INTRODUCTION TO
INTERNATIONAL AGILE PRODUCT OWNER FOUNDATION

Table of Contents

01 INTRODUCTION TO "INTERNATIONAL AGILE PRODUCT OWNER FOUNDATION"

This course is the foundation course to become a professional Product Owner. Before you take this course, you have passed the International Scrum Master Foundation, which has taken you through all the fundamental knowledge of the Scrum Method.

The course gives you a deeper understanding of the Task a Product Owner does. It also gives you techniques and methods for becoming a successful Product Owner.

Who should take this course: As this is a foundation course, everyone who is interested in Scrum should take it.

- **Project leaders**: who are interested in the Product or Scrum Master role, will benefit from this course which gives you access to becoming a certified Advanced Product Owner.

- **Scrum Team member**s: will also benefit from this course, as it gives you understanding of the job the Product Owner has to perform, and what expectation you as a Scrum Team member should have towards your Product Owner.

The Product Owner represents the interests of the Stakeholder community towards the Scrum Team. The Product Owner is responsible for ensuring clear communication of product or service functionality requirements to the Scrum Team, defining Acceptance Criteria, and ensuring those criteria are met. In other words, the Product Owner is responsible for ensuring that the Scrum Team delivers value.

The Product Owner must always maintain a dual view. He or she must understand and support the needs and interests of all Stakeholders, while also understanding the needs and workings of the Scrum Team. Because the Product Owner must understand the needs and priorities of the Stakeholders, including Customers and users, this role is commonly referred to as the Voice of the Customer.

The Product Owner's responsibilities in the various Scrum processes and this course will take you through this in more details:

Process	Product Owner Responsibilities
Create Product Vision	• Defines the Product Vision
Identify People Requirements	• Helps organize Scrum Teams for the project • Identifies Stakeholder(s) and Scrum Master(s)
Form Scrum Team	• Helps select Scrum Team members • Helps develop a Collaboration Plan • Helps develop the Team Building Plan with Scrum Master(s)
Develop Epic(s)	• Creates Epic(s) and Personas
Create Prioritized Product Backlog	• Prioritizes Prioritized Product Backlog Items • Defines Done Criteria
Conduct Release Planning	• Creates Release Planning Schedule • Helps determine Length of Sprint
Create User Stories	• Helps create User Stories • Defines Acceptance Criteria for every User Story
Approve, Estimate and Commit User Stories	• Approves User Stories • Facilitates Scrum Team and commits User Stories
Create Tasks	• Explains User Stories to the Scrum Team while creating the Task List
Estimate Tasks	• Provides guidance and clarification to the Scrum Team in estimating effort for Tasks

Create Sprint Backlog	• Clarifies requirements to the Scrum Team while creating the Sprint Backlog
Create Deliverables	• Clarifies business requirements to the Scrum Team
Groom Prioritized Product Backlog	• Grooms the Prioritized Product Backlog
Demonstrate and Validate Sprint	• Accepts/Rejects Deliverables • Provides necessary feedback to Scrum Master and Scrum Teams • Updates Release Plan and Prioritized Product Backlog
Ship Deliverables	• Helps deploy Product Releases and coordinates this with the Customer
Retrospect Project	• Participates in Retrospect Sprint Meetings

Other responsibilities of a Product Owner are:

- Determining the project's initial overall requirements and kicking off project activities, this may involve interaction with the Program Product Owner and the Portfolio Product Owner to ensure that the project aligns with direction provided by senior management.

- Representing user(s) of the product or service with a thorough understanding of the user community.

- Securing the initial and ongoing financial resources for the project.

- Focusing on value creation and overall Return on Investment (ROI).

- Assessing the viability and ensuring the delivery of the product or service.

02 CREATE PRODUCT VISION

In this graph below, we see different levels of detail according to how far away a feature is from being implemented. In most agile methods, we like to deliver a release—at least an internal release—in less than three months. Therefore, when you are more than three months away from delivery, all you really need is an overall vision for the release. What market is the software going to serve, what kind of value is it going to provide, what are the basic things the software is going to do? You may have a product vision document or a mission statement.

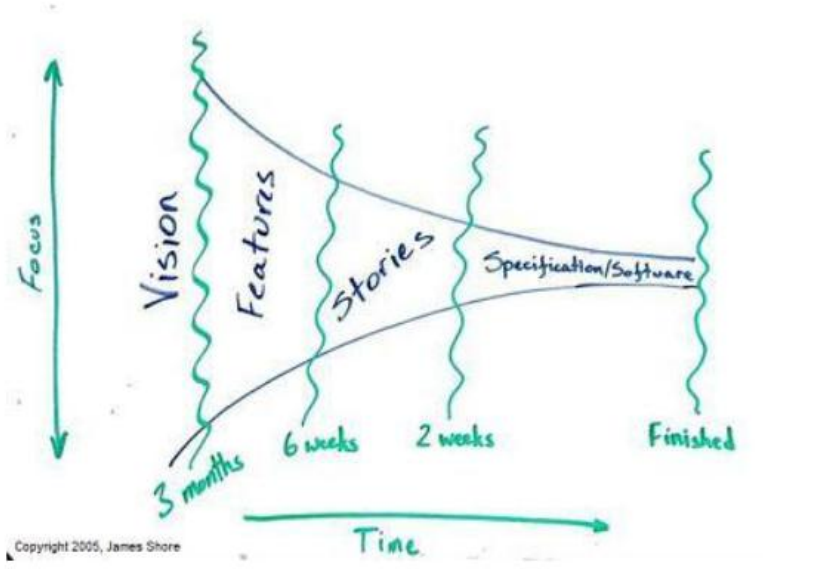


Figure: Focus on more detail over time. Focus: Vision, Features, Stories, Specification and Software; Time: 3 months, 6 weeks, 2 weeks, Finished. Copyright James Shore

This is an idealized view of the world. In practice, we do not proceed smoothly from vision, to features, to stories, to specification, on the clear, crisp timeframe described here. Reality is messier and it can be hard to know when to go into more detail. However, the overall point is valid: you do not need all of the details in advance.

The first stage in an Agile Project is defining your product vision. The *product Vision Statement* is an elevator pitch—a quick summary—to communicate how your product supports the company's or organization's strategies. The Vision Statement must articulate the goals for the product.

The Product Owner is responsible for knowing about the product, its goals, and its requirements throughout the project and takes responsibility for creating the Vision Statement, although other people may have input. The Vision Statement becomes a guiding light, the *"what we are trying to achieve"* statement that the Development Team, Scrum Master, and Stakeholders refer to throughout the project.

Anyone involved with the project, from the Development Team to the CEO, should be able to understand the product Vision Statement.

2.1 Four Steps to create a Product Vision

Four Steps to create a Product Vision:

1. Developing the agile product objective
2. Creating a draft Agile Vision Statement

3. Validating and revising the Agile Vision Statement
4. Finalizing your Agile Vision Statement

2.1.1 Developing the agile product objective

To write your Vision Statement, you must understand and be able to communicate the product's objective. You need to identify:

- **Key product goals:** How will the product benefit the company creating it? The goals may include benefits for a specific department within your company as well as the company as a whole. What specific company strategies does the product support?
- **Customer:** Who will use the product? This may be more than one entity.
- **Need:** Why does the Customer need the product? What features are critical to the Customer?
- **Competition:** How does the product compare with similar products?
- **Primary differentiation:** What makes this product different from the status quo, or the competition, or both?

2.1.2 Creating a draft Agile Vision Statement

After you have a good grasp of the product's objective, create a first draft of your Vision Statement. In creating your Vision Statement, you help convey your product's quality, maintenance needs, and longevity.

One way to make your product Vision Statement more compelling is to write it in the present tense, as if the product already exists. Using present tense helps readers imagine the product in use.

A Vision Statement identifies a future state for the product when the product reaches completion. The vision focuses on the conditions that should exist when the product is complete.

Avoid generalizations in your Vision Statement such as *"make Customers satisfied"* or *"sell more oil"*. Also, watch out for technological specificity, such as *"using HTML5, create a interface with four external calls that..."*.

At this early stage, defining specific technologies may limit you later. A few extracts from unsuitable Vision Statements that should ring warning bells:

- *"Secure additional Customers for the SuperApp application"* (!)
- *"Satisfy our Clients by September"* (!)
- *"Remove all issues and improve quality"* (!)
- *"Create a new application in HTML5"* (!)
- *"Beat Facebook to market by six months"* (!).

2.1.3 Validating and revising the Agile Vision Statement

After you draft your Vision Statement, review it against a quality checklist:

- Is this Vision Statement clear, focused, and written for an internal audience?
- Does the statement provide a compelling description of how the product meets Customer needs?
- Does the vision describe the best possible outcome?
- Is the business objective specific enough that the goal is achievable?
- Does the statement deliver value consistent with corporate strategies and goals?
- Is the project Vision Statement compelling?

These yes-or-no questions help you determine whether your Vision Statement is thorough. If any answers are no, revise the Vision Statement.

When all answers are yes, move on to reviewing the statement with others, including:

- **Project Stakeholders:** The Stakeholders will be able to identify whether the Vision Statement includes everything the product should accomplish.

- **Your Development Team:** Because the Team will create the product, it must understand what the product needs to accomplish.

- **Scrum Master:** A strong understanding of the product helps the Scrum Master remove roadblocks and ensure that the Development Team is on the right path later in the project.

- **Agile Mentor:** Share the Vision Statement with your Agile Mentor, if you have one. The Agile Mentor is independent of the organization and can provide an external perspective, qualities that can make for a great objective voice.

Discover whether others think the Vision Statement is clear and delivers the message you want to convey. Review and revise the Vision Statement until the Project Stakeholders, the Development Team, and the Scrum Master fully understand the statement.

At this stage of your project, you may not have a Development Team or Scrum Master. After you form a Scrum Team, be sure to review the Vision Statement with it.

2.1.4 Finalizing your Agile Vision Statement

Make sure your Development Team, Scrum Master, and Stakeholders have the final copy of the Vision Statement. You can even put a copy on the wall in the Scrum Team's work area, where everyone can see it every day. You refer to the Vision Statement throughout the life of the project.

If your project is more than a year long, you may want to revisit the Vision Statement to make sure the product reflects the marketplace and supports any changes in the company's needs.

03 IDENTIFY PEOPLE REQUIREMENTS

Identifying People Requirements is one of the initial steps in selecting the Scrum Master and the Stakeholder(s). It is important to document the roles and responsibilities of all those who would be involved in completing the Tasks in the project. This includes all individuals involved in the project in any capacity, regardless of whether their role is core or non-core.

3.1 The four main actors in Scrum

In Scrum there are four main actors

- The Stakeholders
- The Product Owner
- The Scrum Master
- The Team (see Chapter 4).

Usually, the Product Owner or the Scrum Master work with the Human Resource Department of the company to determine and finalize the People Requirements for a project.

Prior to selecting the Scrum Master and Stakeholder(s), their availability must be confirmed. Only Team members who will be available and can fully commit to the project should be selected. People Availability and Commitment are commonly depicted in the form of calendars showing when human resources will be available to work throughout the duration of the project.

To be effective, Scrum Teams should ideally have six to ten members; and replacing persons or changing Team members is not advisable in Scrum Core Teams. Therefore, it is important to have persons in the Scrum Core Team who are available and fully committed to the project.

3.1.1 Stakeholder(s)

Program Stakeholder(s) is a collective term that includes Customers, users, and sponsors for a program. They influence all the projects in the program throughout the project's development. Program Stakeholder(s) can also help define the project vision and provide guidance regarding business value.

