see that there is no point in adapting Scrum in this kind of situation until you properly categorize the components of the problem.

5. Interruptions

When you have a project that would be experiencing multiple interruptions, then you may need to use a different approach from Scrum – in an interrupt-driven work scenario, your backlog would be filled up non-stop and you do not have the luxury of time to work on them in the future.

Moreover, your backlog can change frequently. This makes iterations unreliable since you do not know what future iterations would require you to do. You are also likely to receive high-priority backlogs that would prevent you from planning and achieving the next iteration you have in mind.

Because work flows with a lot of interruptions also need an agile approach, you may want to use an agile framework, such as Kanban, that would allow you to limit work in process to help you make sure that you are not handling more than what you can do. An agile framework also allows you to optimize your work flow, measure your progress, and see where you can improve your approach.

6. Complex

Complex problems make you realize that there are more unpredictable results than what you can expect from your approaches, and in this domain, you are more likely to get the right answer when you correct a mistake or hindsight.

This requires you to explore as much as you can about the problem, inspect what is working or not, and then adapt based on what you have learned. This is also the domain that you would encounter when you are trying to make innovations or making enhancements for previous projects that require better features.

This is the domain where Scrum truly shines, since complex situations make use of your ability to probe the situation and inspect what you can do in a particular timeframe. When you learn a better method in the process of iteration, then you can adapt that on the next iteration timebox.

Chapter 3: The Scrum Team

Following what you have read on the previous chapter, Scrum is not a standard that you can follow faithfully in order to make a guaranteed product that would make customers happy, or to develop anything within your budget and deadline. Instead, it is better to understand Scrum as a framework that would enable you to organize and manage work better.

You can think of it as the foundations of a building – you can't simply disregard or make sudden changes to a practice, principle, or value without getting the risk of a collapse. With Scrum, what you can do is to add fixtures and customize a building until you get a process and product that works best for you.

Crucial to keeping the Scrum framework working are the roles that people do when they agree to adhere to this type of management. Making Scrum work means that people within the Scrum team would be able to fulfill their roles efficiently to prevent any problems.

The Product Owner

He is responsible for telling what should be developed and the order of items that needs to be fulfilled. You can consider him as the sole authority that would tell the rest of the team what they need to create and which features should come first. In short, he is the one who tells the other members of the team about what they should be coming up with.

For that reason, he creates the product backlog that contains all the product goals that the development team needs to accomplish when they are ready to start working. The product owner also needs to be available at all times in any case the development team and the ScrumMaster has any question about the goals that he has mentioned in his product backlog. Because of this, the product owner bears the responsibility of making sure that the product would be successful during development and maintenance.

This role entails that the product owner needs to do the following tasks:

1. Manage economics

There are several things that a product owner needs to meet to make sure that he is managing resources well, especially when the entire team is ready to start developing the product. When he manages economics, he needs to make sure that he manages the following in the process.

a. economics at release level

At this point, the product owner makes a series of trade-offs when it comes to date, scope, quality, and budget as he gets information during the product development. For example, if he sees that the team may produce a product that may create extra revenue if they work on an additional week, then he must make a trade-off for the budget and the product's final release.

Also, he may make the decision to not fund an additional week of work if the work planned to be done on that timeframe does not mean creating a better product. It can also be that the resources that the team is working on will not justify additional work. He can

also decide that the goals that they need to work on should change, or that the entire team should stop production because of problems encountered with other stakeholders.

b. economics at sprint-level

It is the product owner's job to make sure that there is a good return of investment (ROI) happening in every sprint. Whenever they do economics at this level, they treat the money that the entire organization is using as if it is their own. This makes them consider if the features that are bound to be spent on is really worth doing.

c. economics in product backlog

Since the product owner is the one who is going to create the product backlog, he needs to make changes in priorities in this list when economic changes happen. For example, if the development team says that they need to work more on a feature that was earlier predicted to take only modest effort, then he sees that the benefits that everyone would get from this feature has already changed, prompting him to put another feature up as a priority in lieu of the heavier task.

2. Take part in planning

During product planning, the product owner deals with the stakeholders to get their help in envisioning the product. Whenever a sprint is accomplished, the product owner goes back to the stakeholders and the rest of the team to define what should be done next. When he is planning the sprint with the rest of the team, he provides the input that the development team needs to look at which items in the backlog can be realistically done within the sprint's timeframe.

3. Create the product backlog

When the product owner grooms the product backlog, he takes care of the refining, prioritizing, and estimating of the items that are listed there, with the help of the members of the Scrum team. While he may not be entirely knowledgeable of the development process, he is available for consultation and clarification when the development team foresees that some amendments must be made to meet their deadlines.

This ensures that goals are updated according to the work that can be done during production and that backlog items can smoothly flow into the next sprints.

4. Make criteria on what is acceptable and check that people are meeting them

The product owner makes sure that all goals that should be met in a sprint session have been accomplished by seeing to it that both non-functional and functional requirements have been met by the team. He may consult with experts to do this or get assistance from the development team.

The acceptance criteria are crucial to the Scrum team because it tells everyone about the project's progress. Without them, the development team would not be able to understand what defines a completed work and would not be able to include better practices on their next sprint.

5. Work with the development team

The job of the product owner becomes an everyday role since he is required to stay engaged in the tasks of the development team in order to prevent delays on essential feedbacks that can already be incorporated within the day.

It also makes it possible for the product owner to see specific features that are initially required but may no longer be needed when other features are done. When the product owner is always within reach and ready to provide feedback, the organization prevents unwanted expenses by creating timely adaptations for better practices.

6. Work with the stakeholders

The product owner is the sole voice of the stakeholder community that speaks to the people involved in the production process. When the product owner is able to work closely with all persons involved in the product creation that are outside the Scrum team, he would be able to gather all the input that he needs to create a coherent vision in the development process. This way, the entire Scrum team prevents unwanted risks in developing features that may not meet client and customer satisfaction.

ScrumMaster

The ScrumMaster takes care of team guidance in developing the product, as well as following the process based on Scrum. He is the one who makes everyone understand practices, principles, and values that everybody needs to stick to in order to achieve the success of the project. In the Scrum team, the ScrumMaster serves as the coach that provides pointers on how to optimize performance by providing the necessary process leadership.

The ScrumMaster also helps resolve potential issues and make improvements on the projects by following Scrum. He also makes sure that the team is protected from any outside interference and removes anything that may hamper productivity. However, this does not mean that the ScrumMaster has total control over the team – he acts as a leader, and not as a traditional project manager.

Since the main task of the ScrumMaster is to serve as coach, there would be a point in the day-to-day activity when the ScrumMaster does not need to actively guide the development team anymore, especially when the Scrum team has accomplished multiple sprints.

Because Scrum is designed to prevent variances and make work in progress more efficient, the development team would arrive to that point when they do not need coaching anymore. However, the ScrumMaster would be very valuable whenever the Scrum team is about to start a new sprint and the entire team needs to incorporate a product backlog that they have not encountered before.

Development team

The development team determines methods on delivering what the product master has asked for. This team is comprised by different people with different job descriptions, such as designer, tester, and database administrator, and architect, which allows them to cross-function and become dynamic when it comes to designing, testing, and building the product required by the product owner. Because of their task, it is important that the development team is capable of being self-organized to decide on the best way to meet the goals that are set by the product owner.

Here are the responsibilities that the development team has within the Scrum framework

1. Execute the sprint

When a sprint happens, all members of the development team do the designing, integrating, building, and testing of all items in the product backlog by working in increments and producing potentially shippable features. In order for them to do all these tasks, they need to be self-organized and work as a collective to make their own decisions when it comes to communicating, managing, planning, and carrying out work. Majority of their time is spent on sprint execution.

2. Grooming the backlog

Whenever a sprint planning happens or before they start producing features, the development team allots its time to assist the product owner when it comes to refining, prioritizing items, and creating items on the product backlog.

3. Planning the sprint

The development team works with the product owner and the ScrumMaster to develop the goals that they need to achieve during the next sprint. The team would be responsible for identifying which subset of the product backlog should be prioritized in order to achieve the goals they have identified.

4. Inspect and Adapt

When a sprint ends, the development team becomes involved in reviewing the features they have accomplished in the previous sprint and what technical and process practices they need to adapt in their next sprint.

Because of these responsibilities, the development team needs to be self-sufficient and capable of cross-functionality. That means that it may be necessary for the team to be made up of people that come from different backgrounds that equip them different cognitive skills. While the members of the team have their own specialties, they should also be able to adapt as individuals to tasks that are outside of their core discipline.

It is also important that the development team has the right size. Scrum prefers that the development team is small (just about five to nine members) in order to maintain its efficiency. By keeping the team small, there is no chance for members to do any social loafing or slack around with the thought that there would always be someone that would do the work for them.

Less time is also spent on communicating efforts on other team and constructive interaction is also promoted in a group with a small number. In addition, everyone in the team realizes how important everyone in the team is, which makes harmful specialization less likely to happen.

What About Managers and the QA?

Scrum does not need to have a number of managers in order to make the entire Scrum team sustainable. All that they need is to make sure that the product owner and the ScrumMaster works together well when it comes to ensuring that the development team is doing all the work needed.

Some organizations try to still implement separate testing or install a QA team while they are adapting the Scrum framework. While there are some instances that a separate team that performs testing on features may be necessary, it is often the case with Scrum that there would be no need for such. Because testing is already interwoven with work that happens in every sprint, the development team can already test the features while they are still working on them.

