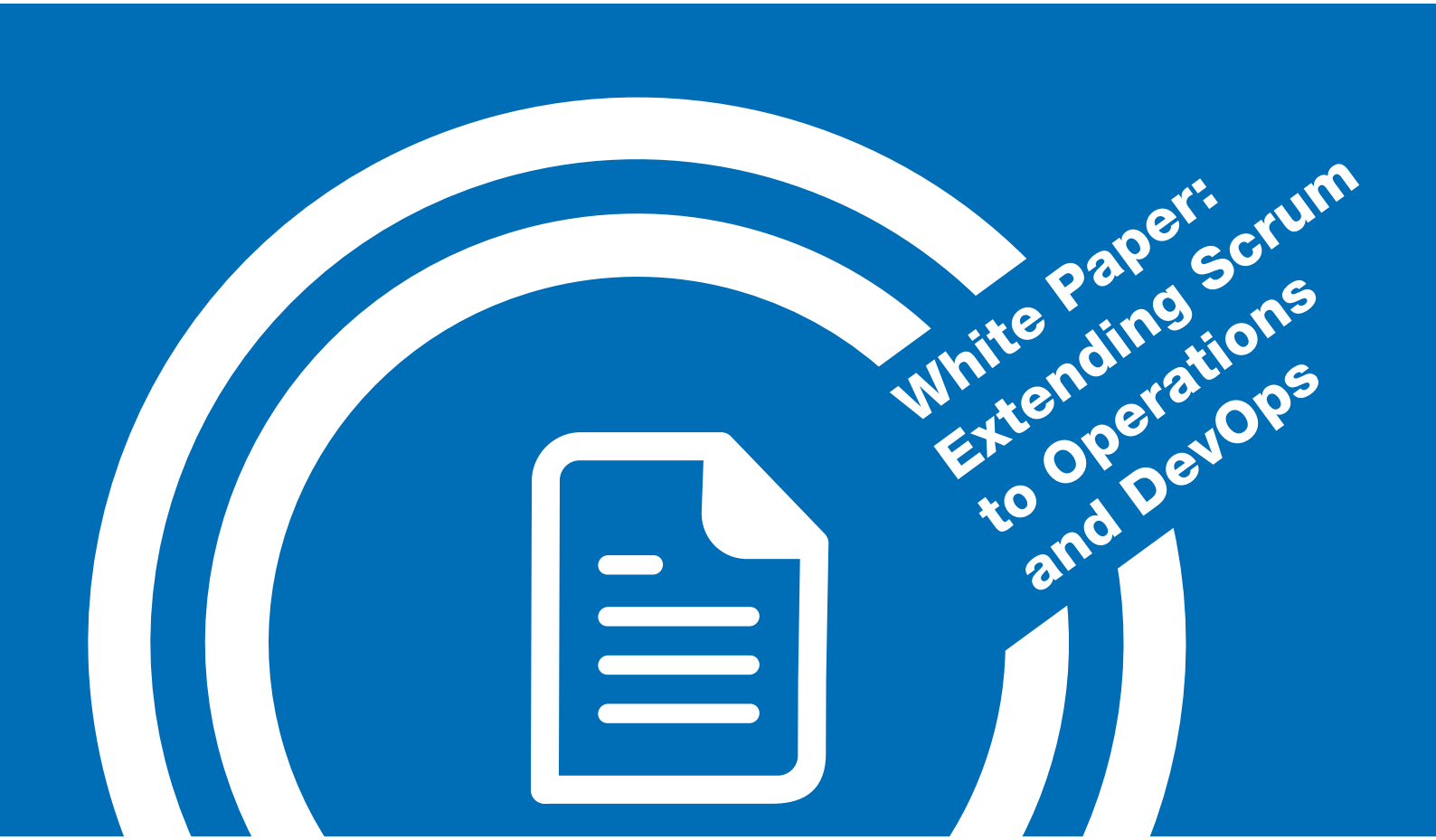
SCRUMstudy

Targeting success



White Paper: Extending Scrum to Operations and DevOps

WHITE PAPER: EXTENDING SCRUM TO OPERATIONS AND DEVOPS 2

1.1 Summary 2

1.2 Need for Agility in Development, Operations, and DevOps Environments 3

1.3 Business and IT Operations and Relevant Use Case Scenarios 4
1.3.1 Operational Work versus Project or Development Work 5

1.4 Extending the Product Backlog to Operational Work 5
1.4.1 Single Team versus Multiple Teams 7
1.4.2 Finite versus Recurring User Stories 9
1.4.3 Operational Work Packages and Tasks 12
1.4.4 Planning for Known and Unknown Operational Work 13
1.4.5 Specific Concepts and Considerations within IT Operations 13

1.5 Setting up an Organization to Extend Scrum to its Operations 17
1.5.1 Organization 17
1.5.2 Scrum Flow 18
1.5.3 Scrum Processes 19

1.6 Setting up an Organization for Extending Scrum to its Large-scale Operations 20
1.6.1 Organization 21
1.6.2 Scrum Flow 22
1.6.3 Scrum Processes 23

1.7 Setting up an Organization for Extending Scrum to its Development and Operations (DevOps) 28
1.7.1 Organization 29
1.7.2 Scrum Flow 30
1.7.3 Scrum Processes 31

1.8 Case Studies 31
1.8.1 Large Development Scrum Project (Multiple Team) 31

1.9 Conclusion 32

White Paper: Extending Scrum to Operations and DevOps

1.1 Summary

In an ever-evolving landscape of business management, the traditional bureaucratic and siloed management approach has become increasingly inadequate. This white paper explores how the Scrum framework can be extended to address the unique challenges within the business and IT operations and DevOps environments.

In this era of rapid software deployment and continuous integration, the convergence of development and operations teams within an organization, commonly referred to as DevOps, has gained prominence. To effectively manage this convergence, many organizations are turning to agile frameworks for a solution. The Scrum framework, with its iterative, collaborative, and customer-focused approach, is a comprehensive solution that addresses the challenges being faced by this integration.

This white paper delves into the fundamental principles, processes, and aspects of the Scrum framework and demonstrates how they can be extended to operations and DevOps environments. By fostering cross-functional teams, promoting frequent collaboration, and emphasizing the importance of continuous improvement, Scrum offers a structured framework that optimizes the flow of work, enhances the quality of products and services, and ensures the alignment of business goals with technical capabilities.

This paper also explores the key processes and activities involved in the Scrum framework, such as sprint planning, backlog management, daily stand-up meetings, and retrospective analyses; and demonstrates how the Scrum framework can provide flexibility to teams to streamline operational processes, increase transparency, and adapt to changing customer needs with agility.

By extending Scrum to operations and DevOps functions, organizations can aspire to achieve continuous improvement and innovation, ultimately enhancing their ability to respond to market demands and deliver value to their customers. This white paper aims to provide comprehensive insights to both novice and experienced practitioners looking to incorporate Scrum practices to not only optimize product development, but to also optimize business and IT operations, and DevOps practices within their organizations.

1.2 Need for Agility in Development, Operations, and DevOps Environments

Scrum has grown in application and has helped companies achieve success across numerous industries. Scrum has been scaled for small and large enterprise applications. Projects that involve the development of new products or services are considered "Development" or "Dev" projects. They generally have a defined beginning and end point. The application of the Scrum framework to projects is detailed in the *SBOK® Guide*.

The *SBOK® Guide* can be used as a reference and knowledge guide by both experienced Scrum and other product and service development practitioners, as well as by persons with no prior experience or knowledge of Scrum or project management approaches. The contents of the *SBOK® Guide* are organized for easy reference by the three Scrum Core Team roles—Product Owner, Scrum Master, and Scrum Team.

The chapters covering the six Scrum principles (chapter 2) and five Scrum aspects (chapter 3 through 7) include a Roles Guide. This guide provides direction regarding the relevance of each section in the chapter to the Scrum Core Team roles. Scrum is a framework that is not meant to be prescriptive, which means there is room for flexibility in its application. All the fundamental Scrum processes detailed in the *SBOK® Guide* (chapters 8 through 12) are required for every Scrum project but can be applied based on the specific needs of the organization, project, product, and/or team. Additional inputs, tools, and outputs apply when Scaling Scrum to Large Projects (chapter 13) and additional processes apply when Scaling Scrum to an Enterprise environment (chapter 14).

Besides developing new products and services, organizations also require operational support and maintenance in their production environments. These "Operational" or "Ops" functions were historically carried out by specialized Ops teams and resources worked differently than the organization's development teams. They also used different IT software/tools and frameworks for managing the organization's operational activities. However, these days, many organizations have incorporated Scrum practices to support their normal Operations (Ops) activities.

Many organizations have opted to combine Dev and Ops functions, aka DevOps, to integrate development and operational support activities while leveraging an Agile/Scrum framework. The SCRUMstudy framework extends Scrum practices from its use with traditional project development to apply to both Operations and DevOps functions.

Some of the following Use Cases may seem familiar.

Use Case A: Managing Operations—An IT Operations team is tasked with the responsibility of monitoring all IT customer support tickets, solving relevant issues within their scope of operations, and also escalating some issues to higher-level teams for further resolution.

Use Case B: Managing DevOps—A DevOps team works with multiple development teams to create the product pipeline. The DevOps team also ensures that products are tested properly in a test environment and effectively transitioned to the live production environment.

Use Case C: Managing Shared Resources—There may be shared resources in the company who do not work full-time for any particular Scrum Team, but are instead shared across multiple Scrum Teams. For example, a specialized Audit and Compliance team or Senior Architects who work with several or all the Scrum development teams on a part-time basis.