Program Stakeholder(s) interface with Portfolio Stakeholder(s) to ensure alignment of the program with the goals and objectives of the portfolio. They are also involved with appointing Stakeholder(s) for individual projects and ensuring that the vision, objectives, outcomes, and releases of individual projects in the program align with that of the program.

3.1.2 Choosing the right Product Owner

Finding the right person to fill the Product Owner role is challenging. How do we as a company choose the right person to be Product Owner?

In general, there are two ways to organize Scrum Teams:

- Feature Teams
- Component Teams

3.1.2.1 Feature Teams

A Feature Team implements a cohesive set of requirements, such as one or more themes of features. The result is an executable vertical slice that cuts across major parts of the software architecture. Feature Teams are organized around the Product Backlog.

3.1.2.2 Component Teams

A Component Team creates a component or subsystem. Component Teams are organized around the software architecture.

Both Team setups have advantages and disadvantages. Component Teams ensure architectural integrity and reuse. Unfortunately, they often cannot utilize Product Backlog items expressed as User Stories or use cases but require detailed technical requirements. Feature Teams, on the other hand, can normally work in parallel. They encounter fewer integration issues and can utilize the requirements stated in the Product Backlog. Ensuring architectural integrity and reuse can be a challenge. As a rule of thumb, companies should employ Feature Teams whenever possible and use Component Teams only if absolutely necessary.

3.1.2.3 Common mistakes when choosing Product Owner

Finding the right Product Owner can be difficult, here a list of common mistakes:

- The underpowered Product Owner
- The overworked Product Owner
- The partial Product Owner
- The distant Product Owner
- The proxy Product Owner
- The Product Owner Committee Product Owner

In the advanced course we will go into detail regarding these mistakes.

3.1.3 Scrum Master(s)

The Program Scrum Master is a facilitator who ensures that all Project Teams in the program are provided with an environment conducive to completing their projects successfully.

The Program Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved in the program.

- Provides guidance to Scrum Masters of individual projects
- Clears away impediments for the different Project Teams
- Coordinates with the Scrum Guidance Body to define objectives related to quality, government regulations, security, and other key organizational parameters
- Ensures that Scrum processes are being effectively followed throughout the program.

The Program Scrum Master interfaces with the Portfolio Scrum Master to ensure alignment of the program with the goals and objectives of the portfolio. He or she is also involved with appointing Scrum Masters for individual projects and ensuring that the vision, objectives, outcomes, and releases of individual projects in the program align with those of the program.

Large projects require multiple Scrum Teams to work in parallel. Information gathered from one Team may need to be appropriately communicated to other Teams—the Chief Scrum Master is responsible for this activity.

Coordination across various Scrum Teams working on a project is typically done through the Scrum of Scrums (SoS) Meeting. This is analogous to the Daily Standup Meeting and is facilitated by the Chief Scrum Master. The Chief Scrum Master is typically responsible for addressing impediments that affect more than one Scrum Team.

To learn about the Scrum Team, turn to Chapter 4.

04 FORM THE TEAM

Formation of the Scrum Team is a very important Task. The Scrum Team is at the heart of the process since these are the people who will realize the Product Owner's vision.

This vision is nothing except a list of features that the Product Owner wants, with no unifying principle or clear goal. This does nothing to guide the Team (or inspire the Team). It also rarely passes the elevator test and usually is difficult to remember. What it adds up to is that it does nothing to align the Team or help them make decisions, and as a result, the "secret sauce" in Scrum (self-organization) never emerges or emerges misdirected.

4.1 Skills

With "self-organization" comes responsibility and commitment. Therefore, the Product Owner have to look at more than just technical skills when selecting his or her Team. Some of the skills that normally are also needed are:

- Self-organization skills
- Communication and collaboration skills.

4.1.1 Self-organization skills

Self-organization is the ability to work in an ordered and methodical manner whilst being efficient and productive. Good self-organizational skills help us to cope with the world around us and are essential if we want to achieve personal goals as well as perform well in our job. These skills help keep us focused on doing the right Tasks, help us set our priorities and give us the confidence that we are following our chosen pathway to our desired destination.

Good self-organization requires the ability to prioritize, plan, manage time and work to deadlines. Self-organization is required for managing our time, resources, relationships, information, our environment, pressure and stress, and our behavior.

Limited self-organization skills may cause difficulties such as:

- Planning: setting achievable and realistic goals
- Implementing a systematic and organized strategy to achieve these objectives
- Identifying priorities
- Organizing one’s workload to maximize results
- Taking personal responsibility to achieve/exceed standards and expectations
- Taking responsibility to enhance one’s professional development by addressing and overcoming any weaknesses and fully utilize one’s strengths.

4.1.2 Communication and collaboration skills

Scrum attempts to simplify the often-confusing web of relationships through defining a clear set of roles and responsibilities.

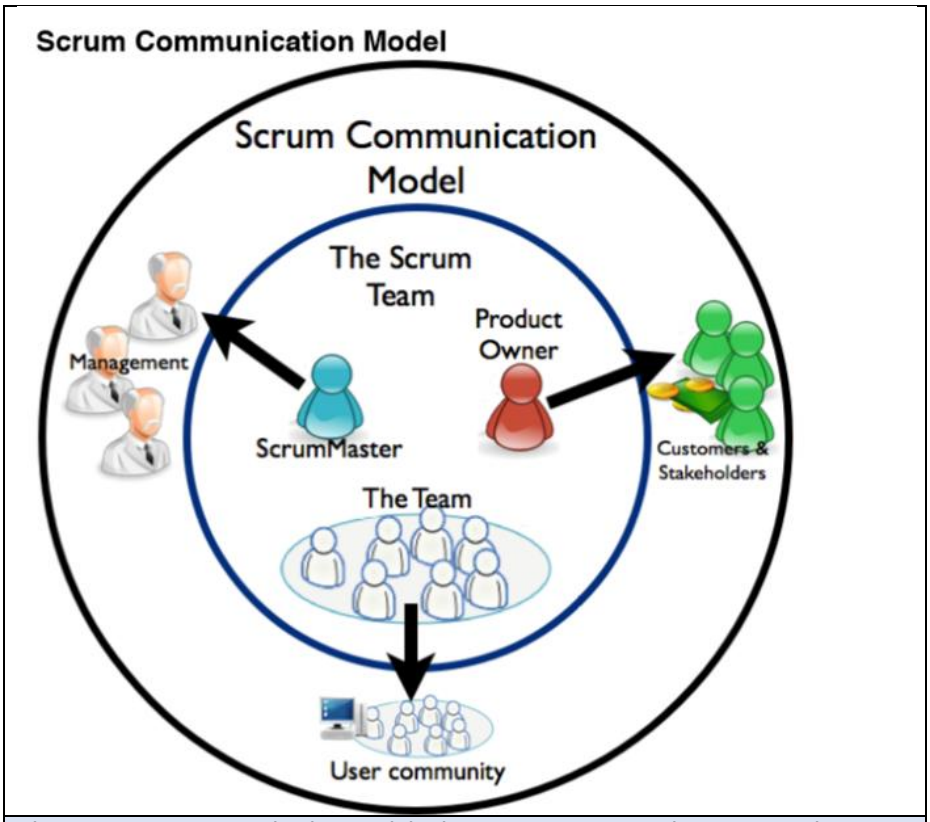


Figure: Scrum Communication Model. The Scrum Team, Product Owner, The Team, Scrum Master; Customers and Stakeholders, User Community, Management

Within the inner circle we find the Scrum Team. This includes the Product Owner, Scrum Master and the Team. Within this boundary, there should be no barriers to communication. This Team is collectively held accountable for the success of the product, and there should be an open dialogue between all its members.

The outward focus of the Scrum Team's communication is also illustrated above. The Product Owner's focus of communication should be with Customers (who pay the bill) and other Stakeholders who may have an interest in the product. Ensuring that their voices are heeded in the development of the product is vital to ensure its successful outcome.

Through the interactions within this group, we can better understand what to build, but even more importantly, when to stop building functionality. The value of code not written is difficult to measure but undeniable; we have no need to build, maintain, debug, test or release this code.

The Team should as far as possible communicate directly with the user community. Sometimes this may be the same as the Customer (if you are dealing with a direct Customer-facing product). However, where this is not the case (as in many enterprise scale applications) having the Product Owner as a proxy or conduit for information from users is not advised. Face-to-face interaction (or at least direct communication) between the user community and the Team reduces the likelihood of miscommunication.

It is often the case that the Scrum Master might interpret his or her mandate to "protect the Team" over-zealously, and attempt to limit or eliminate interaction between the Team and users. However, this information is essential to the Team. The Scrum Master should protect the Team from interruptions, not information.

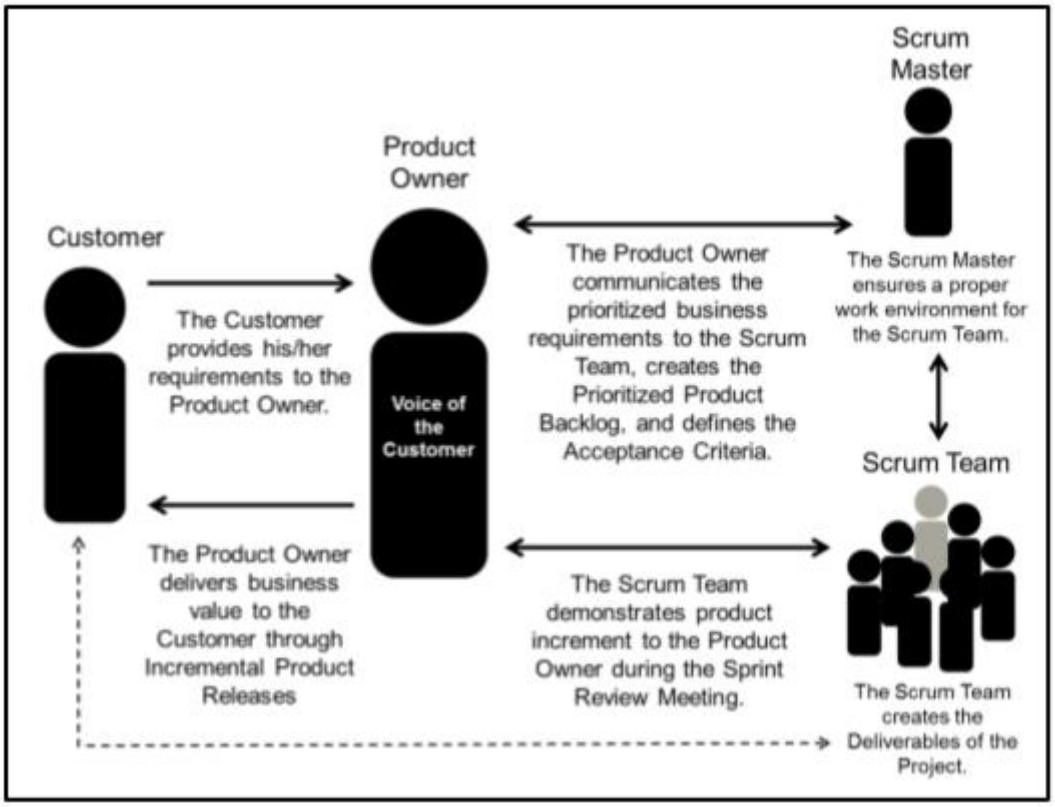


Figure: Organization in Scrum.
Customer—The Customer provides his/her requirements to the Product Owner.
Product Owner—*"The Voice of the Customer"*: The Product Owner delivers business value to the Customer through Incremental Product Releases. The Product Owner communicates the prioritized business requirements to the Scrum Team, creates the Prioritized Product Backlog, and defines the Acceptance Criteria.
Scrum Master—The Scrum Master ensures a proper work environment for the Scrum Team.
Scrum Team—The Scrum Team demonstrates product increment to the Product Owner during the Spring Review Meeting. The Scrum Team creates the Deliverables of the Project.

