DevOps Institute

# The Agile Service Management Guide

By Jayne Groll, DevOps Institute
with Foreword from Denis Esslinger

# Agile Service Management Guide

By Jayne Groll with Foreword from Denis Esslinger
Revised July 2021

# FOREWORD

**Denis Esslinger**
ITSM and DevOps Evangelist

# FOREWORD

Opportunities, plans, and circumstances, both internal and external to organizations, are changing at an ever-increasing pace. External unforeseen circumstances, like the COVID-19 pandemic, can suddenly require significant alterations to business practices and the IT services that support them. Agility is no longer something organizations want to strive for – it's a necessity for business survival.

I met Jayne about 20 years ago at an ITSM conference where she was talking about the importance of breaking down silos in IT. She advocated that IT teams needed to use organization-wide processes to successfully deliver value for customers. Since that time, Jayne has continued to be a leader in advancing people-centered practices to improve value streams, including being a co-founder of DevOps Institute.

This guide grew out of Jayne's recognition that end-to-end IT agility could only be achieved if Agile thinking and practices were instilled into every aspect of the management of products and services before, during, and after deployment. Much of Agile's focus has been on software development and deployment. However, if the deployed software applications are not consistently delivered at needed levels of availability, capacity, security, and continuity, little value will result from the deployed software.

Now, more than ever, organizations need to be agile and quickly adapt their IT services to changing needs. Being agile is a never-ending challenge. This guide helps provide a practical framework to engineer and continually improve your IT service management practices to ensure services provide customer value as needs continue to change ever more rapidly.

**Denis Esslinger**, ITSM and DevOps Evangelist

DevOps Institute

## TABLE OF CONTENTS

# Table of Contents

# INTRODUCTION

# INTRODUCTION

*A lot has changed in just a few short years. The rate of technological adoption has accelerated at a pace unseen before. Automation is integrated into just about everything we do, both personally and professionally. User expectation around technical services is always on, always reliable, always delivering value, and always doing more.*

The mandate remains the same but is more urgent than ever: go faster but control costs and risks. The ability to adapt to changing market trends is essential for every business in order to stay competitive and retain the current customer base. Any vertical market may be the target for disruption by a known competitor, an unseen startup, or on the radar of aggressive organizations such as Amazon. Disrupt or be disrupted. Transformation is no longer an option as we enter the "next normal" following the 2020 global pandemic. The directive from the enterprise to IT is clear. Think like a start-up but respect the bottom line. Be innovative and internally disruptive. Speed, quality, and reliability are your key metrics. Fail fast and learn from it.

When I first authored the Agile Service Management Guide in 2015, my primary objective was to instill agile thinking into IT Service Management (ITSM) process design. Recognizing the similarities between software development and process design cycles, I wondered if the same incremental and iterative approach put forth by Scrum could be adapted to the design, implementation, and management of ITSM processes. Certainly, concepts such as "minimum viable", "just enough" and "user stories" as well as the pillars of transparency/inspection/adaptation could be applied to process engineering.

Since that time, Agile Software Development is now being practiced by millions of more software engineers daily. Shippable increments are getting smaller and smaller. Cloud and hybrid environments are becoming normalized, giving way to cloud-native applications and containerization. Monolithic applications are

being broken down into microservice architectures. Enterprise interest in emerging practices such as continuous delivery/ deployment, automated testing, serverless, Kubernetes, and reliability engineering is on the rise. Delivering value is the prime directive and value stream management is replacing traditional service management Open source applications are being utilized by developers around the world and those same developers may be responsible for writing, building, testing, deploying, and securing their code.

IT culture is shifting too. A few years ago, IT governance was considered the path to business and IT alignment and was reflected in service management processes such as IT Change Management. Today, Change Management is under the microscope as being too restrictive as self-organizing hybrid product teams are supplanting traditional development, operations, and security silos. Process, automation, and "human skills" are considered equally important according to DevOps Institute's annual Upskilling: Enterprise DevOps Skills Survey and Report. Interoperability between humans is as important as interoperability between automation in order to increase flow. Lightweight interoperable processes and tools are needed to support both technical and human interoperability.

IT organizations must adopt a service-oriented approach to efficiently and effectively meet changing business needs. Rapidly changing business requirements require rapidly changing organizational capabilities.

## Is IT Service Management (ITSM) Still Relevant in the Next Normal?

There have been some significant paradigm shifts with approaches such as Agile, DevOps, and Site Reliability Engineering. There will be more. How does ITSM interact with these approaches, if at all? Before we dive into the details of Agile Service Management, let's explore the ongoing relevancy of IT Service Management itself.

## What is a Service?

While there are many definitions of a "service", here's what I believe –

A service enables the ability to do something when and how it is needed or desired. It enables its customers to achieve their objectives more efficiently and/or more effectively than they could without the service.

A service exists purely to "serve" the objective of its end customer (human or technical) and is only perceived to create value if the service delivers on its promise for timeliness and quality. In today's fickle and competitive "app" culture, a service that does not deliver value quickly and efficiently is replaced with one that does.

A service may be comprised of multiple products, applications, databases, infrastructure, platforms, and environments. Separately, most of these elements do not create value for the end customer. However, when federated into a "service", value is perceived because the outcome is achieved on time, cost, and quality.

## What is IT Service Management?

"IT Service Management is adopting a process approach towards management, focusing on customer needs and IT services for customers rather than IT systems, and stressing continual improvement." Source: Wikipedia

IT Service Management focuses on ensuring IT services deliver value by understanding and optimizing their end-to-end value streams. All services need to be managed in order to deliver customer value, regardless of which approach, framework, or methodology is applied. IT Service Management, therefore, underpins just about everything IT does in order to deliver valuable services including:

- **Service Levels**
- **Changes**
- **Releases**
- **Configurations**
- **Incidents**
- **Problems/Root Cause**
- **Requests**
- **Events/Monitoring**
- **Capacity**
- **Availability/Reliability**
- **Security**
- **Continuity**

One consideration affecting IT service management is the evolution from project to product management with its faster, more frequent iterations and no finish line. This paradigm shift has caused some friction between traditional command and control ITSM processes and the self-regulating systems of Agile, DevOps, and Site Reliability Engineering. In order to align with new requirements, organizations should review their ITSM governance model that was once a driving force behind traditional ITSM.

The question, therefore, is not whether ITSM is still relevant or even how to manage services in an Agile, DevOps, and digital transformation world. The real question is how much IT service management is just enough in order to create consistent customer value and compete in a fast-paced disruptive world. That's Agile Service Management.

DevOps Institute

# CHAPTER ONE: Being Agile

# Being Agile

If we can agree that ITSM is still relevant, how then does it become "agile"?

The MacMillan Dictionary defines "agile" as:

**ag·ile**

/ˈajəl/ | adjective

*Able to move quickly and easily; able to think quickly, solve problems, and have new ideas.*

Too often in IT, the term "Agile" is used to describe Agile software development, Scrum or Scaled Agile practices. While these are all excellent frameworks, the application of Agile practices to software development does not in and of itself increase an organization's agility. As many have learned, Agile software development teams are often frustrated by delays from downstream activities.

Agility must span the entire value stream. It is more important to "be agile" than to "do Agile".

Being agile is a state of mind. It is more perspective than prescription. In order for an organization to "be agile," it must also be:

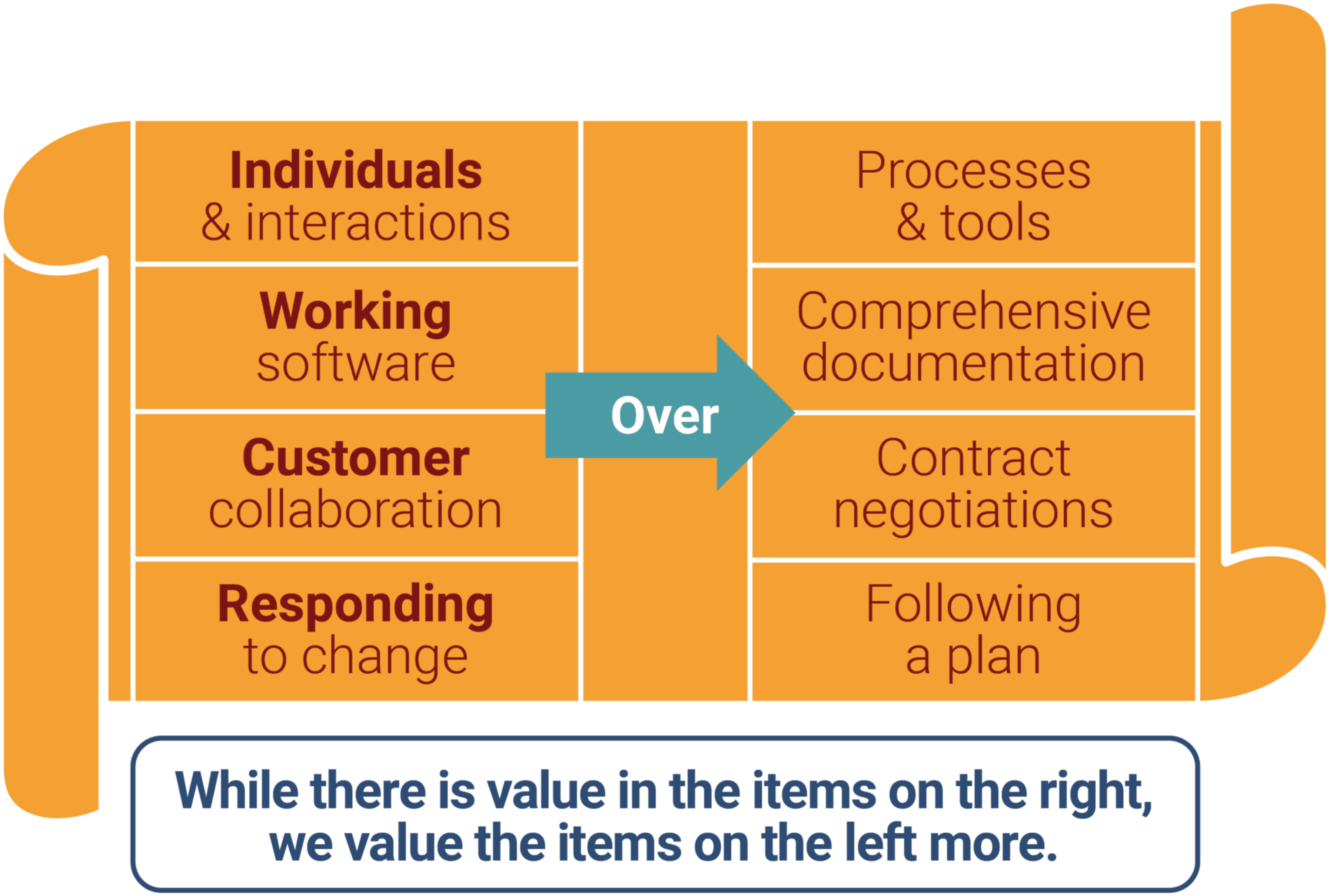
- **Customer-centric**
- **Lean**
- **Collaborative**
- **Communicative**
- **Adaptive**
- **Measurable**
- **Consistent**
- **Results-oriented**
- **Reflective**

## The Agile Manifesto

The underlying concepts of agile software development were first laid out in the Agile Manifesto in 2001.

The Agile Manifesto was supported by twelve principles as paraphrased below:

1. The highest priority is to satisfy the customer
2. Welcome changing requirements even if


Individuals & interactions
Working software
Customer collaboration
Responding to change
Over
Processes & tools
Comprehensive documentation
Contract negotiations
Following a plan
While there is value in the items on the right, we value the items on the left more.

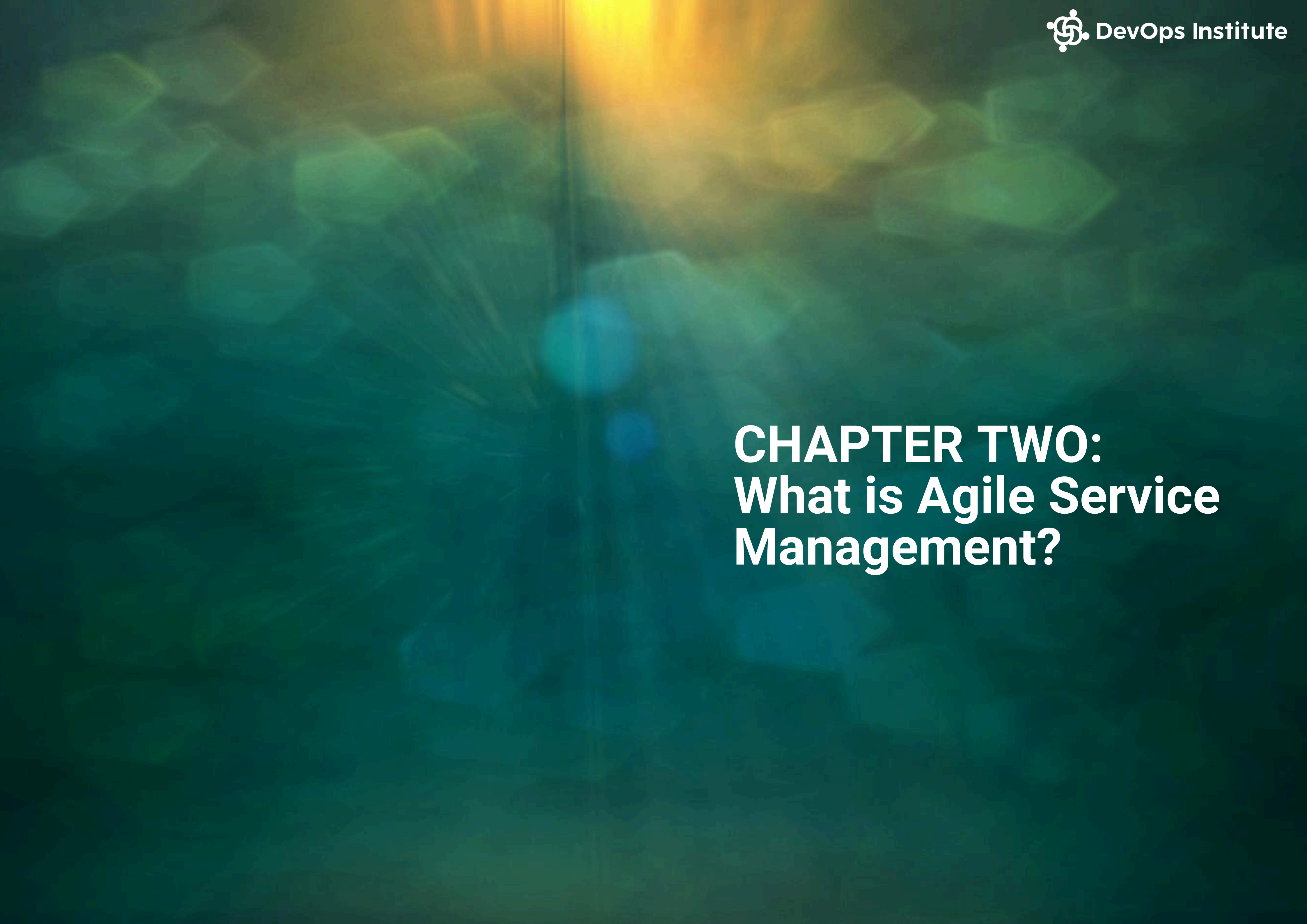
The Agile Manifesto

late in the development cycle

3. Deliver working software frequently
4. Business and IT must work together daily
5. Give motivated people what they need and trust them to get the job done
6. Face to face is the best way to communicate
7. Working software is the most important measurement of progress
8. Agile processes promote sustainable development
9. Be simple – maximize the amount of work NOT done
10. Self-organizing teams create the best architectures, requirements and designs
11. Teams should regularly reflect on and readjust their behavior to become more effective
12. Continually grow the team's technical excellence and design skills

The spirit of the Agile manifesto has never been more relevant and the underlying principles should form the basis for managing the entire IT value stream from ideation to value creation.

To the ITSM community, the Agile Manifesto may initially seem to discredit everything that ITSM stood for including processes, tools, plans, documentation and service level agreements. Not true. The Agile Manifesto does not suggest that the items on the right have NO value. It's a reminder that the items on the left have MORE value and therefore a caution not to prize the artifacts on the right over the outcomes on the left. It's a reminder that we should do "just enough" of the items on the right in order to deliver on the promises of the left.



DevOps Institute

# CHAPTER TWO: What is Agile Service Management?

## What is Agile Service Management?

Agile Service Management (Agile SM) ensures that IT service management processes reflect Agile values and are designed with "just enough" control and structure to enable the delivery of services that enable the ability to do something when and how they are needed or desired.

Agile Service Management adopts Agile principles to IT service management processes by implementing IT Service Management in small, integrated increments. This approach ensures that IT Service Management processes reflect Agile values from initial design through continuous improvement.

**The goals of Agile Service Management are:**

- Ensuring IT Service Management processes help create value for customers
- Ensuring holistically that all IT Service Management processes are coordinated and interoperate smoothly
- Ensuring there is the least amount of process control for the greatest amount of speed, quality, and compliance

**To achieve these goals, Agile Service Management strives to:**

- Optimize processes across the organization's value streams
- Make sure that agile values and systems thinking are instilled in IT Service Management processes
- Enable a faster flow of software delivery by defining "just enough" IT Service Management
- Engineer an integrated Agile Service Management microprocess architecture
- Find the balance between an IT Service Management governance model and a self-regulating system
- Optimize the use of automation to execute process tasks and to manage artifacts

It is important to note that Agile Service Management does not reinvent or replace the guidance from other frameworks such as ITIL®→ or Site Reliability Engineering. By itself, Agile Service Management is not an ITSM framework that defines principles, processes, and procedures. Agile Service Management is an enabler of faster, more adaptable IT Service Management regardless of framework. By embracing the "just enough" spirit of the Agile Manifesto, the incremental and iterative approach of Scrum, key practices from ITIL®/ ITSM, value stream management, Continuous Delivery, and Site Reliability Engineering, Agile Service Management focuses on having the least amount of process control for the greatest amount of speed, quality, and compliance.