If any of these Operations or DevOps Use Cases sound familiar, consider applying the SCRUMstudy framework to these scenarios and others that are similar. When applying Scrum to Operations or DevOps, the Ops teams are divided into Scrum Teams, similar to traditional Scrum Dev project teams, and they are expected to adhere to the Scrum principles, aspects, and processes as defined in the *SBOK® Guide*. All teams within the company, including Dev and Ops teams, work in Sprints with their core teams which include Product Owners, Scrum Masters, and Scrum Teams. The only major change when apply the Scrum framework to Operations or DevOps involves how requirements for Dev and Ops are defined and managed in the Prioritized Product Backlog. The backlog should include all the use cases required for both the project development work and the operations work required within the company, including all the Dev and Ops functions.

This white paper details how the Scrum framework can be extended throughout an entire organization. In addition to delivering successful development projects, Scrum can also support an organization's operational requirements, such as operational tasks, DevOps, IT Support, and so on.

1.3 Business and IT Operations and Relevant Use Case Scenarios

To effectively extend Scrum to Ops and DevOps, a working knowledge of several operational concepts and definitions is essential.

"Operations" is a term that can be used to define a broad range of activities and in the context of business management, it can refer to day-to-day functions that keep a company running smoothly. Operational activities can vastly differ from one company to another, but they generally fall into two categories:

- **Business Operations**—these are all the administrative and business infrastructure maintenance activities that enable a company to keep everything running, such as activities pertaining to human resources, sales, purchasing, payroll, production, and manufacturing.
- **Information Technology (IT) Operations**—these are all the activities to manage, support, and maintain computer systems and devices, such as activities related to managing hardware, software, databases, and network components. IT operational deliverables are often recurring Sprint after Sprint for the life cycle of the IT application or service.

Business and IT operations can either be conducted for the performing organization (internal customers), or for external customers, such as for outsourced or contracted product/services. Scrum can be extended to all types of operations and, unless or otherwise specified in the *SBOK® Guide*, the term "operations" refers to both business and IT operations. A few examples of business operations and IT operations are provided in the table below.

Business Operations	• **Human Resources functions**—recruiting, hiring, terminations, annual performance reviews, quarterly reporting, etc. • **Payroll Processing functions**: bi-weekly employee payments, new employee payroll setup, processing payroll exceptions, etc. • **Manufacturing/Production functions**—Production of widgets for order fulfillment, weekly production reporting, equipment/tool maintenance, etc.
IT Operations	• **Enterprise Resource Planning (ERP) Software system support and maintenance**—add/remove user accounts, logging help desk support requests, addressing routine/minor incidents, usage reporting, performance reporting, etc. • **Mobile Application Support and Maintenance**—add/remove account access, customer billing support, analytics reporting, etc. • **Website Support and Maintenance**—address routine change requests, analytics reporting, etc.

1.3.1 Operational Work versus Project or Development Work

Operational work generally has a recurring or continuing element to it and involves repetitive activities or tasks. This typically includes providing maintenance or support for an established product or service function. This is also true for business operations, such as new employee hiring, financial reporting, payroll processing; as well as IT Operations, such as customer call center support, system monitoring and/or reporting, infrastructure operations, and performing routine procedures. Operational work can be predictable with set rules and procedures in place to address specific types of situations.

Project or development work produces a unique product or service and has a definite beginning and end. By nature, the work is less predictable, since it is a set of unique activities or tasks that pertain to the specific project being undertaken. Examples of project or development work would be creating a new software application or developing new functionality for an existing application.

1.4 Extending the Product Backlog to Operational Work

When extending Scrum to Ops and DevOps initiatives, the Prioritized Product Backlog continues to represent requirements captured in the form of Epics and User Stories that are prioritized by business value. The primary difference between operational and project work is the repetitive nature of operational requirements.

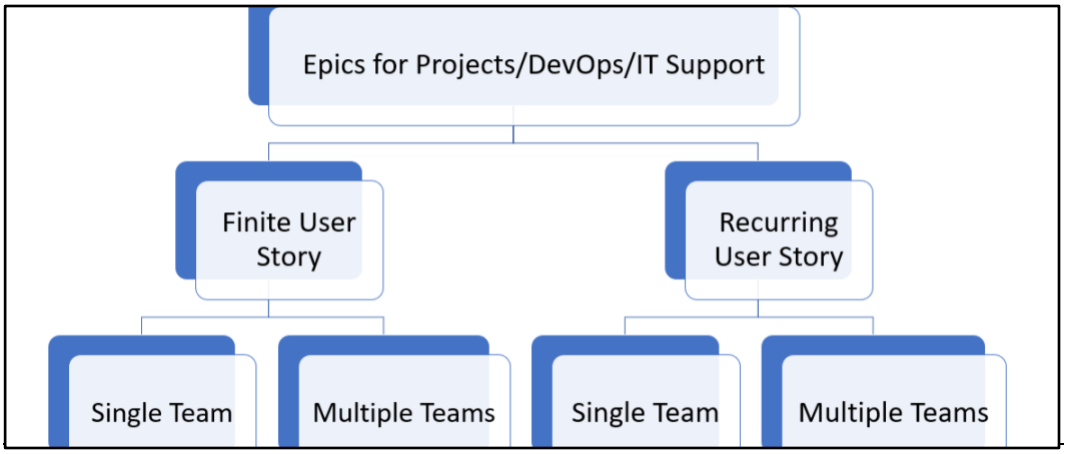
For this reason, ScrumOps User Stories are specified and can be recurring, meaning that they can be relevant to multiple Sprints. ScrumOps User Stories may require multiple ScrumOps teams to work to complete specific tasks.

In the *SBOK® Guide*, the Scrum framework is stated to be applicable to the management of portfolios, programs, and projects, as well as the development of products or services. When Scrum is applied to projects, which inherently have well-defined business objectives to be achieved through the unique

products or services the projects aim to create, User Stories are considered finite, since they have specific development deliverables that can be completed within the duration of a project, typically within a single Sprint. Hence, Scrum projects primarily consist of Finite User Stories (also referred to as User Stories) that are typically worked on by a single Scrum Team.

In the context of operations, the Scrum framework is extended to the entire lifecycle of the product or service. So, in addition to being applied to projects (used for product development), the Scrum framework is also applied to operational activities. In this environment, certain operational requirements of the product or service lifecycle that are recurring in nature are persistent across multiple Sprints and are referred to as Recurring User Stories.

To support ScrumOps, User Stories are broken down into the following categories:



The objective of the above classification is to ensure that the entire Scrum Team (which includes those working on projects and/or operations) works together and coordinates, wherever needed, to complete each User Story. This classification should also incorporate all Use Cases pertaining to operational tasks.

When creating a User Story, the Product Owner considers whether the User Story is either a finite User Story or a recurring User Story. The Product Owner then determines whether a single team or multiple teams is needed to complete that User Story. The default selection will be a finite User Story with a single team (as per how User Stories are defined in the *SBOK® Guide*), as User Stories applicable to projects, are meant to be finite and applicable to a single team (i.e., to be completed within one Sprint by one committed team).

If the User Story is not a finite User Story requiring only one team (i.e., the default), the Product Owner creating the User Story, should specify that the User Story is recurring and then specify whether it requires either a single team or multiple teams to complete the applicable work involved in that User Story. Otherwise, if not specified, the User Story will be considered to be a finite and single team User Story. Specifying a recurring User Story requiring multiple teams could be done through an additional label or category used for the User Story.

A User Story may start out as single team User Story, but if it is later determined that multiple teams are needed to work on it, then it can later be recategorized by the Product Owner as a multiple team User Story.

- A Sprint can have a combination of any of the four types of User Stories. For example, a Sprint can have several finite and single team User Stories and one recurring and multiple team User Story.