4.2 Working with HR on formation of the Scrum Team

Just as in other projects, we normally have to go through the HR department to obtain resources to form the Team, and in some cases, the Scrum Team is also part of a program or a project, which is a layer on top that controls the budget.

If you are a Product Owner you will have to go through the normal administrative processes in the company to assemble your Team. This involves some or all of the below Tasks or some other process:

- People Requirements
- People Availability and Commitment
- Organizational Resource Matrix
- Skills Requirements Matrix

4.2.1 People Requirements

Identifying People Requirements is one of the initial steps in selecting the Scrum Master and the Stakeholder(s). It is important to document the roles and responsibilities of all those who will be involved in completing the Tasks in the project. This includes all individuals involved in the project in any capacity, regardless of whether their role is core or non-core.

Usually, the Product Owner or the Scrum Master work with the Human Resource Department of the company to determine and finalize the People Requirements for a project.

4.2.2 People Availability and Commitment

Prior to selecting the Scrum Master and Stakeholder(s), their availability must be confirmed. Only Team members who will be available and can fully commit to the project should be selected. People Availability and Commitment are commonly depicted in the form of calendars showing when human resources will be available to work throughout the duration of the project.

To be effective, Scrum Teams should ideally have six to ten members; and replacing persons or changing Team members is not advisable in Scrum Core Teams. Therefore, it is important to have persons in the Scrum Core Team who are available and fully committed to the project.

4.2.3 Organizational Resource Matrix

The Organizational Resource Matrix is a hierarchical depiction that combines a functional organizational structure and a project organizational structure. Matrix organizations bring together Team members for a project from different functional departments such as information technology, finance, marketing, sales, manufacturing, and other departments, and create cross-functional Teams.

Team members in a matrix organization fulfill two objectives—functional and project. Team members are directed by Product Owner(s) with respect to project related activities, while the functional managers perform managerial activities related to their departments such as performance appraisals and approving leaves.

4.2.4 Skills Requirements Matrix

The Skills Requirement Matrix, also known as a competency framework, is used to assess skill gaps and training requirements for Team members. A skills matrix maps the skills, capabilities, and interest level of Team members in using those skills and capabilities on a project. Using this matrix, the organization can assess any skill gaps in Team members and identify the employees who will need further training in a particular area or competency.

05 DEVELOPING EPIC STORIES

In Scrum, the Teams that complete the work assign effort estimates to every User Story. Of course, that assumes that a Team can reach a consensus for an appropriate estimate. What happens when a Story includes too many unknowns to tell just how big it is? On the other hand, what if the Story's requirements are known, but its effort is too huge to complete in a single Sprint. We call these stories "Epics". While a Team should be able to tackle a typical Story in four to sixteen hours, an Epic is a Story that would require twelve or many more to complete. Most Scrum experts suggest that any Task requiring twelve or more hours should be decomposed into several smaller Tasks. These stories will not only be smaller in scope, but also more narrowly defined. Breaking down Epics helps the Development Team translate its work into chunks that can be accomplished in a single day.

Is there any danger to estimating an Epic? Quite simply, the answer is yes. Estimating Epics can be harmful because it creates a false sense of certainty for the Product Owner, who begins to believe that the requirements, Tasks, and effort of the Epic are known. When a Team estimates an Epic, that estimation is just that—an estimation—but it seldom remains a best guess. It is often used for forecasting, which, in turn, becomes the basis of a budget. When that happens, that estimate is now an inflexible projection that binds the Team to complete an unknown amount of work while respecting an established budget.

When putting User Stories onto a Product Backlog (or feature list), you should not feel compelled to break everything down until the features are nearing development. Further down the Product Backlog, its fine for items to be fuzzy. It is also fine for items further down the Backlog to be whole projects – large, high-level items that are not so much User Stories but more like Epics!

As an item nears development, the item should be broken down further. In addition, as it nears development, the item on the Backlog should be defined in sufficient detail that the Team could reasonably estimate its size and break it into Tasks. Until that time, however, it is just really a placeholder. A reminder for prioritization and high-level estimating. That is all.

For some people, particularly those used to a more traditional project approach, who are accustomed to getting detailed specifications up-front, this can potentially feel very uncomfortable. It should not. The logic here is simple. There is little point in defining a feature (or set of features) in detail if it may never reach the top of the priority list. The other aspect of this logic is that you tend to know more about your requirements, constraints, etc. as time goes by. Moreover, things change. People come and go. Sometimes the Team has changed significantly since the original requirements emerged, so opportunities can be lost if information is frozen too early. Therefore, it makes business sense to defer details until they are needed.

06 PRODUCT BACKLOG

The Program Product Owner develops the Program Product Backlog, which contains a prioritized list of high-level business and project requirements, preferably written in the form of major Program Backlog Items or Epics. These are later refined by the Product Owners of individual projects as they create and prioritize Product Backlogs for their projects. These Prioritized Product Backlogs have much smaller but detailed User Stories that can be approved, estimated, and committed by individual Scrum Teams.

The Program Product Backlog is continuously groomed by the Program Product Owner to ensure that new business requirements are added and existing requirements are properly documented and prioritized. This ensures that the most valuable criteria for meeting the program objectives get high priority, while others are put lower on the list.

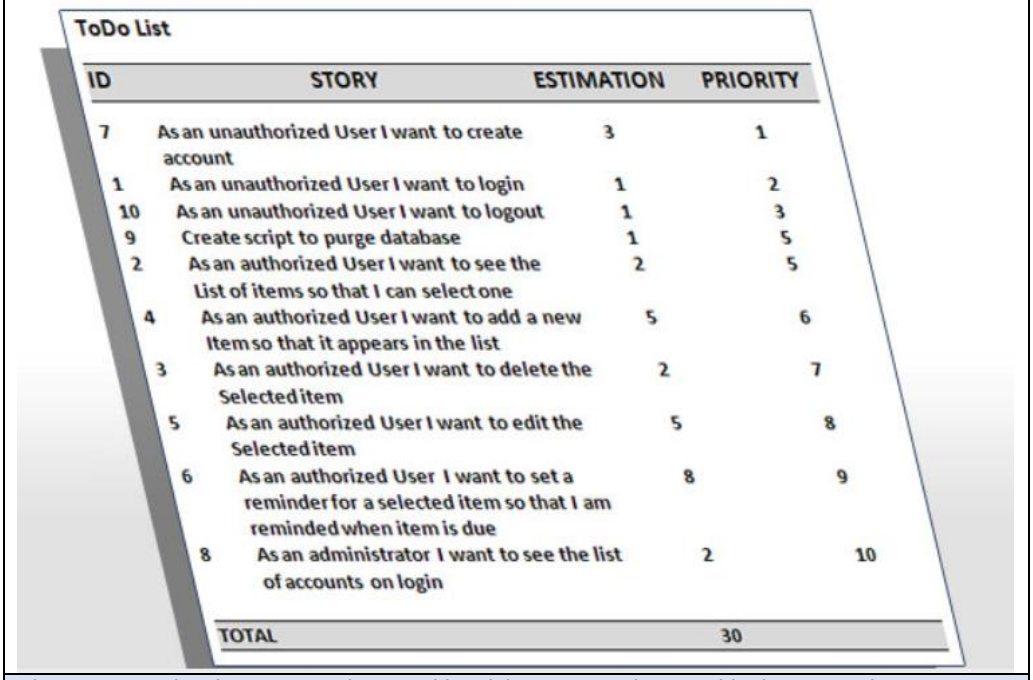
The Program Product Backlog created for the program presents a larger picture of all projects that are part of the program. Therefore, it can provide significant guidance regarding project goals, scope, objectives, and the expected Business Benefits.

The Scrum Product Owner uses the Scrum Product Backlog during the Sprint Planning Meeting to describe the top entries to the Team. The Scrum Team then determines which items they can complete during the upcoming Sprint.

6.1 Product Backlog

Each Scrum Product Backlog has certain properties that differentiate it from a simple to-do list:

- An entry in the Scrum Product Backlog always adds value for the Customer
- The entries in the Scrum Product Backlog are prioritized and ordered accordingly
- The level of detail depends on the position of the entry within the Scrum Product Backlog
- All entries are estimated
- The Scrum Product Backlog is a living document
- There are no action-items or low-level Tasks in the Scrum Product Backlog



ToDo List

ID	STORY	ESTIMATION	PRIORITY
7	As an unauthorized User I want to create account	3	1
1	As an unauthorized User I want to login	1	2
10	As an unauthorized User I want to logout	1	3
9	Create script to purge database	1	5
2	As an authorized User I want to see the List of items so that I can select one	2	5
4	As an authorized User I want to add a new Item so that it appears in the list	5	6
3	As an authorized User I want to delete the Selected item	2	7
5	As an authorized User I want to edit the Selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
TOTAL		30	

Figure: Example of Scrum Product Backlog (also see Product Backlog), To Do List. ID, Story, Estimation, Priority. For instance: ID 7: As an unauthorized User I want to create an account, Estimation 3, Priority 1, etc.

The Product Backlog and the business value of each backlog item are the responsibility of the Product Owner. The size (i.e. estimated complexity or effort) of each backlog item is determined by the Development Team, who contributes by sizing items, either in Story Points or in estimated hours.

The idea that only User Stories are allowed in a Product Backlog is a common misunderstanding. By contrast, Scrum is neutral on requirement techniques. As stated in Scrum Guidelines:

> *Product Backlog items are articulated in any way that is clear and sustainable. Contrary to popular belief, the Product Backlog does not contain "User Stories"; it simply contains items. Those items can be expressed as User Stories, use cases, or any other set of requirements that the group finds useful. However, whatever the approach, most items should focus on delivering value to Customers.*

The Product Owner is responsible for maximizing the value of the product and the work of the Development Team. The Product Owner gathers input, takes feedback and is lobbied by many people, but will ultimately have to make the call on what to build. The Product Owner has sole responsibility for management of the Backlog.

The Product Backlog is used to:

- Capture requests for modifying a product. This can include adding new features, replacing old features, removing features and fixing issues
- Ensure the Delivery Team is given work which maximizes the Business Benefit to the owner of the product.

Typically, the Product Owner and the Scrum Team come together and write down everything that needs to be prioritized and this becomes the content of the first Sprint, which is a block of time set aside for focused work on selected items that can be accommodated within a timeframe. The Scrum Product Backlog is permitted to evolve as new information surfaces about the product and its Customers, and so new work is scheduled for subsequent Sprints.

The following items typically comprise a scrum backlog: features, bugs, technical work, and knowledge acquisition. In the web development sphere, there is confusion as to the difference between a feature and a bug, technically a feature is "wanted", while a bug is a feature that is "unintended" or "unwanted" (but not necessarily a fault). An example of technical work would be: "run virus check on all developers' workstations". An example of knowledge acquisition could be a scrum backlog item about researching WordPress plugin libraries and making a selection.

6.2 Product Backlog between Product Owner and Scrum Team

A backlog, in its simplest form, is merely a list of items to be worked on. Having well-established rules about how work is added, removed and ordered helps the whole Team make better decisions about how to change the product.

The Product Owner prioritizes which of the Product Backlog items are most needed. The Team then chooses which items can be completed in the upcoming Sprint. On the Scrum Board, the Team moves items from the Product Backlog to the Sprint Backlog, which is the list of items they will now build. Conceptually, it is ideal for the Team to only select what they think they can accomplish from the top of the list, but it is not unusual to see in practice that Teams are able to take lower priority items from the list along with the top ones selected. This normally happens because there is time left within the Sprint to accommodate more work. Items at the top of the Backlog, the items that are going to be worked on first, should be broken down into Stories that are suitable for the Delivery Team to work on. The further

down the Backlog goes, the less refined the items should be. As Schwaber and Beedle put it "*The lower the priority, the less detail, until you can barely make out the Backlog item*."