One of the key principles of Agile Service Management is the decoupling of monolithic processes into multiple microprocesses that can be designed and managed separately. Microprocesses can integrate with other microprocesses from the same or other IT Service Management practices. This flexible "plug and play" approach encourages systems thinking and allows activities from multiple IT service management practices to be built in close alignment with each other. A good example of this would be developing a microprocess for recording changes at the same time as a microprocess for recording incidents so that the two could be cross-referenced.

The key benefits of taking an Agile Service Management approach include:

- Increased value to customers
- The ability to meet customer requirements faster and more accurately
- Overcoming constraints in process flows
- Increasing the effectiveness and efficiency of ITSM processes
- Improving the collaboration between development, operations, security, and the business
- Improving the velocity of the process improvement team
- Getting more "done"

Agile Service Management has cultural benefits too. By creating a common approach to practices from multiple frameworks, Agile Service Management helps to instill systems thinking that reduces handoffs, optimizes automation, and creates a common language drawn from multiple sources.

## Key Terms

**Service Management Practice:** an end-to-end capability for managing a specific aspect of service delivery (e.g., changes, incidents, service levels) that is built on a microprocess architecture

**Process:** interrelated work activities that take specific inputs and produce specific outputs

**Microprocess:** a distinct activity that can be defined, designed, implemented, and management independently but is integrated and interoperable with other microprocesses

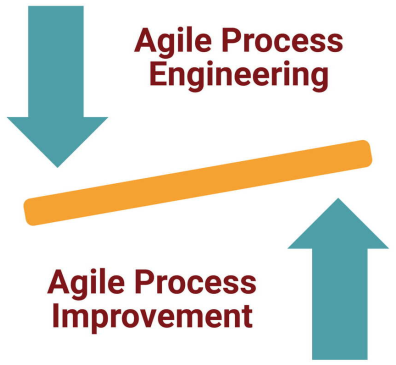
**Microprocess Architecture:** A collection of integrated microprocesses that collectively perform all of the activities necessary for an end-to-end service management practice to be successful.

## The Two Aspects of Agile Service Management

There are two aspects of Agile Service Management: Agile Process Engineering and Agile Process Improvement.

**Agile Process Engineering** is the aspect of Agile Service Management that applies an Agile approach to process engineering similar to an Agile approach to software development.

**Agile Process Improvement** is the aspect of Agile Service Management that aligns Agile values with processes through continuous improvement.



## Agile Process Engineering

Agile Process Engineering (formerly Agile Process Design) is an iterative and incremental approach to designing a process that replicates many of the practices used in Agile software engineering – short, iterative designs of potentially shippable Increments or microprocesses. By focusing on the "minimum viable" or "just enough" level of process control, Agile Process Engineering supports a "shift left" mentality so as to engage ITSM practices much earlier in the value stream. Smaller increments of microprocesses support faster deployments, better compliance, and more knowledge captured at the source. This approach also allows ITSM practices more time to be institutionalized and normalized by the people who execute the practices, microprocesses, and procedures.

Agile Process Engineering requires an IT service management architecture that supports systems thinking that spans the entire value stream. The architecture is engineered much like software where interoperability in a complex system is essential.

Agile Process Engineering also aligns with the goals and objectives of a microservice application architecture that structures an application as a collection of independent "services "

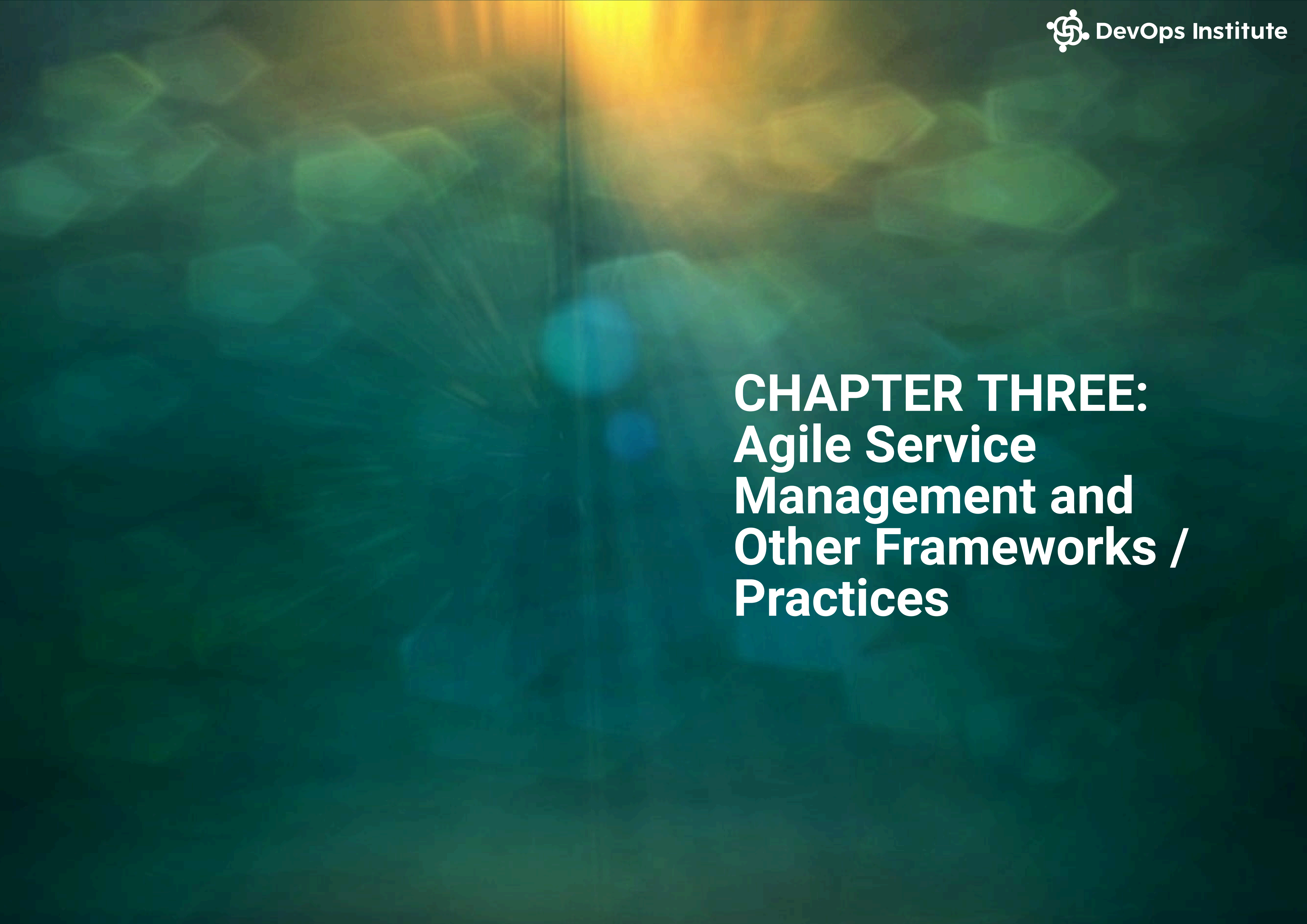
Microservices should be managed by microprocesses. We will define and discuss microprocesses and Service Management practices further in Chapter 4.

## Agile Process Improvement

Agile Process Improvement ensures that ITSM agility introduced through Agile Process Engineering is continually reviewed and adjusted as part of the ITSM's commitment to continual service improvement. The goal is to identify and mitigate any constraints that may be affecting both the flow of the service management practice, microprocess, and general flow of the value stream. Agile Process Improvement is the watch guard over unintentional process bureaucracy.

During this repetitive stage, Practice Owners conduct regular reviews with other teams and stakeholders to ensure that ITSM processes are still providing value and have not drifted from “just enough” to “too much” or “not enough”. Agreed improvements are captured in the practice’s Practice Backlog and engineered as part of Continual Service Improvement (CSI) sprints.

Since Agile Process Engineering creates a service management architecture built on interoperable microprocesses, Agile Process Improvement can manage and deploy improvements to a single microprocess or an entire service management practice, allowing IT service management programs to adapt to changing business requirements faster. We explore Agile Process Improvement further in Chapter 10.



# CHAPTER THREE: Agile Service Management and Other Frameworks / Practices

# Agile Service Management and Other Frameworks / Practices

As mentioned before, Agile Service Management does not redefine the inputs, outputs, or metrics of processes such as IT Change or Incident Management. There are some great frameworks in place that already do this. Agile Service Management does define a model for engineering any process in an incremental and iterative way. The principles and model can be applied to the adoption of one or more IT or even enterprise service management frameworks.

When I first authored the Agile Service Management Guide, ITSM and ITIL® were often considered to be virtually synonymous. Since then, frameworks and methods such as DevOps, Site Reliability Engineering (SRE), Lean IT, Scaled Agile (SAFe) and a new release of ITIL® have emerged. Each of these have inspired new ideas and new approaches to the problem of delivering and managing services in a technology-centric world. It is important to note that while each of these frameworks have merit, were authored by talented industry experts, and contain very useful advice, none are perfect and none are a complete end-to-end solution. Agile Service Management encourages the exploration and adoption of guidance from multiple sources in order to create a proprietary "just enough' ITSM program that is customized to the organization's requirements.

Let's look at the most prominent process frameworks and models trending today.

## Agile Software Development

### Scrum

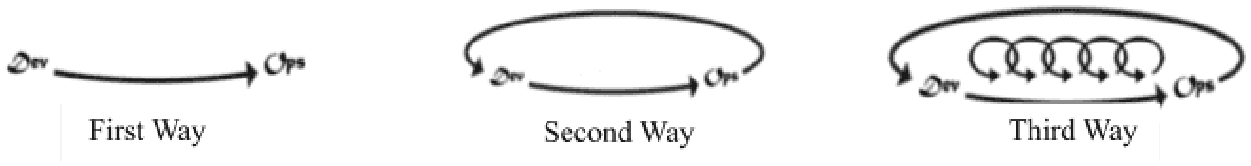
According to the 2017 Scrum Guide, Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum is:

- Lightweight
- Simple to understand
- Difficult to master

Recognizing the similarities between software engineering and process engineering, Agile Service Management loosely adopts the principles and processes from the Scrum Framework and applies them to IT Service Management. We will explore this further in Chapter 5.

### DevOps

DevOps is a cultural and professional movement that stresses communication, collaboration, and



*Source: The Phoenix Project, Kim, et al, IT Revolution*

integration between software developers and IT Operations professionals while automating the process of software delivery and infrastructure changes. Gene Kim, lead author The Phoenix Project, IT Revolution Press

On its own, DevOps is not a framework, standard, or methodology. It is a set of guiding values and principles that crosses multiple domains, tools, and practices including development, operations, security, and reliability.

Agile Service Management aligns with the principles of The Three Ways of DevOps as described in The Phoenix Project by Gene Kim, et al. An incremental approach to process engineering can remove constraints and increase flow across multiple processes (First Way), shorten feedback loops on process performance and improvement (Second Way) and encourage a collaborative, experimental and learning environment between various IT teams and frameworks (Third Way).

## Continuous Delivery/Deployment

Continuous delivery is a methodology that focuses on making sure software is *always in a releasable state* throughout its lifecycle. Continuous Deployment automatically releases the software into production.

Continuous Delivery is one of the main crossroads where Agile Service Management and DevOps can meet and align. Continuous Delivery relies heavily on automation to perform many of the activities associated with Release Management including building, testing, staging, securing, and releasing software. However, intelligent automation needs intelligent processes as shown in the Upskilling: Enterprise DevOps Skills Report. Agile Service Management can underpin Continuous Delivery/Continuous Deployment by defining lightweight interoperable microprocesses while Continuous Delivery automation can reduce IT service management toil. Best yet, since both are engineered (or re-engineered) in increments, the process, the automation, the metrics, and the artifacts can be defined and implemented simultaneously.

## Lean Practices

### Value Stream Management

Value stream management is a management approach that focuses on the end-to-end flow of customers, their challenges, and ideas (as input) to target business value (as output) through best practices and by the elimination of wasted time and resources.

Value stream management helps determine the value of software development and delivery efforts and resources. It also helps to improve the flow of value to the organization, while managing and monitoring the software delivery life cycle from end to end. *Source: Forbes Technology Council*

A value stream is the sequence of activities required to design, produce, and deliver a specific product or service and streams typically span multiple processes. Value streams allow teams to identify areas of non-value creation waste and to identify, prioritize and measure improvements.

Value stream mapping and value stream management have emerged as key tools for visualizing, understanding, and managing how value is created for the customer. This is not merely renaming a flowchart or replacing a service lifecycle diagram. By managing a value stream, the entire organization focuses on the customer's perception of value and how the organization manages value creation on an ongoing basis. The net result is that value stream management requires everyone involved to shift towards systems thinking and away from silos.

Since value streams cross multiple processes, taking a microprocess approach as defined by Agile Service Management allows processes to be built incrementally and in alignment with each other. Sometimes the biggest constraint to flow may be unintended bureaucracy or complex end-to-end processes that were designed in isolation. Agile Service Management helps to overcome those constraints.

### Kanban

Kanban is a method of work that pulls the flow of work through a process at a manageable pace. A Kanban board, therefore, makes work visible, reduces work in progress, helps teams collaborate, and supports the "definition of done". Kanban boards are used by many Agile software development teams to manage user stories, backlogs, and sprints. Since Agile Service Management adopts the concept of user stories, backlogs, and sprints, a Kanban can be a useful tool for managing practice backlogs, strategic, process, and continuous improvement sprints. Most importantly, whether in use for software or process engineering, a Kanban can be particularly helpful in understanding and managing team velocity.

## IT Service Management Frameworks

### ITIL® 4

For over three decades, ITIL® has been the most widely accepted and adopted framework for managing IT Services. ITIL® provides a framework that organizations can adapt to deliver and maintain IT services to provide optimal value for all stakeholders including the customer. It provides guidance and structure to practices such as change enablement, service configuration, deployment, release, incident, and problem management.

With an incremental release of the library starting in 2019, ITIL® 4 remains true to its heritage as an IT governance model but with closer alignment to modern practices such as Agile and DevOps.

ITIL® 4 embeds elements of V3's Service Lifecycle into a Service Value System (SVS) with a more visible focus on service value and outcomes. Where ITIL® V3 had 26 processes, ITIL® 4 defines 34 "practices" for managing services in a complex multi-domain environment. The migration from "process" to "practice" may seem semantic, but the goal was to instill a sense of capability instead of a sense of rigid process.

Like SRE or Scrum, ITIL® 4 embeds a set of core principles into ITIL® 4, adapting from those originally introduced in the ITIL® Practitioner:

The introduction of ITIL® 4 may inspire organizations to reimagine their IT Service Management models and re-engineer some of their processes or practices to line up with Agile, DevOps, SRE, and Continuous Delivery ecosystems. I would encourage those looking at ITIL® 4 to review the current state of their process adoption and analyze whether they are at "just enough", "too much" or "not enough" based on individual requirements, transformation efforts, and compliance.

### Site Reliability Engineering

Site Reliability Engineering (SRE) is a discipline that incorporates aspects of software engineering and applies them to infrastructure and operations problems. Inspired by Google's book series of the same name, SRE has gained global interest due to its goal of creating ultra-scalable and highly reliable software systems. The role of a Site Reliability Engineer has also emerged as a desirable and hirable job whose team is empowered to regulate its own workload, make tomorrow better than today and use failure as an opportunity to improve.

By its own admission, SRE is Google's approach to service management and is likely the most innovative approach to ITSM since the early days of ITIL®. The guidance provides insight on service management practices such as service level, change, incident, capacity, availability and other traditional ITSM practices.

The principles of SRE are founded on a self-regulating system that aligns closely with Agile software development and DevOps

While Agile Service Management is built around the pillars of Scrum, there is a close alignment to the Site Reliability Engineering principles of eliminating toil, empowering teams, optimizing automation and maintaining simplicity. The creation of a microprocess architecture that allows SRE and Development teams to autonomously adopt a small set of rules and "just enough" structure should appeal to Site Reliability Engineering teams.

## Observability

The concept of observability is quickly gaining acceptance as the more modern way to establish monitoring and event management practices.

Based on control theory, observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. This approach is now being adopted by global IT organizations that are managing complex interoperable environments and need more than reactive monitoring to understand the state of their systems.

Observability takes monitoring and event management to a whole new level. No longer are we monitoring for reactive issues such as capacity, availability or throughput thresholds. No longer are we satisfied with proactive system health monitoring. Observability is much more than being reactive or proactive – it now creates an environment where operations and development can infer the state of the internal system based on external conditions. It does not replace monitoring but adds a level of insight and abstraction that may not have amalgamated before.

## Enterprise Service Management

The success of IT service management practices has inspired organizations to extend the concepts beyond IT as Enterprise Service Management (ESM). ESM applies many of the same principles, practices and processes to business services – particularly those that the business supplies to its internal customers from business units such as HR or Finance. Many of the well-established ITSM vendors such as BMC, Cherwell and ServiceNow have extended their products to accommodate ESM.

Since Agile Service Management is not aligned to a particular ITSM framework, the same approach can be applied to any process engineering effort including Enterprise Service Management.

## ISO/IEC 20000:1 2018

ISO/IEC 20000 is the auditable international standard for IT Service Management that specifies the "requirements for an organization to establish, implement, maintain and continually improve a service management system (SMS) including the planning, design, transition, delivery and improvement of services to meet the service requirements and deliver value."