Chapter 4: Activities and Artifacts

When an organization decides to adhere to the Scrum framework, everyone involved, including all stakeholders and members of the Scrum team must make sure that all tasks are completed. To measure that, they need to follow a protocol that would allow them to keep track of activities and artifacts that they need to accomplish.

While Scrum makes it a point that the workflow in any given project is dynamic, it still needs to follow a specific of events. By following this flow, people would be able to determine what tasks they are doing and what needs should be fulfilled in the process of producing a product. This flow also shows them when they need to adapt a better practice, scrap a feature altogether, or launch the product for the end users to receive them.

Here are the activities and artifacts that each member of the team needs to take care of in order to create a product or complete a project:

a) Product backlog

This is the list that contains all the things that are needed to be accomplished in a product or project. To obtain this list, the product owner, with the help of the input from the stakeholders and the Scrum team, creates a sequence of tasks that are arranged according to importance. The product backlog contains the initial vision of the product master before any work is done, and after a few iterations, may contain changes to alter features that are not working out, required repairs, possible improvements, and so on.

Some Scrum teams refer to their product backlog as “user story” since that reminds them better that they build products or finish projects to satisfy the end user’s needs. In creating the items or stories in the backlog, the product owner defines the following:

i. which types of users will the item benefit
ii. what needs to be built
iii. the reason why the feature is important
iv. how much work the item needs to be implemented
v. standard that will tell that the item has been implemented correctly

Before all these are finalized, you need to look at the size of each item that you have placed in the backlog. If the item in the backlog is huge, then it entails a higher cost. In any project, cost needs to be determined in order to tell if it should be a priority or not.

For that reason, a lot of teams use a term called relative size measure, which you may encounter as ideal days or story points. Using this measure, you can tell the overall size of an item in the backlog and then compare it to the size of other items.

b) Sprints

In Scrum, the sprint is the iteration that exists within a particular timeframe. Within sprints, the team aims to create something that is already valuable to the end user.

Within the sprint is a sprint backlog which serves as the to-do list that needs to be done within a particular duration. Within the sprint timeframe, the ScrumMaster makes it a point that there would be no reason for the development team to not meet any of the items in the sprint backlog.

To do this, it becomes a practice to make sure that there is no alteration in the goals that have been expressed earlier – there should be no changes done in the sprint backlog and there should be no changes in personnel involved. For that reason, the product master and the ScrumMaster agrees on what should be in the sprint backlog, and then the development team reviews which high-priority items can be realistically achieved while doing the job in a sustainable pace.

That means that the development team has a say on what kind of pace they should be working on to comfortably achieve what is said in the backlog. To be able to determine the suitable pace, the development team divides the items that the product owner and the ScrumMaster agreed to do on a specific sprint into smaller tasks, which becomes the items on the sprint backlog.

When the development team plans their sprint backlog, they usually define how many hours they need in order to accomplish the task. Since sprints are supposed to be done fast, they do not want to take too much time in planning – the sprint plan for a month should be completed in less than 8 hours.

The development team should be able to see if the tasks they need to do would fit in the sprint timeframe that the ScrumMaster gave them. If it does, they would be able to repeat the work and check if they can accommodate more tasks.

c) Execution

Once the Scrum team is done with planning and everyone is set with the tasks that they need to accomplish on the next sprint, the development team goes to work under the guidance of the ScrumMaster. The team would only consider a task done if they are confident that all tasks necessary to create quality features are accomplished.

At this phase, nobody tells the development team what to do in order to achieve all items specified in their sprint backlog. Instead of being directly managed, the development team organizes itself according to what can make them achieve their tasks efficiently.

d) Daily Scrum

The development team members hold a specific portion of the day to inspect their methods and see if there is a better practice they can adapt to improve the quality of the product they are making on a particular sprint. This normally happens as a 15-minute (or less) huddle within the development team. The ScrumMaster assists them on their huddle by asking each member the following:

i) Accomplishment since the last daily scrum

ii) Plans for the next scrum

iii) Any obstacles that prevents progress

When team members address these issues, they become better aware of the bigger picture – they see if they are making any progress or if there are any modifications they need to make in their activities and immediately put them into action without having to go back to the drawing board and redo the plan all over again. By taking note of any improvements or additional good practices that they can perform, they know that they can do a much better product on the next sprint.

e) Done

All people in different roles need to agree on the concept of “done” before they move on to other items in the backlog that they agreed to resolve on the next sprint. When everyone agrees on the concept of “done”, they should be able to reach that level of confidence that the amount of work they have completed is good quality work and has the potential to be shipped.

When you think about the concept of potentially shippable product, it does not have to be a product that the team would already deploy to the client or customers. It only means that when all other things are satisfactory in business (the customers can absorb another change that may happen to a product they have deployed or the organization has reached a good amount of features to justify deployment to customers), the product that the team has created is ready to go public.

In Scrum, a product is potentially shippable when everyone is aware that the amount of work accomplished in sprint is satisfactory and there is no work or feature neglected in the work process. That means that the work accomplished is already enough to generate feedback, which would determine if they need to add features in the product backlog or that they are ready for launch.

f) Sprint Review

When a sprint is accomplished, it becomes necessary for everyone to inspect and adapt the project or product that everyone is building. At this point, the Scrum team, sponsors, customers, stakeholders, and other interested parties hold a conversation to review the completed features based on the context of the development's overall effort.

If the sprint gets a successful review, everyone who is not on the Scrum team would be contributing ideas on how the development can improve. This is that point wherein stakeholders, customers, sponsors, and other concerned parties can pitch in to the development's direction.

g) Sprint Retrospective

When the Scrum team has received the review and feedback from others, then they would be doing their own inspect and adapt process. At this point, everyone in the team would be discussing what worked and what they need to throw out of the process.

Once they are done with this phase, they are ready to plan their next sprint based on the assessment of their previous workflow and technical processes.

Chapter 5: Agile Principles at Work

How do you know if the Scrum framework is going well? Since this is an agile framework, you would need to see if the practices that all stakeholders and members of the Scrum team works well with the agile principles that they need to follow.

The Principles of Agile

Going back to the traditional Waterfall system, following a plan-driven development only works when every challenge an organization meets are already predictable, well-defined, and very unlikely to change. While a plan-driven approach such as the Waterfall would be useful for many scenarios, you can think of it as an extremely measurable, accountable, and orderly means of embarking into a project. However, you would need agile methods when you are already aware that things are not likely to be predictable.

For this reason, frameworks like Scrum adhere to an entirely different set of practices and principles in order to address uncertainty. Agile principles are divided into these categories

1. Variability and uncertainty

Agile processes are created to embrace any variability that may help produce a better feature. In order to address variances and the unknown, they apply iterations and pursue developments in increments in order to make sure that they are going to meet well-defined goals in time.

By embracing variability, they turn it into a leverage to pursue adaptation during work by transparency in processes and constant inspection. Because variances would become predictable in time by adapting it when necessary, it is possible to reduce different forms of uncertainty simultaneously in an agile framework such as Scrum.

2. Prediction and adaption

In an agile framework such as Scrum, you would need to keep all options open and accept the fact that it may be impossible to get flawless features in a single go. In order to achieve a potentially shippable feature, frameworks like Scrum would need to explore and adopt changes, as long as they are economically sensible. It also means that agile processes need to create a balance between predictable work upfront and an adoptive work that meets the schedule.

3. Validated learning

In an agile framework, all people involved need to validate all their assumptions and they need to do it fast. They also need to leverage different simultaneous learning loops to make work rendered faster. Because of this, it is necessary for a very organized workflow to make timely feedbacks possible.

4. Work in Process (WIP)

Work in Process refers to any work that has already started but is not yet finished. When dealing with WIP, it is important that you commit to work that is economically sensible so that teams can finish batches of work within the timeframe and budget that they need. In an agile framework such as Scrum, small batches of work are generally favored to promote the following benefits:

1. Reduced cycle time – small amount of work produces lesser amount of tasks that needs to be processed in a sprint, which means that there is less waiting time for a team member when he needs to get another person's output before he gets started. This means that everyone in the team gets work done faster.

2. Reduced flow variability – when smaller amount of work is done in batches, they flow nicely within team members since they can count on resources to accommodate a batch.

3. Accelerated feedback – work done in small batches can already receive feedback since they are accomplished faster. This minimizes error risk

4. Reduced overhead – when teams work in smaller batches of work, they do not need to spend too much on their overhead to complete a cycle,

5. Increased sense of urgency and motivation – small batches makes it possible for everyone involved to see possible causes of delays, mistakes, and development progress. Since they can already correct mistakes and see their progress, people get more motivated to work on the next batch immediately.

When you consider all these, you would also be able to compute the cost of delays more effectively by seeing the amount of resources that you have lost because of unfinished work. This allows you to reprioritize items that you need to take care of.

At the same time, you also get to focus on idle work instead of having the management feel bothered by idle workers. By looking at work that is still not done, you would be able to calculate overhead costs that go to waste and delegate tasks more efficiently.