As the Team works through the Backlog, one must always be mindful that "changes in the world can happen"; the Team can learn about new market opportunities to take advantage of, competitor threats that arise, and feedback from Customers that can change the way the product was intended to work. All of these new ideas tend to trigger the Team to adapt the Backlog to incorporate new knowledge. This is part of the fundamental mindset of an Agile Team. The world changes, the Backlog is never finished.

07 WORKING WITH PRODUCT BACKLOG

One of the most important jobs for a Product Owner is to work with the Product Backlog, and here we will look into the following aspects:

- DEEP
- Grooming the Product Backlog
- Prioritizing the Product Backlog
- Getting ready for Sprint planning
- Estimation
- Scaling the Product Backlog.

7.1 DEEP

The Product Backlog is a living document, and it is up to the Product Owner to keep the Product Backlog updated. We normally say that the Product Backlog should be DEEP, which means:

- Detailed Appropriately
- Estimated
- Emergent
- Prioritized.

Let us have a look at this Product Backlog diagram;

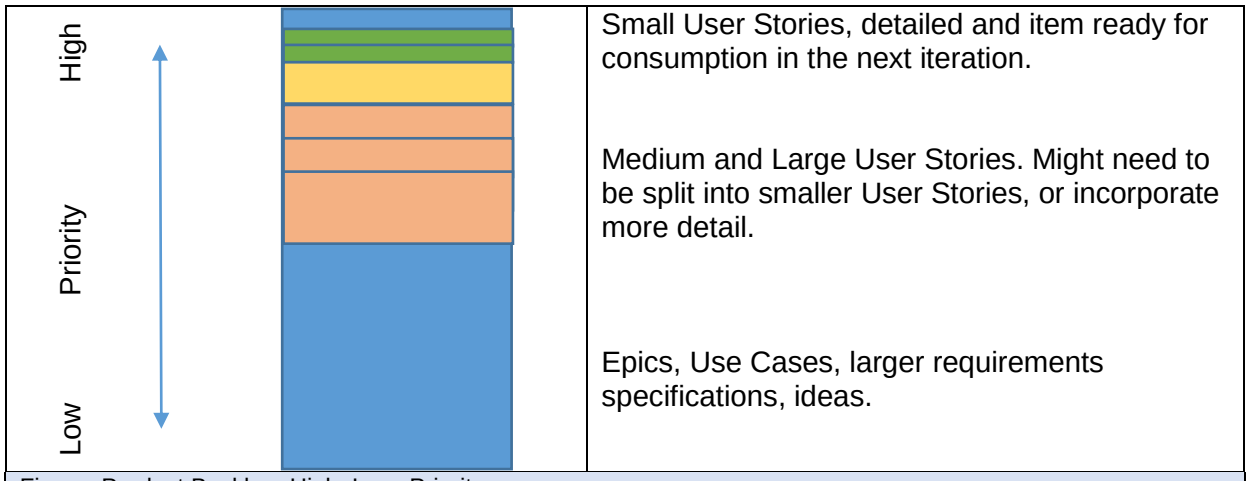


Figure: Product Backlog. High, Low, Priority.

7.1.1 Detailed Appropriately

The top items on the Backlog list should have enough detail for the Delivery Team to start delivering. Items further down the Backlog should have enough detail to allow them to be planned.

7.1.2 Estimated

Scrum does not prescribe an estimation technique, Extreme Programming suggests estimating User Stories as either 1, 2, or 3 weeks in ideal development time. The common practice in Agile Teams is to estimate in relative units, typically Story Points.

Regardless of the estimation technique used, Product Owners often require items to be estimated before they are ordered. We believe that there are two estimates that should be captured, the Cost Estimate and the Business Benefit Estimate. The Cost Estimate represents the amount of effort it will take for the Delivery Team to complete a backlog item and only the Delivery Team should add a Cost Estimate. The Product Owner should supply the Business Benefit; this is a representation of value returned to the business when the item

is complete. The relationship between the cost and the benefit represents the return on investment (ROI).

Many Teams do not create Business Benefit Estimates and instead have the Product Owner use their expert knowledge to calculate the benefit estimate.

7.1.3 Emergent

The Backlog is a dynamic artifact that changes over time as we have detailed in "How does it change over time".

7.1.4 Prioritized

The Backlog is an ordered list, which means that every item in the list is part of a sequence with no two items holding the same position. This rule is an attempt to break the notion that every item in the Backlog is "Priority One". By applying this rule consistently, it will force Product Owners into making decisions about the importance of one Story relative to another.

It is common to order the Backlog by some combination of return on investment, risk, priority and necessity—but ultimately it is up to the Product Owner to make the tradeoff decisions.

7.2 Grooming the Product Backlog

The Team (or part of the Team including the Product Owner) meet regularly to *"groom the Product Backlog"*, in a formal or informal meeting which can lead to any of the following:

- Removing User Stories that no longer appear relevant
- Creating new User Stories in response to newly discovered needs
- Re-assessing the relative priority of Stories
- Assigning estimates to Stories which have yet to receive one
- Correcting estimates in light of new information
- Decomposition of User Stories which, although high priority, are too broad to fit in an upcoming iteration.

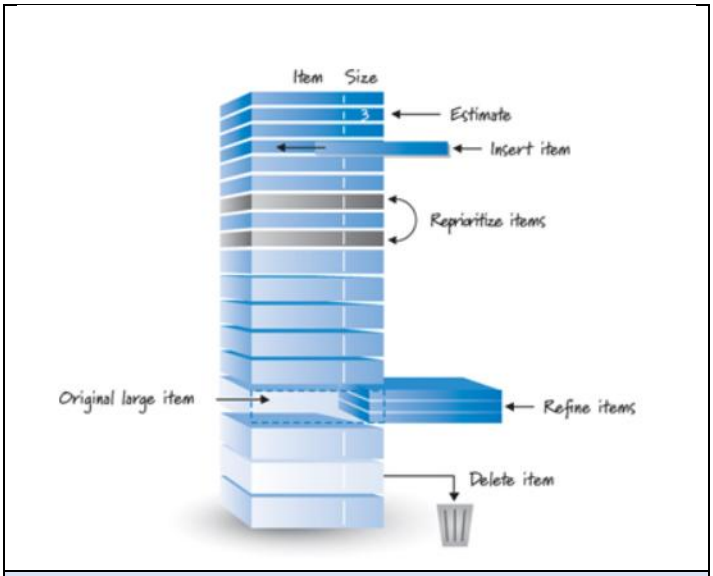


Figure: Item, Size, Estimate, Insert item, Reprioritize items, Original large item, Refine items, Delete item.

Expected Benefits

The intent of a *"grooming"* meeting is to ensure that the Backlog remains populated with items that are relevant, detailed and estimated to a degree appropriate with their priority, and in keeping with current understanding of the project or product and its objectives.

Unlike a more formal *"requirements document"* the Backlog is understood as a dynamic body of information. For instance, not all User Stories need to have been decomposed to a detailed level at the onset of the project; nor to all User Stories need to be estimated in detail; but it is important that at any moment a *"sufficient"* number of stories should be ready for scheduling in the next few iterations.

An Agile Project is, no less than any other, subject to *"mission creep"*, in the form of User Stories, which do not really yield substantial value but were thought *"good ideas at the time"*, and entered into the Backlog lest they be forgotten. In the absence of explicit efforts aimed at managing this inflation, the result will be the all-too-common pathologies of schedule and budget overruns.

7.3 Prioritizing the Product Backlog

The Product Backlog in Scrum is normally a prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it is not necessary to start a project with a lengthy, upfront effort to document all features in detail.

Typically, a Scrum Team and its Product Owner begin by writing down everything they can think of for the Backlog prioritization. This kind of Product Backlog is usually more than enough for a first Sprint. The Scrum Product Backlog will then grow and change as more is learned about the product and its Customers.

A typical Scrum backlog comprises the following different types of items:

1. Features
2. Bugs
3. Technical work
4. Knowledge acquisition.

A Scrum Team normally express features on the Product Backlog in the form of User Stories, which are short, simple descriptions of the desired functionality told from the perspective of the user (more about User Stories in Chapter 9).

Consider the following when prioritizing:

- The value of the feature to the general user
- The value of the feature to the specialist user
- Story cost
- Estimated time to implement
- Impact on other Stories
- Risk and opportunity costs.

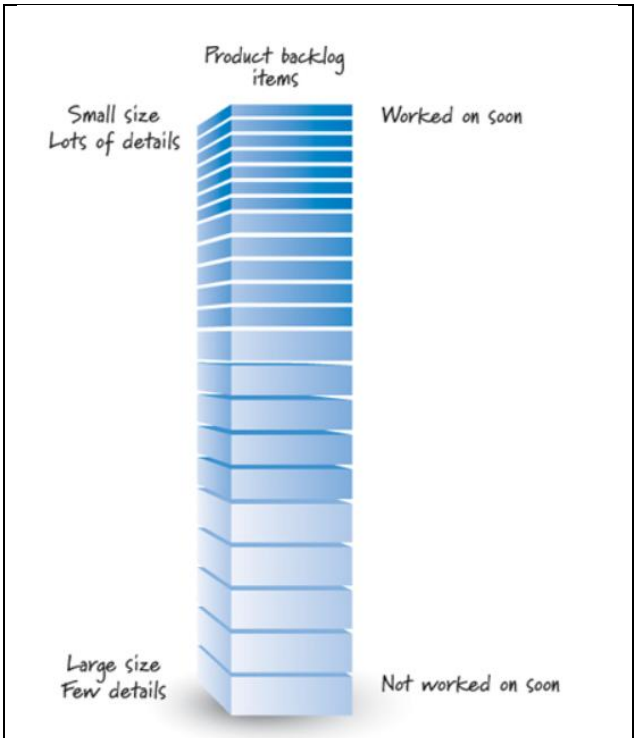


Figure: Product backlog items, Small size—Lots of details, Worked on soon, Large size—few details, Not worked on soon

User Story Prioritization can be done in many ways, but here is a brief list of common techniques used to prioritize the User Stories and specific requirements in the Prioritized Product Backlog:

- **Relative Prioritization Ranking**: A simple listing of User Stories in order of priority is an effective method for determining the desired User Stories for each iteration or release of the product or service. The purpose is to create a simple, single list with the goal of prioritizing features, rather than being distracted by multiple prioritization schemes.

  This simple list also provides a basis for incorporating changes and identified risks when necessary. Each change or identified risk can be inserted in the list based on its priority relative to the other User Stories in the list. Typically, new changes will be included at the expense of features that have been assigned a lower priority.

  Defining the Minimum Marketable Features (MMF) is extremely important during this process, so that the first release or iteration happens as early as possible, leading to increased ROI. Normally, these User Stories would rank highest in priority.

- **MoSCoW Prioritization scheme**: The MoSCoW prioritization scheme derives its name from the first letters of the phrases *"Must have"*, *"Should have"*, *"Could have"* and *"Won't have"*. This prioritization method is generally more effective than simple schemes. The labels are in decreasing order of priority with *"Must have"* User Stories being those without which the product will have no value and *"Won't have"* User Stories being those that, although they would be nice to have, are not necessary to be included.

- **Paired Comparison**:In this technique, a list of all the User Stories in the Prioritized Product Backlog is prepared. Next, each User Story is taken individually and

compared with the other User Stories in the list, one at a time. Each time two User Stories are compared, a decision is made regarding which of the two is more important. Through this process, a prioritized list of User Stories can be generated.