# CHAPTER FOUR: Scrum Basics

## Scrum Basics

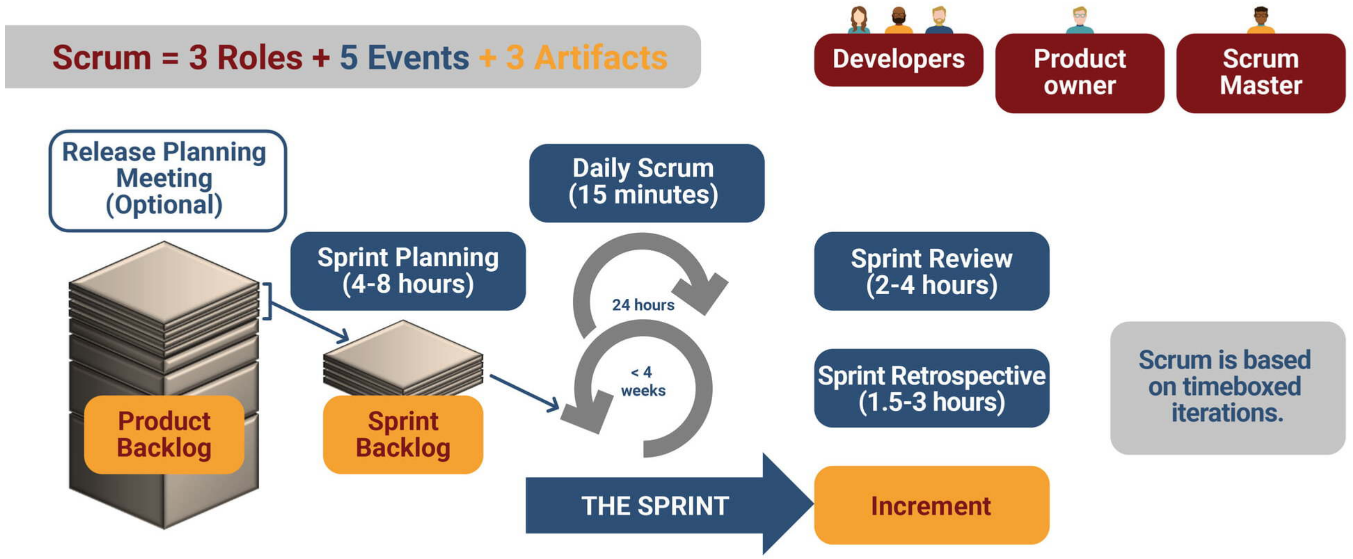
While there are several solid frameworks and methods for Agile software development, I have chosen to loosely model Agile Service Management after the Scrum Framework. Scrum is lightweight, simple to understand, embodies empirical thinking, and reflects the core values of the Agile Manifesto. While the context may be different, the core Scrum iterative and incremental guidance can be adapted to process engineering and improvement.

The Scrum Guide defines Scrum as:

Scrum is founded on empirical process control where knowledge comes from experience, decisions are based on what is known and three pillars (Transparency, Inspection, and Adaptation) underpin the entire framework.

The Scrum framework is built around interactions and rules that govern roles, artifacts, and events.

- 3 Roles (Product Owner, Scrum Master, Development Team)
- 3 Artifacts (Product Backlog, Increment, Sprint Backlog,)
- 5 Events (Sprint Planning, The Sprint, Daily Scrum, Sprint Review, Sprint Retrospective)

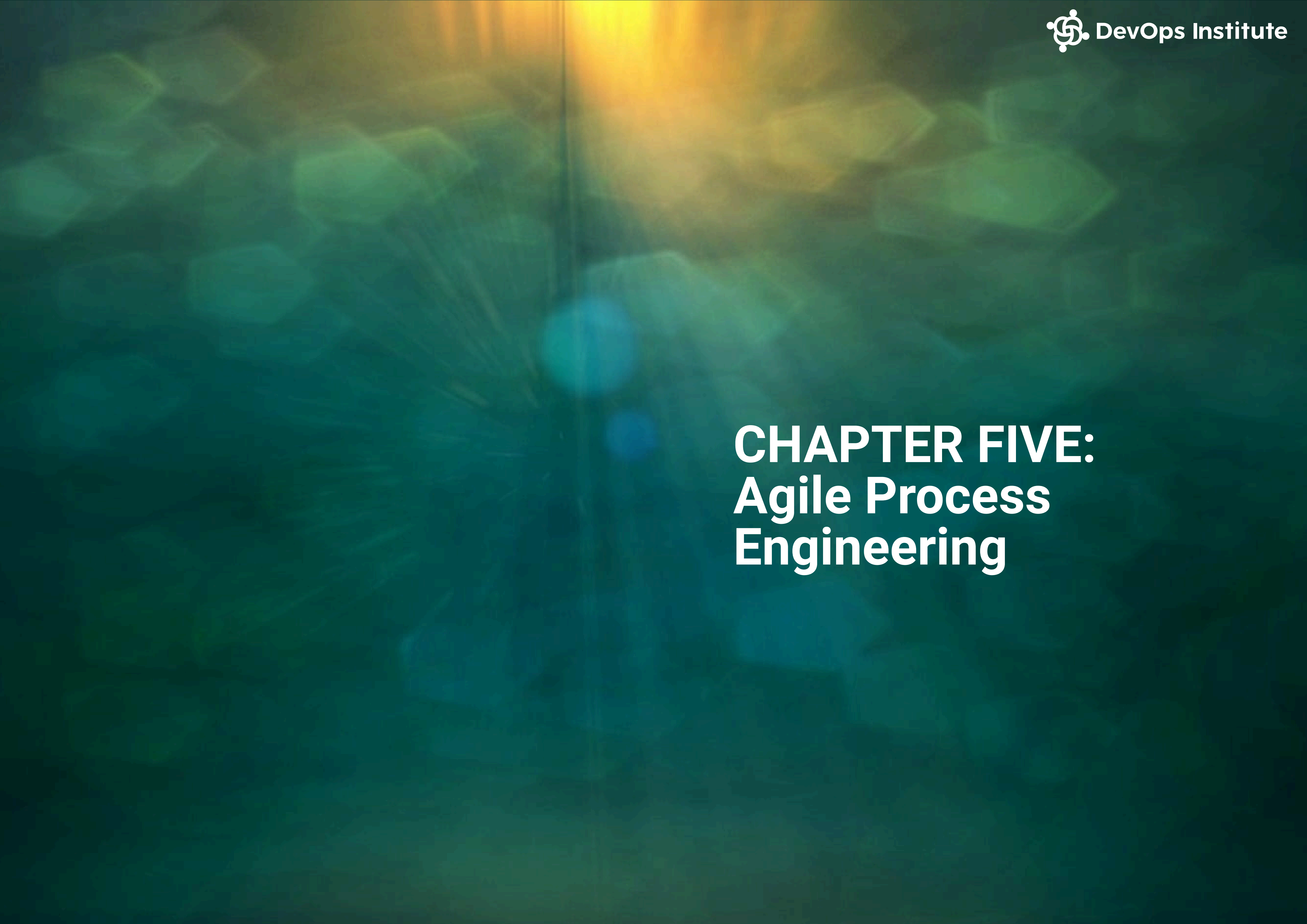


Scrum = 3 Roles + 5 Events + 3 Artifacts
Developers
Product owner
Scrum Master
Release Planning Meeting (Optional)
Daily Scrum (15 minutes)
Sprint Planning (4-8 hours)
24 hours
< 4 weeks
Sprint Review (2-4 hours)
Sprint Retrospective (1.5-3 hours)
Scrum is based on timeboxed iterations.
Product Backlog
Sprint Backlog
THE SPRINT
Increment

Agile Service Management retains the interactions and rules but adapts the roles, artifacts, and events to IT service management process engineering

- 3 Roles (Practice Owner, Agile Service Manager, Agile Service Management Team)
- 3 Artifacts (Practice Backlog, Increment, Sprint Backlog)
- 6 Events ( Planning, Sprint, Sprint Planning, Process Standups, Sprint Review, Sprint Retrospective)

I would encourage those interested in Agile Service Management to develop a deeper understanding of Scrum in order to internalize the similarities between the process of developing software and the process of developing ITSM processes. You can download the full Scrum Guide for free at https://www.scrumguides.org/.



# CHAPTER FIVE: Agile Process Engineering

# Agile Process Engineering

Agile Processing Engineering promotes a more adaptive approach by:

- Breaking a monolithic process into an architecture of independent microprocesses
- Taking a holistic approach to building, maturing, and integrating related microprocesses
- Designing each microprocess in smaller, more frequent iterations
- Encouraging shortened feedback and feed-forward loops
- Shaping future increments based on current business conditions
- Giving process practitioners more time to absorb new behaviors
- Getting more "done" and delivering value more quickly

## Engineering Agile Processes

Agility is a state of mind that centers around speed, collaboration and adaptability. Agility requires a conscious effort to limit delay, overcome impediments and avoid unnecessary complexities. Agility is where it is more important to "be agile" than "do Agile"

Since early ITSM adoption defined processes that were designed to provide evidence of IT Governance and controls, some IT service management processes such as IT Change Management grew layers of bureaucracy that were accompanied by inherent delays and in many cases a "one-size-fits-all" approach. While the intentions were honorable, the resulting processes were overburdened and not fit for the purpose intended. If unchecked, even agile processes can become unwieldy particularly in large, regulated organizations looking to scale.

An agile process is one that delivers "just enough" structure and control to enable the organization to achieve its service outcomes in the most expeditious, effective, and efficient way possible. An agile process is easy to understand, easy to follow, and prizes its collaboration and outcomes more than its artifacts.

The characteristics of an Agile process include:

- Having an accountable owner (Practice Owner)
- Clarifying everyone's roles and responsibilities
- Allowing for self-regulation, with consequences
- Benchmarking itself against Agile values and principles
- Being as simple, lean, efficient, and expedient
- Being scalable
- Adapting to change
- Optimizing automation for repetitive tasks
- Encourages accountability and autonomy
- Can be executed by people, other processes, or automation

Agile processes must be engineered much like software where interoperability in a complex system is essential.

## Practices vs Processes

Taking a page from ITIL® 4, what was previously represented as an ITSM process might today be considered a broader “practice”. A service management practice is a complete end-to-end capability for managing a specific aspect of service delivery (e.g., changes, incidents, service levels).

Under Agile Service Management, a service management practice is built on a microprocess architecture.

Examples of practices or practice areas would include:

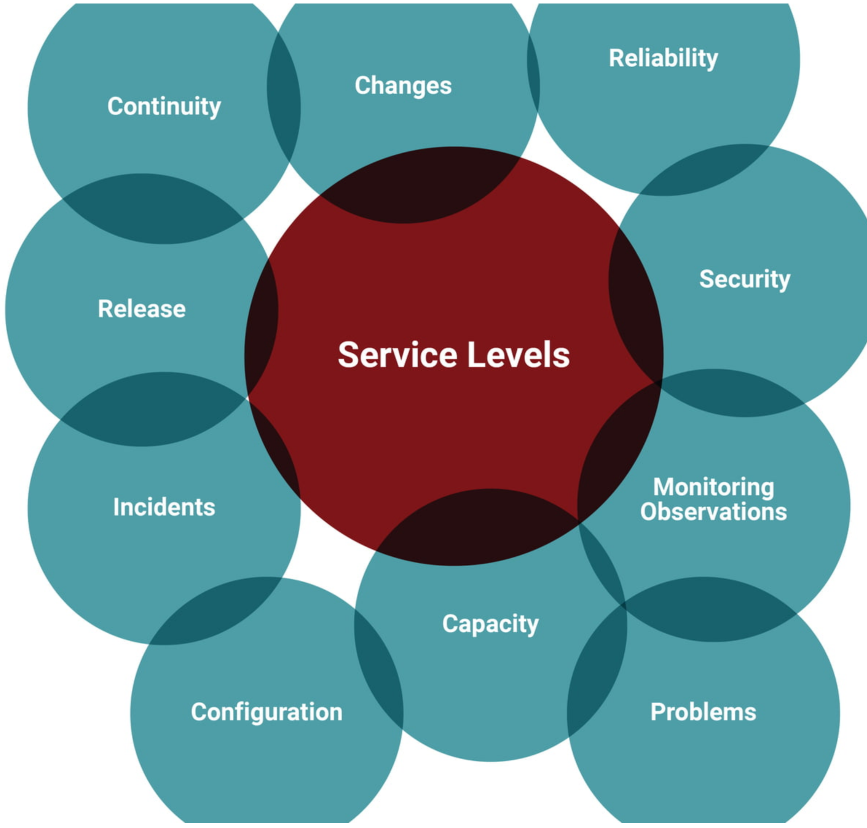
- Service Level Management
- Change Management
- Release Management
- Incident Management

This is more than semantics. A service management practice is successful not because of a flat and linear flowchart but from the combined power of the people, microprocesses, and automation that manage a specific aspect of services – whether it's managing changes or incidents, or requests. Looking at service management as a matrix of integrated and inclusive practice areas ensures that agility is embedded into every aspect of the organization’s value stream.

In Agile Service Management, a practice is built on an architecture of microprocesses that can be defined, designed, implemented, adapted, and improved independently. A Practice Owner is accountable for the service management practice and is an accountable collaborator on the structure and use of the microprocess architecture.

## Service Management Practices

Specific integrated capabilities for managing the necessary aspects of a service.



## Microprocesses

A microprocess is a distinct activity that can be defined, designed, implemented, and managed independently. A microprocess is generally associated with a primary service management practice but may be integrated with other service management practices. A microprocess can be defined as having its own inputs, outputs, activities, triggers, and defined outcomes. It may have its own artifacts or be part of one or more shared artifacts such as plans, tools, or maps.

Microprocesses are the foundation upon which an incremental and iterative approach to Agile Service Management can be based. Treating service management activities as microprocesses allows the organization to build service management practices in small, frequent releases that can deliver immediate value to the overall ITSM goal.

Why are microprocesses important to service management? Traditionally, service management practices are defined with a sequential series of activities that take an input and produce an output. In a microprocess architecture, each activity delivers its own value and can stand on its own. Its input can come from different sources and its outputs may be inputs to another microprocess or even another service management practice.

For example, the first microprocess for IT Change Management might be about “recording all changes”. This is a distinct activity with its own inputs, outputs, activities, triggers. The benefit of having more transparency into changes would not only affect IT Change Management but would be useful to practice areas such as Incident Management, Service Level Management, among others. Approaching this activity as a microprocess also encourages dialogue about how much is “just enough” detail about each change to provide value. This microprocess could provide metrics on change success, team velocity, etc.

## Microprocess Architectures

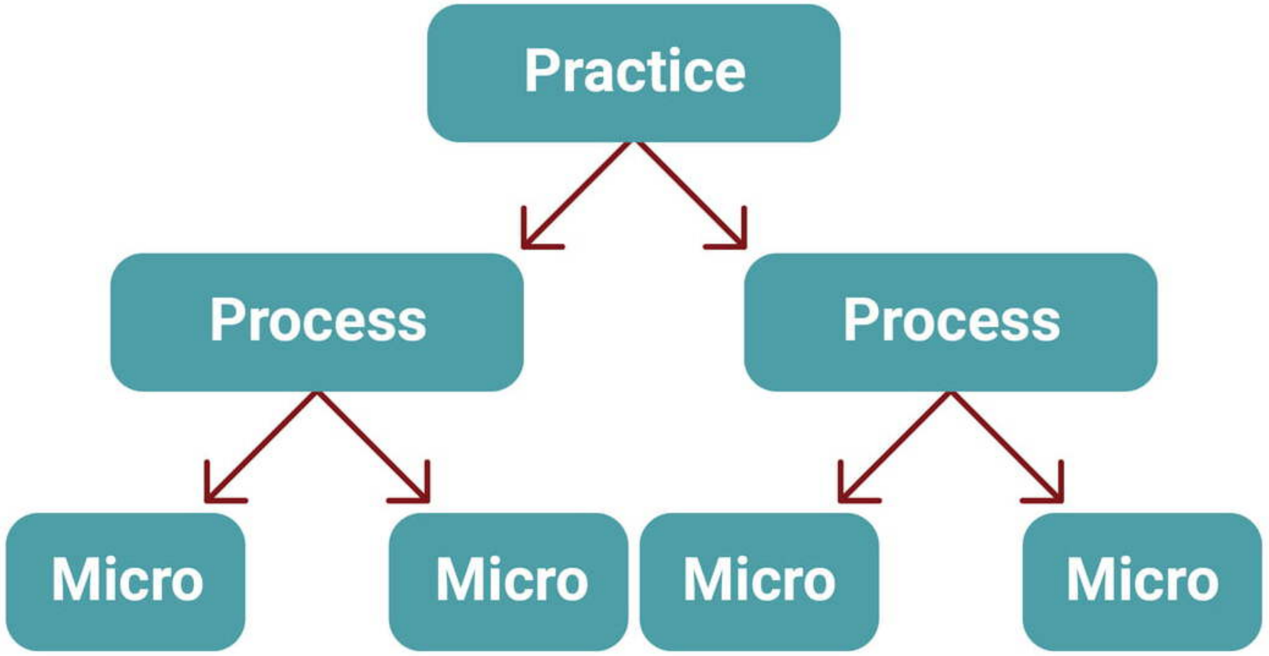
A microprocess architecture is a collection of integrated microprocesses that collectively perform all of the activities necessary for an end-to-end service management practice to be successful. Microprocess activities could be linear or nonlinear so that some are done in parallel or be recursive..

As DevOps, containerization, cloud architectures, and service management practices have evolved, Agile Service Management has similarly evolved and now decouples monolithic service management practices into a microprocess architecture. The microservice architecture enables ITSM to adapt to changing business requirements or to improve discrete aspects of a process without having to re-engineer the entire process (but with interoperability and integration always in consideration.)