5. Progress

When you use Scrum as a framework, you would be able to measure progress at work according to what teams have already delivered and what is already validated. That means that you do not need to be too concerned about how far teams are in a production phase or if things are going according to plan. When you look at progress at work this way, you do the following things:

a. Adapt to what you know now

Since you do not have to follow a plan so faithfully, you are already aware that things may not go as expected. In Scrum, you need to make provisions for replanning as soon as you get the information that you need to adapt to certain changes, especially if you have an economically important detail that you need to consider within the development.

b. Measure through validation of working assets

You would only be able to measure progress when you create validated assets that work. They should be able to deliver the value that your clients and end-users need, instead of simply meeting your budgets and deadlines. In the end, this would allow you to receive the feedback that you need to identify what your next step in the project should be.

c. Focus on value-centric delivery

A process does not matter if the development plan would not be able to deliver anything of value. Since agile frameworks such as Scrum define progress according to satisfaction and value, artifacts that are defined earlier can change if the situation has already changed for the customer.

That means that when you adhere to Scrum, you would always have to validate assumptions and turn them into knowledge of what is perceived to give customer value. In Scrum, every artifact that does not present a value does not provide you any useful assumption to aid you in knowing what you should do next.

6. Performance

In Scrum, there are different work characteristics that describe what good performance is. You are aware that you are doing excellent performance when you:

a. Do work fast, but not in a hurry

While agile processes require quick feedbacks, the quickness of doing work in a framework such as Scrum does not mean that you quickly move from Step 1 to Step 2. That only happens in a scenario with zero variances, which is while desirable, less likely to happen. In Scrum, your goal is to be speedy in development, without neglecting the need to be flexible and highly adaptive.

Time is very important for Scrum to work, since people are working within timeboxed

sprints. However, you would not want to rush things in this framework in order to make sure that you are working in a sustainable pace. When you stick to the principle of sustainable pace, you stick to the idea that you would work at a pace that would allow you to keep on working for an extended amount of time. This prevents exhaustion from team members, unpredictable expenses, and the risk of making quality suffer.

b. Create quality products

Traditional frameworks like Waterfall believe that careful planning and doing everything according to plan is the secret sauce in creating a high-quality product. However, it is important to know what quality really is, and you cannot do that until a latter phase in the process. If testing tells that the product lacks quality, then it would be important to do the very expensive test-and-fix method.

In Scrum, a cross-functional Scrum team owns the quality of the results and they make it a point that they build according to it in every sprint. When they produce increments of the product, everyone is already aware that it is done with a high level of confidence and it already has the potential for shipping. Because of this good practice, there is no more need to test features later in the process and the risk of incurring additional costs for revising is dramatically reduced.

c. Observe ceremonies as minimal as possible

Ceremonies or formalities often play a big role in tradition-driven working processes. In methods like Waterfall, the success of a phase is often measured by adapting checkpoints, creation of documents, or making a new team to work on the product’s aspect. This increases the costs of delays and blow up the overhead.

In an agile framework like Scrum, organizations make it a point that they minimize these ceremonies as often as possible. Since there is no need for any checkpoint, apart from the end of the sprint wherein teams meet up with the stakeholders, work is done faster. Moreover, confidence among stakeholders already exist and they already are aware that features produced in each sprint are about to be ready for production.

By learning these principles, you become aware of how to turn your organization into a Scrum environment. Now, it’s time to understand how to optimize performance and ensure confidence in a Scrum process.

Chapter 6: The Product Backlog

As you have already read in an earlier chapter, the product backlog is the list that contains all the priorities in a project. It also is the very thing that provides a shared understanding of what people should be creating and in what way they should be building it. In Scrum, as long that there is a system or a product that is in the process of being created, improved, or supported, the product backlog exists.

What Should Be In It?

Product backlog items, or PBIs, are functionality items or features that have a real value to a customer or user. These items are often told as “user stories”. PBIs may also be in the form of defects that need to be repaired, knowledge-acquisition work or research, or technical enhancements.

Good and efficient product backlogs all have the same traits. These are the following:

1. Detailed appropriately

Not all PBIs will all have the same detail that the Scrum team and the stakeholders are aware of at the same time. The PBIs that are needed to be worked on as soon as possible are those that are small in size and are very detailed so that they can be accommodated in the nearest sprint. Those that are not yet defined and still not divided into smaller sets of tasks are those that belong to the bottom of this priority list.

2. Emergent

The product backlog should not be frozen or seen as complete as long as there is still a product being built, repaired, or maintained. It should be continuously updated according to all the information that an organization receives, and that means that it will be updated as often as possible, especially when competitors make a bold move to challenge the product, customers provide poor feedback about a feature, or an unforeseen technical problem arises.

While these scenarios may be problematic, the product backlog must always be flexible enough to adapt to these scenarios.

3. Estimated

Each PBI should have a size estimate which equals to the amount of effort everyone involved in the product creation should exert to make it possible. These estimates would be one of the most important factors that the product owner needs to consider in making priorities in the product backlog.

For a product owner to put an item above the rest means that that item has already been measured to be within the working capacity and budget of the organization and has foreseen economic benefits that can be cashed in as soon as the next sprint is done.

4. Prioritized

While the product backlog is known to be a list of project priorities, you need to understand that not all items in there will be prioritized in the sprints. It is however very

useful to prioritize PBIs that everyone has already decided to belong to the foreseen and upcoming sprints, or items that needs to be taken care of until the initial release of the product. However, afterthought features that may be done after the release should not be included in the product backlog.

Grooming the Product Backlog

Grooming refers to the following activities:

1. creating and refining PBIs

2. estimating items

3. ordering items in the backlog according to priority

All items in the backlog need to be estimated in order to identify the rank in the backlog and decide whether the Scrum team needs to do any additional work to refine an item. At the same time, when an important information comes in, new items would be created in the backlog and inserted in this list according to how they should be prioritized.

When you get closer to a large item, you would need to refine it into smaller items to know how you can fulfill it. You might also just decide that an item in the backlog is not necessary to spend resources on and should be deleted from the list.

When the product owner, the development team, and stakeholders decide to groom the product backlog, they do not agree on when grooming should take place. When the work has already started, grooming happens as an out-of-the-flow activity, since Scrum has an uncertain environment where everyone should be prepared to constantly inspect features that are being worked on and adapt changes or good practices when necessary.

Taking the time to stop work and meet up for grooming should only happen when it is very necessary; otherwise, it would disrupt Scrum’s fast workflow.

When is the Backlog Ready?

When the product owner is done grooming the backlog and the development team is confident that they can achieve assigned products on the next sprint, then the backlog is considered as ready. That means that the following criteria have been satisfied:

1. the business value of the items are clearly stated

2. details of items are understood by the development team and they are able to make a decision that they can complete items in the backlog

3. team is adequately and appropriately staffed to fulfill in the backlog

4. the PBI is small enough and estimated to be fulfilled in a sprint

5. the product owner has made clear and testable acceptance criteria

6. the Scrum team knows how to demonstrate the item during the sprint review.

Chapter 7: Estimating Work and Measuring Velocity

When you are managing or planning how a product should be built, you need to answer questions like “How many features should be done?” “When will all the work become finished” and “How much budget will the tasks use?”

To use Scrum to answer these questions, you would need to have an estimate of the size that you need to build and measure the velocity or the rate at which you will be able to get the work accomplished. Once you have all those details, you are likely to come up with the overall time needed for development and its corresponding cost.

How Long and Costly is Work?

You can easily derive the product development duration by simply following this formula:

(estimated size of feature set) / (team’s velocity)

Wherein:

Estimated size of feature set = the total of size estimate for each product backlog item (PBI)

Team’s velocity = the amount of work the Scrum team completes each sprint

The velocity is easily measured by adding the size estimates of all the PBIs that was accomplished during the sprint.

What You Need to Measure

In Chapter 4, you have read that relative size measure is done in order to calculate the amount of work to be done. Since items in the backlog are also called “stories”, the estimated amount of effort that needs to be done in order to complete a feature is called “story points”. Alternatively, you can also get the relative size measure by knowing the “ideal days” of work. Here is how these measurements work:

1. Story points

These are numbers assigned to a particular item in the backlog, which shows the relationship of other items that also needs to be accomplished. For example, if you need to create a title for your project, it could have a story point of 2. Getting the details for publicity copy for your project may have a story point of 8.