- **100-Point Method**: Dean Leffingwell and Don Widrig (2003) developed the 100-Point Method. It involves giving the Customer 100 points they can use to vote for the User Stories that are most important. The objective is to give more weight to the User Stories that are of higher priority when compared to the other available User Stories. Each group member allocates points to the various User Stories, giving more points to those they feel are more important. On completion of the voting process, prioritization is determined by calculating the total points allocated to each User Story.

- **Kano Analysis**
  Consider the following when prioritizing:
  - The value of the feature to the general user
  - The value of the feature to the specialist user
  - Story cost
  - Estimated time to implement
  - Impact on other stories
  - Risk and opportunity costs

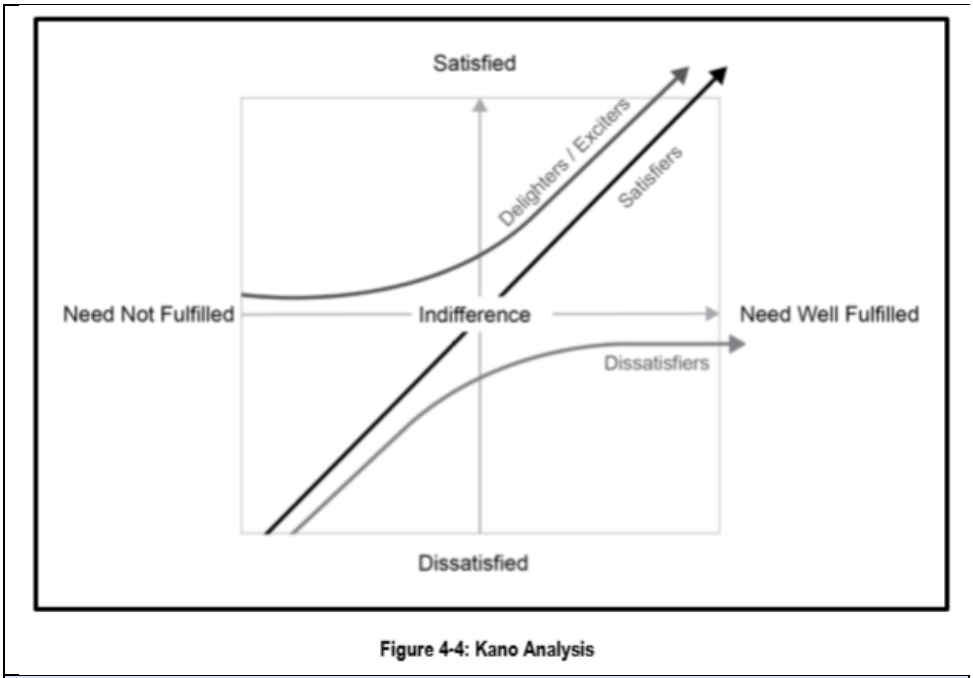


Figure 4-4: Kano Analysis

Figure 4-4: Kano Analysis. Satisfied, Dissatisfied, Need Not Fulfilled, Indifference, Need Well Fulfilled. Delighters/ Exciters, Satisfiers, Dissatisfiers.

Interestingly, features usually move down the classification list over time; Customers will come to expect features (e.g., cameras on phones) and these features will move from being exciters and delighters to satisfiers and eventually to dissatisfiers. There are other methods like Theme Screening, Theme Scoring, Relative Weighting and Financial Analysis, which will be handled in depth in the Advanced Level Product Owner course.

08 RELEASE PLANNING

Release Planning Sessions are conducted to develop a Release Plan. The plan defines when various sets of usable functionality or products will be delivered to the Customer. In Scrum, the major objective of a Release Planning Meeting is to enable the Scrum Team to have an overview of the releases and delivery schedule for the product they are developing, so that they can align with the expectations of the Product Owner and relevant Stakeholders (primarily the project sponsor).

7.1 Release planning

Many organizations have a strategy regarding release of products. Some organizations prefer continuous deployment, where there is a release after creation of specified usable functionality. Other organizations prefer phased deployment, where releases are made at predefined intervals. Depending on the organization's strategy, Release Planning Sessions in projects may be driven by functionality, in which the objective is to deliver a release once a predetermined set of functionality has been developed; or the planning may be driven by date, in which the release happens on a predefined date.

Since Scrum framework promotes information-based, iterative decision making over the detailed upfront planning practiced in traditional waterfall style project management, Release Planning Sessions need not produce a detailed Release Plan for the entire project. The Release Plan can be updated continually, as relevant information is available.

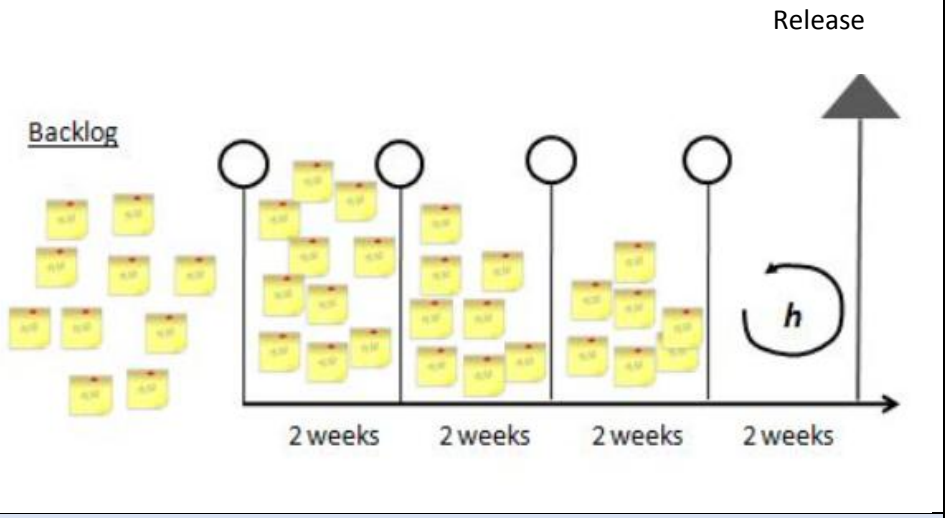


Figure: Backlog, 2 weeks, 2 weeks, 2 weeks, 2 weeks, Release

8.2 Release Prioritization Methods

Release Prioritization Methods are used to develop a Release Plan. These methods are industry and organization specific and are usually determined by the organization's senior management.

8.3 Release Planning Schedule

A Release Planning Schedule is one of the key outputs of the Conduct Release Planning process. A Release Planning Schedule states which Deliverables are to be released to the Customers, along with planned intervals, and dates for releases. There may not be a release scheduled at the end of every Sprint iteration. At times, a release may be planned after a group of Sprint iterations is completed. Depending on the organization's strategy, Release Planning sessions in projects may be driven by functionality, in which the objective is to deliver once a predetermined set of functionality has been developed, or the planning may be driven by date, in which case the release happens on a predefined date. The Deliverable should be released when it offers sufficient business value to the Customer.

8.4 Length of Sprint

Based on the various inputs including business requirements and Release Planning Schedule, the Product Owner and the Scrum Team decide on the Length of Sprint for the project. Once determined, the Length of Sprint often remains the same throughout the project.

However, the Length of Sprint may be changed if and when the Product Owner and the Scrum Team deem appropriate. Early in the project, they may still be experimenting to find the best Sprint length. Later in the project, a change in the Length of Sprint normally means it can be reduced due to improvements in the project environment.

A Sprint could be time-boxed for from 1 to 6 weeks. However, to get maximum benefits from a Scrum project, it is recommended to keep the Sprint time-boxed to 4 weeks, unless there are projects with very stable requirements, when Sprints can extend up to 6 weeks.

8.5 Target Customers for Release

Not every release will target all Stakeholders or users. The Stakeholder(s) may choose to limit certain releases to a subset of users. The Release Plan should specify the target Customers for the release.

8.6 Refined Prioritized Product Backlog

The Prioritized Product Backlog, developed in the Create Prioritized Product Backlog process, may be refined in this process. There may be additional clarity about the User Stories in the Prioritized Product Backlog after the Scrum Core Team conducts Release Planning Sessions with Stakeholder(s).

09 USER STORIES

9.1 A User Story

A **User Story** is one or more sentences in the everyday or business language of the consumer or end-user of a system that captures what a user does or needs to do as part of his or her job function. User Stories are harnessed with agile software development methodologies as the basis for defining the functions a business system must provide, and to facilitate requirements management. It captures the 'who', 'what' and 'why' of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a small paper notecard.

The Product Owner, based on his or her interaction with the Stakeholders, business knowledge and expertise, and inputs from the Team, develops User Stories that will form the initial Prioritized Product Backlog for the project. The Prioritized Product Backlog represents the total sum of what must be completed for the project. The objective of this exercise is to create elaborated and refined User Stories that can be approved, estimated, and committed to by the Scrum Team. At times, the Product Owner may bring a Business Analyst to assist with writing User Stories.

Although the Product Owner has the primary responsibility for writing User Stories and often carries out this exercise on his or her own, a User Story Writing Workshop can be held if desired. In real life situations, User Stories are often written by or for business users or Customers as a primary way to influence the functionality of the system being developed. User Stories may also be written by developers to express non-functional requirements (security, performance, quality, etc.) However, primarily it is the Task of a product manager to ensure User Stories are captured.

Here are some examples of User Stories, with Story Points:

#	Backlog Item (User Story)	Story Point
1.	As a Teller, I want to be able to find clients by last name, so that I can find their profile faster	4
2.	As a System Admin, I want to be able to configure user settings so that I can control access.	2
3.	As a System Admin, I want to be able to add new users when required, so that...	2
4.	As a data entry clerk, I want the system to automatically check my spelling so that...	1

9.2 Creating User Story

User Stories adhere to a specific, predefined structure, and are thus a simplistic way of documenting the requirements, and desired end-user functionality. A User Story tells you three things about the requirement; **Who**, **What**, and **Why**. The requirements expressed in User Stories are short, simple, and easy-to-understand statements. The predefined, standard format results in enhanced communication among the Stakeholders and better estimations by the Team. Some User Stories may be too large to handle within a single Sprint. These large User Stories are often called Epics. Once Epics come up in the Prioritized Product Backlog to be completed in an upcoming Sprint, they should be decomposed into manageable User Stories.

The Prioritized Product Backlog is a dynamic list that is continuously updated because of reprioritization and new, updated, refined, and sometimes, deleted User Stories. These updates to the Backlog are typically the result of changing business requirements.

As the Customer representative conceives a User Story, it is written down on a note card with a name and a brief description. If the developer and the Customer representative find a

User Story deficient in some way (too large, complicated, or imprecise), it is rewritten until satisfactory—often using the INVEST guidelines (see Chapter 8.5 below). Commonly, User Stories should not be considered definitive once they have been written down, since requirements tend to change throughout the development lifecycle, which agile processes handles by not carving them in stone upfront.

9.3 Format for creation of User Story

A useful format for creating a User Story is the following:

"As a *<role>,* I want *<goal/desire>* so that *<benefit>".*

Here is an example using the format: As a Database Administrator, I should be able to revert a selected number of database updates so that the desired version of the database is restored.

9.4 User Story Acceptance Criteria

Every User Story has associated Acceptance Criteria. User Stories are subjective, so the Acceptance Criteria provide the objectivity required for the User Story to be considered as Done or Not Done during the Sprint Review. Acceptance Criteria provide clarity to the Team on what is expected of a User Story, remove ambiguity from requirements, and help in aligning expectations. The Product Owner defines and communicates the Acceptance Criteria to the Scrum Team.

In the Sprint Review Meetings, the Acceptance Criteria provide the context for the Product Owner to decide if a User Story has been completed satisfactorily. It is important and the responsibility of the Scrum Master to ensure that the Product Owner does not change the Acceptance Criteria of a committed User Story in the middle of a Sprint.