A microprocess architecture is built on an interrelated collection of microprocesses that are:

- Highly maintainable and testable
- Loosely coupled

- Independently deployable
- Have their own feedback loops
- Interoperable with multiple processes
- Organized around business capabilities
- Owned by a small team

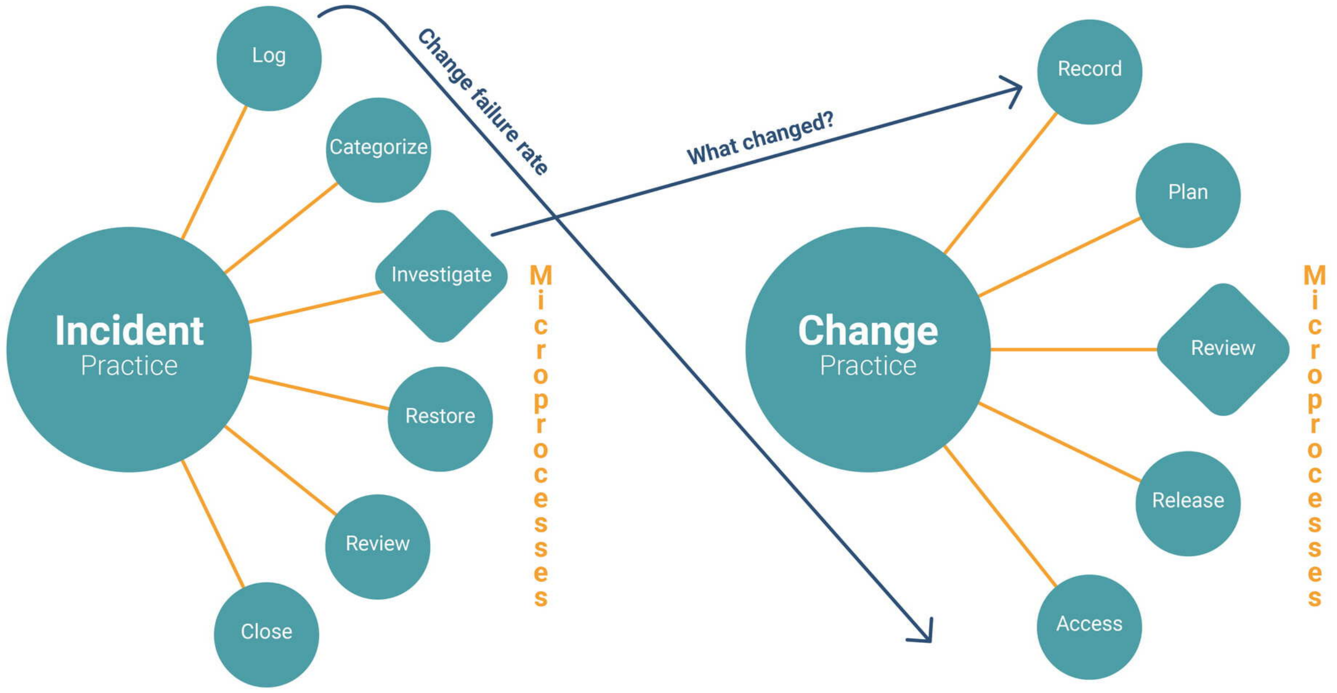


**Microprocess Architecture**

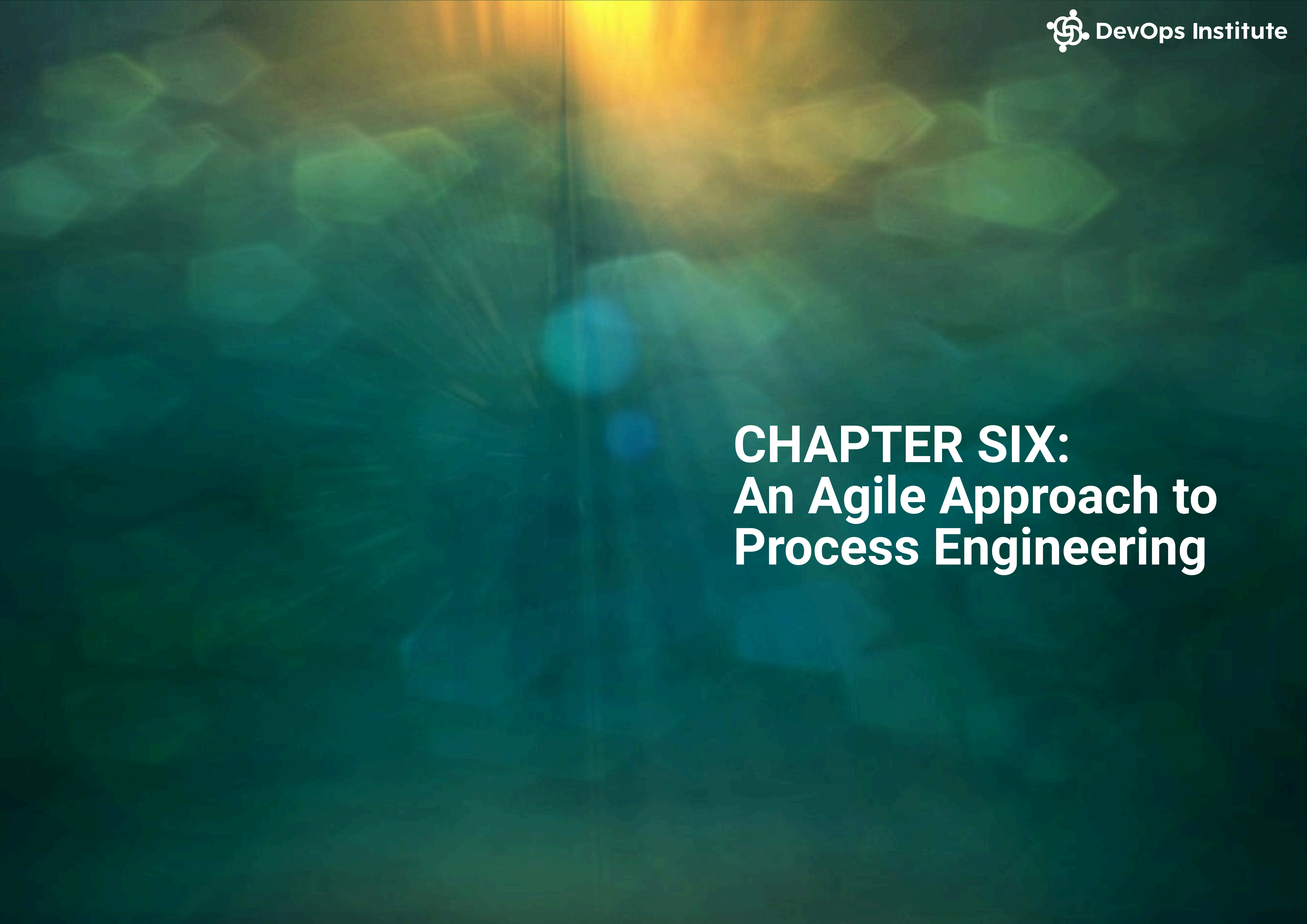
## Service Management Architecture

The service management architecture will be a matrix of integrated practices (and their respective microprocesses) that collectively ensure that services deliver the expected value when and how they are needed. The service management architecture supports systems thinking and therefore strives to adapt to all of the people, processes, automation, and information that span the entire value stream.

Ownership of the Service Management Architecture is a collaborative effort by the individual Practice Owners who will collectively adopt and integrate microprocesses and other practices into their ITSM "recipe".



**Service Management Architecture**



# CHAPTER SIX: An Agile Approach to Process Engineering

## Waterfall vs Agile Process Engineering

The waterfall model is a sequential approach to software development where each phase of the development flows the project further downward until the product is built, tested, deployed and ready to maintain.

While the waterfall model is associated with software development, earlier ITSM frameworks were authored when waterfall development and project management were common so the process design projects were built in a similar cadence. While the bodies of knowledge behind ITIL® and other ITSM frameworks did not necessarily promote a waterfall or bureaucratic approach, typical ITSM process roadmaps were positioned as multi-year where one or two monolithic processes were defined, designed, and deployed sequentially.

There are several challenges when applying a waterfall model to designing a complete service management practice or even a microprocess. These include:

- The rigidity of a sequential approach
- User feedback that comes late in the process
- The delays, rework, and additional costs resulting from user feedback and testing errors
- The need for integration with processes not yet in design
- The extensive time required to build and deploy an entire end-to-end process
- The learning curve that users will experience when trying to normalize an entire process and its procedures
- The risk that the resulting process will no longer fit the needs of its customers or deliver on promised outcomes when and how needed

Agile Process Engineering promotes a more adaptive approach by:

- Breaking a monolithic process into an architecture of independent microprocesses
- Designing each microprocess in smaller, more frequent iterations with its dependencies and successors in mind
- Encouraging shorter feedback and feed-forward loops
- Shaping future increments based on current business conditions
- Taking a holistic approach to building, maturing, and integrating related microprocesses
- Giving users time to absorb and institutionalize new behaviors

- Getting more "done" and delivering value more quickly

The net result will be a service management practice architecture that delivers "just enough" structure and control while

- Tying success measures to business outcomes
- Engaging stakeholders and soliciting input and feedback
- Enabling effective communication
- Integrating with other processes and frameworks
- Introducing timely improvements
- Having simple documentation
- Applying product management techniques over project management plans

How much is "just enough" process? The answer will vary from organization to organization, service to service, and practice to practice. Business requirements, governance, risk, and compliance will be important factors. Identifying the balance between "just enough" and "too much" process will take time, experimentation, and stakeholder engagement. Service Level Objectives will also likely be a driving force.

To start, it is best to create a Minimum Viable Process (MVP).

## Minimum Viable Process

A Minimum Viable Process is the least amount needed in order for this process or microprocess to meet its Definition of Done. Every practice and microprocess should therefore first strive to find its Minimum Viable Process.

Like a Minimum Viable Product in software development, a Minimum Viable Process has three characteristics:

1. It has enough value that people are willing to use it initially
2. It demonstrates enough future benefit to retain early adopters
3. It provides a feedback loop to guide future capabilities

It is much easier to add to a practice or microprocess gradually for scale than it is to revert to a lighter level later. A MVP approach ensures that the core elements and goals of a practice or microprocess are discussed, understood, designed and introduced first. It strips away the "wants" from the "needs" and provides a basis for ongoing dialogue and feedback so that future development will provide continued value to those who rely on the practice or microprocess.

One of the other key advantages of starting with a Minimum Viable Process is the ability to experiment with microprocesses and remediate quickly if necessary. Not only is this in line with the Three Ways of DevOps, but it also allows MVP microprocesses to be designed and deployed much more quickly. Minimum Viable Process is the launchpad for identifying "just enough" and can be used for new or re-engineered processes.

**To find the Minimum Viable Process, you may want to start with this question:**

*"What is the least amount that we would need to do in order for this practice or microprocess to meet our immediate goals or requirements?"*

Pick an easy microprocess that is a distinct activity that everyone recognizes such as documenting incidents.

Sample questions:

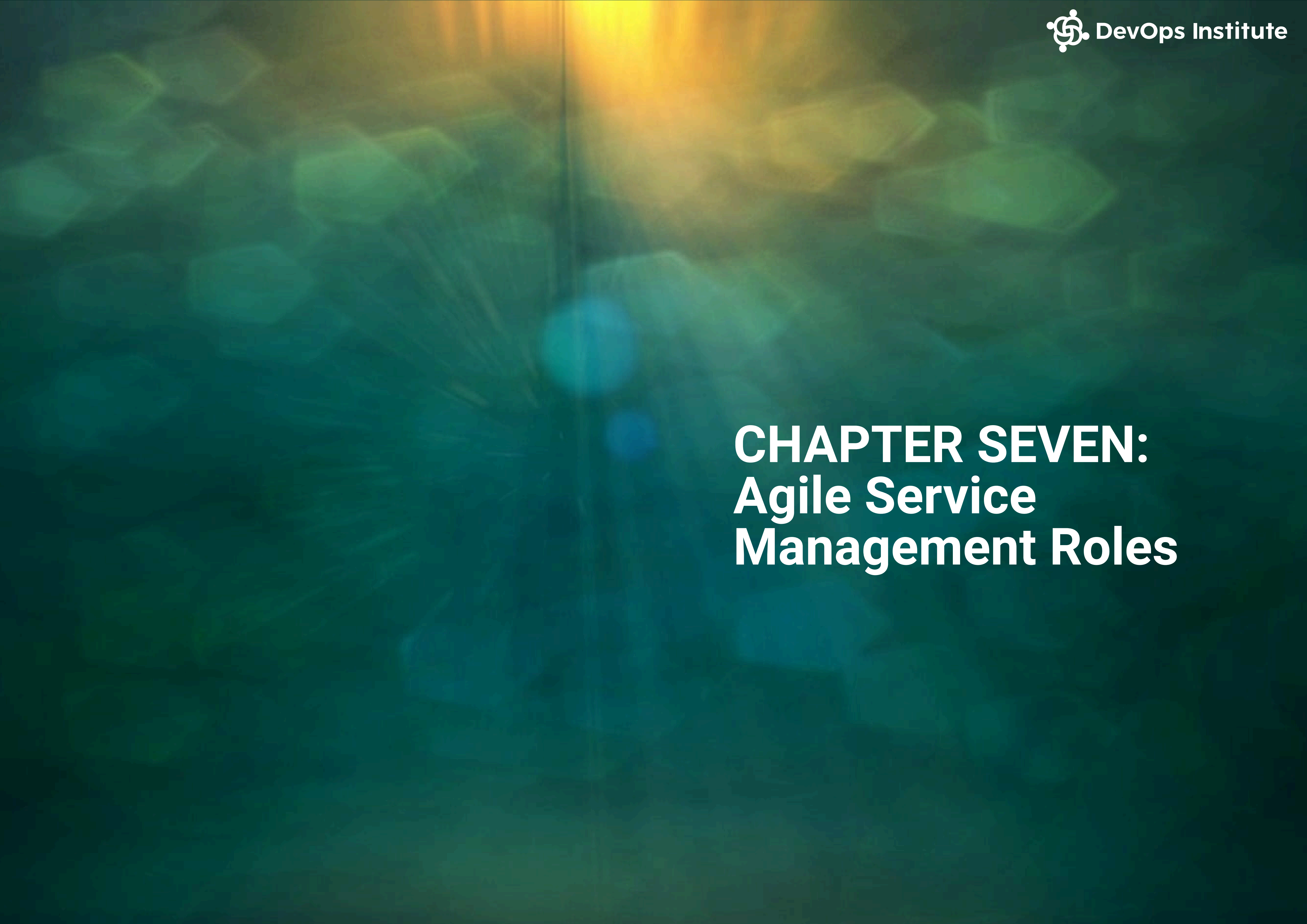
- "What is the least amount of information that we need to capture an incident?
- "What are the minimum time requirements for when the incident should be recorded?
- "What is the fastest, most efficient way of communicating the incident to key stakeholders or teams that need to be engaged?
- "What is the easiest and most sustainable tool for collecting incident data and to follow its progress?"
- "How will we know this microprocess is successful?
- "What are the expected outcomes?"
- "How will this microprocess support the Incident Management practice as a whole and other service management practices?"
- "Are there any risks or requirements that we are not considering at this point?"

While the questions are important, the dialogue is more important, particularly if the right people are asked the right questions. A skilled Practice Owner will be able to keep the Agile Service Management Team and stakeholder discussions focused on "least" and "minimum" to avoid the risk of "just in case" complexity layers that make a process more difficult and cumbersome.

The Agile Service Management Team should be comfortable deploying an MVP microprocess quickly but must also recognize that the first MVP release is likely not perfect. Soliciting constructive input and feedback from those that are executing the microprocess will help with the next iteration.

Each MVP microprocess contributes to the overall success of the service management program. By taking an experimental, MVP approach, the Practice Owners, Agile Service Managers, and Agile Service Management Team continuously learn and improve.

Now that we have an understanding of the basics, let's go into more detail about adapting the roles, artifacts, and events of Scrum to Agile Process Engineering.



# CHAPTER SEVEN: Agile Service Management Roles

# Agile Service Management Roles

The essence of Scrum is a small team of flexible and adaptive people. Small teams then become part of the fabric of interactive and collaborative networks of humans that define, design, deploy and improve agile practices across the value stream.

The same holds true for Agile Service Management. Small teams of people can focus on a microprocess as part of a network of other teams that are working on microprocesses from the same or different practice.

| Scrum Role | Agile Service Management Role |
|---|---|
| Product Owner | Agile Product Owner |
| Scrum Master | Agile Service Manager |
| Scrum Team | Agile Service Management Team |

In Agile Service Management, there are three clearly defined roles.

- Agile Practice Owner
- Agile Service Manager
- Agile Service Management Team (the "Team")

A "Team" may be accountable for a single microprocess or a complete service management practice. The Agile Service Management Team may also participate in Agile Service Management events for a related practice or microprocesses.

## Characteristics of an Agile Service Management Team

An Agile Service Management Agile Service Management Team is:

- Self-managing
- Cross-functional and cross-skilled
- Multi-domain
- Without egos or titles
- Without sub-teams
- Accountable for the work produced as a whole regardless of individual skills or experience

A self-managing team understands what it takes to get things done. For each increment of work, they are provided a goal, a backlog of tasks, a completion date, and a clear and shared Definition of Done. The Agile Service Management Team agrees on an approach for completing the work and meeting the goal. Essentially, the Agile Service Management Team is given the "what"; they collectively determine the "how."

Successful self-managing teams are:

- Stable
- Trusting
- Empowered
- Motivated
- Accountable
- Focused
- Business-centric
- Collaborative and communicative
- Diverse and Inclusive
- Empathetic
- Quality driven

Different perspectives and cross-functional skills are essential to an Agile Service Management Team. Roles should include a:

- Agile Practice Owner
- Agile Service Manager
- Customer and/or process practitioner
- Process architect
- Tool administrator or software engineer
- Change Manager
- Scribe or Documenter

NOTE: The Agile Service Management Team MUST include a customer or practitioner representative.

Each member of the Agile Service Management Team will work on items from the Practice Backlog. None are observers.

The Team should have at least three members but no more than nine to ensure sufficient skills and the ability to self-organize. Members may be on multiple teams, although it is recommended that an individual not work on more than two Agile Service Management Teams at any given time.

## Velocity

Velocity is a metric that estimates how much of the Process Backlog an Agile Service Management Team can handle in a single Sprint.

Velocity is often measured by work accomplished during past Sprints and serves as a predictor of future Team performance.

It is important to be realistic when first establishing the goals, the Increments, and planning the Sprint Backlog for any specific practice or microprocess. Unlike Scrum for software engineering, members of an Agile Service Management Team most likely have other roles and responsibilities. The cadence of work may ebb and flow according to dependencies on other teams, tools, or stakeholders. As Agile Service Management practices mature, the velocity of each Agile Service Management Team will naturally increase as more microprocesses are built, deployed, and integrated into the microprocess architectures.