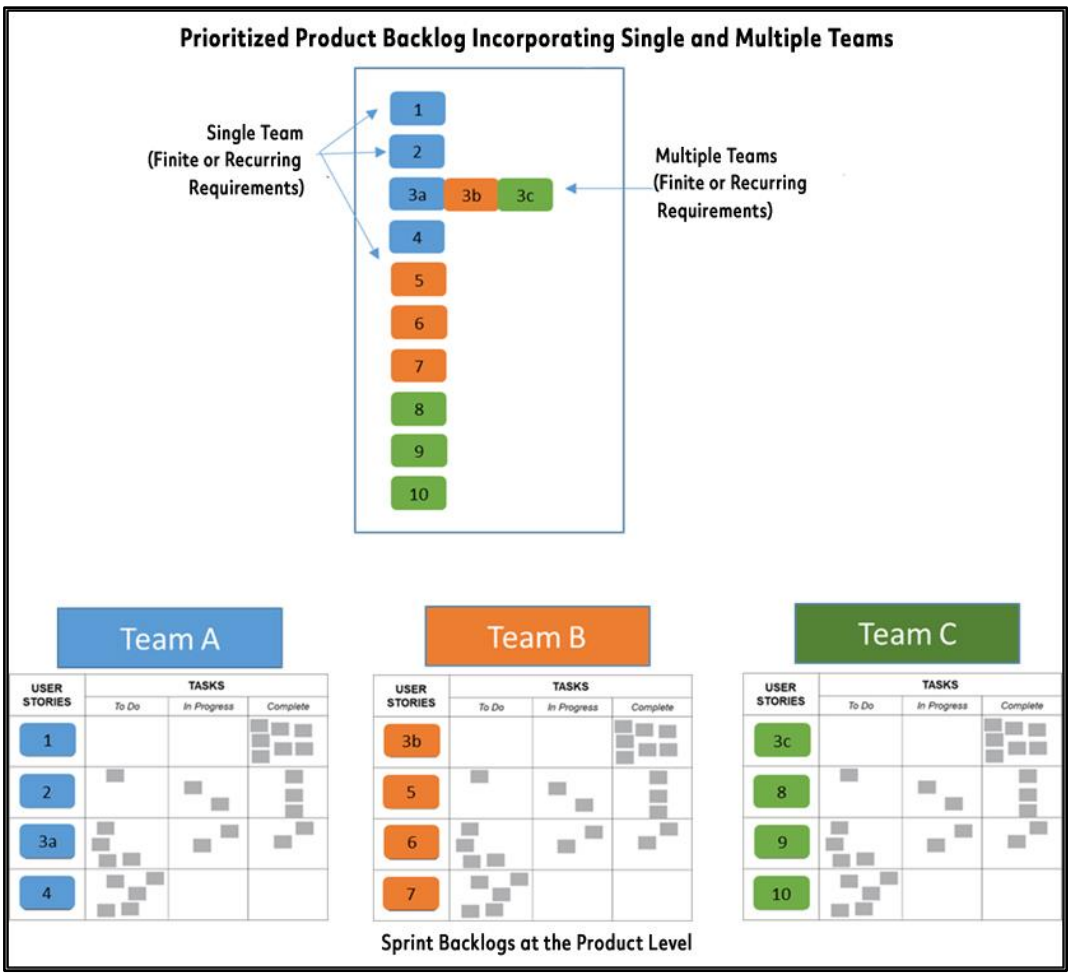
1.4.1 Single Team versus Multiple Teams

1.4.1.1 **Single Team**

A single team involves one team working on a single User Story. Single team User Stories are the most commonly used User Stories in Scrum projects and they should be used exclusively for all Dev projects. Unless otherwise specified, the term "User Story" refers to a single team with a finite User Story.

1.4.1.2 **Multiple Teams**

The use of multiple teams facilitates the sharing of resources among several Scrum teams. Multiple teams are used when more than one Scrum Team is needed to deliver a single User Story, or when there are one or more teams of specialists who perform a specific type of work needed for many User Stories (e.g., a team of Enterprise Architects or Risk Specialists, who provide their expertise across various Scrum Teams throughout the company). It is the responsibility of the Scrum Masters to ensure that all teams involved with completing a User Story are engaged at the appropriate times and work on their portions of each User Story. Typically, the Scrum Master for the Scrum Team that is performing most of the work for the relevant User Story takes the lead to ensure that all other teams working on the User Story are appropriately engaged. For multiple team User Stories, all the applicable teams can hold Scrum of Scrum meetings at appropriate time intervals to discuss the current status of each respective team's tasks and blockers, if any.



Single team implies that the User Story will be done by one Scrum Team. In this case, the User Story (and all the relevant tasks associated with it) will be completed by the same Scrum Team.

Multiple teams implies that the User Story will be done by multiple Scrum Teams. However, each User Story still has only one Product Owner assigned. Each team that will work on such a User Story commits to a portion of that User Story and identifies the relevant tasks necessary to create the deliverables associated with its portion of the User Story. In this case, it is important that these tasks include both the associated Dev and/or Ops tasks. The multiple team concept underlies the fact that some User Stories will require the work of multiple Scrum teams with the ability of the relevant teams to work together. Each team will create their own estimate for their own portion of the User Story. When portions of a User Story are committed to be delivered by one team, the work continues to be visible by other teams working on the other portion of the same User Story. Each team will have edit access for the tasks that its own team members have created and will have view access for the tasks created by other teams. Each Scrum Team estimates how much effort they will require to complete their own portion of the User Story.

The use of multiple teams allows the flexibility for a Dev team to also work on Ops activities during each Sprint. For example, a Scrum Team working on a Sprint can commit to assigning a certain percentage of its work effort towards supporting an existing application such as fixing bugs or monitoring the application. In such a case, the Scrum Team can commit to work on a User Story that's assigned to a multiple team, such as "Fixing Bugs in XYZ Website" during a Sprint, and can budget for certain number of hours or Story Points towards that User Story. So, if a bug comes up, the Dev Team can create appropriate tasks for that specific bug and work on that User Story, while at the same time continuing to work on the other commitments that are a part of the Sprint.

There is one Product Owner for each User Story who continues to provide the business requirements. In the case of User Stories that are finite, the Product Owner will sign off once each User Story is fully completed by the relevant team. In the case of User Stories that are recurring, the Product Owner will review the status and work done by the relevant teams in specific time intervals, and will continue to provide inputs as required.

The use of multiple teams also facilitates shared resources among the teams. User Stories that require multiple teams can also be used in situations where there are one or two people who do specialist work for multiple user stories across teams such as one Enterprise Architect or one Risk Specialist who need to provide their inputs for multiple User Stories which are primarily being done by multiple Scrum Teams inside the company. It is the responsibility of the Scrum Masters to ensure that all teams involved with completing a User Story get engaged at appropriate times and pick up their portions of the User Stories. Typically, the Scrum Master of the Scrum Team which is doing most of the work for the User Story would take a lead to ensure that all other teams required to complete the User Story are also engaged appropriately. If required, the Scrum Master may interface with the Product Owner for the User Story to ensure that all teams required for the User Story are engaged at appropriate times to complete the User Story.

1.4.2 Finite versus Recurring User Stories

1.4.2.1 **Finite User Story**

This is how a User Story is traditionally understood and is the same as how a User Story is defined in the *SBOK® Guide*. User Stories that are finite can be worked on in a single Sprint by one Scrum Team. User Stories are defined by the Product Owner (who may also be a Technical Architect or someone who heads an Ops function). They are committed to, and completed by the Dev/Ops/DevOps/IT Support teams as part of a Sprint and eventually approved by the Product Owner. User Stories that are finite involve work that is expected to be completed within a given time frame and therefore do not include any recurring work.

1.4.2.1.1 Single Team Finite User Story

This refers to one Scrum Team working exclusively on one finite User Story, such as a team creating deliverables for a new product feature, or a team setting up the test environment for a project. Single team User Stories that are finite are most effectively used for the development of new products or new features, and are therefore used with most Scrum development projects and other product development initiatives. Unless specifically clarified, the term User Stories will refer to single team finite User Stories. User Stories as discussed in the *SBOK® Guide* are referring to single team finite User Stories. It is preferable that each Scrum Team has all the required skill sets needed to perform the work required for the User Story in a Sprint. In this way, one single team has the capabilities and expertise to deliver the User Stories planned by them in each Sprint, with no dependencies on other teams.

1.4.2.1.2 Multiple Teams Finite User Story

This refers to multiple Scrum Teams working on one User Story that consists of specific deliverables that are finite in nature. For example, releasing a product feature may require the Scrum Team developing the product to work with a separate Ops team that will perform the work involved in releasing the product. Another example is a User Story that involves setting up the infrastructure for a project that will require work to be performed by multiple Scrum Teams (but still have only one Product Owner assigned). One Scrum Team may be procuring and setting up the hardware, a second Scrum Team may be procuring and setting up the software products, a third Scrum Team may be configuring the hardware and software, and a fourth team may be testing and finetuning the hardware to ensure everything works well.

With User Stories that require multiple teams to complete the associated work, tasks created by one team should be visible by the other teams and all the team members working on the User Story (even if they are a part of a different Scrum Team) should coordinate and communicate together to get the User Story and its relevant tasks completed. When multiple teams are assigned to a User Story, it is preferable to have the Sprints for all the teams aligned to have the same start and end dates. The Scrum Masters for the different teams will need to coordinate (e.g., through Scrum of Scrum (SoS) meetings, direct messaging, etc.) to ensure that all the teams are aligned to work together to deliver the User Story.

User Stories consisting of finite work that require multiple teams to complete the work should be used for Dev and Ops Scrum projects where it is impossible to have all skill sets to complete the User Story within one team. It is important to have as few teams involved with Scrum project initiatives as possible, so that dependencies between teams can be minimized and waiting and planning time is reduced.

1.4.2.2 **Recurring User Story**

This refers to User Stories that need to be delivered by a single or multiple teams over several Sprints, which may involve several months or years to complete the required work. Such User Stories typically include operational requirements. Examples include identifying and fixing bugs, support activities, monitoring applications, infrastructure optimization, process improvements, and so on. This type of recurring work typically needs to be performed if a product or application continues to be used.

User Stories that involve repetitive operational work can be chosen by any capable team to complete in a Sprint. However, these types of User Stories will not get completed at the end of just one Sprint, although the associated tasks are completed during the Sprint. A team that works on the tasks associated with these types of recurring User Stories may need to estimate the work effort required during the Sprint for the particular User Story and should be aware that all tasks in the User Story may not be known in advance. So, there is a possibility that some User Stories will take more or less time than what was estimated earlier during Sprint Planning. There will be one Product Owner for each recurring User Story. However, instead of approving User Stories at the end of each Sprint (as is done with the work performed for User Stories that are finite in nature), the Product Owner should be able to view the underlying tasks of User Stories at regular intervals, including their completion status, and provide his/her inputs to ensure that tasks in the User Story get completed as planned.

When Scrum is applied to operations work, the associated User Stories could include typical work requirements of operations teams. Examples include fixing bugs in an application, monitoring an application, resolving customer queries, and so on.