9.5 INVEST criteria for User Stories

The INVEST system features the following characteristics.

Letter	Meaning	Description
I	Independent	The User Story should be self-contained, in a way that there is no inherent dependency on another User Story.
N	Negotiable	User Stories, up until they are part of an iteration, can always be changed and rewritten.
V	Valuable	A User Story must deliver value to the end user.
E	Estimable	You must always be able to estimate the size of a User Story.
S	Scalable (small sized)	User Stories should not be so big as to become impossible to plan or Task or prioritize with some level of certainty.
T	Testable	The User Story or its related description must provide the necessary information to make test development possible.

10 APPROVE, ESTIMATE AND COMMIT USER STORIES

The User Stories, which are input to this process, have high-level estimates from the Create Prioritized Product Backlog and Create User Stories processes. These estimates are used by the Product Owner to approve User Stories for the Sprint.

It must be noted that it is the Product Owner's responsibility to ensure that approved User Stories deliver value and meet the needs and requirements of the Project Stakeholders. Once approved, the User Stories are estimated by the Team using the various estimation techniques discussed in this section. After estimation, the Team commits to a subset of approved and estimated User Stories that they believe they can complete in the next Sprint. These User Stories are Approved, Estimated, and Committed User Stories that will become part of the Sprint Backlog.

Although the Product Owner approves the initial User Stories for a Sprint, the final decision about which specific User Stories (among those approved by the Product Owner) should be chosen for the Sprint lies with the Scrum Team. The Scrum Team (with consultation from the Product Owner, if required) finalizes which User Stories they will be working on during the Sprint.

10.1 Planning Poker

Planning Poker, also called Estimation Poker, is an estimation technique that uses consensus to estimate relative sizes of User Stories or the effort required to create them.

In Planning Poker, each Team member is assigned a deck of cards. Each card is numbered in a sequence and the numbers represent the complexity of the problem, in terms of time or effort, as estimated by the Team member. The Product Owner chooses a User Story from the Prioritized Product Backlog and presents it to the Team. The Scrum Team members assess the User Story and try to understand it better before providing their estimate for developing it.

Then, each member picks a card from the deck that represents his or her estimate for the User Story. If the majority or all Team members select the same card then the estimate indicated by that card will be the estimate for that User Story. If there is no consensus, then the Team members discuss reasons for selecting different cards or estimates. After this discussion, they pick cards again. This sequence continues until all the assumptions are understood, misunderstandings are resolved, and consensus or agreement is reached.

Planning Poker advocates greater interaction and enhanced communication among the participants. It facilitates independent thinking by participants, thus avoiding the phenomenon of groupthink.

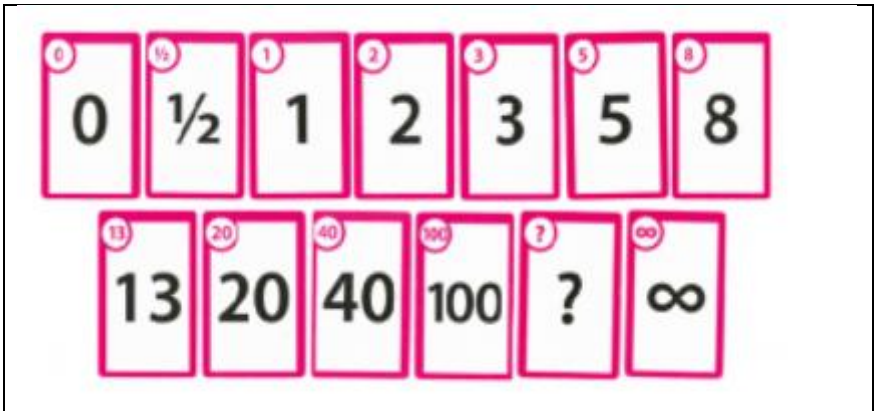


Figure: Fibonacci Series Cards for Planning Poker

- Agile Planning Poker cards can be used for the activity, or simple numbers are written on paper prior to the meeting and distributed to Scrum Team members.

- Sizing and estimation using Planning Poker is an abstract method that takes the focus off the actual time in hours or days and instead puts the focus on describing the relative expense and complexity of a Task as compared to other Tasks.

The Scrum Team use Fibonacci-like numbers to estimate effort: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100:

- Number estimates are based on relative sizing that the Team feels is appropriate.

- Zero "0" means no effort required to complete this Story (functionality might get created due to completion of some other User Story); 1 means the Story is trivial; 5 means it's average; 20 means it's extremely difficult or much remains unknown.

- If the Team provides an estimate of 3 or 5 or 8 then those User Stories are perfect for this Team.

- 13 means the Team is asking nicely for you to break the Story down into smaller pieces if possible.

- 20 or 40 means the Team is telling you that Story is too big and it must be broken down into multiple smaller stories.

The Planning Poker process is conducted as follows:

1. The Scrum Master allows the Team to read the Story, and, if necessary, the Product Owner or Functionality Expert explains the Story.

2. The Scrum Master facilitates the discussion on dependencies, and what technical stories or technical Tasks need to be accomplished to support the Story completion. The technical stories or Tasks are noted.

3. For User Stories or derived technical stories, the Scrum Master asks the developers to vote using Planning Poker method, with a *"one-two-three-go!".*

4. The Scrum Master then challenges the highest and lowest votes to explain their position and rationale behind their selection, i.e. why they believe it is more or less difficult than their peers expect.

5. The Scrum Master repeats the votes until the numbers converge and the Team as whole can agree on a number or small group of numbers.

6. If the Team has not agreed on a single number then the Scrum Master decides the best number based on the number of votes, or sometimes by affording more weight to the votes of Team members who are most likely be working on that particular User Story.

7. The Scrum Master assigns the effort points to the Story, a.k.a the Plan Estimate.

8. Repeat until there are no more Stories or the allocated time for the meeting is exhausted.

10.2 Fist of Five

The Fist of Five is a simple and fast mechanism to achieve consensus in a group and drive discussion. After initial discussion on a given proposal or a pending decision, the Scrum Team members are each asked to vote on a scale of 1 to 5 using their fingers. The value in using this technique is not only consensus building, but also driving discussion because each Team member is asked to explain the reason for their ranking. They are also given the opportunity to express any issues or concerns. Once the Team has discussed it, a collective decision will be made.

The number of fingers used to vote indicates the level of agreement and desire for discussion:

Fingers shown	Meaning
One finger:	I disagree with the group's conclusion and have major concerns.
Two fingers:	I disagree with the group's conclusion and would like to discuss some minor issues.
Three fingers:	I am not sure and would like to go with the group's consensus conclusion.
Four fingers:	I agree with the group's conclusion and would like to discuss some minor issues
Five fingers:	I wholeheartedly agree with the group's conclusion.

The Fist of Five method is also useful in certain cases to find out if people understand the discussion subject or the requirement in detail.

Figure: I do not understand at all; I need to go over this again; I think I get it, but am not completely comfortable; I got it; I can explain it to someone else.

10.3 Points for Cost Estimation

Cost estimation can be accomplished using relative units (e.g., effort estimates) rather than absolute units (i.e., actual costs incurred). In order to estimate the cost to implement a User Story, the Scrum Team can use Story Points. When this is done, the Cost Estimated for each Task will be in the form of Story Points, rather than monetary units. In order to do this successfully, the Scrum Team should identify a baseline User Story that all Team members can relate to. Once this baseline is identified, all Cost Estimates for User Stories should be done compared to that baseline. These estimates remain fixed throughout a Sprint because Teams are not supposed to change during a Sprint.

10.4 Wideband Delphi

Wideband Delphi is a group-based estimation technique for determining how much work is involved and how long it will take to complete. Individuals within a Team anonymously provide estimations for each feature and the initial estimates are plotted on a chart. The Team then discusses the factors that influenced their estimates and proceed to a second round of estimation. This process is repeated until the estimates of individuals are close to each other and a consensus for the final estimate can be reached.

10.5 Relative Sizing and Story Points

In addition to being used for estimating cost, Story Points can also be used for estimating the overall size of a User Story or feature. This approach assigns a Story Point value based on an overall assessment of the size of a User Story with consideration given to risk, amount of effort required, and level of complexity. The Scrum Team will conduct this assessment and a Story Point value will be assigned. Once an evaluation is done on one User Story in the Prioritized Product Backlog, the Scrum Team can then evaluate other User Stories relative to that first Story. For example, a feature with a 2-point Story value must be twice as difficult to complete as a feature with a 1-point Story; a 3-point Story should be three times as difficult to complete as a 1-point Story.

10.6 Affinity Estimation

Affinity Estimation is a technique used to quickly estimate a large number of User Stories. Using sticky notes or index cards and tape, the Team places User Stories on a wall or other surface, in order from small to large. For this, each Team member begins with a subset of User Stories from the overall Prioritized Product Backlog to place by relative size. This initial placement is done in silence. Once everyone has placed their User Stories on the wall, the Team reviews all of the placements and may move User Stories around as appropriate. This second part of the exercise involves discussion. Finally, the Product Owner will indicate some sizing categories on the wall. These categories can be small, medium, or large, or they may be numbered using Story Point values to indicate relative size. The Team will then move User Stories into these categories as the final step in the process. Some key benefits of this approach are that the process is very transparent, visible to everyone, and is easy to conduct**.**

10.7 Estimate Range

Estimates for projects should be presented in ranges. Precise figures may give an impression of being highly accurate when in fact they may not be. Indeed, estimates by definition are understood not to be precisely accurate. Estimate ranges should be based on the level of confidence the Team has in each estimate. The range can be narrow when the Team is confident and wide when the Team is less confident.

11 CREATE AND ESTIMATE TASKS

11.1 Sprint Planning Meeting

The Product Owner, Scrum Master, the entire Scrum Team, and any interested and appropriate management or Customer representatives attend the Sprint Planning Meeting.

During the Sprint Planning Meeting the Product Owner describes the highest priority features to the Team. The Team asks enough questions during this meeting so that they can go off after the meeting and determine which Tasks they will move from the Product Backlog to the Sprint Backlog.

Collectively, the Scrum Team and the Product Owner define a Sprint goal, which is a short description of what the Sprint will attempt to achieve. The success of the Sprint will later be assessed during the Sprint Review Meeting against the Sprint goal, rather than against each specific item selected from the Product Backlog.

After the Sprint Planning Meeting, the Scrum Team meets separately to discuss what they heard and decide how much they can commit to during the upcoming Sprint. In some cases, there will be negotiation with the Product Owner, but it will always be up to the Team to determine how much they can commit to completing.

The Task Planning Meeting is normally divided into two sections, with a designated purpose and broad agenda for each.

TPM First Part	• The Product Owner suggests User Stories that should be part of the Sprint • The Scrum Team determines how many User Stories it can perform in the Sprint • A consensus is achieved on the User Stories to be included in the Spring
TPM Second Part	• The Scrum Team determines how to turn the selected User Stories into a Product Increment by breaking them down into Tasks • The Scrum Team commits to the Deliverables for the Sprint

Figure: Task Planning Meeting, Broad Agenda

11.2 Index Cards

In Scrum, User Stories are written on small Index Cards. Only essential details are documented on the cards, which can be used by the Scrum Team to collaborate and discuss. These Index Cards, often described as Story Cards, increase visibility and transparency, and facilitate early discovery of any problems that may arise.

11.3 Decomposition

Decomposition is a tool whereby high-level Tasks are broken down into more detailed, low-level Tasks. Members of the Scrum Team decompose the User Stories into Tasks. Prioritized Product Backlog User Stories should be sufficiently decomposed so that the Scrum Team has adequate information to create Deliverables from the Tasks itemized in the Task List.