## The Agile Practice Owner

The Agile Practice Owner is accountable for the end-to-end results of the service management practice. Their key responsibility is to create, manage, prioritize and own the Practice Backlog. The Practice Backlog is the single source of current or future requirements including activities, tools, plans, interfaces, documentation, training, and improvements.

The Agile Practice Owner has ultimate authority over the items in the Practice Backlog and ensures that the items are clear and visible. This role understands how to prioritize items in the Practice Backlog and helps the Agile Service Management Team understand the goals and objectives of the next Increment. The Agile Practice Owner is the only individual who can change the Team's direction and/or add, remove or cancel items in a Sprint.

The primary responsibilities of the Agile Practice Owner are:

- Setting and communicating the practice's vision and practice goal
- Ensuring the practice supports the needs of stakeholders and the
- creation of value
- Staying informed about changes in business direction and needs
- Aligning the practice with other practices, frameworks, and methods
- Ensuring that Agile values and thinking are embedded into the practice
- Measuring and assessing the value, quality, and relevance of the practice

Additional responsibilities include:

- Prioritizing items in the Practice Backlog
- Helps the Team to understand the goals and objectives of the next Increment
- Clarifying a Definition of Done for each Increment or backlog item
- Inspecting the progress and status after each Sprint
- Identifying opportunities to optimize automation and reduce manual activities
- Auditing and reviewing the practice on a regular basis
- Avoiding unnecessary layers of bureaucracy and complexity
- Working with other Practice Managers to own and ensure that the Service Management Architecture is aligned and efficient

The Agile Practice Owner is not necessarily responsible for performing any or all of the tasks associated with managing a practice. Depending on the size and complexity of the organization, the Agile Practice Owner may assign one or more roles to oversee day-to-day execution. An Agile Practice Owner may also be accountable for one or more related practices.

## The Agile Service Manager

In the early days of ITIL®, an individual who possessed the most knowledge about service management strategies and tactics was designated as a Service Manager. The same holds true for the Scrum Master in Agile Software Development. Given the adoption of multiple frameworks, tools, and methodologies, the Agile Service Manager is tasked with expertise on Scrum, Agile Service Management, other service management frameworks, and related practices. This individual is the subject matter expert, coach, and protector of the Agile Service Management Team.

Responsibilities of the Agile Service Manager include:

- Facilitating Agile Service Management events
- Helping the Agile Practice Owner create and maintain a "just enough" architecture of their

practice
- Helping the Team understand, adopt and adapt Scrum and Agile Service Management principles and methods
- Ensuring that the Agile Service Management Team focuses on outcomes over artifacts
- Protecting the Team by removing impediments whenever possible so that the Agile Service Management Team is successful
- Instilling agile thinking, empirical principles, Scrum knowledge, and ITSM objectives into the Team's culture and behaviors

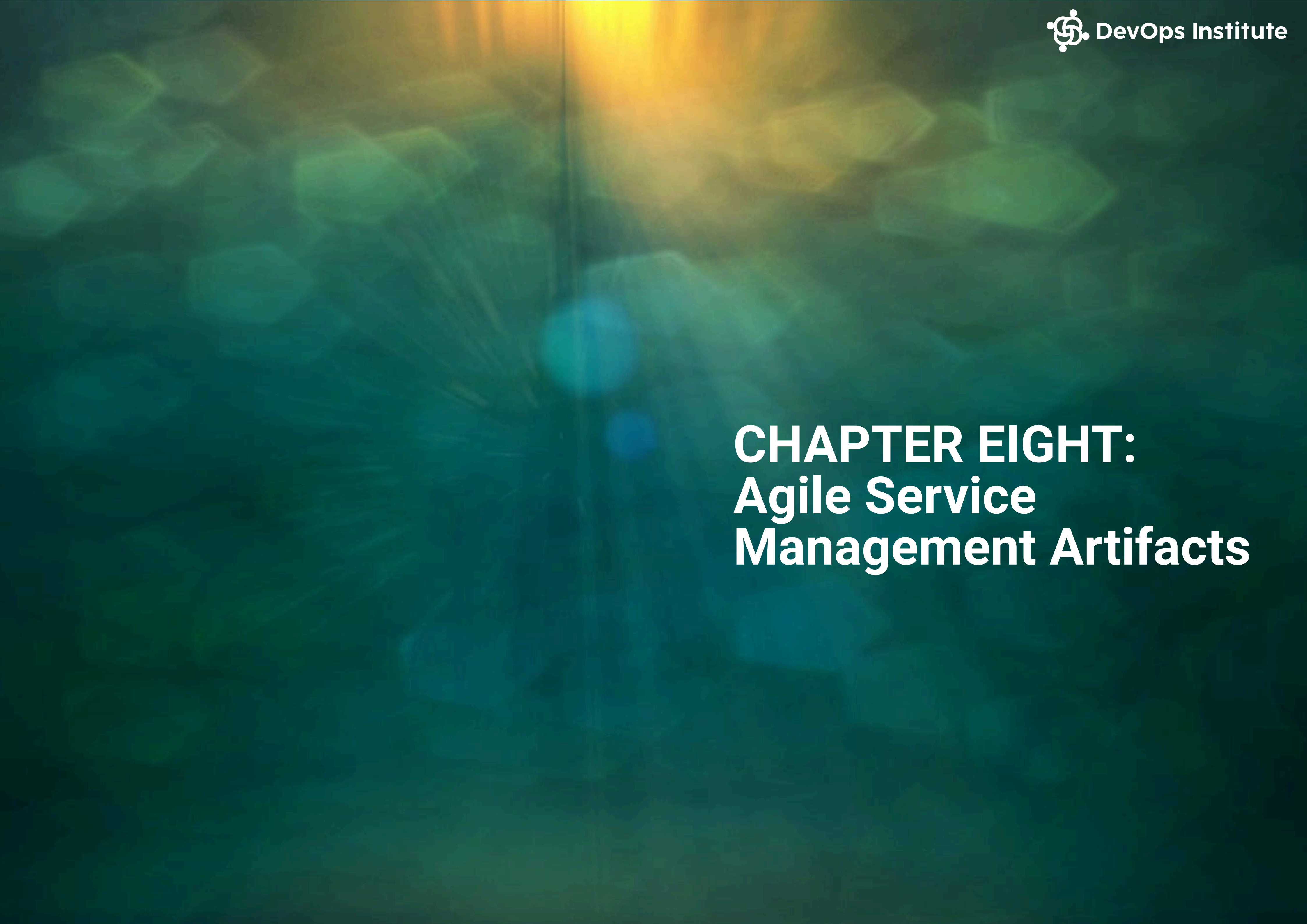
The Agile Service Manager does not manage the Agile Service Management Team. The Team is self-managing. The Agile Service Manager is a servant-leader that helps the Agile Practice Owner create a "just enough" architecture of Agile Service Management practices and microprocesses by maintaining an accurate and relevant Practice Backlog. The Agile Service Manager coaches the Team, helps the members write effective process-related user stories, and encourages them to think small but act big.

The Agile Service Manager serves the Agile Practice Owner by:

- Helping ensure the practice is in alignment with other practices and supports value streams
- Sharing techniques for effective Practice Backlog management
- Helping ensure the processes reflect Agile values and principles
- Facilitating stakeholder collaboration as requested or needed

Most importantly, the Agile Service Manager protects the Team and does everything possible to ensure its success. This includes helping those outside the Team understand how to (and how not to) interact with the Team. The Agile Service Manager educates the organization on Agile values, Scrum practices, ITSM processes, and the Agile Service Management approach to process design and development so that everyone knows what to expect.

The Agile Service Manager bridges a relationship with developers, Scrum Masters, DevOps teams, Site Reliability Engineers and automation architects to ensure cross-pollination of taxonomy, tools and activities. Collaboration across multiple domains helps to create and maintain a unified Agile, DevOps and ITSM/SRE culture.



# CHAPTER EIGHT: Agile Service Management Artifacts

# Agile Service Management Artifacts

| Scrum Artifact | Agile Service Management Artifact |
|---|---|
| Product Backlog | Practice Backlog |
| Increment | Increment |
| Sprint Backlog | Sprint Backlog |
| Burndown Chart | Burndown Chart |

## The Practice Backlog

The Practice Backlog is a prioritized list of everything that needs to be designed or improved for a service management practice including current and future requirements.

The Practice Backlog is the single source of truth for a service management practice where each item is expressed first as a user story. It includes activities, tool updates, plans, interfaces, documentation, training and improvements. The Practice Backlog continually evolves, is regularly re-prioritized and is never complete. It exists as long as the service management practice exists. It is solely owned and managed by the Agile Practice Owner.

The form and format of the Practice Backlog is not prescribed – entries can be captured in anything from a Kanban to a spreadsheet to a service management tool. It should be visible to all stakeholders and readily available for inspection.

## The Practice Backlog and User Stories

An Agile Service Management user story is a simple statement that describes what a user or process practitioner wants from an aspect of the service management practice. It is always written from the user's perspective and in their words. It is not meant to include all of the details about the aspect but is intended to encourage further dialogue, detail and collaboration. User stories are generally captured on index cards or sticky notes (physical or digital) but may point to more comprehensive documents or diagrams. The user story is generally written in business terms and may evolve over time with collaboration.

**User stories generally follow the formula:**

“As a (role), I want to (do something) so I can (achieve something)

In 2003, Bill Wake recommended the INVEST model to describe the elements of a good user story

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

An Agile Service Management user story can be written for any aspect of the practice or microprocess including an activity, a procedure, a complete microprocess or process artifact. A process user story is the quickest and easiest way to understand how the process, microprocess or procedure will be consumed, by whom and what the expected value outcome is. User stories are very powerful tools that provide a wealth of insight in a single sentence. The Agile Practice Owner is responsible for maintaining the Practice Backlog of user stories and facilitating their refinement.

## Epics

An Agile Service Management epic is a collection of related user stories that may need to be worked on across multiple sprints. Most microprocesses would likely be considered an “epic” to accommodate multiple requirements.

## Practice Backlog Refinement

The Practice Backlog should be refined regularly to add detail, estimates and prioritization to the user stories or epics that comprise Practice Backlog items. **The Practice Goal is also maintained in the Practice Backlog as a long-term target for the service management practice that the Agile Service Management Team can use to help planning. This may also be refined as business conditions change.**

The Agile Practice Owner and the Agile Service Management Team will determine when and how the backlog items should be reviewed and refined. As items become higher priorities, the amount of detail needed will become greater and therefore refinement more necessary. Details can come from a variety of sources, but the Agile Service Management Team is responsible for updating the work estimates as important inputs into Sprint Planning.

Each user story or epic in the Practice Backlog should be refined with at least the following details:

- A unique reference number for identification and querying
- The primary stakeholders or customers
- An assigned priority
- The estimated number of hours to complete its design
- Its dependencies and successors
- Who the story has been assigned to?
- The anticipated Sprint that will include this

story
- An approximate date of completion

It is important to retain the spirit of the Agile Manifesto when entering or refining items in the Practice Backlog. Keep the process as simple as possible and be wary of prioritizing the Practice Backlog items over the value of work that the items will facilitate.

## The Sprint Backlog

The Sprint Backlog is a subset of the Practice Backlog and forecasts what increment of the service management practice or microprocess will be tackled during the next Sprint. It is created during Sprint Planning and documents all of the backlog items that will be necessary in order to meet the Sprint Goal. It should be highly visible and available for inspection.

The Sprint Backlog provides a central artifact around which the Agile Service Management Team can self-manage in order to meet the Sprint Goal. It should have enough detail so that the Team understands the Definition of Done and can inspect progress during the Daily Scrum.

The Sprint Backlog expires at the end of the Sprint – hopefully with all items completed. Outstanding items do not automatically carry over to the next Sprint. They are reprioritized with other Practice Backlog items and considered during the next Sprint Planning.

## Increments

An Increment is the potentially releasable completed work that is the outcome of a Sprint. It is one element of a service management practice or microprocess. It may be an entire microprocess or a discreet aspect of an epic or service management practice. Increments are considered potentially releasable if they can deliver value. The increment could be an enhancement, correction or an element of one or more microprocesses. The Increment is built upon user stories.

The Increment and its outcome is defined during Sprint Planning from items in the Sprint Backlog.

An Increment is considered finished when it meets the agreed Definition of Done. It is demonstrated and discussed during the Sprint Review. The Agile Practice Owner then decides whether and when the Increment should or can be released. If the Increment is part of an epic, then it will be reviewed in line with the progress of that epic.

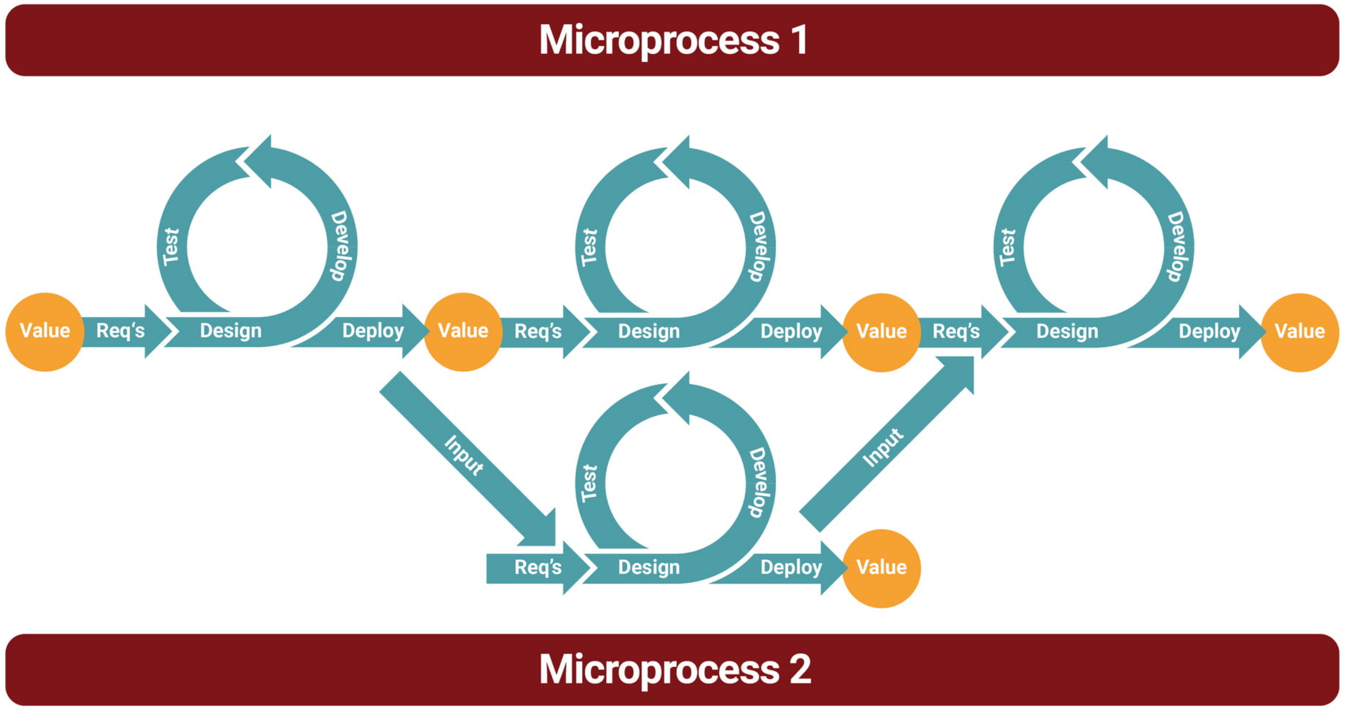
## What's the Difference Between an Iteration and an Increment?

An increment is a potentially releasable piece of a service management practice or microprocess whereas an iteration is the repetition of building increments and microprocesses in an Agile Service Management way. Defining a process or microprocess in increments allows it to be designed gradually so that it is transparent, adaptive and inspected (the pillars of Scrum). Iterations are usually repeated as “sprints”. Each iteration of Agile Service Management (e.g., the sprints) enriches the practice either by adding more value/ease of use or by removing manual work or toil.

## The “Definition of Done”

The Agile Service Management Team and process stakeholders must share an understanding of the “definition of done” for each Practice Backlog item, Increment or microprocess. A service management practice is never “done”.

The Definition of Done is critical to Sprint Planning in that it defines when the increment is complete. It guides how many user stories can be added to the Sprint Backlog and reasonably accomplished during a Sprint. As the Agile Service Management Team’s velocity increases, their ability to get more “done” in each Sprint will also increase.