Epics are initial, high-level business requirements that are too large for a single Sprint and therefore need to be broken down into smaller, more refined User Stories. This definition of Epics still applies for operational User Stories that are recurring in nature. In this case, the set of tasks (also referred to as Work Packages) associated with an operational User Story should ideally be small enough to be completed within a single Sprint. A few examples of ScrumOps Epics and User Stories are illustrated below:

Example A: Human Resources Operational Services

- **Epic**—Fill open employment needs for the company with qualified candidates.
    - **Operational User Story #1**—As an [HR representative], I want to [recruit and hire full-time employees] so that [the department I represent will have the necessary skilled resources to fulfill company needs]. **Acceptance criteria:** Employment contract signed and completed for each specified job opening.
    - **Operational User Story #2**—As an [HR representative], I want to [recruit and hire part-time student workers] so that [the department I represent will have the necessary overflow resources to supplement their full-time staff]. **Acceptance criteria:** Employment contract signed and completed for each specified job opening.

Example B: Call Center Support Services

- **Epic**—Answer phone calls from customers during business operations hours
    - **Operational User Story #1**—As a [Call Center representative], I want to [assist customers with submitting support tickets into the company's incident reporting system] so that [issues or service requests can be tracked and resolved]. **Acceptance Criteria:** Successful resolution or escalation as per the company's service level agreements, procedures, and documentation requirements.
    - **Operational User Story #2**—As a [Call Center representative], I want to [assist customers with password reset requests] so that [customers can regain access to their services]. **Acceptance Criteria:** Customer successfully logs into the system or application with a new password.

In the examples above, the User Stories represent operational requirements that are recurring within the organization. Operational Epics/User Stories and their related Acceptance Criteria are created by the Product Owner and incorporated into the Prioritized Product Backlog.

Some types of ScrumOps User Stories will remain dormant in the Prioritized Product Backlog to be scheduled when appropriate and prioritized for a particular Sprint time period (e.g., only when there is a hiring need). In certain situations, ScrumOps User Stories are perpetually recurring in every Sprint.

1.4.3 Operational Work Packages and Tasks

In ScrumOps, User Stories represent general operational requirements that are persistent or recurring, while the underlying tasks capture the specific activities to be performed by Scrum Team members. For instance, a business requirement for monthly financial reporting for the comptroller's office would be considered an operational User Story that would be recurring in each Sprint time interval that coincides with the end of month reporting period. The related tasks would include creation of the same report at the end of each month. Often, there may be a set of the same tasks involved, for instance (1) finalizing data for the month, (2) generating a draft report, (3) and obtaining department approval. In these cases, the repetitive tasks can be standardized into Work Packages. Below is an example to further illustrate the concept of Work Packages pertaining to recurring operational work.

Example: An IT operational User Story for account access to an internal software system includes a Work Package with the following tasks:

1. Collect new user information.
2. Validate user information is complete and has authorization approval for new account.
3. Add new user account to system database.

Standardized Work Package templates can be defined and applied for repetitive sets of tasks. This can help predefine standard estimated effort and the approval steps involved. For example, a Scrum Team working on Customer Support requirements can create Work Package templates for common queries

that are received regularly, such as creating a new user account, or updating the name or email addresses of customers. Work Packages can be tracked and evaluated for performance over a period of time to identify process efficiency improvements.

Tasks in ScrumOps are treated the same as in traditional Scrum projects. Effort estimation of tasks in the Estimate Tasks process is optional, but recommended, and tends to be more accurate for operational tasks since they are repetitive in nature. Estimates can be made in Story Points or ideal time, based on the preference of the Scrum Core Team.

Templates for Epics and operational User Stories can also be created so that similar Epics and User Stories can be created easily. For example, a company that is involved with providing outsourced help desk services to other companies can define a template for creating new operational services. That template can then be used as a starting point for working on other Epics or User Stories for creating different types of operational services. Similarly, the company may create a template for an operational User Story related to monitoring and providing support to any website that is launched into production. Then that template can be used as a starting point to create operational User Stories for monitoring and providing support for specific websites. Similarly, work performed earlier by the team for previously completed Epics, User Stories, or tasks could be used as a template for similar User Stories that need to be delivered in the future.

1.4.4 Planning for Known and Unknown Operational Work

At the User Story level, operational work can be planned and committed for a Sprint. However, the underlying tasks or Work Packages may or may not be known at the beginning of the Sprint.

For example, Work Packages and/or tasks related to monitoring an application may be known in advance and can be defined as the User Story is broken down into specific tasks as part of the *Identify and Estimate Tasks* process. However, for a User Story related to answering call center support requests, the team would not be able to know how many requests, and what types of requests will be incoming at the beginning of the Sprint. The User Story for addressing support calls would be committed for the Sprint, but the specific tasks or Work Packages for the User Story would be unknown. Although the details are not known, an estimated amount of work would be committed in order to ensure that a team does not over commit or under commit for a Sprint. As calls come in during the Sprint, the specific tasks and details pertaining to the call would be incorporated into the operational User Story. These additional task details are not considered changes to the Sprint since they are considered essential for fulfilling the operational User Story requirements and the amount of work required had already been committed. Also, based on the actual work done in the Sprint for the tasks or Work Packages, better estimates can be provided in future Sprints with similar tasks or Work Packages.

1.4.5 Specific Concepts and Considerations within IT Operations

Important concepts and considerations specific to IT operational deliverables are provided below.

- **Incident Severity Levels**—This concept is relevant for operational User Stories involving IT production system monitoring and support. The types of system issues identified or reported are considered "incidents" and are categorized by the following severity levels:

Severity 1 (SEV1)	A critical incident with very high production impact
Severity 2 (SEV2)	A major incident with significant impact
Severity 3 (SEV3)	A minor incident with low impact
Non-production Defect	No impact to production

  Work Package templates can be created for each incident severity level to outline the standard set of tasks involved in addressing the incident. This can include tasks for investigation, resolution, testing, documentation, and so on.

- **Service Level Agreements (SLAs)** —IT production services typically have pre-established timeline targets or obligations for resolving support requests or incidents. These timeline targets can range from informal agreements to contractual obligations in the form of a Service Level Agreement (SLA). In the case of contracted IT production monitoring or maintenance support for a product or service, the SLA is a legal agreement to provide specified service expectations within established time limits. Each severity level can have associated SLAs. For example, SEV1 Work Packages or tasks need to be resolved within 6 hours, SEV2 within 24 hours, etc.

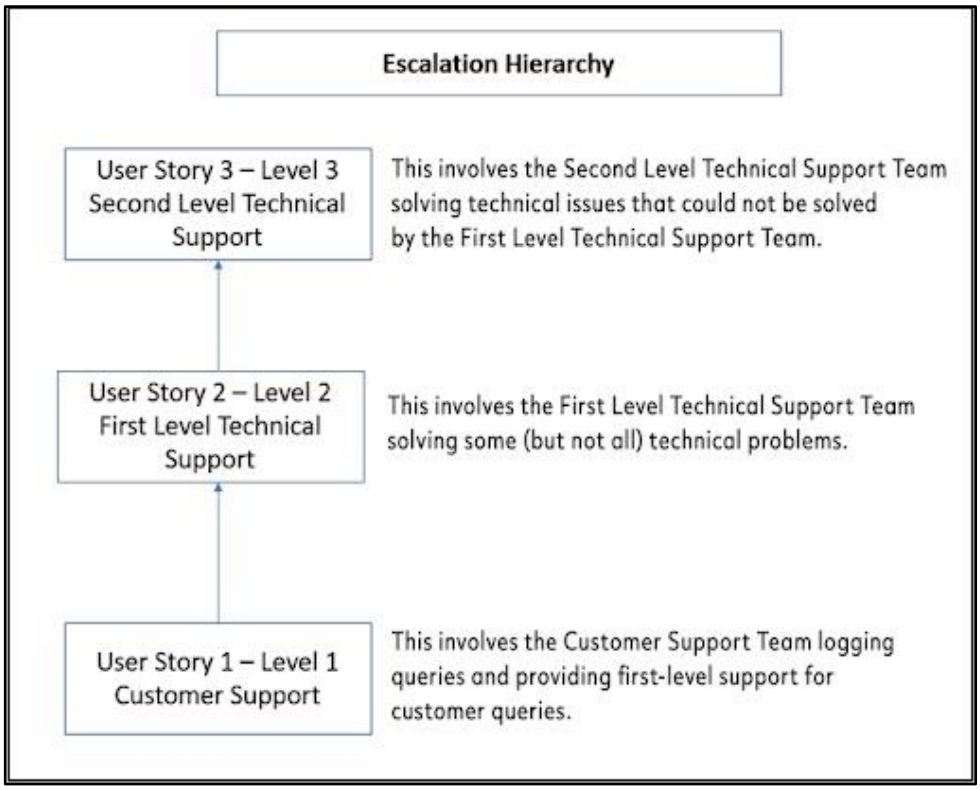
- **Swarming**—This is not strictly an IT operations concept; however, the term is most commonly associated with technical support situations where an urgent or critical requirement triggers involvement of as many team members as possible to work simultaneously on the same priority item until it is resolved. In the case of multiple Scrum Teams, there may be team members from more than one team involved in swarming.

- **Escalations**—In Scrum, tasks are usually completed by one Scrum Team that is assigned to complete a User Story. However, some operational User Stories may allow for Work Packages that can be escalated. Such User Stories should mandatorily be set up to allow for multiple teams to work on them. In this case, any Work Package that allows for escalations, should provide a link to the User Story that the Work Package can be escalated to. It is important to note that although Work Packages can be escalated, tasks within the Work Package are still defined and completed by Scrum Team members who are working on the User Story.