11.4 Dependency Tree

Once the Scrum Team has selected User Stories for a given Sprint, they should then consider any dependencies, including those related to the availability of personnel as well as any technical dependencies. Properly documenting dependencies helps the Scrum Teams to determine the relative order in which Tasks should be executed to create the Sprint

Deliverables. Dependencies also highlight the relationship and interaction between Tasks, both within the Scrum Team working on a given Sprint, and within other Scrum Teams in the project.

There are numerous types of dependencies: mandatory and discretionary, internal and external, or some combination thereof. For example, a dependency may be both mandatory and external.

- **Mandatory dependencies:** Dependencies that are either inherent in the nature of the work, as a physical limitation, or that may be due to contractual obligations or legal requirements. For example, work on the first floor cannot begin until the foundation of the building is complete. Mandatory dependencies are also commonly described as hard logic.

- **Discretionary dependencies:** Dependencies that are placed into the workflow by choice. Typically, the Scrum Team determines discretionary dependencies based on experience or best practices in a particular field or domain. The Team may decide to complete one Task before working on another because it is the best practice, although not a requirement. For instance the Team may choose to build the door and window frames before the full structure of the wall is in place.

- **External dependencies:** External dependencies are those related to Tasks, activities, or products that are outside the scope of the work to be executed by the Scrum Team, but are needed to complete a Project Task or create a Project Deliverable. External dependencies are usually outside the Scrum Team's control. For example, if the Scrum Team is not responsible for procuring the materials required for building the walls, then those materials and Tasks related to their procurement are considered external dependencies.

- **Internal dependencies**: Internal dependencies are those dependencies between Tasks, products, or activities that are under the control of the Scrum Team. For example, installing dry-wall must be completed before painting the wall can begin. This is an example of an internal dependency because both Tasks are part of the project. In this case, it is also mandatory because it is based on a physical limitation. It is not possible to paint the wall before it is dry-walled.

11.5 Output from the Sprint Planning Meeting

- **Task List:** This comprehensive list contains all the Tasks to which the Scrum Team has committed for the current Sprint. It contains descriptions of each Task along with estimates derived during the Create Tasks process. The Task List must include any testing and integration efforts so that the Product Increment from the Sprint can be successfully integrated into the Deliverables from previous Sprints.

  Even though Tasks are often activity-based, the Scrum Team decides the level of granularity to which the Tasks are decomposed.

- **Updated Approved, Estimated, and Committed User Stories**: The User Stories are updated during this process. Updates can include revisions to the original User Story estimates based on Tasks creation and complexity factors discussed during the Sprint Planning Meeting

- **Dependencies:** Dependencies describe the relationship and interaction between different Tasks in a project. They can be classified as mandatory or discretionary; or internal or external.

There are numerous ways to identify, define, and present the Tasks and their dependencies. Two common methods involve the use of Product Flow Diagrams and Gantt Charts.

11.6 Task Estimation in the Sprint Planning Meeting

Task Estimation enables the Scrum Team to estimate the effort required to complete a Task or set of Tasks and to estimate the human and other resources required to carry out the Tasks within a given Sprint. In the Task Estimation Workshop, the Scrum Team members use the Task List to estimate the duration and effort for the User Stories to be completed in the Sprint.

One of the key benefits of this technique is that it enables the Team to have a shared perspective of the User Stories and requirements, so that they can reliably estimate the effort required. The information developed in the Task Estimation is included in the Estimated Task Effort List and it is used to determine the velocity for the Sprint.

In this workshop, the Scrum Team may use various techniques such as decomposition, expert judgment, analogous estimation, and parametric estimation. All the techniques in Chapter 10 can be used in this Workshop.

11.6 Estimation criteria

The primary objective of using Estimation Criteria is to maintain relative estimation sizes and minimize the need for re-estimation. Estimation Criteria can be expressed in numerous ways, with two common examples being Story Points and Ideal Time. For example, an Ideal Time normally describes the number of hours a Scrum Team member works exclusively on developing the project's Deliverables, without including any time spent on other activities or work that is outside the project. Estimation Criteria make it easier for the Scrum Team to estimate effort and enable them to evaluate and address inefficiencies when necessary.

12 CREATE SPRINT BACKLOG

12.1 Sprint Backlog

The Task List to be executed by the Scrum Team in the upcoming Sprint is called the Sprint Backlog.

It is common practice that the Sprint Backlog is represented on a Scrum Board or Task Board, which provides a constantly visible depiction of the status of the User Stories in the Backlog. Also included in the Sprint Backlog are any risks associated with the various Tasks. Any mitigating activities to address the identified risks would also be included as Tasks in the Sprint Backlog.

Once the Sprint Backlog is finalized and committed to by the Scrum Team, new User Stories should not be added. If new requirements arise during a Sprint, they will be added to the overall Prioritized Product Backlog and included in a future Sprint.

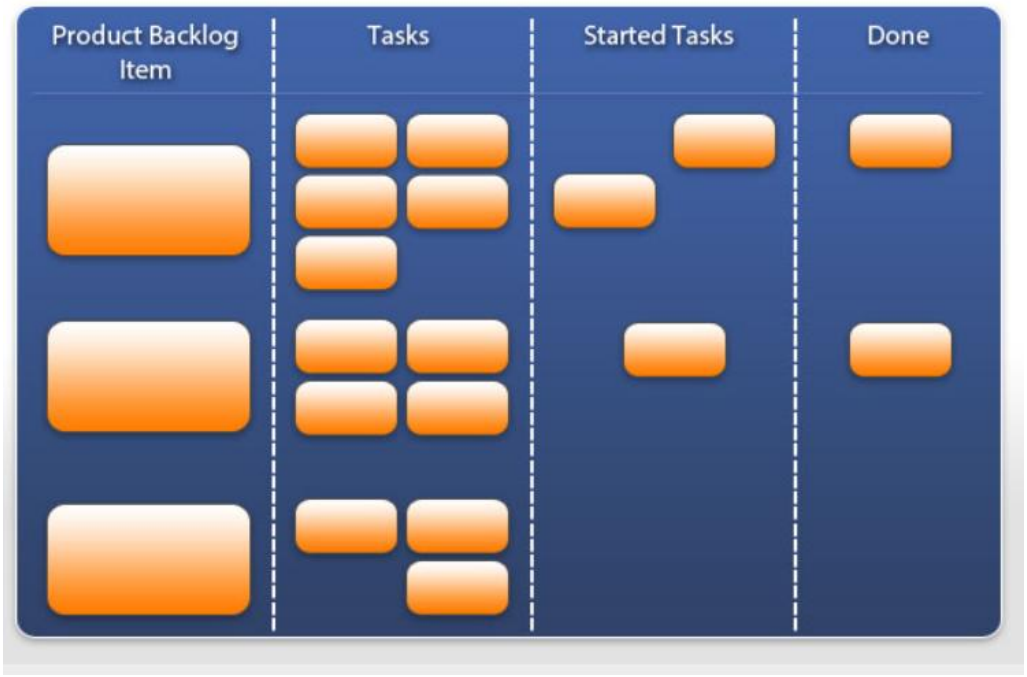


Figure: Example of Sprint Backlog. Product Backlog Item, Tasks, Started Tasks, Done.

12.2 Sprint Burndown Chart

The Sprint Burndown Chart is a graph that depicts the amount of work remaining in the ongoing Sprint. A Planned Burndown accompanies the initial Sprint Burndown Chart. The Sprint Burndown Chart should be updated at the end of each day as work is completed. This chart shows the progress that has been made by the Scrum Team and allows estimation errors to be discovered. If the Sprint Burndown Chart shows that the Scrum Team is not on track to finish the Tasks in the Sprint on time, the Scrum Master should identify any obstacles or impediments to successful completion and try to remove them.

A related chart is a Sprint Burnup Chart. Unlike the Sprint Burndown Chart which shows the amount of work remaining, the Sprint Burnup Chart depicts the work completed as part of the Sprint.

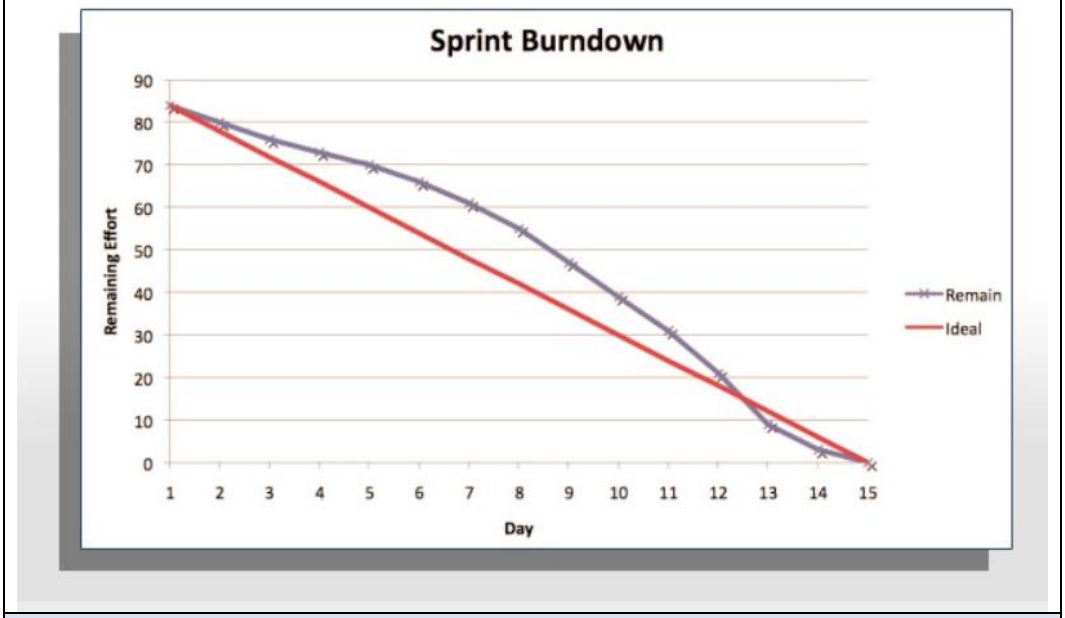


Figure: Example of Sprint Burndown chart. Sprint Burndown, Remaining Effort, Day. Remaining, Ideal.

The initial Sprint Backlog defines the start-point for the remaining effort. The remaining effort for all activities is compiled on a daily basis and added to the graph. In the beginning, the performance is often not as good as predicted by the ideal burndown rate, due to inaccurate estimation, or impediments that have to be removed in order to get up to full speed.

12.3 Sprint Velocity

Sprint Velocity is the rate at which the Team can complete the work in a Sprint. It is usually expressed in the same units as those used for estimation, normally Story Points or Ideal Time. A record of the Sprint Velocity of the Team for each Sprint is maintained and used as a reference in future Sprints. The Previous Sprint Velocity becomes the most important factor in determining the amount of work the Team can commit to in a subsequent Sprint. Any changes in the situation, or variations in conditions since the last Sprint must be taken into account to ensure accurate estimation of Sprint velocity for the upcoming Sprint.

Generally, velocity remains fairly constant during a development project, making velocity a useful metric for estimating how long it will take a Team to complete a software development project. If the Product Backlog has 300 Story Points, and the Team is averaging 30 Story Points per Sprint, it can be estimated that the Team will require 10 more Sprints to complete the work. If each Sprint lasts two weeks, then the project will last 20 more weeks. If a Team member is moved to another project, however, or new members are added, then the velocity must be recalculated.