## When is an Increment Done?

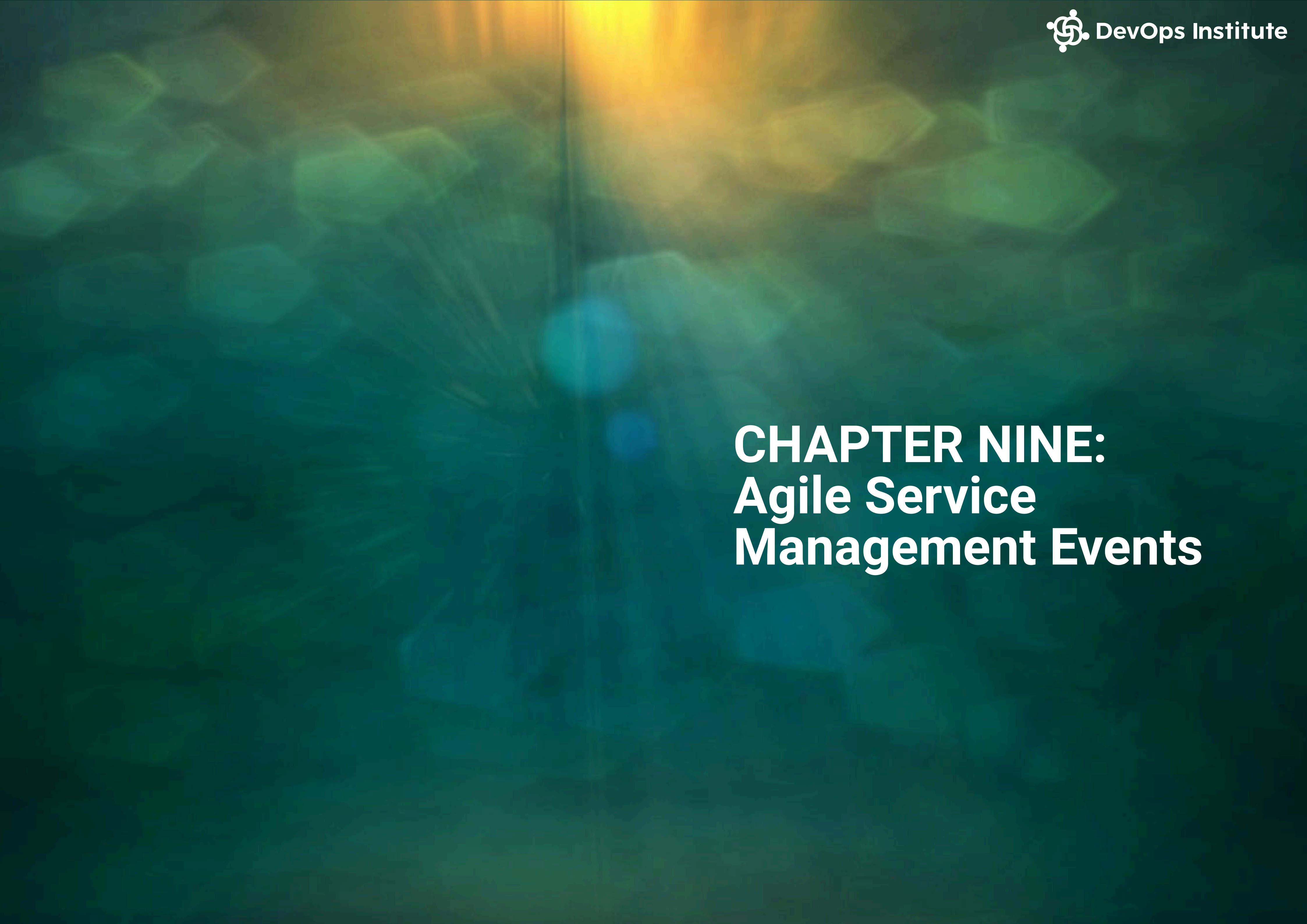
The Definition of Done will vary from Increment to Increment depending on the scope of work in the Sprint Backlog. Microprocesses should only be considered “done” when the following questions have been answered:

- Have the benefits and value been defined and communicated? (Why)
- Have roles and responsibilities been clarified? (Who)
- Have the inputs, outputs, triggers and outcomes been defined? (What)
- Are procedures easy to understand and do? (How)
- Have tools and automation been updated? (Toil reduction)
- Have policies been reviewed and updated if necessary? (Consequences)

- Have related Agile Service Management Teams been informed? (Interoperability)
- Have stakeholders been given a chance to test? (Feedback)
- Is it ready to stand on its own and deliver value? (Go/NoGo)

In simple terms, the Definition of Done is when the Team does not need to think about this increment or microprocess anymore and the completed increment should be potentially shippable for release.

Other than the Practice Backlog (which is owned by the Agile Practice Owner), all Agile Service Management artifacts are owned by the Team. The Agile Service Manager protects the Agile Service Management Team, removes impediments and provides expertise on service management and Scrum but is not accountable for managing the artifacts.



DevOps Institute

# CHAPTER NINE: Agile Service Management Events

## Agile Service Management Events

| Scrum Event | Agile Service Management Event |
|---|---|
| | Practice/Microprocess Planning |
| Sprint Planning | Sprint Planning |
| The Sprint | The Sprint |
| Daily Scrum | Process Standups |
| Sprint Review | Sprint Review |
| Sprint Retrospective | Sprint Retrospective |

**Open full table in browser:**
https://devops.turtl.co/story/the-agile-service-management-guide/page/13/1

### Timeboxes

A Timebox is the maximum duration for each event. The timebox range depends on the length of the Sprint (usually less than one month).

| Event | Timebox |
|---|---|
| Planning Event | Not timeboxed |
| Sprint Planning Event | 2 to 4 hours |
| The Sprint | 2 to 4 weeks |
| Process Standups | 15 minutes |
| Sprint Review | 2 to 4 hours |
| Sprint Retrospective | 1.5 to 3 hours |

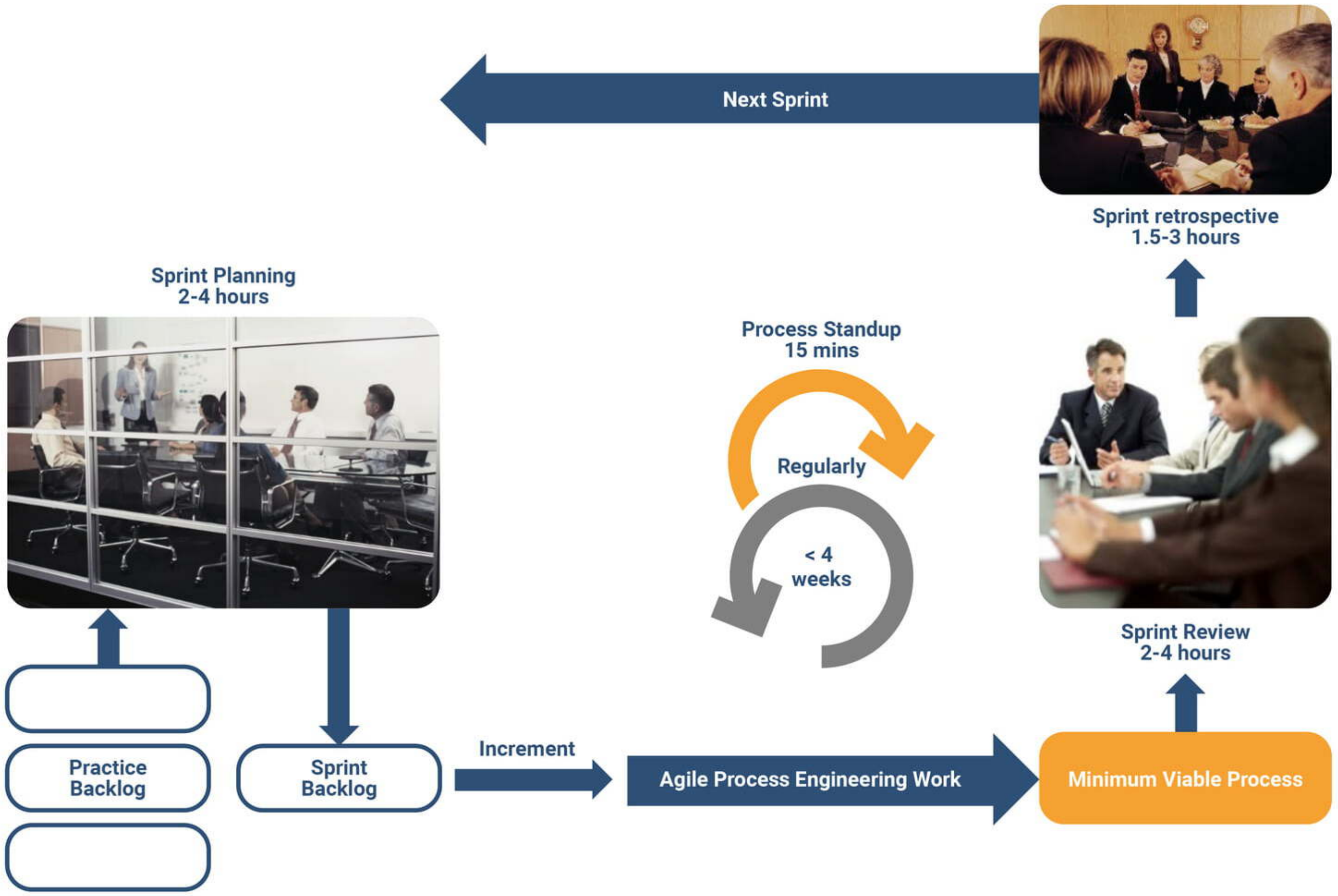
**Open full table in browser:**
https://devops.turtl.co/story/the-agile-service-management-guide/page/13/1

### Practice/Microprocess Planning

Deming's Plan-Do-Check-Act model starts with "Plan" for good reason – planning ensures that the value and critical elements of a service management practice or microprocess are considered, discussed, resourced, and agreed upon. Planning events fall into two primary categories: Practice Planning and Microprocess Planning. Regardless, planning events are not optional in Agile Service Management and are essential to ensuring that "minimum viable or "just enough" are always in the spirit of the plan. One of the most important outcomes of any planning event is the metrics that will be used to measure the progress or value of a practice or microprocess.

## Practice Planning

Service management practices (e.g., IT Change Management) must be planned in order to understand and create the underlying microprocess architecture. This is particularly important in order to understand and map the order in which each microprocess will be designed and the relationships between the microprocesses both within and outside the same service management practice. Practice or microprocess planning should stay focused on high-level strategic aspects and would likely span multiple events. The result should be one or more Strategic Sprints. Attendees at Practice/Microprocess Planning events would likely be more managerial than practitioner and must include business stakeholders and IT management since areas such as service levels, resource allocation, business value, timelines,



and financing would likely be discussed.

Some of the areas to address during Practice/Microprocess Planning include:

- The business value and benefit of the practice
- Goals, objectives, inputs, and outputs of the processes or microprocesses
- The microprocess architecture
- Prioritization of microprocess design
- Expected integration and dependencies with other practices and microprocesses
- Stakeholders
- Automation opportunities (existing and potential)
- Regulatory, governance, or policy requirements
- Major risks
- The "minimum viable" or "just enough" level to meet goals
- Definition of Done
- Metrics

The output of an early Practice Planning event should be a Practice Definition Document for the specific service management practice. It should be concise, easy to understand, and include a preliminary map of the microprocess architecture for that service management practice.

## Microprocess Planning

The primary goal of Microprocess Planning would be to answer this question:

"What is the least amount of effort available for this microprocess to deliver value to our organization and our service management practices?"

Microprocess planning is likely faster and less complex than practice planning since the focus is on a single well-defined activity. Microprocesses are small, easier to understand, visualize and therefore may be easier to identify a "minimum viable" plan for that activity.

Some other areas to be planned for a single microprocess would include:

- What are we trying to achieve and why? (Value)
- Roles and responsibilities
- Timelines
- Relationship and dependencies with other microprocesses and service management practices
- Constraints
- Policies and consequences
- Major risks
- Communication
- Metrics

One of the risks of a microservice architecture is that each microprocess becomes more complex or cumbersome than necessary. Remember, the goal is to define a "just enough" state for any service management practice built upon an architecture of agile microprocesses that contribute to the overall speed and quality of the practice and services. If each microprocess is not easy to execute, the entire service management practice will become unwieldy and bureaucratic.

Microprocess planning should not be done in isolation with the Agile Service Management

Team. Good planning requires the Agile Service Management Team to actively solicit input, ideas, and feedback from those that will execute the microprocess under a variety of circumstances. Whereas Practice Planning usually engages management, microprocess planning should include IT and business practitioners. They are the best resources for identifying "minimum viable" or "just enough", particularly when they are invited to be part of the solution. The same stakeholders should be invited to Sprint Reviews for microprocess development.

Planning events are not timeboxed but should take on the same Agile Service Management spirit of "just enough".

## Sprint Planning

Sprint Planning is timeboxed for 2 to 4 hours which demonstrates the importance of proper Sprint Planning. The Agile Service Manager facilitates the event and the Agile Practice Owner describes the next Increment or microprocess to be completed. The entire Agile Service Management Team collaborates on planning the details of the next Sprint.

The primary purpose of Sprint Planning is to:

- Establish the Sprint Goal
- Define what increment of the Practice Backlog will be completed during the Sprint
- Determine how the increment will be done
- Ensure that the Agile Service Management Team has all of the necessary skills and resources for this Sprint
- Consider any automation opportunities or requirements
- Define any dependencies or integrations with other microprocesses or practices
- Clarify the Definition of Done for each backlog item and the Sprint itself
- Create a Sprint Backlog

Inputs to Sprint Planning include the Practice Backlog, the past velocity of the Agile Service Management Team, the availability of team members and the dependencies on other processes and tools. Only the Team can determine how much it can accomplish during the next Sprint.

Sprint Planning is also where the Team begins to self-manage by determining how they will accomplish the Sprint Goal. They plan their approach and prioritize the items going into the Sprint Backlog. By the end of Sprint Planning, the Team should be able to articulate what they are going to accomplish and how they are going to do it.

## The Sprint

A Sprint is a fixed period of less than 4 weeks during which the work needed to meet the Sprint Goal is performed. Other events including Sprint Planning, Process Standups, Sprint Review and Sprint Retrospective all happen within the Sprint.

The Team builds an increment from the Sprint Backlog items that were agreed to during Sprint Planning. The Sprint is guided by the Sprint Goal and the Definition of Done.

During the Sprint, the Agile Service Manager keeps the Team focused, coaches the members and stakeholders on Scrum and service management practices and protects the Team from outside distractions. The Agile Service Manager also removes impediments whenever possible. The Agile Practice Owner ensures that no one else attempts to change the Team's priorities or tasks during the Sprint.

No changes can be made to the Increment during the Sprint that would endanger the Sprint Goal. Progress is inspected during the Process Standups. The Agile Practice Manager can clarify any questions about the scope of the Sprint.

Agile Service Management embraces the Scrum principle of being iterative and incremental. Every Sprint is considered an iteration that progresses the service management practice forward in an incremental way. When one iteration is completed, another is planned and repeated until all increments of the practice or microprocess are done.

## Sprint Types

Springs can be used for many different aspects of engineering processes including:

- Designing microprocesses and components
- Identify requirements
- Communication and Training
- Benchmark current performance
- Documentation
- Tools and Automation
- Assess performance and continually improve
- Implementation

To ensure that all aspects of the practice or microprocess are considered, Agile Service Management defines three basic types of Sprints.

### Sprint Type 1: Strategic Sprint

A Strategic Sprint is a timebox committed to working on the critical high-level items in the Practice Backlog. These are usually the output of Practice Planning. These items do not usually appear on flowcharts, architectures, value stream maps or other artifacts. Yet they are essential for the service management practice and its microprocesses to be effective, efficient and sustainable.

Items accomplished during Strategic Sprints can include:

- Creating a concise Practice Definition Document to establish measurable goals and objectives for the service management practice
- Planning and allocating resources (human, technical and financial)
- Inventorying and assessing existing tools

- Creating new or updating existing policies
- Conceiving the microprocess architecture or defining relationships and dependencies
- Identifying and mapping stakeholders and practitioners
- Drafting communication plans

Strategic Sprints follow the rules of any other type of Sprint. They are guided by a Sprint Goal, agreed Definition(s) of Done and produce an Increment that is demonstrated during a Sprint Review.

Strategic Sprint iterations are not necessarily sequential. Strategic Sprints can be planned when they make sense to do so but always after the first Practice Planning event. Planning integrated Strategic Sprints for related practices or microprocesses may help to ensure alignment and integration.

## Sprint Type 2: Increment Sprint

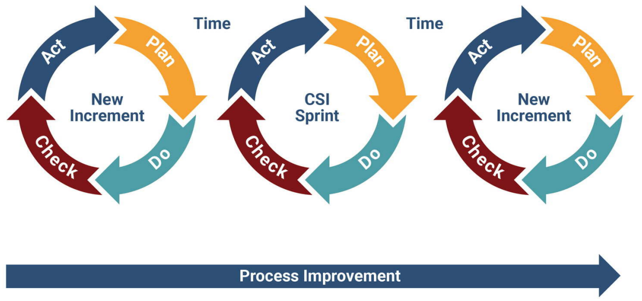
An Increment Sprint is planned in order to complete an Increment that could be a whole microprocess, a piece of a microprocess or epic or an improvement or correction to a practice or microprocess based on Practice Backlog user stories.

The Increment to be completed during an Increment Sprint should be identifiable, logical and be considered potentially shippable depending on its nature. It should not be a random collection of Practice Backlog items that are not interrelated. There should be a clear Definition of Done and Sprint Goal for an Increment Sprint.

## Sprint Type 3: Continual Service Improvement (CSI) Sprint

A CSI Sprint commits a cycle of work to implementing prioritized improvements from the Practice Backlog for a service management practice or a microprocess.

A CSI Sprint is usually undertaken as part of Agile Process Improvement. It is an opportunity to adapt the input and feedback from prior sprints, stakeholders, practitioners and customers. A CSI Sprint is a good opportunity to level the service management practice or microprocess to a "just enough" level. Sprint Planning for a CSI Sprint is particularly important in order to discuss, resource and prioritize improvements.



CSI Sprints should be regularly planned throughout the lifecycle of the practice or microprocess to maintain or increase agility and to ensure that services are being managed according to speed and quality expectations of the end consumer.

One caution: Change fatigue can occur when too many changes are made to a practice or microprocess in rapid succession. The risk is particularly strong after a new microprocess or practice is introduced or re-engineered. It is extremely important to allow time for people to adapt to a new way of working with ample time for feedback and ideas. While there will likely be no shortage of comments or suggestions about a newly implemented process, it is important to collect and analyze all input and feedback to recognize patterns and anti-patterns.

Defining different types of Sprints is offered solely for the purpose of ensuring that all aspects of the service management are addressed. There is no limit to the number or frequency of each type of Sprint or the order in which they are done. There may also be other Sprint cycles that do not fall into a particular type and are just iterations to progress the service management practice forward.

## Process Standups

The Process Standup is timeboxed for 15 minutes during an active Agile Service Management sprint. It is not a status meeting but a frequent opportunity to inspect progress towards the Sprint Goal and identify impediments as quickly as possible. The term "Standup" demonstrates that the event is intended to be short and not necessarily seated.