When defining operational User Stories, the Product Owner should have the ability to specify if escalations are permitted. In this case, if a Work Package cannot be completed by team members working on the current User Story, then the team can escalate the Work Package to another User Story involving other team members who are better able to handle it. The Work Package escalated could spawn off new tasks for another User Story team to which the Work Package was escalated to but will

still be linked to the initial Work Package created by the team that escalated the Work Package. The history of Work Packages and their associated tasks are also identified and documented so this is visible to all teams involved in the escalation. Thus, in this way, the team that escalated the Work Package can follow through to ensure that the escalated Work Package is being addressed properly by the team to which the Work Package was escalated to.

ScrumOps can allow for multiple levels of escalations as depicted in the diagram below:



Note that a Work Package from an operational User Story can only be escalated to another operational User Story if the User Story is in an ongoing Sprint. If the escalated Work Package is not complete and the Sprint ends, the corresponding operational User Story should be transferred to the next Sprint.

At the User Story level, the Product Owner should have the ability to define if the Work Packages under that User Story will have SLAs as defined at the project level or if they will not have any SLAs.

A Work Package can only be escalated to another operational User Story if both User Stories have the same SLA definitions (i.e., meaning that both User Stories should either have SLAs allowed or not allowed). Therefore, if a User Story has project-level SLAs defined, then Work Packages from other User Stories without any SLAs defined cannot be escalated to this User Story. Likewise, if a User Story does not have any SLAs defined, then Work Packages from other User Stories with SLAs cannot be escalated to this User Story.

Once a Work Package is escalated to another operational User Story, the team can then create new tasks for that Work Package. The Scrum Team working on the User Story to which the Work Package

was escalated to, may add their own estimates. However, the SLA for the task (if any) remains the same throughout the entire escalation process. For example, there could be one operational User Story that states “First Level Support for Customer Queries” that has the objective of logging customer queries and providing first-level support towards understanding and solving the query. There might be an associated Work Package1 that states “Update Customer Address,” which is relevant to when a customer is facing issues when updating his/her address. Tasks associated with this Work Package1 may be as follows:

Task 1.1—Check if customer is valid
Task 1.2—Check if address is valid
Task 1.3—Update new address
Task 1.4—Inform customer that address has been updated

If there is an issue with updating the address that cannot be solved by the First Level Customer Support Team, then the Work Package “Update Customer Address” can be escalated to the Second Level Technical Support Team. In this case, there could be a second operational User Story, such as “Provide Level 2 Technical Support”. Once the Work Package is escalated, the Second Level Technical Team can create new tasks such as:

Task 2.1—Check if address is valid
Task 2.2—Determine if database connections are working properly
Task 2.3—Fix any errors in the address update program
Task 2.4—Update address

It is important to note that even though the Work Package remains the same, the tasks identified and performed by the two teams are different. Once the Second Level Customer Support Team successfully completes the task “Escalation1—Update Customer Address,” the First Level Customer Support Team gets notified and they can then complete any pending work for the Work Package “Update Customer Address” and then mark the Work Package as complete. For example, they can then perform the Task 1.4—Inform Customer that the address has been updated.

Depending on the Work Packages, if required, the same User Story could be escalated to different User Stories (and each User Story could be worked on by different Scrum Teams). For example, it may be defined in the company SLAs that any website issues will be escalated to an operational User Story “Second Level Technical Support for Website” and would be assigned to a Scrum Team that is responsible for managing Website issues. Similarly, any mobile application issues would be escalated to another User Story “Second Level Technical Support for Mobile Applications” and would be assigned to a Scrum Team that is responsible for managing mobile applications. Finally, any issues related to social media accounts would be escalated to a third User Story “Second Level Technical Support for Social Media Accounts” and would be assigned to a Scrum Team that manages Social Media accounts. Therefore, depending on the type of Work Package issue identified by the First Level Customer Support

Team, the Work Packages would be escalated to the appropriate User Story to be taken on by the relevant team.

1.5 Setting up an Organization to Extend Scrum to its Operations

In the most straight-forward application of Scrum to an operation environment (referred to as ScrumOps), there would be one Product Owner, one Scrum Master, and one Scrum Team. In this context, there would be a single Prioritized Product Backlog containing operational Epics and User Stories to be carried out by the Scrum Core Team on behalf of an internal or external customer, product, or service. This would be considered a small-scale ScrumOps setup which could expand to up to three Scrum Teams, still with only one Product Owner and one Scrum Master. Beyond three Scrum Teams, additional inputs, tools, and outputs are recommended and would then be considered a large-scale ScrumOps, which would be similar to the setup of Scrum for Large Projects involving development work.

1.5.1 Organization

The roles and responsibilities for ScrumOps are essentially the same as when the Scrum framework is used for project development. However, there are a few additional considerations and complexities that are discussed in this section.

The core roles of Product Owner, Scrum Master, and Scrum Team remain unchanged when Scrum is applied to operational work.

- **Product Owner**—The Product Owner is the person responsible for maximizing business value. He or she is responsible for articulating customer requirements and maintaining business justification for the operational functions. The Product Owner is the *Voice of the Customer*, which can represent an internal or external customer, a product, or a service. The Product Owner also defines the Acceptance Criteria and is responsible for accepting or rejecting the operational deliverables.

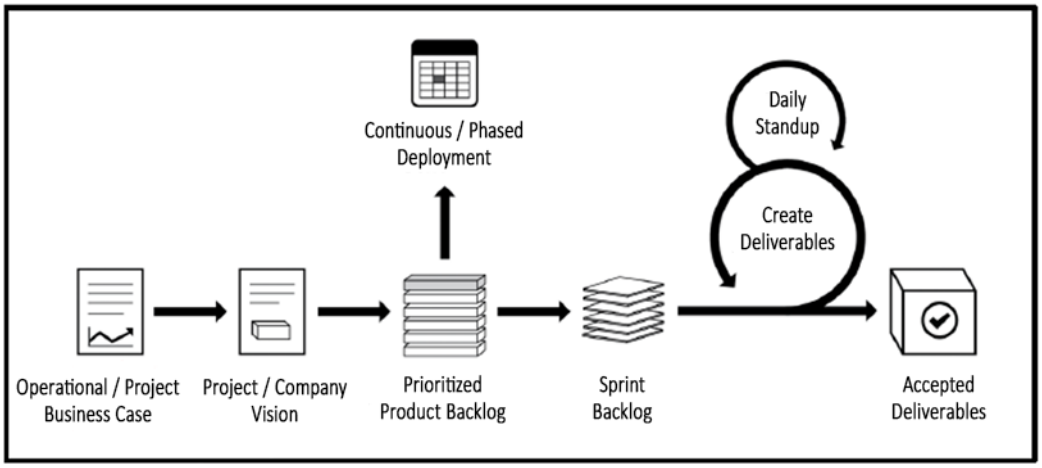
- **Scrum Master**—The Scrum Master is a facilitator who ensures that the Scrum Team is provided with an environment conducive to completing the operational activities successfully. The Scrum Master guides, facilitates, and teaches Scrum practices to everyone involved; clears impediments for the team; and ensures that Scrum processes are being followed.

  Scrum Team—The Scrum Team is the group of people who are responsible for understanding the business requirements specified by the Product Owner, estimating User Stories, and creating the operational deliverables. A Scrum Team typically consists six to ten members and possesses all the essential skills required to deliver the operational deliverables.

1.5.2 Scrum Flow

For small-scale ScrumOps, all the standard meetings and artifacts would apply as part of the fundamental Scrum Flow. Sprint Planning, Daily Standup, Sprint Review, and Retrospect Meetings are conducted, as previously defined in the *SBOK® Guide*. Due to the repetitive and predictable nature of operations, the length of Sprints can be longer than Sprints used for projects. Release Planning may not be as relevant, since templated Work Packages and standardized processes may allow pre-defined approval or Product Owner sign-off to deploy operational deliverables to customers and/or end users. Releases, or delivery to production, may not necessarily coincide with the end of a Sprint, therefore, ScrumOps allows for a more continuous deployment model when applied to operational deliverables.

Even though operational deliverables are generally ongoing, operational goals and targets are typically tied to fixed time periods, such as quarterly, bi-annually, or annually. The Project Vision Statement in ScrumOps represents these operational goals for the specific time period.



1.5.3 Scrum Processes

In the most simplistic ScrumOps situation with one Product Owner, one Scrum Master, and one Scrum Team, the fundamental Scrum processes described in chapters 8 to 12 of the *SBOK® Guide* remain essentially unchanged. However, there are additional considerations that are highlighted below for each process. Although this section is for single Scrum Team situations, this general process approach may be applied to situations with one Product Owner, one Scrum Master, and up to three Scrum Teams.

The tables below highlight a summary of impacts (if any) when the Scrum processes (as defined in the *SBOK® Guide*) are extended to the work being completed in an operations environment.

Initiate	Summary of Impacts
8.1 Create Project Vision	The vision statement would include operational objectives and goals.
8.2 Identify Scrum Master and Business Stakeholder(s)	No change.
8.3 Form Scrum Team	No change.
8.4 Develop Epic(s)	Epics would reflect operational requirements.
8.5 Create Prioritized Product Backlog	Epics, or high-level User Stories, would reflect operational requirements.
8.6 Conduct Release Planning	Release planning may be less relevant for continuous deployment situations.