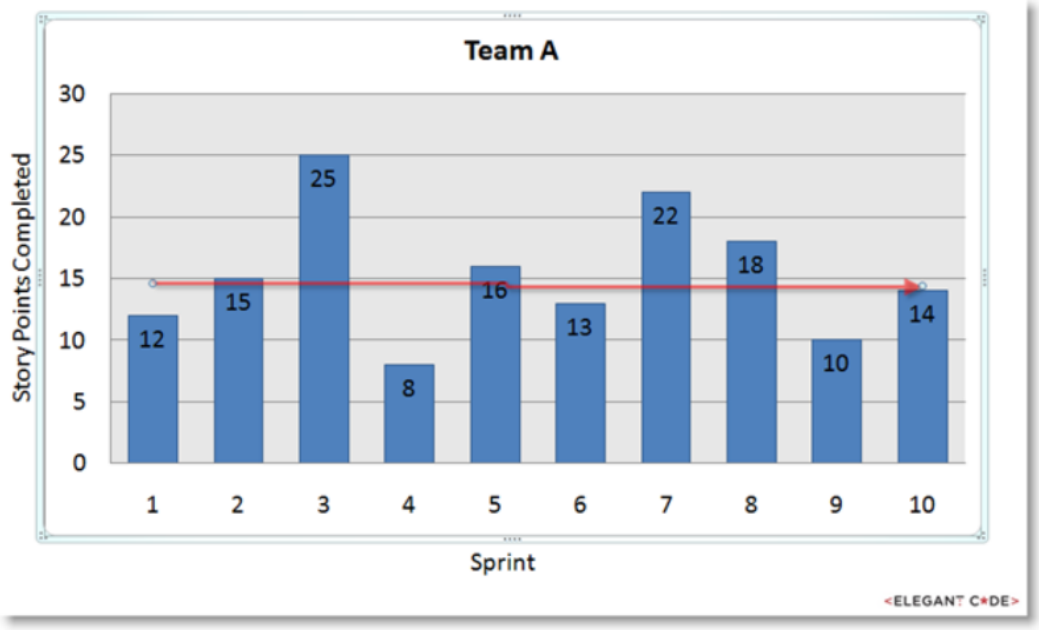


Figure. Example of Sprint Velocity Chart for Team A. Story Points Completed, Sprint.

13 DEMONSTRATE AND VALIDATE SPRINT

13.1 Sprint Review Meeting

The Scrum Core Team members and relevant Stakeholder(s) participate in Sprint Review Meetings to accept the Deliverables that meet the User Story Acceptance Criteria and reject those that do not. These meetings are convened at the end of every Sprint. The Scrum Team demonstrates the new features completed during the Sprint, including the new functionalities and product iterations. This provides an opportunity for the Product Owner and Stakeholder(s) to inspect what has been completed so far and to determine if any changes should be made in the project or processes in subsequent Sprints.

13.2 Accepted Deliverables

The Product Owner accepts those Deliverables which meet the User Story Acceptance Criteria. The objective of a Sprint is to create potentially shippable Deliverables, or product increments, which meet the Acceptance Criteria defined by the Customer and Product Owner. These are considered Accepted Deliverables that may be released to the Customer if desired. A list of Accepted Deliverables is maintained and updated after each Sprint Review Meeting. If a Deliverable does not meet the defined Acceptance Criteria, it is not accepted and will usually be carried forward into a subsequent Sprint to rectify any issues. Non-acceptance is an undesirable result since the objective of every Sprint is to produce Deliverables that meet the Acceptance Criteria.

13.3 Rejected Deliverables

Deliverables that do not meet the Acceptance Criteria will be rejected. The User Stories behind these rejections are returned to the Prioritized Product Backlog, and will be considered for inclusion in a future Sprint.

14 SHIP DELIVERABLES

14.1 What to ship

The Release phase focuses on delivery of the Accepted Deliverables to the Customer and on identifying, documenting, and internalizing the lessons learned during the project. Release is the relevant option for the following:

- Portfolios, programs, and/or projects in any industry
- Products, services, or any other results to be delivered to Stakeholders
- Projects of any size or complexity.

Scrum can be applied effectively to any project in any industry—from small projects or Teams with as few as six Team members to large, complex projects with up to several hundred Team members.

14.2 Working Deliverables Agreement

Deliverables that meet the Acceptance Criteria receive formal business sign-off and approval by the Customer or sponsor. To get formal Customer acceptance is critical for revenue recognition and the responsibility for obtaining it will be defined by the company policies and is not necessarily the responsibility of the Product Owner.

14.3 Working Deliverables

This output is the final shippable Deliverable for which the project was sanctioned. As new product increments are created, they are continually integrated into prior increments, so there is a potentially shippable product available at all times throughout the project.

14.4 Product Releases

The Product Releases should include the following:

- Release Content: This consists of essential information about the Deliverables for the use of the Customer Support Team

- Release Notes: These should include external or market-facing shipping criteria for the product to be delivered.

14.5 Piloting Plan

A Piloting Plan is an optional input that can be used to map out a pilot deployment in detail. The scope and objectives of the deployment, the target deployment user base, a deployment schedule, transition plans, required user preparation, evaluation criteria for the deployment, and other key elements related to the deployment are specified in the Pilot Plan and shared with Stakeholders.

15 RETROSPECTIVE

15.1 Retrospect Sprint

The Retrospect Sprint Meeting is an important element of the 'inspect-adapt' Scrum framework and it is the final step in a Sprint. All Scrum Team members attend the meeting, which is facilitated or moderated by the Scrum Master. It is recommended, but not required for the Product Owner to attend. One Team member acts as the scribe and documents discussions and items for future action. It is essential to hold this meeting in an open and relaxed environment to encourage full participation by all Team members. Discussions in the Retrospect Sprint Meeting encompass both what went wrong and what went right. Primary objectives of the meeting are to identify three specific items:

1. Things the Team needs to keep doing—**best practices**
2. Things the Team needs to begin doing—**process improvements**
3. Things the Team needs to stop doing—**process problems and bottlenecks**

These areas are discussed and a list of Agreed Actionable Improvements is created

15.1.1 Explorer—Shopper—Vacationer—Prisoner (ESVP)

This exercise can be conducted at the start of the Retrospect Sprint Meeting to understand the mindset of the participants and set the tone for the meeting. Attendees are asked to anonymously indicate which best represents how they feel regarding their participation in the meeting.

- **Explorer**—Wants to participate in and learn everything discussed in the retrospective
- **Shopper**—Wants to listen to everything and choose what he takes away from the retrospective
- **Vacationer**—Wants to relax and be a tourist in the retrospective
- **Prisoner**—Wants to be elsewhere and is attending the retrospective because it is required

The Scrum Master then collates the responses, prepares, and shares the information with the group.

15.1.2 Speed Boat

Speedboat is a technique that can be used to conduct the Retrospect Sprint Meeting. Team members play the role of the crew on a speedboat. The boat must reach an island, which is symbolic of the project vision. The attendees to record engines and anchors use sticky notes. Engines help them reach the island, while anchors hinder them from reaching the island. This exercise is Time-boxed to a few minutes. Once all items are documented, the information is collated, discussed, and prioritized by way of a voting process. Engines are recognized and mitigation actions are planned for the anchors, based on priority.

15.1.3 Metrics and Measuring Techniques

Various metrics can be used to measure and contrast the Team's performance in the current Sprint to their performance in previous Sprints. Some examples of these metrics include:

- **Team velocity**: Number of Story Points done in a given Sprint
- **Done success rate**: Percentage of Story Points that have been Done versus those Committed
- **Estimation effectiveness**: Number or percentage of deviations between estimated and actual time spent on Tasks and User Stories
- **Review feedback ratings**: Feedback can be solicited from Stakeholder(s) using quantitative or qualitative ratings, providing a measurement of Team performance

- **Team morale ratings**: Results from self-assessments of Team member morale

- **Peer feedback**: 360 degree feedback mechanisms can be used to solicit constructive criticism and insight into Team performance

- **Progress to release or launch**: Business value provided in each release, as well as value represented by the current progress towards a release. This contributes to the motivation of the Team and to the level of work satisfaction.

15.1.4 The outcome of the Retrospect Sprint Meeting

- **Agreed Actionable:** Improvements: Agreed Actionable Improvements are the primary output of the Retrospect Sprint process. They are the list of actionable items that the Team has come up with to address problems and improve processes in order to enhance their performance in future Sprints.

- **Assigned Action Items and Due Dates:** Once the Agreed Actionable Improvements have been elaborated and refined, action items to implement the improvements may be considered by the Scrum Team. Each action item will have a defined due date for completion.

- **Proposed Non-Functional Items for Prioritized Product Backlog:** When the initial Prioritized Product Backlog is developed, it is based on User Stories and required functionalities. Often, non-functional requirements may not be fully defined in the early stages of the project and can surface during the Sprint Review or Retrospect Sprint Meetings. These items should be added to the Prioritized Product Backlog as they are discovered. Some examples of non-functional requirements are response times, capacity limitations, and security related issues.

- **Retrospect Sprint Log**: The Retrospect Sprint Logs are records of the opinions, discussions, and actionable items raised in a Retrospect Sprint Meeting. The Scrum Master could facilitate creation of this log with inputs from Scrum Core Team members. The collection of all Retrospect Sprint Logs becomes the project diary and details project successes, issues, problems, and resolutions. The logs are public documents available to anyone in the organization.

- **Scrum Team Lessons Learned**: The self-organizing and empowered Scrum Team is expected to learn from any mistakes made during a Sprint. These lessons learned help the Teams improve their performance in future Sprints. These lessons learned may also be documented in Scrum Guidance Body Recommendations to be shared with other Scrum Teams.

  There may be several positive lessons learned as part of a Sprint. These positive lessons learned are a key part of the retrospective, and should be appropriately shared within the Team and with the Scrum Guidance Body, as the Teams work towards continuous self-improvement.

- **Updated Scrum Guidance Body Recommendations**: As a result of a Retrospect Sprint Meeting, suggestions may be made to revise or enhance the Scrum Guidance Body Recommendations. If the Guidance Body accepts these suggestions, these will be incorporated as updates to the Scrum Guidance Body documentation.

15.2 Retrospect Project

The Retrospect Project Meeting is a meeting to determine ways in which Team collaboration and effectiveness can be improved in future projects. Positives, negatives, and potential

opportunities for improvement are also discussed. This meeting is not Time-boxed and may be conducted in person or in a virtual format. Attendees include the Project Team, Chief Scrum Master, Chief Product Owner, and Stakeholder(s). During the meeting, lessons learned are documented and participants look for opportunities to improve processes and address inefficiencies.

Some of the tools used in the Retrospect Sprint process can also be used in this process. Examples include:

- Explorer—Shopper—Vacationer—Prisoner (ESVP) exercise
- Speed Boat
- Metrics and Measuring Techniques

15.2.1 The outcome of the Retrospect Project Meeting

- **Agreed Actionable Improvements**: Agreed Actionable Improvements are the primary output of the Retrospect Project Review. They are the list of actionable items that the Team has come up with to address problems and improve processes in order to enhance their performance in future Sprints.

- **Assigned Action Items and Due Dates:** Once the Agreed Actionable Improvements have been elaborated and refined, action items to implement the improvements may be considered by the Scrum Team or Stakeholders. Each action item will have a defined due date for completion.

- **Proposed Non-functional Items for Program Product Backlog and Prioritized Product Backlog**: When the initial Program Product Backlog or Prioritized Product Backlog are developed, they are based on User Stories and required functionalities. Often, non-functional requirements may not be fully defined in the early stages of the project and can surface during the Sprint Review, Retrospect Sprint or Retrospect Project Meetings. These items should be added to the Program Product Backlog (for the program) and Prioritized Product Backlog (for the project) as they are discovered. Some examples of non-functional requirements are response times, capacity limitations, and security related issues.