That means that the effort that needs to be done to do research for your publicity is 4 times larger than creating the title. Take a look at this diagram to have a visual idea on how the story points are assigned depending on the size of a construction:

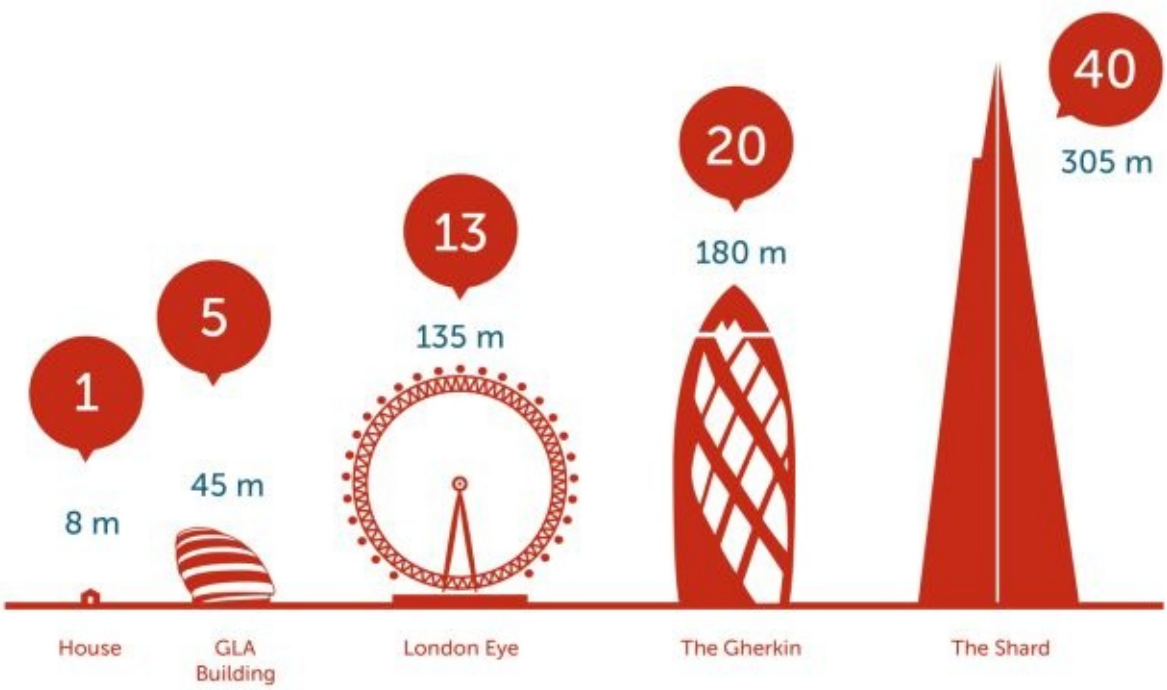

Image from manifesto.co.uk

2. Ideal Days

Ideal days are the number of days that you need to work on a specific PBI. Take note that ideal days are not necessarily consecutive calendar days – it represents the number of days that a person needs to work on a story, and is not the same as the elapsed time.

You can take a football game as an example for this – while four quarters into the game are supposed to last for only 15 minutes each, it may take the game three hours to play. By measuring work done according to ideal days, you have room for possible interruptions or distractions that may happen within the sprint that you are planning to do.

What should you use as a measure in your organization? There is no right or wrong answer to this, and you need to find out the method that works best for you. If there is no reason for people to misunderstand a personnel's idea of ideal days, then you can adopt that measure with no problem. However, if you think that there would be a misunderstanding when you use this term, then you can stick to using story points instead.

Playing the Planning Poker

The Planning Poker is a technique employed to measure PBI sizes. In this method, you are using a technique that relies on the consensus to have an estimate on the effort that needs to be exerted for every PBI. Experts or knowledgeable people are also tapped by the product manager to expose assumptions about the effort and for everyone in the Scrum team to have a more accurate understanding on how much work should really be done.

In this planning activity, the entire Scrum team works together to group items in the backlog that have the similar size. Afterwards, the team will use all the details that they have found out from the Planning Poker to estimate the rest of the items in the product backlog.

These are the cards that are used in Planning Poker:

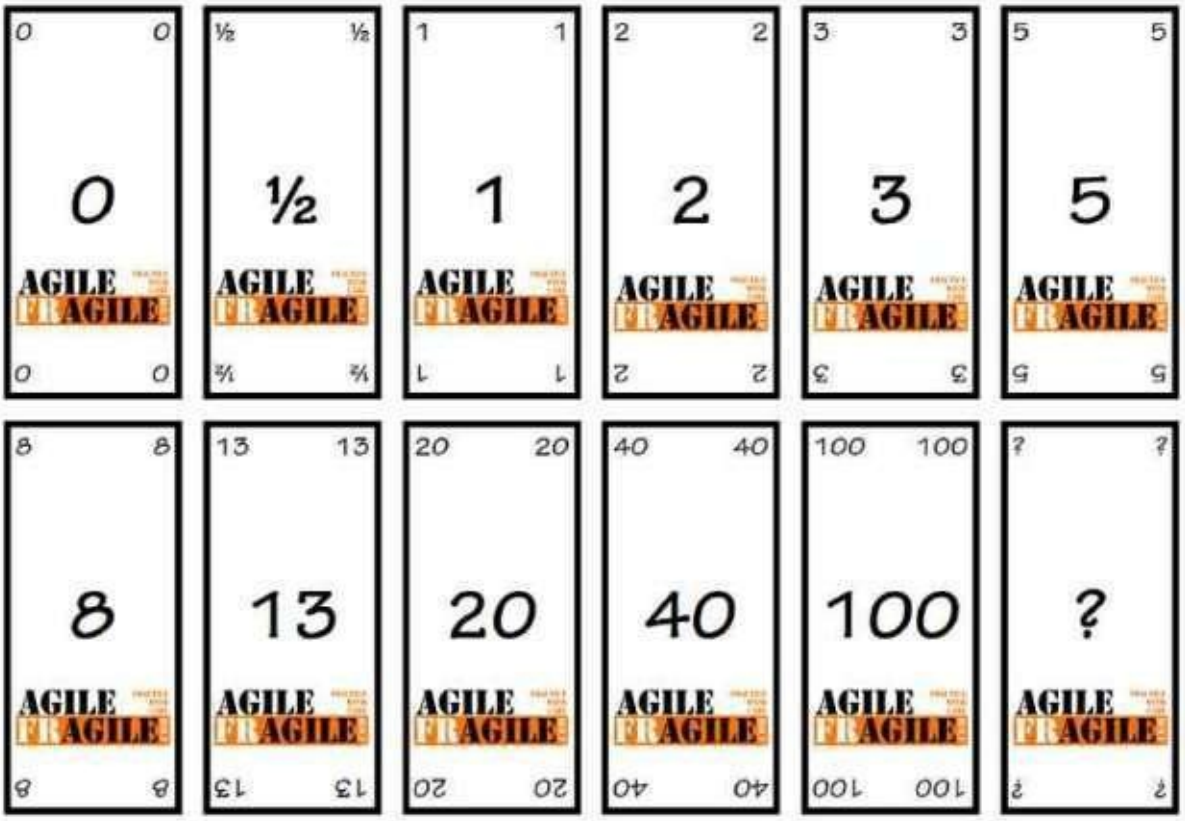

Image from: check.burdenis.net

These numbers are ordered through the Fibonacci sequence, but there are some cards that simply show the relationship of sizes for a particular PBI. When you see numbers on a Planning Poker set, they simply mean the following:

a. 0 – the item is already done or is too small to assign it a number.

b. ½ - the item is tiny

c. 1, 2, 3 – points assigned for small items

d. 5, 8, 13 – points assigned for medium-sized items. A lot of teams consider that 13 points worth of an item would be the biggest size that they would put in a sprint. If an item scores more than 13, they would want to break it into smaller items instead.

e. 20 and 40 – points assigned for large items. Organizations typically call these story items a feature or theme-level stories

f. 100 – points assigned to a very large story which is also referred to as an epic

g. ∞ - indicated that an item is too large that it doesn't correspond to any number

h. ? – shows that the team member does not comprehend what the item is and is asking the product owner to define or clarify it for him.

i. π – this doesn't mean the mathematical pi, but rather somebody wants to take a break, or have a "pie". Some decks use a coffee cup image instead. When someone raises this card, the team needs to take a break.

Here are the rules for the Planning Poker:

1. The product owner reads the PBI to the team

2. The development team discusses the item and asks for any clarifications to the product owner

3. Each member of the team selects a card that represents his estimate. Each member should hide their card from his teammates.

4. Once everyone is done selecting their card, the team members simultaneously show their cards to expose private estimates.

5. If everyone on the team shows the same card, then a consensus is reach. The number on the card would be the PBI estimate.

6. If a different card is shown by a member, the team needs to discuss any assumptions or misunderstanding regarding the PBI. This discussion usually starts by asking why the person who showed a different card made his estimate.

7. When the discussion is done, repeat step 3 until the team makes a consensus.

Defining Velocity

As mentioned earlier, the term velocity refers to the amount of work completed in a sprint. It is normally measured by adding the sizes of the PBIs that are done by the end of a sprint session. The numbers to be added will not include any partially done PBI since they do not get any value out of it anyway.

Because of this definition, velocity is a way to measure the output and not the outcome or the value of what is delivered. It is a way for the team to measure how many sprints should they make in order to complete everything on the backlog and have the entire discussed feature released.

It is also a way to determine the capacity of everyone to commit to work during the next sprint. Because of this, the team's velocity becomes a very useful diagnostic measure that

the Scrum team can use to improve and evaluate the usage of the Scrum framework to deliver what the customer or end user truly needs. By taking a look at the team's velocity over time, the team can see how any process change affects how they can deliver an item that has measurable value to customers.

Calculating for the Velocity Range

Whenever you plan an upcoming sprint, you would benefit from expressing the range by saying "the team can most likely accomplish 30 to 40 points every sprint". This way, you can remain accurate in your estimate and not stick to a precise time on when the PBIs can be accomplished.

To calculate the range, you would need to have two velocities from the team, which is the estimate of the team's faster velocity and their slower velocity. You can also infer that when the team would be required to do more sprints on their slower velocity, and vice versa. If the team has been working using Scrum for a long time, it would be easier to predict its future velocities. However, if there are new members on the team, there may be some discrepancies in this forecast.