Plan and Estimate	Summary of Impacts
9.1 Create User Stories	User Stories created for ScrumOps would define operational requirements, which are generally recurring and continuous.
9.2 Estimate User Stories	For operational User Stories, historical estimates would be leveraged to reflect known and unknown work.
9.3 Commit User Stories	No change for the commitment of known work. For unknown work, a certain amount of work is committed or allocated for the Sprint. The overall amount of work committed for a Sprint matches the team's velocity.
9.4 Identify Tasks	Since recurring User Stories typically involve repetitive tasks, standardized Work Package templates can be used to facilitate the identification of tasks for example, when monitoring an application or performing similar recurring work, the associated Work Package(s) normally are already known and well defined and can be taken from an existing template. Unknown work needs to be addressed when the respective trigger (e.g., an incident) happens. The relevant Work Package template can be applied for the team to execute during the Sprint.
9.5 Estimate Tasks	Estimates for repetitive tasks for recurring User Stories may be leveraged from historical data.
9.6 Update Sprint Backlog	No change.

Implement	
10.1 Create Deliverables	No change.
10.2 Conduct Daily Standup	When an issue occurs during a Sprint (unknown work), the work related to the handling of that incident will become the predominant topic of the Daily Standup Meeting. Otherwise, this process is applied as described in section 10.2 of the *SBOK® Guide*.
10.3 Refine Prioritized Product Backlog	In ScrumOps, the recurring User Stories definition is usually stable and fixed; however, priorities may change and will need to be refined. The volatility is primarily at the task level instead of at the User Story level.

Review and Retrospect	
11.1 Demonstrate and Validate Sprint	Sprint deliverables are demonstrated to the Product Owner and either approved or rejected. Any incomplete or rejected Work Packages remain associated to a recurring User Story to be included in the planning for the next Sprint. At times, the Product Owner may approve or reject deliverables during the Sprint, particularly in the case of continuous deployment.
11.2 Retrospect Sprint	No change. An additional consideration would be to evaluate Work Package templates for improvements.

Release	
12.1 Ship Deliverables	This process is used the same way as described in section 12.1 of the *SBOK® Guide*, but special attention to operational support should be given.
12.2 Retrospect Release	This process is used the same way as described in section 12.2 of the *SBOK® Guide*. In addition to normal topics of any Retrospect Release Meeting, the team should pay attention to potential issues with recurring User Stories or Work Packages.

1.6 Setting up an Organization for Extending Scrum to its Large-scale Operations

In practice, operational deliverables captured in a single Prioritized Product Backlog may at times require four or more functional Scrum Teams due to the size of the operational initiatives. The specific operational requirements and functional organization structure can result in a combination of single and/or multiple Product Owners, Scrum Masters, and Scrum Teams.

1.6.1 Organization

For large operational initiatives requiring the efforts of four or more Scrum Teams with multiple Product Owners and Scrum Masters, the additional roles of Chief Product Owner and Chief Scrum Master are required.

- **Chief Product Owner**—This role helps to resolve any conflicts in prioritization when multiple Product Owners are involved. One of the Product Owners can take on the role of Chief Product Owner, or this role can be assigned to a different person, often a higher-level executive, to make priority judgement calls only when conflicts arise.

- **Chief Scrum Master**—This role aids cross-team coordination efforts when multiple Scrum Masters and Scrum Teams are involved. Typically, one of the Scrum Masters is identified to take on the role of Chief Scrum Master and is responsible for facilitating the Scrum of Scrums Meetings for overall coordination among Scrum Teams.

The additional considerations related to multiple Product Owners, Scrum Masters, and Scrum Teams are described below:

- **Multiple Product Owners**—Each Product Owner is responsible for articulating the requirements and priorities for a specific internal or external client, product, service, or domain. For this reason, each User Story captured in the Prioritized Product Backlog for the large operational initiative would have one Product Owner. Each Product Owner would therefore own a sub-set of the requirements in the overall Prioritized Product Backlog. Since there are multiple Product Owners working with a single Prioritized Product Backlog, a Chief Product Owner role (as described earlier) is essential to resolve any conflicts in prioritization. One of the Product Owners can take on the role of Chief Product Owner, or this role can be assigned to a different person, often a high-level executive, to make priority judgement calls only when conflicts arise.

  Multiple Scrum Masters—This is typically required when multiple Scrum Teams are involved. Dedicated Scrum Masters are in place to support each team or set of teams. A Chief Scrum Master role (as described earlier) is essential for cross-team coordination efforts and to facilitate Scrum of Scrums Meetings. Typically, one of the Scrum Masters is identified to take on the role of Chief Scrum Master. Sprint schedules should preferably be aligned, so that all Scrum Teams have the same start and end dates for their respective Sprints.

- **Multiple Scrum Teams**—In ScrumOps, Scrum Teams often represent a specific set of functional skills and can be somewhat less diverse than project Scrum Teams. This is due to the repetitive and predictable nature of operational User Stories that allow a narrower definition of skill sets required by the team. When multiple Scrum Teams are involved, coordination of activities and dependencies between teams is critical. Scrum Teams commit to deliver User Stories in the Prioritized Product Backlog for each Sprint. At times, a Scrum Team or team member can commit

to complete specific tasks or Work Packages within an operational User Story. Hence, multiple Scrum Teams may be working on a single User Story.

- **Shared Resources**—At times, resources may be shared between multiple teams. This is typically due to limited resource specialists that have a specific set of skills and expertise required by multiple teams. For example, an Enterprise Architect or Risk Specialist who provides input for multiple User Stories being worked on by Scrum Teams within the company. It is the responsibility of the Scrum Masters to ensure that all teams completing the required work for a User Story are engaged at appropriate times. Typically, the Scrum Master for the Scrum Team that is performing most of the work for the User Story takes the lead and ensures that all other Scrum Teams working on the User Story are appropriately engaged. If required, the Scrum Master may interface with the Product Owner to ensure that all teams required to complete the Work Packages associated with the User Story are engaged at appropriate times to effectively complete the User Story deliverables.

1.6.2 Scrum Flow

For large-scale operational initiatives involving multiple Product Owners, Scrum Masters, and Scrum Teams, additional meetings and artifacts are required to coordinate the work efforts in the overall Prioritized Product Backlog. Many of the additional inputs, outputs, and tools from chapter 13, Scaling Scrum for Large Projects in the *SBOK® Guide* would apply to large-scale operational initiatives, primarily due to the added organizational complexity.

The following additional artifacts and tools used for large Scrum projects are relevant for large-scale operational initiatives and are applied in much the same way.

- 13.2.2 Product Owners Collaboration Plan
- 13.2.3 Scrum Masters/Scrum Teams Collaboration Plan
- 13.2.4 Shared Resources
- 13.2.5 Team Specialization
- 13.2.6 Environment and Environment Schedule
- 13.3.1 Large Project Communications Plan
- 13.3.2 Large Project Resource Planning
- 13.3.3 Environment Identification
- 13.3.4 Prioritized Product Backlog Assignment
- 13.3.5 Scrum of Scrums (SoS) Meeting
- 13.3.8 Scrum Project Tool

Scrum Project Tool—When multiple Product Owners, Scrum Masters, and Scrum Teams are working with one overall Prioritized Product Backlog, it is essential to have a Scrum Project Tool or process in place that allows visibility to all streams of work while allowing individual Scrum Teams and team members to view and edit progress on their tasks during a Sprint. All team

members working on the User Story (even if they are part of different Scrum Teams) should be able to coordinate and communicate together to get the Work Packages and associated tasks in the User Story completed.

The Scrum Project Tool should allow Work Packages and their associated tasks to be categorized by type, such as issues, bugs, defects, monitoring, reporting, etc. Additional categories may also be configured to provide insights when tracking Work Packages and tasks for improvement opportunities. These categories can include information such as task creator or submitter, severity, and so on.

1.6.3 Scrum Processes

In the case of large operational initiatives involving four or more Scrum Teams, many of the additional Scrum inputs, tools, and outputs described in chapter 13, Scaling Scrum for Large Scrum Projects *SBOK® Guide*, would apply. This section highlights the specific ScrumOps considerations for large-scale operational initiatives as it relates to the fundamental Scrum processes within each work phase.

Initiate	Summary of Impacts
8.1 Create Project Vision	**Updated Output: Identified Product Owner*** (see section 8.1.3.1)—This process would be updated to identify multiple Product Owners. **Additional Output: Identified Chief Product Owner*** (see section 3.7.2.1)—When multiple Product Owners are involved, a Chief Product Owner should be identified. **Additional Output: Product Owners Collaboration Plan*** (see section 13.2.2) —For large operational initiatives, it is essential for the entire team of Product Owners to collaborate and embrace Scrum processes to successfully deliver the associated Work Packages in the User Story.
8.2 Identify Scrum Master and Business Stakeholder(s)	**Updated Output: Identified Scrum Masters*** (see section 8.2.3.1) —This process would be updated to identify multiple Scrum Masters. **Additional Output**: **Identified Chief Scrum Master*** (see section 3.7.2.2)—A Chief Scrum Master is identified with the role of focusing on multi-team interaction and synchronization. **Additional Output: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3)—The Scrum Masters/Scrum Teams Collaboration Plan defines how the Scrum Masters and Scrum Teams participate in the refining of the Prioritized Product Backlog. This plan should also define which team representatives should be involved in the refining process, and how they will be selected. **Additional Output: Shared Resources*** (see section 13.2.4)—Knowledge of any shared resources available to the Scrum Teams would be a necessary input in forming the individual Scrum Teams.