Because the Team is not necessarily working on the Agile Service Management sprint full-time, the timing of Process Standups may not be daily but should at least be held weekly.

During the Process Standup, each Team member in turn shares:

- What they have accomplished since the last standup
- What they are going to do before the next standup
- What obstacles are in his/her way

While observers and stakeholders may attend, Team members are the only ones allowed to speak. Questions are not allowed during the timebox. The Agile Service Manager facilitates the event.

The importance of a Process Standup should not be undervalued – the faster deviations and impediments are identified, the greater the opportunity to meet the Sprint Goal and get more done. Fifteen minutes during an active Sprint is usually time well spent.

## The Sprint Review

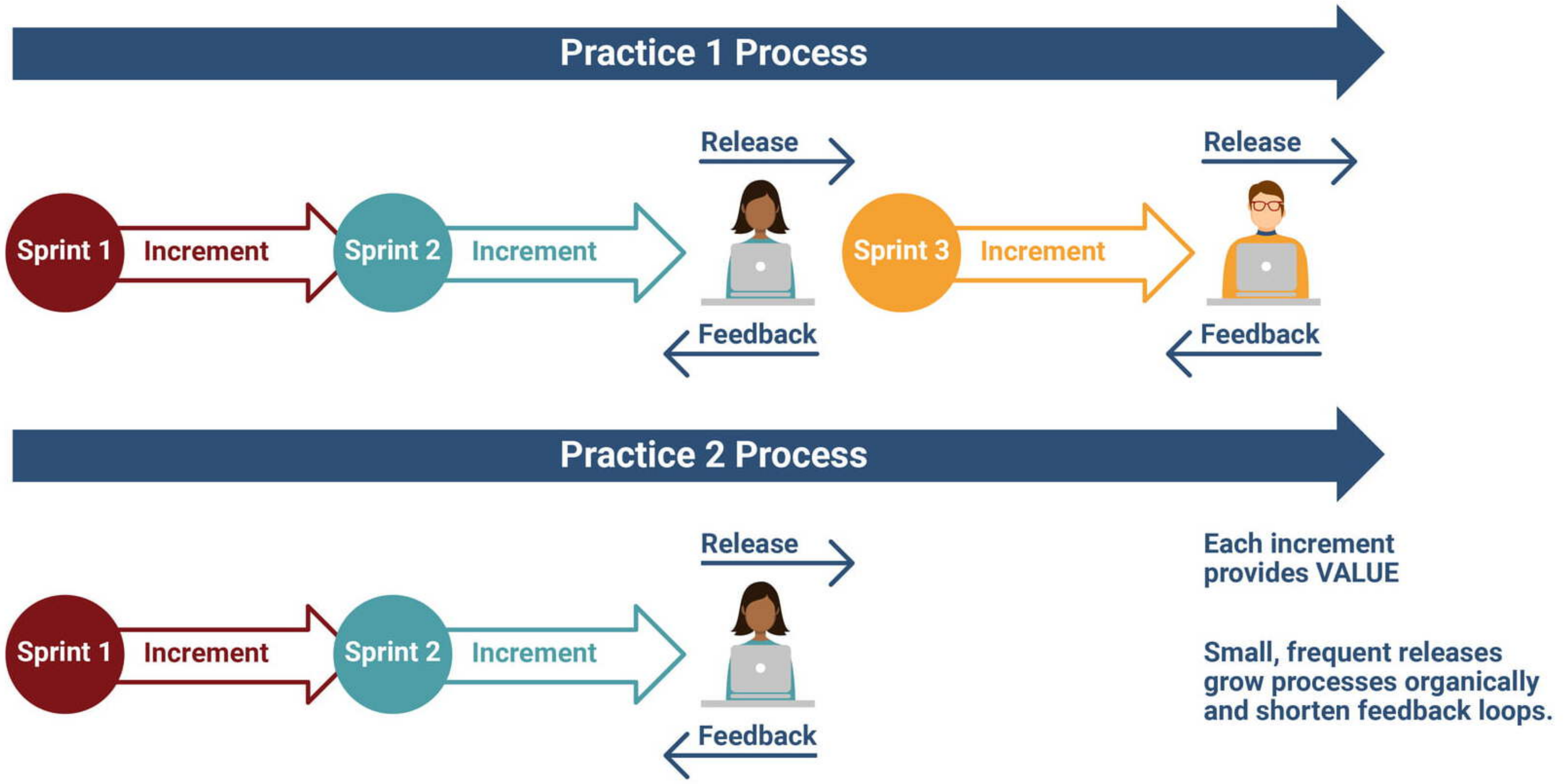
The Sprint Review is timeboxed for 2-4 hours and is attended by the Team and stakeholders or customers. It is an important opportunity for transparency, inspection and adaptation (the pillars of Scrum). It is facilitated by the Agile Service Manager.

During the Sprint Review, the Team demonstrates the aspects of the practice or microprocess that were engineered during the last Sprint. The Team shares the challenges they faced, successful resolutions and outstanding issues. The Agile Practice Owner explains the current state of the practice or microprocess and the Practice Backlog. The Agile Practice Owner also describes any feedback received from practitioners about any previously released Increments. A decision on whether the current Increment will be released is made.

The Sprint Review allows the Team and stakeholders to discuss the next steps for the microprocess or practice as input to the next Sprint Planning.

## Should An Increment or Microprocess be Released?

One of the key decisions made during the Sprint Review is whether or not to release the Increment. While releasing aspects of service management processes incrementally gives the organization time to adopt and adapt to new



behaviors, there are several considerations that should be discussed including whether

The increment stands alone on its own merit or enhances or improves another increment:

- The organization is ready and receptive
- It won't confuse practitioners
- It does not add extra toil while waiting for automation
- It delivers business value
- There is no risk to its dependencies and related practices or microprocesses
- Metrics have been adjusted if needed
- The Increment will not affect the accuracy or validity of data or reporting
- It does not contribute to "change fatigue"

Some of the benefits of releasing an Increment include:

- Changing organizational behaviors one increment at a time (maybe making ITSM easier)
- Capturing more data or information
- Shortening feedback loops and using feedback to influence future Increments
- Helping the practice adapt to changing requirements as it is slowly being matured
- Identifying and aligning dependencies on other processes
- Encouraging an integrated approach to service management
- Keeping tools relevant and updated
- Finding the right level of "just enough" ITSM

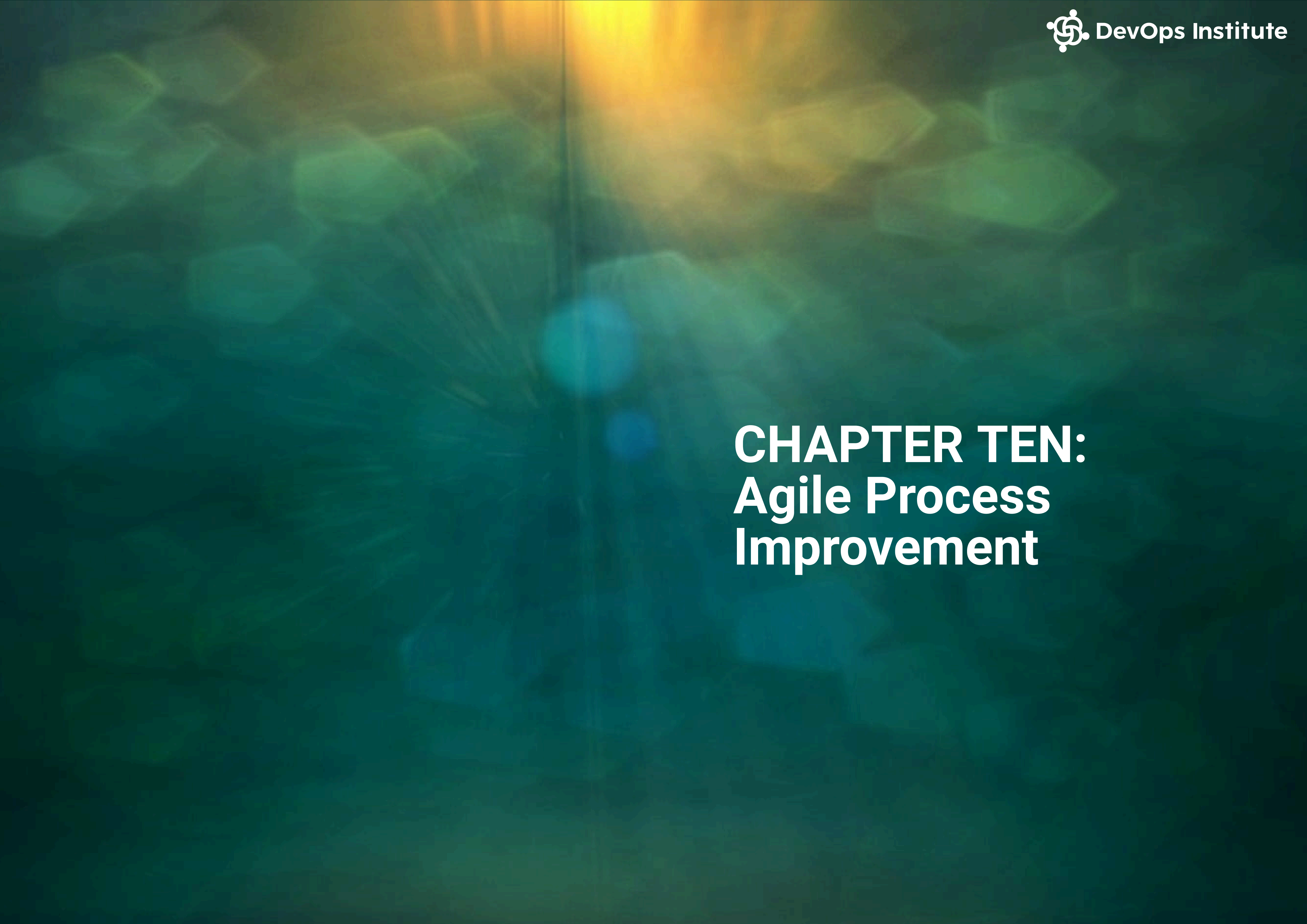
## Sprint Retrospective

The Sprint Retrospective is an internal opportunity for the Team to reflect on and inspect the progress and organization of the last Sprint. In some ways, it resembles the form and format of a blameless post-mortem review in that it addresses:

- What did we do right?
- What could we have done better?
- What have we learned?
- What will we do differently next time?
- In the spirit of continual improvement, the Team also discusses
- Team composition and skill sets
- Tools
- Logistics
- The Definition of Done
- Internal and external communications
- Input and feedback from stakeholders
- Velocity

The Sprint Retrospective is timeboxed for 1.5 to 3 hours and is facilitated by the Agile Service Manager. While the temptation may be to go from the Sprint Review directly into the next Sprint Planning, it is important for the Team to take the time to review and discuss their past performance. Incremental improvements will absolutely increase the Team's maturity and velocity.

Sprint retrospectives have to be inclusive and blameless. Everyone's input and feedback is important so long as it is constructive and well-intentioned.



# CHAPTER TEN: Agile Process Improvement

# Agile Service Improvement

Not surprisingly, engineering agility into a service management practice the first time around is easier than maintaining a "just enough" structure in the long term. If left unchecked, microprocesses can become complex and bureaucratic over time. There is also a risk that people will revert to old ways and potentially there will be "not enough" or too many variations on the microprocess architecture to meet the organization's goals. The leap from "just enough" to "too much" process can seem to happen almost overnight. The drift can result in confusion, more toil, delays, and lower quality services.

Since Agile Service Management is loosely based on Scrum and its basis in empirical thinking, Agile Process Improvement aligns with Scrum's three pillars of Transparency, Inspection, and Adaptation.

Agile Process Improvement's goal is to ensure that service management practices and microprocesses are not:

- Bureaucratic
- Unclear
- Constrained
- Time-consuming
- Irrelevant
- Circumvented
- Nice on paper, but…

Remember, Agile Service Management practices and processes are purely theoretical until someone actually performs the tasks. It is therefore extremely important to plan a review cycle while keeping an open line of communication with practitioners. Those that are actually doing the work and are measured for their personal success are the best surveyors of how the practice or microprocess is delivering value in the form of quality IT services.

Agile Process Improvement can be applied to any service management practice or microprocess regardless of the frameworks or methods used.

## Agile Process Improvement Reviews

Agile Practice Owners should plan regular surveillance reviews of their service management practices and microprocesses. The Agile Service Manager would facilitate the review and provide guidance on Agile, ITSM, and Agile Service Management best practices.

The review is an important opportunity to connect again with stakeholders, practitioners, and the Agile Service Management Team. In some cases, a formal audit demonstrating control or for ISO/IEC certification purposes may be necessary but Agile Process Improvement no longer addresses the topic of formal ITSM audits.

## The Goals of Agile Process Improvement

- Identify and eliminate waste and bottlenecks
- Detect practice or microprocess drift
- Benchmark against Agile values and principles
- Scale up or back to “just enough” structure and control
- Ensure alignment with other microprocesses and practices
- Confirm ongoing relevance and value creation
- Ensure ease of use
- Solicit input and feedback from practitioners and stakeholders
- Improve effectiveness, efficiency, and agility

Agile thinking encourages us to value customer interactions over processes and tools. Agile Process Improvement embraces that spirit by ensuring that the voice of the customer is instilled into improvement efforts. Focus groups, surveys, and face-to-face discussions are the best way to draw out suggestions, challenges, and insight. It is also a great way to clearly identify waste and the manual, repetitive work known as toil.

Be sure to include a cross-section of business and IT managers, practitioners, and suppliers who regularly engage with the practice or microprocess.

Sample questions include:

- Is this practice or microprocess simple to understand and apply? If so, how? If not, why?
- Is it timely?
- How does it help (or hinder) your job performance?
- How does it enable the management and quality of services?
- Is it relevant to the current business environment?
- How does it work with other practices or microprocesses?
- Are there any manual tasks that you think we can or should automate?
- How can we improve the flow, value or ease of use of this practice or microprocess?

Conducting a 360-degree surveillance will help the Agile Practice Owner understand how to keep or improve the value of the process in the management of IT services. The Agile Service Manager would facilitate the reviews and assist the Agile Practice Owner in collecting and evaluating the output in line with Agile values and principles and ITSM guidance.

The improvement suggestions collected as part of the Agile Process Improvement review should take the form of User Stories and be added to the Practice Backlog for prioritization and potential inclusion in upcoming CSI Sprints.

One of the key advantages of Agile Service Management is that the microprocess architecture makes conducting reviews easier and more accessible than in traditional end to end ITSM processes.

Thinking of microprocess reviews as more of a check-in than an audit helps the Agile Practice Owner develop a reasonable and achievable review cycle, particularly when there are a greater number of microprocesses in use.

There are certain events or situations that would be appropriate for an Agile Process Improvement review:

- About a month or two after a microprocess release
- Quarterly for microprocesses, annually for service management practices
- When a related microprocess is released
- When automation is being considered or introduced for one or more tasks
- When a formal audit is required
- Whenever it is beneficial to do so

## Sustaining Improvements

It is important to take a holistic approach when making improvements sustainable. Start by baselining "as is" performance so that performance changes can be monitored.

Feedback should be solicited when new microprocesses are released or changes to existing microprocesses are made. Actively listen to and act on complaints and suggestions. Communicate successes, failures and lessons learned both within the Team and to stakeholders.

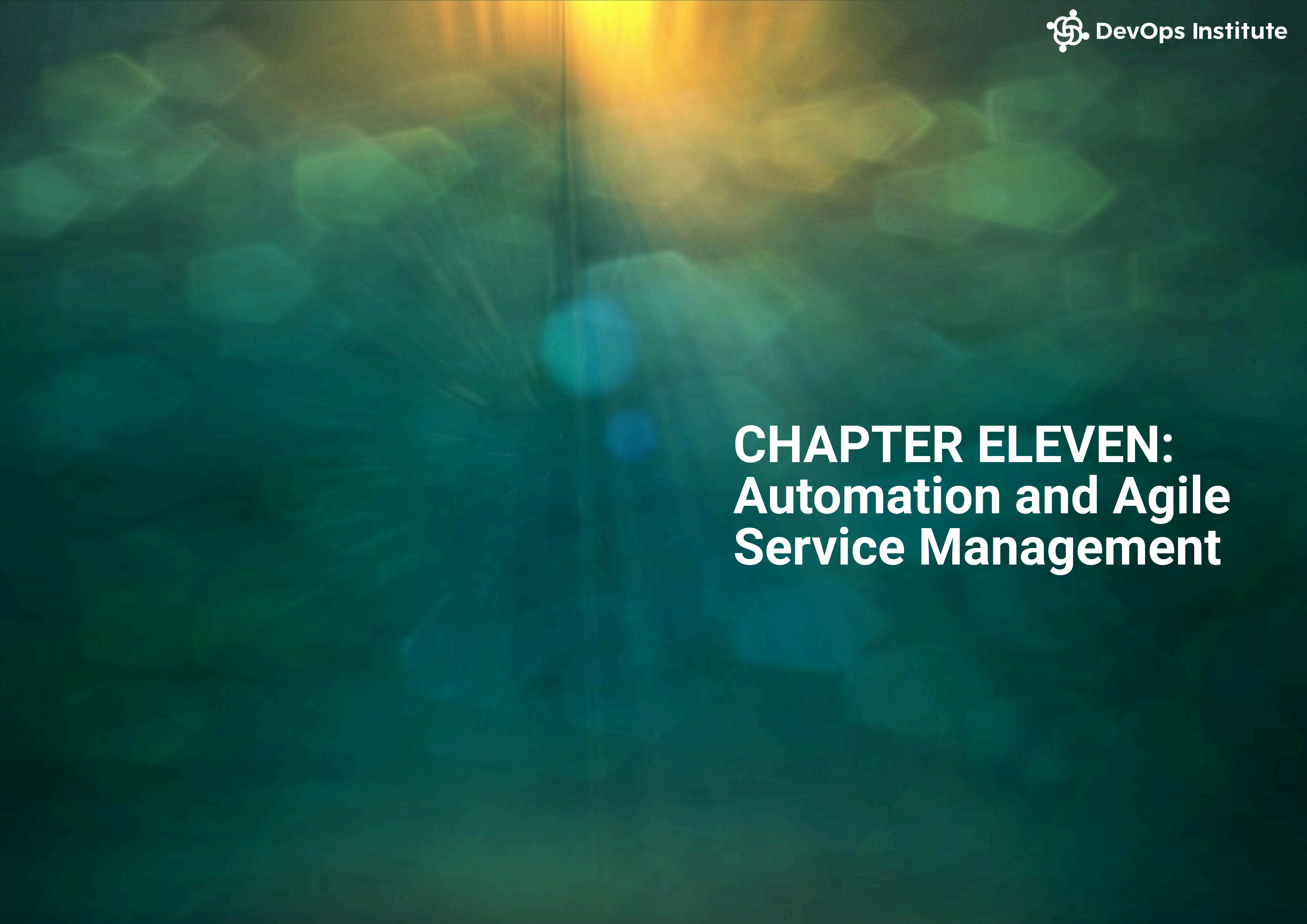
Clear roles and responsibilities and tools that match new or existing rules are essential.