One way to tell a team's velocity is to have them perform a sprint planning to tell what PBIs the team can commit to in a single sprint. If the commitments are reasonable enough, then you can add the sizes of these PBI commitments and use the forecasted velocity.

Now, since you also want to know the velocity range, you can have the team plan out two different sprints. You can use one of the estimated velocity number as the high, and have the other one as your low number. You can also make some adjustments once you can already measure the actual velocities and use that as your team's historical velocity.

What Affects the Velocity?

Most people that are starting out with Scrum think that a team's velocity is bound to improve over time. This trend is often based on the reason that if a team constantly inspects and adapts, and in the process assumes more good practices in the sprints, the velocity should go up after a couple or so sprints.

While teams can have that aggression when it comes to improving and focusing on delivering quality features with low technical debt, then it is reasonable to judge that their velocity would improve. However, the trend would not always be in an upward direction – in some cases, the trend might plateau.

It doesn't mean, however, that when a team's velocity plateaus means they no longer have the potential to improve their velocity. There are different ways to improve a team's velocity, such as activating new policies that would lessen distractions, or creating provisions for technology to come in and improve development time. However, introducing changes like this can also mean that the velocity would drop for a time, and then improve once the team becomes used to these changes.

Some organizations make it a point that they improve their development time by making the team take overtime shifts; however, there is an associated risk that comes with this. While it is possible that the team's velocity may improve during the first instances of

overtime, it is also likely that the velocity would drastically drop and the quality would suffer after consecutive overtime shifts.

When you want to determine factors that may increase a team’s velocity, you may want to find methods that would have long-term benefits to the organization.

When is Velocity Misused?

Velocity is good as a planning tool and in diagnosing a team’s metrics. However, it is not great as a performance metric that would gauge any team’s productivity. When velocity is used in an organization as a gauge of productivity, velocity can motivate bad performance behaviors.

For example, if you have made it a point to give a large performance bonus to a team that has the best velocity, you might be thinking that it is only fitting to reward a great work behavior.

However, if you are comparing teams by velocity and not the size of the PBIs that they are taking care of in a sprint, While Team A may be faster than Team B, it would depend on how they are taking the same amount of tasks in the same way. For example, team A may give a PBI a story point of 2, while the other team may feel that it is worth 20.

At the same time, the velocity of the team should be judged according to the definition of done in order to produce a better feature or product, and not merely to meet a high velocity. While teams need to improve on the numbers that they are hitting in a particular duration, it is also important that organizations motivate teams to lower technical debt.

In the end, you should think of velocity on how it should assist development by using accurate planning and how it should help teams to promote development within their groups. Otherwise, it would promote behaviors that would be wasteful to the organization.

Chapter 8: Going Bigger with Scrum

Scrum teams are the probably the best assets of every Scrum-based organization. By structuring them and making sure that they work well with one another, you can guarantee that the Scrum-based project that you are running is going to achieve success. However, what if your organization is planning to release more than one product?

If you are only trying to create one product with a single deadline for release that you are trying to meet, then you would only need to build one cross-functional team. However, if you are going to work using the Scrum framework for a long time, you would want your scrum team to turn into a high-performance group that would deliver high business value.

You are also likely to experience growth in your organization. That would require you to manage multiple Scrum teams, which will intersect with one another in order to generate bigger business value.

Feature vs. Component Teams

When you have a feature team, you have a cross-component and cross-functional team that can create end-user features out of the product backlog and create them. A component team, on the other hand, is largely focused on the development of a subsystem or a component that can be utilized in creation of a single part of an end-user feature.

For that reason, component teams are also called as asset teams. These people are normally bound together by similar skills. Members of this type of team are also likely to report to a single manager and use a centralized resource shared by others. Most organizations favor building these teams in order to put all experts at work and create a part of a product, until every component is done and ready to be assembled in a timely manner.

However, the problem with component teams is that they need to rely on another team to work on a feature. There is no way you can operate a business with the thinking that production should be put on halt because the other component team is not finished with their share of work. Instead of an idle component team being able to solve the problem of delay, the entire business would have to compensate for the time that they need to wait for the other team to finish their backlogs.

Scrum, on the other hand, puts favor to feature teams, while most organizations that are running on traditional framework are likely to favor component teams. Traditional organizations think that when there is a person that is unfamiliar with how a code, for example, is produced, then there is a possibility of an error.

However, when there are two or more teams that are exclusively concerned about the components that they are supposed to deliver, you can probably predict that the teams are more likely to prioritize their own backlogs and increase the risk that the feature that you are trying to build won’t get finished.

The reason is that the more component teams you have, the more points of failure exist in creating that feature. Since Scrum dictates that you only have a single feature team, you only have to be concerned with one possible failure location.

The solution to this problem is to create feature teams that are capable of cross-functioning because they have the skills needed to work on different end-user features and achieve them, without having to supply the pieces of the feature to several component teams.

In time, you would be creating more developed feature teams that are capable to become trustworthy custodians of the feature, instead of sourcing out experts that can only work on a particular component. To produce these feature teams, you need to establish an approach that would allow your organization to transition to creating a multi-feature team that will manage the logistics in getting the feature done.

How to Produce the Feature Team

If your organization is created with several component teams with different priorities, you can create a feature team that has all the skills that you need to get a desired feature done. The component teams that have established themselves to be trustworthy in achieving their tasks can remain to maintain an individual component's integrity. To make sure that component teams are working with each other to meet the needs of a feature, a member of each component team can be assigned as member of the feature team.

Component team members that are also assigned to the feature team will fulfill two roles:

1. Pollinator – the component team member will provide feature team members with the knowledge that they share in his component team. By doing so, he shares ownership of his knowledge and allows shared ownership in the feature team.

2. Harvester – the component team member collects all the changes that the feature team need to make within component areas, and makes sure that his component team knows all the changes that they need to make to meet the requirements of the feature team. When the component team discusses these changes, the team makes sure that they can avoid any conflicts by making sure that their tasks are coherent with the feature team backlog.

If you have a large organization that has about 50 component teams, you may be thinking that you would be putting 50 people in a single feature team, which goes against Scrum's tradition of keeping large teams. If you have multiple teams that intersect with a creation of a single feature, you can produce multiple feature teams instead and divide the number of component teams among them.

This way, you can form feature teams around smaller clusters of component teams and promote better coordination within these clusters. If your organization has small cluster teams and it doesn't make sense that you assign one of its members to a feature team, then you can reduce the number of products that are being produced simultaneously, or hire more people who have expertise in a component area.

In the end, there is no single solution that organizations can use when solving the issue of creating feature teams while maintaining multiple component teams. Large Scrum organizations that has remained successful over time tend to adapt a blended model which are composed of multiple feature teams, and then produce component teams only when needed, which is when the economics of producing a component team for a central

resource is sensible.

Making Multiple Teams Work Together

In Scrum, scalability happens not by making larger development teams, but by creating multiple Scrum teams that are made up with the right number of people. When you have more than one Scrum team, you would need to create a method on how these teams can coordinate. You can use these techniques to do that:

1. Scrum of Scrums (SoS)

In an earlier chapter, you have read that development teams perform a daily scrum during sprint execution. To coordinate multiple teams and make sure that they are committed to having backlogs that work well with each other, these teams can perform a Scrum of Scrums, or SoS.

In this coordination technique, teams may opt to send a member of the development team and their ScrumMaster (which can actually be shared by two or more Scrum teams) to the SoS. However, they need to be sure that the number of members attending the SoS is not too large.

In SoS, the members answer questions like:

a. What has the team done since the last meeting that could affect other teams?

b. What will the team do before the next meeting that can affect the other teams?

c. What problems is the team having that can be resolved with another team's help?

Like the individual team's daily scrum, the SoS can be timeboxed to be done in just about 15 minutes. However, the SoS can extend beyond this time period if the teams need to do collective problem solving before heading to their corresponding sprints.

2. Release Train

A release train is a coordination method that is useful when it comes to aligning planning, interdependence, and visions of feature teams. With this method, a cross-team synchronization happens, which allows teams to have a fast and flexible flow when they are tasked to achieve a large product.

The term "train" refers to the published schedule on when agreed-upon set of features are supposed to leave its "station". All teams that participate in the development of a product should fulfill their backlogs, or place their "cargo" on this train at a specific schedule. Scrum makes it a point that all these cargoes gets into the train at a specific schedule; however, if a team misses the train schedule, there's no reason to panic since another train will arrive at the station anyway.

a) An effective release train follows these rules:

b) Planning should be periodic and frequent, and the release dates for solutions should be fixed. While the schedule and the quality is fixed for potentially shippable increments (PSI), the scope can be variable

c) All teams need to apply the same iteration lengths.

d) Objective, global, and intermediate milestones should be established.

e) PSIs should be available at regular intervals for system-level quality analysis, internal review, and customer preview

f) Use of system-level iterations is available to reduce any technical debt and for teams to have more time for specialty release-level testing and validation.

g) Some infrastructure components should typically track ahead to enable build on top of the same constructs

h) A continuous integration of the system is implemented at all levels

You can divide a really large enterprise backlog into three levels: portfolio backlog (portfolio management with epics), program backlog (program management has features), and team backlogs (product owners has user stories that they can work out in a sprint). Using a Scrum of Scrums, you can coordinate and integrate all the tasks of feature teams that are within the feature area and integrate their jobs at ever sprint.