	Updated Output: Identified Supporting Services (see section 3.3.2)—Some additional supporting services may be needed to coordinate activities between all Product Owners, Scrum Masters, and Scrum Teams.
8.3 Form Scrum Team	**Additional Input: Chief Product Owner*** (see section 3.7.2.1) **Additional Input: Chief Scrum Master*** (see section 3.7.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3) **Additional Input: Team Specialization*** (see section 13.2.5)—Some Scrum Team members may require specialized skills to work on specific issues related to large scale operations. **Additional Tool: Multi-team Communications Plan*** (see section 13.3.1)—This plan highlights how to manage effective communication between multiple teams. **Additional Tool: Multi-team Resource Planning*** (see section 13.3.2)—This plan helps manage the complexity of allocating various types of resources to the multiple Scrum Teams working in parallel. **Additional Tool: Environment Identification*** (see section 13.3.3)—In large projects, it is important to identify the number and types of environments needed because multiple Scrum Teams will be working simultaneously to carry out the work in their respective Sprints. **Additional Optional Tool: Scrum Tool** (see section 13.3.8)—A Scrum Tool helps facilitate communications and coordination between multiple teams. **Additional Output: Environment and Environment Schedule*** (see section 13.2.6)—After environments are identified, an Environment Schedule is created which is used for the coordination of Sprint activities across teams. **Additional Output: Updated Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3)—As Scrum Teams are formed,

	inputs from the teams and additional considerations will result in updates to the Scrum Masters/Scrum Teams Collaboration Plan.
8.4 Develop Epics	**Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3) **Additional Optional Tool: Scrum Project Tool** (see section 13.3.8)
8.5 Create Prioritized Product Backlog	**Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Tool: Prioritized Product Backlog Assignments*** (see section 13.3.4)—Prioritized Product Backlog Assignments ensure effective allocation of Epics and User Stories to all the Product Owners.
8.6 Conduct Release Planning	**Additional Input: Chief Scrum Master*** (see section 3.7.2.2) **Updated Output: Release Planning Schedule** (see section 8.6.3.1)—To reflect the importance of operational aspects, the Release Plan could also include information about any support required from the Scrum Team after a release. **Additional Output: Release Readiness Plan*** (see section 13.2.7)—The Release Readiness Plan includes specific activities that need to be performed shortly before release planning.

Plan and Estimate	Summary of Impacts
9.1 Create User Stories	Epics are decomposed into User Stories, as described in chapter 9 of the *SBOK® Guide*. Operational User Stories can be perpetual or recurring Sprint after Sprint. **Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3) **Additional Tool: Environment Identification*** (see section 13.3.3) **Additional Optional Tool: Scrum Project Tool** (see section 13.3.8)
9.2 Estimate User Stories	The estimation of User Stories is done as described in chapter 9.2 of the *SBOK® Guide*. **Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3) **Additional Optional Tool: Scrum Project Tool** (see section 13.3.8)
9.3 Commit User Stories	Scrum Teams commit to recurring User Stories based on past velocity. Since underlying task details may not be known at the beginning of a Sprint,

	the team commits to an allocation of time or quantity of Story Points dedicated to the User Story.
9.4 Identify Tasks	For known work (including fixes for known bugs), this process is used in the same way as described in section 9.3 of the *SBOK® Guide*. Unknown work, associated with recurring User Stories, is also committed to in this process, but in a different way. Known work would be contained in Work Package(s). These Work Packages are committed to in this process at the beginning of the Sprint. Some recurring work (e.g., handling incidents), is unknown at the beginning of the Sprint and therefore details are not available before an incident actually happens. Even when unknown work is included in a Sprint, it remains important that the team does not over commit for the Sprint. Because details are not yet known, the team commits to a certain amount of Story Points for any unknown work. The overall amount of work committed, including both known work and unknown work, needs to match the team's velocity. Example: The team's velocity is 30. The team commits to spending 10 Story Points worth of effort on handling incidents. The team commits to 20 Story Points to User Stories and known Work Packages. Thus, the overall commitment matches the team's velocity.
9.5 Estimate Tasks	If tasks are estimated, this optional process is used the same way as described in section 9.5 of the *SBOK® Guide*.
9.6 Update Sprint Backlog	This process is applied in the same way as described in section 9.6 of the *SBOK® Guide*.

Implement	**Summary of Impacts**
10.1 Create Deliverables	The regular creation of Sprint deliverables may be interrupted when an incident occurs so the team can address the incident. How the Scrum Team continues to create the Sprint deliverables therefore depends heavily on whether any unknown work for recurring User Stories arises during the Sprint, which may impact the usual deliverables planned for the Sprint.. **Escalation**—The request to work on a specific incident typically comes in from a customer support team via an escalation, as mentioned for the *Plan and Estimate* process. **Swarming**—If an incident occurs, several team members, or even the entire Scrum Team members may need to interrupt their planned work to focus on the handling of the incident. This is referred to as swarming. **Additional Tool: Scrum of Scrum Meetings*** (see section 13.3.5)—These are focused meetings where Scrum Team representatives meet to share the work status of their respective Scrum Teams.
10.2 Conduct Daily Standup	Each team conducts a Daily Standup Meeting. However, because each Scrum Master may be working with multiple Scrum Teams at the same time, some

	coordination effort is required to avoid scheduling conflicts between the Daily Standup Meetings of their respective Scrum Teams. **Additional Optional Tool: Scrum Project Tool** (see section 13.3.8)
10.3 Refine Prioritized Product Backlog	Product Owner interaction with the customer and other business stakeholders is handled by the Chief Product Owner and/or the other Product Owners, rather than a single Product Owner. How this interaction is divided is defined in the Product Owners Collaboration Plan. Additionally, final prioritization decisions are made by the Chief Product Owner. Any collaboration or interaction among Scrum Masters and/or Scrum Team members occurs as defined in the Scrum Masters/Scrum Teams Collaboration Plan. Otherwise, the refining of the Prioritized Product Backlog is handled as in a typical Scrum project. **Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2)—The Product Owners Collaboration Plan defines how the Product Owners make updates to the Prioritized Product Backlog. **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3) **Additional Input: Team Specialization*** (see section 13.2.5) **Additional Tool: Scrum Project Tool** (see section 13.3.8)

Review & Retrospect	**Summary of Impacts**
11.1 Demonstrate and Validate Sprint	This process is carried out individually by each Scrum Team. For each team the respective Product Owner approves the completed User Stories. However, this can be somewhat complex due to inter-dependencies. There may be times when the Chief Product Owner attends Sprint Review Meetings for some Scrum teams which have inter-dependent deliverables. **Additional Input: Chief Product Owner*** (see section 3.7.2.1)
11.2 Retrospect Sprint	Each Scrum Team meets with its Scrum Master for a Retrospect Sprint Meeting. Because a single Scrum Master and a single Product Owner may work with multiple Scrum Teams, some coordination effort is required to avoid scheduling conflicts between the Retrospect Sprint Meetings of different Scrum Teams. Also, the Chief Product Owner and the other Product Owners meet for a Retrospect Sprint Meeting to discuss their collaboration and other aspects of the Sprint. Additionally, Scrum Masters and/or other representatives from each Scrum Team meet for a special Scrum of Scrums Meeting (SoS) to share best practices and other results of the Retrospect Sprint Meetings of the different teams; for example, issues with inter-team collaboration. Often, best practices and problems stem from the collaboration between the multiple Scrum Teams and the team of Product Owners, so it is common practice for the Chief Product Owner and other Product Owners to participate in this meeting. **Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section

	13.2.3)

Release	Summary of Impacts
12.1 Ship Deliverables	All Accepted Deliverables from previously completed Sprints are generally delivered or transitioned to business stakeholders. However, additional preparation steps may be necessary to prepare for the Release. **Additional Input: Chief Product Owner** (see section 3.7.2.1) **Additional Input: Chief Scrum Master** (see section 3.7.2.2) **Additional Input: Release Readiness Plan** (see section 13.2.7) **Additional Optional Tool: Release Readiness Sprint** (see section 13.3.7)— Sometimes a Release Readiness Sprint may need to be planned. Requirements of such a Sprint are then added to the Prioritized Product Backlog.
12.2 Retrospect Release	**Additional Input: Chief Product Owner*** (see section 3.7.2.1) **Additional Input: Chief Scrum Master*** (see section 3.7.2.2) **Additional Input: Product Owners Collaboration Plan*** (see section 13.2.2) **Additional Input: Scrum Masters/Scrum Teams Collaboration Plan*** (see section 13.2.3)

1.7 Setting up an Organization for Extending Scrum to its Development and Operations (DevOps)

As described in previous sections, Scrum has grown in application and has helped organizations achieve success across numerous industries. Scrum is often used to deliver small projects; however, the framework can be scaled to manage large enterprise applications for both project (or development) and operational work. It is no surprise that many organizations have discovered value in combining development and operations

work, referred to as “DevOps” in the software/IT industry, and then applying Scrum practices to these integrated functions, in order to move towards further efficiencies and agility.

The Software Development Life Cycle (SDLC) is a traditional software development and implementation process, which emphasizes significant upfront planning, and subsequently the development of an entire software application, usually spanning several months or years. The product is eventually deployed into production through monolithic releases. SDLC is increasingly being replaced by Agile practices such as DevOps and Scrum for managing IT Operations.

DevOps is the most prevalent framework for IT implementations. This includes Dev frameworks for developing new functionality and Ops practices, which are used to effectively manage IT operations. The emphasis of DevOps is to create small functionality or product increments through iterative and continuous delivery of software, which are frequently deployed into production, thereby delivering high Returns on Investment in short time cycles.

Agile DevOps practices, such as continuous integration, continuous delivery, building a single deployment pipeline, continuous monitoring and logging, continuous learning, and experimentation, have become critical for successful IT implementations.