Other ways to sustain Agile Process improvements include:

- Continue allocating resources to continual improvement
- Hire, promote and develop employees and managers who support the vision
- Provide ongoing training to existing employees
- Conduct regular reviews and audits
- Innovate – take calculated risks
- Claim and promote the wins

Agile Process Improvement is an essential element of continuous service improvement and scaling for ITSM. A service management practice could be effective (gets the job done), efficient (least amount of effort) but not necessarily agile (flexible, adaptable, plays well with others). Agile Process Improvement ensures that each of these are reviewed against all three criteria to maintain a "just enough" level.

DevOps Institute



# CHAPTER ELEVEN: Automation and Agile Service Management

## Automation and Agile Service Management

Automation reduces toil by consistently performing repetitive tasks that allows humans to do more productive, innovative and satisfying work such as weighing evidence, solving problems, making decisions and using their skills, judgment and experiences.

Agile Service Management invites automation into the Agile Service Management Team to help instill an engineering mentality when defining and designing practices and microprocesses. Automation expedites the identification of “just enough” structure and control, particularly when the automation produces the results without the human effort. Automation not only facilitates Agile Service Management’s alignment with other practices such as Continuous Delivery, DevOps and Site Reliability Engineering, the use of consistent and automated controls may also help demonstrate governance, risk management and compliance.

Automation also supports:

- More frequent releases
- Fewer errors
- Higher quality IT services
- Improved security and risk mitigation
- Faster recovery from incidents
- Increased business and customer satisfaction

Intelligent automation requires intelligent microprocesses. The Agile Service Manager should assess tools and automation across the entire organization to identify ways to optimize and integrate existing tools. Avoiding replication of effort and engaging others in dialogue about similar goals and mutual enablement also supports cultural transformation.

Most of the ITSM tools have also recognized the need to be more agile and have introduced integrations, enhancements or new features that make the work of IT easier and faster. The rise of artificial intelligence and machine learning are helping these efforts.

Some automation opportunities to consider:

- ITSM suites (some of which have Agile modules)
- Automated builds, testing, staging and deployment
- Monitoring, observability and event management
- Dashboards
- Metrics and analytics
- Cloud native applications and cloud

infrastructure
- Flowcharts and drawing
- Kanban tools
- Value Stream Management tools
- Product Management to track user stories and the Product/Practice Backlog

Rely on the knowledge and experience of those that do a job when assessing tool selection. Be careful to avoid discussions that turn into a “favorite tech” talk. This can lead to implementing multiple tools that do the same thing because of personal preferences. While people will typically be more productive with tools they know, the automation landscape is constantly changing so it is unrealistic to expect that the tools in use today will be the same tools in use for years to come.

Despite all of the above, it is important to note that technology alone will not instill agility into an organization particularly as it relates to service management. As the Agile Manifesto reminds us, we should not prize the artifacts (tools) over the outcomes (higher quality services).

# CHAPTER TWELVE: Getting Started

# Getting Started

Agility does not happen overnight. Regardless of which Agile, DevOps, SRE, or ITSM frameworks that an organization adopts, moving from a traditional ITSM approach to an Agile Service Management approach is a journey that takes practice and perseverance. Humans require time to absorb change, normalize and excel at new ways of working. It is therefore essential that the Scrum values of Commitment, Focus, Respect, Openness, and Courage are embraced and demonstrated in Agile Service Management environments.

## Value Stream Management

The starting point of any Agile, DevOps, or service management initiative should be the mapping, negotiation, and agreement of the organization's value streams. This increases everyone's awareness of how value is delivered to customers in the form of services. Not only does Value Stream Management increase flow by helping to eliminate waste, delays, or bottlenecks, it is an excellent engagement tool for overcoming some of the cultural constraints that have plagued development, operations, security, ITSM, and the business. Aligning Value Stream Management with service management is a solid recipe for success and can instill a common mindset, common processes, common vocabulary, and a common understanding of what value means to the end customer.

## How Many Service Management Practices and Microprocesses Are Needed?

This is a tricky question – and of course, the obvious answer is "it depends". Version 2 of ITIL® described 11 processes that grew to 26 in version 3 and 34 in ITIL® 4. The Site Reliability Engineering books describe XXX of key practices. Does an enterprise have to implement all? Of course not. Most enterprise IT service management programs focus on 10-15 core practices that have the highest impact on the quality of service. Please avoid process for process' sake.

In my opinion, essential service management practices include

- Service Level Management
- IT Change Management
- Release Management
- Configuration Management
- Incident Management
- Problem/Root Cause Management
- Event Management/Monitoring
- Request Management
- Capacity and Performance Management
- Knowledge Management
- Information Security Management
- Business Continuity Management

**Start simple and stay simple.** Pick one service and one service management practice. Identify an Agile Practice Owner, Agile Service Manager, the Agile Service Management Team, stakeholders, and practitioners. Capture user stories into a Practice Backlog. Plan your Sprints. Experiment and learn. Measure constantly. Engage with your customer community and encourage feedback. Continually look for ways to reduce toil through engineering. Create a cycle of review and refinement. Avoid bureaucracy and change fatigue.

**Don't rush.** Start with a Minimum Viable Process and move forward from there. Introduce the new or improved practice in small, frequent increments. Give the organization time to absorb, adopt and adapt to new behaviors. Mature the processes holistically and organically. Small, short-term wins will deliver greater wins in the long term.

**Most importantly, remember – regardless of what service management framework you adopt or customize, it is always more important to "be agile" than "do Agile."**

# APPENDIX: Agile Service Management Taxonomy

# Agile Service Management Taxonomy

| TERM | DEFINITION |
|---|---|
| **A** | |
| **Agile** | A project management method for complex projects that divides tasks into small “sprints” of work with frequent reassessment and adaptation of plans. |
| **Agile Manifesto** | A formal proclamation of four key values and 12 principles to guide an iterative and people-centric approach to software development. |
| **Agile Process Engineering** | The aspect of Agile Service Management (Agile SM) that applies the same Agile approach to process design as developers do to software development. |
| **Agile Process Improvement** | The aspect of Agile SM that aligns Agile values with ITSM processes through continuous improvement. |
| **Agile Service Management (Agile SM)** | Agile Service Management (Agile SM) ensures that IT service management processes reflect agile values and are designed with “just enough” control and structure to enable the delivery of services that enable the ability to do something when and how they are needed or desired. |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/1

| TERM | DEFINITION |
|---|---|
| **Agile Service Management Team** | A team of at least 3 people (including a customer or practitioner) that is accountable for a single microprocess or a complete service management practice. |
| **Agile Service Manager** | An Agile Service Management subject matter expert who is the coach and protector of the Agile Service Management Team. |
| **C** | |
| **Continuous Delivery** | A software development practice where software is always in a releasable state. |
| **D** | |
| **Definition of Done** | A shared understanding of what it means for work to be complete. |
| **Devops** | A cultural and professional movement that stresses communication, collaboration and integration between software developers and IT operations professionals. |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/2

| TERM | DEFINITION |
|---|---|
| **I** | |
| **Increment** | Potentially shippable completed work that is the outcome of a Sprint. |
| **ITIL®** | Set of best practice publications for IT service management. Published in five core books representing the five stages of the IT service lifecycle: Service Strategy, Service Design, Service Transition, Service Operation and Continual Service Improvement. |
| **INVEST** | A mnemonic was created by Bill Wake as a reminder of the characteristics of a quality user story |
| **K** | |
| **Kanban** | A method for visualizing and communicating workflow in order to reduce or eliminate work in progress. |
| **L** | |
| **Lean Thinking** | The goal of lean thinking is to create more value for customers with fewer resources and less waste. Waste is considered any activity that does not add value to the process. |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/3

| TERM | DEFINITION |
|---|---|
| **M** | |
| **Microprocess** | A distinct activity that can be defined, designed, implemented and managed independently and is generally associated with a primary service management practice. A microprocess may be integrated with other service management practices. |
| **Microprocess Architecture** | A collection of integrated microprocesses that collectively perform all of the activities necessary for an end-to-end service management practice to be successful. |
| **Minimum Viable Product** | The most minimal version of a product that can be released and still provide enough value that people are willing to use it. |
| **P** | |
| **Plan-Do-Check-Act** | A four-stage cycle for process management and improvement attributed to W. Edwards Deming. Sometimes called the Deming Cycle or PDCA. |
| **Practice Backlog** | A prioritized list of everything that needs to be designed or improved for a process including current and future requirements. |
| **Practice Owner** | Role accountable for the overall quality of a service management practice and owner of the Practice Backlog. |

**Open full table in browser:**
https://devops.turtl.co/story/the-agile-service-management-guide/page/17/4

| TERM | DEFINITION |
|---|---|
| **Practice/Microprocess Planning Meeting** | A high-level event to define the goals, objectives, inputs, outcomes, activities, stakeholders, tools and other aspects of a process. This meeting is not timeboxed. |
| **Process** | Interrelated work activities that take specific inputs and produce specific outputs that are of value to a customer. |
| **Process Customer** | A recipient of a process' output. |
| **Process Standups** | A daily timeboxed event of 15 minutes or less for the Team to re-plan the next day of work during a Sprint. |
| **Product Backlog Refinement** | An ongoing process of adding detail, estimates and order to backlog items. Sometimes referred to as Product Backlog grooming. |
| **Product Owner** | An individual who manages the Product Backlog and ensures the value of the work that the Team performs. |
| **R** | |
| **Release Planning Meeting** | A non-timeboxed event that establishes the goals, risks, features, functionality, delivery date and cost of a release. It also includes prioritizing the Product Backlog. |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/5

| TERM | DEFINITION |
|---|---|
| S | |
| **Scrum** | A simple framework for effective team collaboration on complex projects. Scrum provides a small set of rules that create “just enough” structure for teams to be able to focus their innovation on solving what might otherwise be an insurmountable challenge. |
| **Scrum Components** | Scrum’s roles, events, artifacts and the rules that bind them together. |
| **Scrum Guide** | The definition of Scrum concepts and practices, written by Ken Schwaber and Jeff Sutherland. |
| **ScrumMaster** | An individual who ensures that the Team adheres to Scrum practices, values and rules. |
| **Scrum Team** | A self-organizing team consisting of a Product Owner, Development Team and ScrumMaster. |
| **Scrum Values** | A set of fundamental values and qualities underpinning the Scrum framework:; commitment, focus, openness, respect and courage. |
| **Self-organizing** | The management principle that teams autonomously organize their work. Self-organization happens within boundaries and against given goals. Teams choose how best to accomplish their work, rather than being directed by others outside the team. |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/6

| TERM | DEFINITION |
|---|---|
| **Service Management Practice** | A complete end to end capability for managing a specific aspect of service delivery (e.g., changes, incidents, service levels). |
| **Sprint** | A period of 2-4 weeks during which an increment of product work is completed. |
| **Sprint Backlog** | Defines the work that must be completed during the Sprint. |
| **Sprint Goal** | The purpose and objective of a Sprint, often expressed as a business problem that is going to be solved. |
| **Sprint Planning Meeting** | A 4-8 hour timeboxed event that defines the Sprint Goal, the increment of the Product Backlog that will be done during the Sprint and how it will be done. |
| **Sprint Retrospective** | A 1.5-3 hour timeboxed event during which the Team reviews the last Sprint and identifies and prioritizes improvements for the next Sprint. |
| **Sprint Review** | A timeboxed event of 4 hours or less where the Team and stakeholders inspect the work resulting from the Sprint and update the Product Backlog. |
| **Strategic Sprint** | A 2-4 week timeboxed Sprint during which strategic elements that were defined during the Process Planning Meeting are completed so that the Team can move on to designing the activities of the process. |

**Open full table in browser:**
https://devops.turtl.co/story/the-agile-service-management-guide/page/17/7

| TERM | DEFINITION |
|---|---|
| T | |
| **Timebox** | The maximum duration of an event. |
| U | |
| **User Story** | A statement written from the user's business perspective that describes how the user will achieve a goal from a feature of the product. User stories are captured in the Product Backlog. |
| V | |
| **Velocity** | How much Product Backlog effort a team can handle in a single Sprint. |

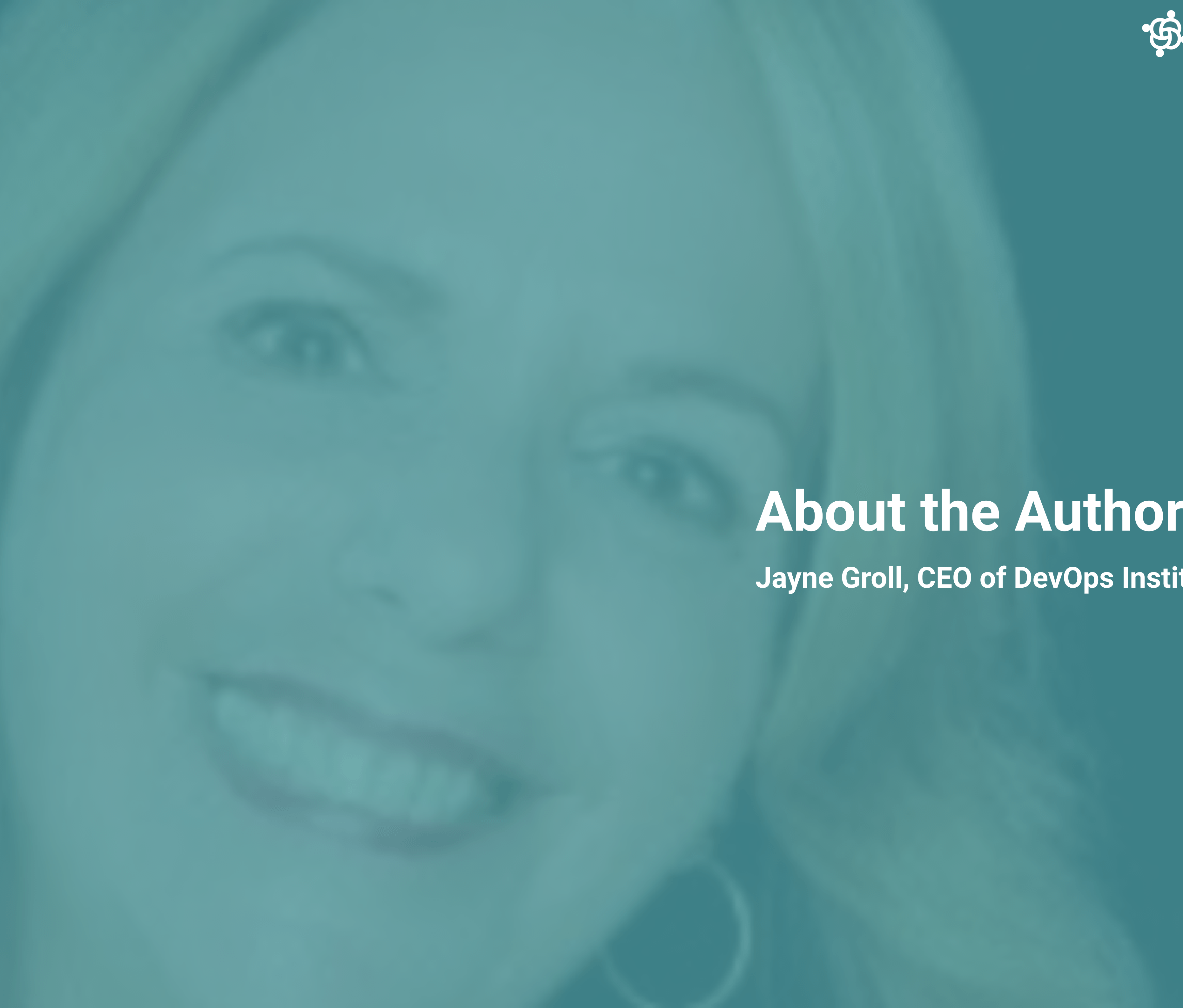
**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/8

| TERM | DEFINITION |
|---|---|
| **W** | |
| **Waste** | Any activity which does not add value to a process. |
| **Waterfall** | A linear and sequential approach to software development |

**Open full table in browser:**

https://devops.turtl.co/story/the-agile-service-management-guide/page/17/9



## About the Author

Jayne Groll, CEO of DevOps Institute

## About the Author

**Jayne Groll**

Jayne Groll is co-founder and CEO of the DevOps Institute whose mission is to advance the Humans of DevOps. Jayne was also one of the founding partners in ITSM Academy, spending over 15 years educating and speaking on IT Service Management, ITIL®, ISO 20000, and process design.

Jayne has had an extensive IT Operations career starting with implementing and managing Service Desks to leading IT organizations across multiple verticals. She achieved her ITIL® V2 Service Manager certification (with distinction) in 2005 and later her ITIL® V3 Expert. She is also a certified ScrumMaster.

Jayne is a recognized thought leader, author, and speaker in the ITSM, DevOps, and SRE communities.

DevOps Institute

DevOps Institute

Thank you for reading

# The Agile Service Management Guide