Whenever it is practical, testing and a full system-wide integration should happen across feature areas. Some teams would want to use their last sprint to send their work on a “train” in order to have some time to harden what they have developed over the previous sprints and integrate and test the results across different feature areas. When teams have already matured, they would lessen their need for a hardening sprint.

Since all sprint durations of different teams participating in the release train are the same, they all start and end on the same schedule. This allows them to synchronize not only within the feature area, but also with other teams that are working on the product.

Chapter 9: How to Cater to Multiple Products

Since most organizations are required to produce more than one product at a time, they need to create choices that are economically sensible when it comes to managing product portfolios. To do that, they need to create governance or management processes that go well with core agile practices to prevent any disconnect from agile approaches that are done at individual product levels. In this chapter, you will see how organizations can strategize portfolio planning and how to determine whether teams can still accommodate additional work.

What is Portfolio Management?

Portfolio management is used to determine which portfolio backlog items should the organization work on. In creating this plan, it is necessary that managers to determine the order of priority PBIs and how long groups should work on them.

When you are doing portfolio planning, you need to consider these aspects to ensure that teams would be able to work on your portfolio's backlog items:

1. Timing

Planning the portfolio is a continuous activity in any organization – as long as you have a product that you need to maintain or develop you have a portfolio that requires managing.

Since planning a portfolio requires dealing with a collection of products and is much larger in scope compared to individual portfolios, managers need to consider new products that they are going to incorporate in the portfolio. However, it does not mean that portfolio planning should go before product planning. By making use of the data collected in product envisioning, the portfolio management makes it possible for you to know whether you should fund the product and how you can prioritize its items in the portfolio backlog.

Portfolio planning does not happen only when there is an envisioned product. It happens at scheduled regular intervals to review products that are already in development, in production, or being sold.

2. Participants

Because portfolio management deals with new and in-process products, the planning participants would include product owners of individual products and internal stakeholders. Adding technical leads and senior articles can be a welcome addition to the planning team.

Stakeholders need to have enough wide business perspective to enable them to make decisions when it comes to prioritizing items in the portfolio backlog and what should happen to the in-process products. Some organizations opt to have stakeholder approval committee or an equivalent group to oversee the portfolio planning process.

Product owners are needed in planning the portfolio to see that their own products are being prioritized well within the portfolio backlog. According to how the portfolio management goes, they would also be able to advocate necessary resources needed to create products that they own.

Process

Two outputs are produced during portfolio planning, which are the portfolio backlog (list of items for future products, according to priority) and the set of active activities (new products that are approved and are decided to be developed immediately, as well as products that are in-process). In order to determine what these outputs should, contain, participants need to do a strategy for these activities:

1. Scheduling

This refers to having a great plan when it comes to determining the sequence of products that should go in the portfolio backlog. With this plan comes the consideration that the organization has limited resources to produce these products in an economical way. While there are a lot of strategies when it comes to planning product sequence in the portfolio backlog, these three strategies may prove to be very effective to your organization.

a. Make Schedules Optimized for Lifecycle Profits

The key to this strategy would be having the right decisions on which variable to measure to see whether optimization efforts are working. When you are creating scheduling strategies, you would want to look at all trade-offs that come with all decisions using a standard unit of measure, which is the lifecycle profit. In this strategy, you would be able to schedule the items in your portfolio backlog in such a way that lifecycle profits are maximized.

Lifecycle profits are the total profit potential of a product in its entire existence. However, in the case of portfolio planning, you are interested in making the entire portfolio optimized for profit, instead of focusing on a single product. Because you are not interested in the success of just a single product, you will want to find the sequence of backlog items that will give you the most profit possible.

Two variables are considered in assessing lifecycle profits: the cost of delay, and the cost of duration. Based on these variables, you can use these scheduling strategies:

a. Cost of delay is same across all products, but products have different sizes – take the shortest job first

b. Cost of delay varies among products, but all products have the same size – take the high delay cost first

c. Both cost of delay and size varies among products – take the weighted shortest job first (calculated by this formula: cost of delay / duration)

2. Calculate for the Cost of Delay

When you sequence items on the portfolio backlog, you need to work on some products over others. However, you need to remember that those products that you are putting off for later will have a delayed start and a delayed delivery date. Those delays come with a cost. When determining the schedule for the portfolio product backlog items, you need to answer this question: What is the cost of delay in lifecycle profits when you delay a product deployment by ___ days/months?

When you look closely at this strategy, you would know that taking the highly profitable item first is not always the right strategy. For example, if you have Project X that has 20% ROI, and a Project Y that has 15% ROI, you may immediately think that you need to take care of Project X first. However, the planning should not stop there.

If you are aware that delaying Project X would mean losing $5,000 and pushing back Project Y for a month will make you lose $70,000, you know that this huge cost of delay discrepancy will eventually impact the lifecycle profitability of your portfolio.

How do you calculate for the cost of delay, anyway? To do that, consider these product attributes and assign a value for the cost of delay (1 being the lowest; 10 the highest):

a. user value – potential value that customer thinks the product has

b. time value – how the customer's perceived value of the product decays over time

c. opportunity enablement or risk reduction – the value in terms of taking advantage of opportunities or mitigating risks

The total product cost of delay would be equal to the sum of these individual delay costs.

If you think that there is no fixed value that you can assign for individual delay costs, then you can create a product delay portfolio and use that in determining scheduling decisions. You can use these profile descriptions:

a. Linear – product will have a cost of delay that will increase at a constant rate

b. Large fixed cost – product will accumulate a one-time cost if not done immediately. This may happen if you get a large portion of payment only after you have delivered the product.

c. Fixed date – product should be delivered by a particular date in the future and will have zero cost of delay until that date is reached and the product is still not fulfilled.

d. Logarithmic – the product's cost of delay is the highest at a very early time. The cost will also have less incremental cost with additional delays.

e. Intangible – product has no obvious cost of delay for a long time, but you will experience a high delay cost during an unexpected time.

3. Have Inflow Strategies

Inflow strategies largely deal with how an organization should balance the rate wherein portfolio backlog items are inserted and the rate items are pulled out. By having this strategy, you will be able to see whether additions to the portfolio backlog are already causing bottlenecks. You will also be able to balance the amount of products coming out by having smaller yet frequent releases.

a. Economic Filter Application

When you have a product vision that goes along with the information that you need in order to produce with confidence, you can make decisions on whether you should fund a development of a particular product. When you do this, you are creating an economic

filter and apply it to see if it meets the organization's funding requirements.

Although every organization out there is required to create an economic filter that goes well with its funding policies, you know that you have a working economic filter of you are able to quickly tell that you should approve an opportunity because it will deliver an overwhelming value compared to its development cost. If you achieve this certainty, then there is no need to talk about whether a project being laid out should be developed or not.

b. Arrival and Departure Rate Balance

Any organization desires to create a steady stream of products that enter and leave the portfolio backlog. However, you don't want to overload the portfolio backlog by having too many products that you need to take care of at the same time.

A lot of businesses conduct strategic planning events annually, which usually takes place during the third quarter of their fiscal year. One of the usual results of this strategic planning is a list of products that they will work on for the next year. These products are usually placed with their existing portfolio, which tends to overwhelm the portfolio-planning process.

It doesn't mean that organizations shouldn't do any strategic planning – organizations should define their strategic direction, but they do not have to identify all the details that they need to take to do that strategy. Deciding what the portfolio of products at one time also violates the principle of using economical sensible batch sizes.

Processing a large group of products right away and determining how they should be sequenced on the portfolio backlog can be expensive and possibly wasteful since a large number of products can possibly complicate the scheduling. Finding out what the sequence of what should be placed in your portfolio backlog is a lot simpler if you have fewer items to manage.

To prevent overwhelming the portfolio backlog, you can opt to introduce new products to your portfolio at frequent intervals instead. When you have more instances of introducing new products, you effectively reduce the cost and effort you need to spend to review and insert products in the portfolio, making the planning stable and predictable.

Now, when the size of the product backlog begins to grow, you can start throttling the product flow. To do this, you can tweak the economic filter to improve product approval standards in such a way that only high quality products are allowed to pass through. This will reduce the tendency of new and random products from coming in and establish better balance with the product departure rate.

c. Embrace Opportunities Right Away

When planning portfolios, you need to embrace emergent opportunities, or opportunities that are previously unknown to your organization or unlikely to occur in your organization; hence, something not worth spending for today.

For example, if your organization thrives in an online betting marketplace, you know that your business operates on an environment that is highly regulated by jurisdictions and regulations. You may be aware that regulations when it comes to casinos can be

unpredictable, depending on government policies. Now, if these regulations change, you need to embrace any opportunities that might suddenly emerge.

If you see that there is an unexpected change in your organization's environment and it would likely affect profit or the amount of effort that a team needs to spend for items in their backlogs, then you need to make the appropriate insertions and releases in the portfolio backlog right away. Doing so would not only allow you to create better working plans, but also stay ahead of the competition who might not take action regarding opportunities right away.

Now, if you have a frequent schedule for evaluating these opportunities right away, such as a monthly meeting, and you have enough economic filters in your organization to work on them right away, then you do not have to spend a lot of time when it comes to considering emergent opportunities.

d. Go for Smaller, More Frequent Releases

As discussed earlier, there are a lot of economic benefits that any organization can enjoy whenever they opt for smaller and frequent releases. One important benefit is that they can increase profit that they can get from lifecycles they divide a product into a series of smaller and incremental releases. It also helps prevent the portfolio from experiencing a convoy effect.