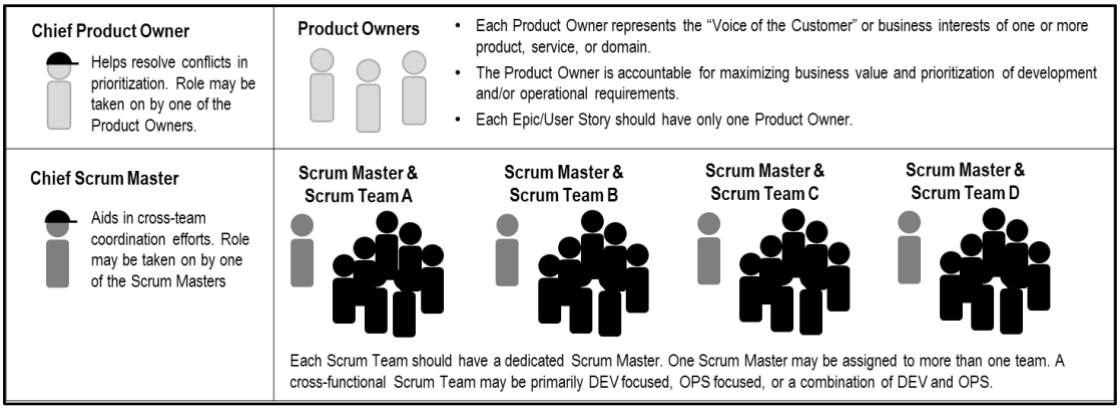
DevOps is most relevant for organizations that provide internal and/or external IT product or service development along with operational support and maintenance in production environments. The operational or “Ops” functions were traditionally carried out by specialized Ops teams and resources that work differently than development teams. Many organizations have opted to combine Dev and Ops functions (into DevOps) to integrate development and operational support activities, while leveraging an Agile/Scrum framework.

The six Scrum principles of Empirical Process Control, Self-organization, Collaboration, Value-based Prioritization, Time-Boxing, and Iterative Development are all still applicable when extending Scrum to an operations setting (ScrumOps) combined with development activities.

It is important to note that the Scrum framework (as defined in the *SBOK® Guide*), when extended to operations (ScrumOps) is very flexible, and can be used for both Ops and Dev roles within any company regardless of the industry. The framework also allows for Dev teams to support Ops requirements (such as a Dev team helping to fix critical bugs in programming code previously written by them), or for Ops teams to work on Dev projects (such as an Ops team working on setting up a testing environment or working on a process improvement project).

1.7.1 Organization

The Scrum roles and responsibilities remain unchanged when extended to DevOps. However, the organizational structure can be more complex when dealing with a combination of Ops and Dev teams. The following figure illustrates an example structure for a large-scale DevOps scenario:



1.7.2 Scrum Flow

In many cases, the combination of development and operational work in the same Scrum Flow is a natural progression. For example, the initial creation and launch of a new website would be managed as a development project with a defined end point—the website production deployment or launch date. Post-launch, the website would transition to a production state where ongoing user support and website maintenance would be required. This is often a combination of operational and development activities, such as software bug fixes or perhaps some additional desired new functionality or development requirements that did not get implemented with the initial website launch. The initial development Scrum Team, or some members of the team, would continue post-launch to work with the operational Scrum Teams in the DevOps Scrum flow.

In DevOps, the Prioritized Product Backlog will contain development-focused User Stories and tasks as well as operational User Stories and tasks. If multiple Scrum Teams are involved, the Scrum Flow operates the same as in a large-scale ScrumOps initiative (as described in section 1.6.2).

Multiple Team User Stories

Wherever possible, requirements should be broken down into separate User Stories so that they can be committed to by a single Scrum Team. However, this may not always be realistic. Some recurring, operational User Stories will require the involvement of multiple Scrum Teams. In this case, the same User Story (and its associated Work Packages or tasks) are completed by multiple Scrum Teams that need to work together to address an issue (e.g., investigating a bug in a software application). For example, a cross-functional development Scrum Team creates a module, a release team installs the software, and a hardware team sets up and manages the hardware on which the application is running. Each Scrum Team estimates and assigns a level of effort it will require to complete its associated tasks for the User Story. For example, the Dev Team that wrote the programming code for the module may estimate 8 Story Points in their Sprint for any bug fixes, the release team may budget for 5 Story Points, and the hardware team may budget for 13 Story Points. Estimates are provided because the work involved in fixing bugs are considered unknown Work Packages or tasks, so very reliable estimates cannot be provided.

1.7.3 Scrum Processes

Since DevOps typically involves multiple Scrum Teams, the Scrum processes generally follow the same processes used in a large-scale ScrumOps initiative (as described in section 1.6.3).

1.8 Case Studies

1.8.1 Large Development Scrum Project (Multiple Team)

The following are some case studies for Large Development Scrum Projects:

Case Study 1: Cross-functional Scrum Teams performing all the development work for the User Stories in a Sprint.

The Chief Product Owner and other Product Owners create User Story requirements for a large development project. Five Scrum Teams, facilitated by the Chief Scrum Master and other Scrum Masters, are responsible for committing to User Stories captured in the Prioritized Product Backlog (per the large project Scrum Flow). Once User Stories are allocated to a Sprint, they are broken down into tasks by the Scrum Team. The work associated with the Sprint is then completed by the Scrum Teams. The deliverables of each User Story are approved by the appropriate Product Owner.

Case Study 2: ScrumOps Help Desk (Multiple Team)

An operations Help Desk team supporting a company's product (Product A) works on several operational requirements (e.g., logging customer queries, resolving standard queries within SLAs, and escalating appropriate queries to a Second Level Technical Support Team) inside the company.

A Product Owner (PO1) for the Help Desk defines an operations User Story in the Prioritized Product Backlog, labelled "Log, solve, or escalate queries received from Product A customers." The User Story is a recurring one that may require involvement from multiple Ops and/or Dev Scrum Teams.

Another Product Owner (PO2) for the Second Level Technical Support Team defines a User Story for his team labelled "Provide second-level support for Product A to solve escalated Help Desk queries." SLAs are defined for the Work Packages and/or tasks in the User Story.

While defining the first User Story, PO1 also specifies that the tasks in the User Story could be escalated to the User Story created by PO2. Therefore, PO2 will accept relevant escalated Help Desk requests from PO1.

During the Sprint, the operations Help Desk team will take on the User Story "Log, solve, or escalate queries received from Product A customers". Any customer query about Product A will be created as a new Work Package in the User Story and a severity level may be assigned depending on SLA definitions for the Work Package. If a relevant Work Package template exists, then that template can be used to create the Work Package. Once the Work Package is created, it is broken down into the

tasks required to complete the Work Package. The Work Package with its defined tasks can be taken on by either the same person who created the Work Package, or by another person on the Scrum Team working on the User Story. The person who takes on the task will try to resolve the query (as per the SLAs for the Work Package). If unable to resolve the query, it can be escalated to the second User Story "Provide second-level support for Product A to solve escalated Help Desk queries." In this case, someone from the second level Scrum Team will take on the escalated Work Package and create different tasks to ensure that the Work Package is completed. After the task is completed by the second level Scrum Team, it will inform the first level Scrum Team about the task completion, and the First Level Scrum Team can then confirm the same with the customer.

Case Study 3: DevOps

A Scrum team typically spends 70 percent of its time doing development work and 30 percent of its time on support activities, such as fixing bugs or monitoring applications.

The Scrum Team selects the relevant recurring User Stories related to fixing bugs or monitoring applications and budgets 30 percent of its Sprint effort for such activities. The remaining 70 percent of effort for the Sprint is allocated to development-related User Stories (i.e., finite User Stories to be completed within the Sprint). These may be finite User Stories that require either a single or a multiple team.

1.9 Conclusion

In the fast-paced world of product development, business and IT operations, and DevOps, where adaptability, collaboration, and efficiency are of paramount importance, extending the Scrum framework to these management aspects has been demonstrated to be a transformative process. This white paper has made it increasingly clear that the integration of Scrum principles, Scrum processes, and Scrum Aspects offers a multitude of benefits for organizations seeking to succeed in the current age of continuous delivery and rapid technological change. Extending Scrum to operations and DevOps can bring structure, efficiency, and adaptability to not only project or product development, but also to an organization's operational and DevOps activities. Scrum can help align technical capabilities with business goals, improve collaboration, and ultimately drive innovation.

This white paper has highlighted how Scrum's core principles, namely, iterative development, empirical process control, value-based prioritization, time-boxing, collaboration, and self-organization, provide a solid foundation for bringing order to a complex and chaotic landscape of project delivery, operations, and DevOps. By fostering cross-functional teams and promoting constant collaboration, Scrum breaks down traditional silos, allowing teams to work seamlessly toward achieving common objectives. Moreover, the consistent feedback loops and regular retrospective analyses inherent in the Scrum framework encourage a culture of continuous improvement.

There are many benefits of extending Scrum to operations and DevOps environments within an organization. When Scrum is applied to projects, User Stories are considered finite and are completed

within the duration of a project, with a primary goal to deliver high value in a short duration. In the context of operations, the Scrum framework can be extended to the entire lifecycle of the product or service. This helps bridge the gap between product development, deployment/release, and operational support. By extending Scrum to DevOps, development teams are able to support the operational requirements of the Ops teams (such as when the Dev team fixes critical bugs in programming code they previously wrote) and Ops teams can work on development projects (such as when the Ops team sets up a testing environment or works on process improvement projects). This integration can help ensure seamless synchronization of both development and operational activities within the organization.

Organizations should be encouraged to consider the journey towards extending Scrum to their operational and DevOps activities. It is a journey marked by challenges and opportunities; one that demands a commitment to change, a willingness to adapt, and a culture of continuous learning and improvements. By doing so, organizations can position themselves to not only survive but thrive in a world full of increasing uncertainties and challenges.