What is a convoy effect? This phenomenon is similar to the experience of having to drive behind a large, slow moving vehicle. If you are trapped behind one, then there is a chance that other drivers of smaller vehicles will get the same experience when they all pile up behind you.

When you allow large products into your portfolio backlog, you are likely to create a long queue of smaller products behind it. Since all products are going to be delayed because of the large product, the smaller ones would begin to accrue considerable costs of delay over time, which will make profit suffer.

Creating Outflow Strategies

When you want to create better management that will help you decide when you should pull out products from your portfolio backlog, you can use the following strategies:

1. Pay attention to idle work

Traditional organizations tend to release products into production by taking these steps:

a. Select a product from the backlog and assign people to deal with it.

b. If everyone is not working at 100 percent capacity, repeat the previous step.

This approach is only designed to make everyone busy, but it does not mean that people will work faster and error-free on a product. Instead of sticking to this old and risky method, you can instead aim to work on a product when you are sure that this item on the portfolio backlog can truly ensure that it will not cause any disruption in the flow of work that has already begun.

2. Limit the WIP

You have read in an earlier chapter that it is wise to only get work from the product backlog when you are sure that the development team has the capacity to work on it. The same rule applies in the portfolio backlog – you should only start pulling out from this list if you have a feature team that can accommodate it.

Knowing how many Scrum teams are available and learning what types of products they can handle will help you get the information you need to know about how much products can feature teams accommodate on their individual backlogs without requiring them to spread too thinly to meet release deadlines.

3. Get a Complete Team

Having a couple of people freed up from their tasks does not mean that you need to give them tasks already. If you want to pull out a product from the portfolio backlog, then see to it that you have an entire team ready to commit to work on a new set of tasks.

Even if you have two or three developers that are ready to deal with the next product, they are not the only ones who are going to work on it. Making them start making progress that requires an entire team to work on only increases the risk of miscommunication and planning revision, which will only slow down work.

Managing In-Process Items

Having the right strategy that caters to products that are already in process will give you guidance on whether you still need to deliver, preserve, or revise a product that teams are working on or just terminate it. Making these decisions should be made regularly (every end of sprint) or occasionally during off-cycle times, especially when you are getting results that you do not expect from products that are being worked on.

Using marginal economics is one strategy that you can use to guide you in decision-making and remain aligned with core Scrum principles. Using marginal economics, you will be making decisions about in-process items according to the return of investment that it will generate. Since you are thinking about the profit, you will need to make a decision on whether you can afford to spend additional resources to continue developing these products.

In-process products can be dealt with using these options:

1. Preserve – continue product development. You want to take this option when the product justifies the resources it has taken from you and you know that continuing development will give you better ROI.

2. Deliver – cease development and deliver the product. You can take this option if the product has the minimum releasable features that bring value to the end-users and great ROI on your end, without needing additional investment.

3. Pivot – change directions according to new data you have learned. This is an option that is only viable when your investment is not justified and the product has no releasable features yet, but you have another path that you can try to ensure the success of the

product.

4. Terminate – cease development and kill the product. This is the option that you have if the resources spent are not justifiable, the product has no foreseeable value to you and the end-user, and you do not have any other reasonable idea to change its process path to make it work.

However, there are a lot of subjective and foolish behaviors that may prevent you from making the right decisions for products that you have already spent time and effort for. For example, there are just too many people who will not dare terminate a project that has already spent too many resources but does not create a clear picture on whether it will work or not. Some will not even think of terminating a product just because they already spent their first dollar on it.

Accounting systems have a lot to do with these risky assumptions. For example, if a company has started developing a software that they initially thought would have a 100% value to their customers and will cost about $500,000 to develop. After spending half a million, they found out that the product will only provide 15% value to their consumers and would actually cost $5,000,000 to develop.

Despite that overwhelming difference, some would still pursue the development even with the knowledge that the value of the product that they are producing is a lot lower than they have expected and the cost is not justifiable at all. To some accounting systems, they are willing to protect expense budgets of their departments and then wait for their expenditures to be capitalized. Of course, this move is not sensible at all.

When you stick to using marginal economics, you would be able to expose every wasteful behavior that you have and use your common sense on what will truly bring you and your customers' value. Instead of wasting your time and money on something that will not give anyone a good return for their investments, you can have the right strategy on whether you should spend more time on development or save your resources for something better.

Chapter 10: What Products Should You Produce?

Before you even start a sprint and start developing anything, you know that you need to have a product backlog. However, to generate a product backlog, you need to have a product vision. You will never know what you will want to accomplish if you do not do any envisioning about the very thing that you want to create.

Envisioning's Goal

When you envision a product using Scrum, you are not doing a ceremonial and very intense chartering of a project. After all, you do not believe that you can possibly know everything that you need to do right away. You will have to have sufficient confidence to subject the idea of portfolio planning and make decision if you can decide whether to fund the next level of development to add more details to the product.

At this point, you know that you understand that funding cannot move forward without first having a vision of the product and if you do not have enough details that will lead you to understand high-quality solutions, features, customers, and cost.

You also do not want to spend too much time and effort on envisioning because you want to get past this stage where there is too much guesswork and thinking that you are already aware of the cost and customer needs. You would want to move immediately towards sprints where you can immediately get feedback and know whether your solutions are working or not.

After all, you can only start implementing real solutions when you have validated learning based on environments in which you want your product to exist.

Envisioning's Timing

Envisioning is an ongoing activity – you do not only envision your product at the beginning of making plans. However, it will begin with the process of ideation, or that moment wherein someone or some people has generated an idea on how a product can be created or improve.

This idea is passed through your organization's strategic filter to see if it is consistent with the organization's existing strategies. If the idea goes with your organization's strategies, then that is the time that people will want to start investigating on why it is worth the investment.

Once that seed of idea has successfully passed the strategic filter, then that is the time to start initial envisioning or the process where you get just enough understanding on how the desired future product's initial release should be. When you do that, you can immediately have a vision of how you can deliver quality value to your customers without spending too much.

That also gives everyone a tangible idea on how this product will appeal to end users and customers and give you enough details to allow you to give feedback or refute assumptions that you have about customers and your overall solution. When this happens, you can see if a project that you want to develop will allow you to persevere with your

current vision, or of you should pivot away from your original solution and modify plans.

Who Should Participate?

The product owner is the only person that is required to attend initial envisioning. The product also normally oversees the initial envisioning together with a stakeholder who serves as a collaborator in performing this process. Specialists such as market researchers, user-design experts, and architects can also participate in different envisioning tasks.

The ScrumMaster and the rest of the development team are also ideal participants in initial envisioning. This is because these people will provide the feedback that stakeholders, experts, and the product owner needs to have a more empirical data on how a product would proceed into development, without having the need to hand off the vision that is going to be created to another team.

However, what happens often is that the organization waits for the initial envisioning being finished before it can move to funding the Scrum team, making it hard to include the Scrum team to be part of this process. However, when the product development is already underway, the entire Scrum team should be participants in any product re-envisioning.

Envisioning Process

The main input needed to start initial envisioning is an approved idea which has cleared your organization's strategic filter. When your product needs to be re-envisioned, you need to have a pivoted idea, or an idea about the product that has been revised or updated based on changes that influence how the product would do once it goes through continued development or releases. These influences can be customer feedback, funding changes, competitor strategies, and a whole lot more.

When envisioning or re-envisioning products, you need to have other inputs, such as

1. indication of planning horizon, or a time frame to consider for envisioning

2. deadline for completion of envisioned products, if applicable

3. availability of resources needed for envisioning

4. confidence threshold or how to determine whether an envisioned product can be done right, which gives stakeholders and other decision makers enough trust that the product should push through.

Envisioning is also composed of different kinds of activities that are made to produce an important output, such as the exact product vision or how the initial product backlog should be like. It is also ideal to create a simple product roadmap that will show a set of near-term releases that can be produced in increments.

Here is an illustration of how the initial envisioning should be performed, using a fictional product idea called One Big Onion:

The company, which is called Nameless Media, is known as a leader in producing quality documentary and PR marketing materials, with its core business being a provider of AVPs and similar services to its clients. The revenues of Nameless Media are moving at a

modest pace for a couple of years, but it remains profitable.

However, its main competitors have launched features in a surprising frequency, which may make Nameless Media's clientele move to them instead. Now, what Nameless Media needs is to have a brand new innovative service offer that will allow them to strategically leapfrog the steep competition.

The marketing team of Nameless Media thought of One Big Onion, which is a way to use both music and graphics revolutionarily in materials that clients will receive. The marketing team believes that this idea could be an innovative service to their clientele, which is fond of using rehash photos and remixed music in their ads.

The marketing team then sent their idea, together with a one-page description of One Big Onion that states its high-level target features, targeted niche, and advantages, to the New Product Approval Committee. This committee reviews the new product in a regularly schedules Idea Review Meeting, which is done every third Thursday of the month.

The senior management agrees that the One Big Onion represents a great opportunity to make the company stand out in the marketplace. After making this decision, the committee assigns Lauren, a business representative from the company's strategic marketing, as the product owner of One Big Onion.

The management has also allowed two weeks to complete the product envisioning, after which the approval committee members will review the results and decide if the project's initial development should be funded or not. In addition to Lauren, the management has also assigned two subject matter experts to do the filtering, a group of stakeholders, and also a market researcher to join the envisioning. However, the management did not authorize the larger expenditure of the full Scrum team during the envisioning.

Lauren is asked to use the resources given to her to produce these items:

1. Initial product backlog, product roadmap, and product vision
2. Evidence for the initial assumption that users want One Big Onion's ability to offer premium graphics and sound design as added services
3. Description of other critical assumptions about potential users of the product and the feature set that should be tested on the first product release
4. A few actionable and important measures that will be used to test assumptions and to see whether the initial release of One Big Onion is meeting expectations or not
5. A list of unknown items that should be answered

The management needs this information to assume confidence in making an informed decision on whether the company should allow One Big Onion to go through initial development.

Visioning

Taking the example scenario for One Big Onion, the first thing that Lauren and the stakeholders need to put together is a compelling and shared vision for the product that they are working on.

However, they do not have to create a 100-page document for their vision – since they are using Scrum, they are aware that they do not need to create a complex document to solidify a vision. Visions, even for the most complex products, should be stated simply and presented with a coherent direction to the people who will be tasked to realize them.

Visions are commonly expressed in terms that tell how stakeholders achieve value. You can use this format to write your vision:

Format	Description
Elevator statement	Have a short pitch (about 30 seconds to one minute) of the product vision.
Product datasheet	Try to come up with the product datasheet on the first day on the front of a one-page marketing piece
User conference slides	Create 2-3 presentation slides that you can use to introduce your product in a public presentation.
Press release	Write a press release that you want to be released once your product is available. It should contain newsworthy content and as long as a page or less
Magazine review	Draft a fictional magazine review bylined by the solution reviewer

When you look at this format, you would notice that everything that the management needs to know is already in there: what the product is about, what features it would contain, how you think the customers would see it, how it is going to be launched to the public, and a projection of customer feedback.

Making a High-Level Backlog

Once you already have a vision, you are ready to create product backlog items. Using the terminology of user stories, and since you are working on a high level, you would want to create epics or extremely large user stories that go well with the product level planning. These epics align with the vision and provide the next level of product detail for both the Scrum team and the senior management.

The people who will create these stories will be the same ones who made the vision – the product owner and the stakeholders, and if possible, with the development team and the ScrumMaster who will work on the product. However, if the project development has not yet been funded or approved, like in the case of One Big Onion, there is a chance that the Scrum team may not be available in making the backlog for the envisioning.

When that happens, the product owner may want to make it an option to ask for help with technical personalities available that may have an interest in the product development to help him out.

Defining the Product Roadmap

Once you are done creating the initial vision and the backlog, you are ready to define how you can do series of releases that will make you achieve some, or possibly all, of your product vision. A product roadmap will serve as an overview of all the incremental deployments that you will perform. Of course, it also follows that if you are doing a small product that can be delivered in a single release, then there is no need to do a product roadmap.

Keep in mind though that even if you intend to make frequent releases, you do not need to be strict about deadlines. What you should focus on instead is the minimum releasable features (MRFs), which is a small set of features that is deemed to be must-haves in order to meet quality expectations and deliver immediate value to stakeholders and end users.

You may also call MRFs as your minimum viable product (MVP) or minimum marketable features (MMFs). When you plan your releases this way, customers and clients will be able to perceive that there is value to the product that you want to develop. At the same time, you can also couple this plan by releasing other features in a fixed and periodic manner; say every quarter, to set expectations on when end users can expect to receive the complete product.

Other Activities in Envisioning

Envisioning can also include any other tasks that may be deemed relevant in meeting the confidence threshold that needs to be targeted. You may want to do additional market research or even do a competitive analysis to see how your product would do on the market when pitted against products of competitors. You may also want to develop a rough business model of the product to see if it will pass your organization’s economic filter and convince stakeholders to proceed to funding your Scrum team.

Making Envisioning Economically Sensible

Envisioning is an investment – you need to do it to get all the information that the management requires to make the decision that your project is worth funding. If you do too little work in envisioning, you may find yourself clueless on what you should do when you do the first sprint and fail to meet customer-quality value that you are expected to achieve.

If you do too much work on envisioning, you may unintentionally create too much product artifacts and be prompted to revise or discard them when you proceed to validated learning, which you would do when you are ready to trim down your high-level product backlog.

While you may think that you need a lot of data in preparing your envisioned product, keep in mind that you are still using Scrum – you only need to have upfront predictive planning that will allow everyone involved to understand what you are trying to create and the risk that everyone may be taking once the project is underway.

That means that you are only aiming to obtain reasonable information that you can achieve within the budget and time that you have been given. Moreover, you have to remember that in Scrum, the product can and will change once you are already in the process of sprinting and once you have received the necessary feedbacks required to achieve quality.

Aim for a Realistic Confidence Threshold

The confidence threshold is the minimum level of information that decision makers need to receive from you in order for them to be confident enough to release their decision on whether they should fund or not fund the product that you want to propose.

You can think of it as the bar that you need to pass so that you can exit envisioning and enter the portfolio planning. As you can remember, portfolio planning is the part wherein you need to apply economic filters to a product to see if it fits your organization's funding criteria. If it does, then you can proceed to validating assumptions and proceed building the product.

The height of this bar has legitimate economic consequences. If you set the bar high, then you need more effort to clear it – you need more time for envisioning, which will delay the product shipping and entail costs to your organization especially if you are trying to envision a product that is truly needed.

At the same time, you need to think that envisioning is already being paid for, and the longer it takes, the more costly it becomes, since predictive tasks like this adds up to the entire organization's WIP. Additionally, setting the bar higher does not mean that you become more certain about the product that would be eventually released, or inform the management better that approval is their best decision to make. Because of these reasons, you need to set the bar to just the helpful level.

You need to do the following to set the bar to a helpful level:

- Look at a short horizon

  Don’t try to envision too much about the product at a single moment. Focus on what features you need to produce on the first candidate release to prevent wasting your time thinking about features that may never be released in the future. Since most of your assumptions are not yet validated and you cannot subject the product yet for customer validation, better think that you can adapt knowledge that you will find out later on to improve the product that you have in mind.

- Act fast

  Envisioning should be fast and efficient, just like everything else in Scrum. The sooner you get done with this phase, the sooner you get to build something tangible that will help you find out if your assumptions are true or not. By acting quickly, you create a sense of urgency for everyone to reach a product decision.

  You can create this urgency by setting a realistic time for your team to finish envisioning, which tells everyone that you are determined to cut the costs of planning and move on to validation instead.

- Spend for validated learning

  When you are doing predictive tasks, such as trying to create hypothesis on how customers will probably like your product or how much revenue your product could make, you are trying to tell the management what they are comfortable with in order to perceive your data as the baseline plan. However, you are aware that you may just be wasting time trying to present data that can be just a wild guess out of your assumptions, which can be found invalid when you get to portfolio planning and product development.

  If there is something that you need to spend for during envisioning, better spend for validated learning and get as much certain information as you can. Work on the core value that you are trying to present to your organization – if you think that users

prefer a feature that you are trying to offer in your product, then conduct a research on that hypothesis as early as possible to make the right adjustments to your roadmap.

- Use provisional or incremental funding

  Always see to it that you are using your funding in increments. When you use incremental funding, you would just be allotting resources to the initial small part of development and then go back to your budget when you learn that what you have spent for is validated. This will allow you to limit your spending on envisioning and also lessen the time to complete it.

  You only need a portion of the money that you are going to get for validated learning, which prevents you from falling into the temptation of spending time and budget for all possible assumptions all at once. It also follows that when you have validation for the minimum assumptions that you need to meet to hit confidence threshold, then there is no point to spend for the others.

- Fail fast

  You do not want to start a product that you're not sure of its success rate – while you cannot know everything that is going to happen with your project, you would want to get as much certainty as possible during the envisioning process.

  Fail fast is one of the things that you can do to prevent too much spending in the future for things that would probably not do. When you try to learn as quickly as possible and make revisions early into envisioning, you can produce a more appropriate product, or decide to kill the idea and do something that would most probably work out better. You do not want to spend resources for something just for the sake of approval, to be just killed off in portfolio planning later on.

Once you are able to do efficient product envisioning, you would be able to think of products that work best, without merely relying on guesswork. The more you use an efficient envisioning process, the more products that will give you quality ROI you can produce.

That means that when your ideas get approved and you are ready to conduct sprints, you will have the confidence that the idea that you wanted to develop is realistic and relevant enough to your organization's growth, and that it would return substantial benefits to your organization.

At this point, you have all the skills that you need to learn in order to produce anything fast and efficient. Once you are done with envisioning your product, you are ready to start building with confidence!

Conclusion

Thank you for reading this book!

I hope that this book has helped you learn how you can use Scrum when you tackle activity, whether with a small group of people or involving an entire organization. I also hope that this book has helped you make planning and building almost anything more efficiently, even when you are starting with zero knowledge about what you are supposed to do.

Learning how to do tasks using Scrum (or any other agile framework out there) effectively takes time, since a lot of people that you encounter are probably still doing the traditional method of planning and building products.

The next step that you can do is to inform people that you are working with how they can be more efficient and resourceful by adhering to Scrum principles. Scrum also requires a lot of practice to get it right, but once you commit to its principles habitually, you will be able to sprint your way to achieving goals.