# Contents

# 1 Introducing Scrum

*This chapter explains how developing novel products for complex markets requires an iterative design and incremental development approach, like Scrum.*



# Introduction

This book is intended for anyone interested in understanding the Scrum framework for agile product development. It serves as an introduction to those forming or joining their first Scrum Team. It provides a useful compendium of techniques to use at each stage of product Discovery and Delivery. And finally, it acts as a reference for anyone who needs to interact with Scrum Teams.

The use of agile approaches for product development continues to evolve as organizations face uncertainty and disruption. Scrum is the most widely adopted agile framework, often together with other methods, such as Extreme Programming (see page 105).

## Learning Scrum in stages

As you begin using Scrum, you will go through development stages, often described using the martial arts term *Shu-Ha-Ri*:

守破離

- **Shu** (follow the rules): during the first stage of adoption, teams should stick to the basic techniques and rules of Scrum; this book is an essential companion to that journey.
- **Ha** (break the rules): in the second stage, teams rise above the basics to work together and drive organization change; Chapters Nine and Ten focus on how teams continually improve.
- **Ri** (form new rules): finally, teams who have mastered Scrum will typically require little outside guidance, such as provided by this book, and instead become role models for other teams.

## Learning Scrum in easy steps

The In Easy Steps series of books is designed to take readers step-by-step through new topics, learning through the experience of doing, rather than forcing readers to work their way through pages of theory and being left to find their own way.

The best way to learn Scrum is to be in a Scrum Team and to experience it first-hand. For those who want to understand Scrum but are not yet in a team, the next best way is to see it play out step-by-step. **Scrum in easy steps** guides you through forming a team and then taking them through the whole product development life-cycle, from Discovery to Delivery and beyond.

To illustrate this, as you work your way through the book you will also be following the experience of a pizza takeaway business – called Dante's – who want to start taking orders online.

All pages that include an example from Dante's, the pizza takeaway used for illustration, will be highlighted with this pizza slice in the margin.

# About Scrum in easy steps

This first chapter shows how the Scrum framework will help you to cope in today's challenging environment. Chapters Two through to Ten walk through Scrum, step-by-step.

For those looking for a **quick reference** to the whole Scrum framework, start with Chapter Eleven.

### Forming your first Scrum Team

One of the biggest shifts you will face when adopting Scrum is how people work together and the changes in role this requires. Chapter Two discusses the three key roles in Scrum: Product Owner, Scrum Master, and the Delivery Team.

### Discovering the product

A key factor in successfully delivering great products is to prepare a good breakdown of the work required. Chapters Three to Five cover how Product Owners work with others to discover, define, and plan delivery of the product through the Product Backlog.

### Delivering the product

The next three chapters step through the typical development cycle (the Sprint), detailing the events, artifacts, and techniques used. Chapter Six covers how a Sprint starts with the Delivery Team agreeing a Sprint goal, and the work required to achieve it.

Chapter Seven covers the typical daily routine of the team as they design, build, and test the product, tracking progress, handling impediments, and re-planning to achieve their Sprint goal.

Chapter Eight shows how the team reviews their completed work with their stakeholders and gets feedback on what should be their next priorities for the Product Backlog.

### Continually improving and scaling Scrum

Just as teams review the product each Sprint, they also need to consider their approach. Chapter Nine explores how Scrum Masters work with the team to continually improve – while Chapter Ten deals with how they support the organization getting better at Scrum, especially at working with multiple teams.

### The Scrum reference

The final chapter wraps up by providing a quick reference of the whole Scrum framework, the roles, the events, the artifacts, and the rules that explain how they interact and work together.

# Why development is hard

Identifying a new product that will solve a problem, and getting it right first time is full of complexity, uncertainty, and risk. There is no single correct approach that will guarantee success.

When you are developing novel products in a new and growing market, there is much that remains unknown. You need to get feedback from potential customers as fast as possible, and follow an approach that allows you to improve and re-release the product as you learn more about what works.

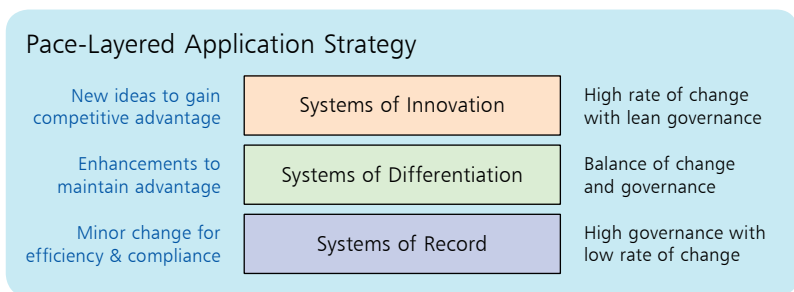
As markets mature, however, the product becomes more defined and uncertainty reduces, at which point you will require a more scalable reliable approach. Should the product ever reach the point of becoming a commodity, you will then tend to focus more on reducing cost by making your processes as simple as possible.

Knowing what product you are developing and what market you are targeting will help guide your product development approach.

## Understanding your product

The **Pace-Layered Application Strategy** was developed to help us choose the right approach to managing our products.

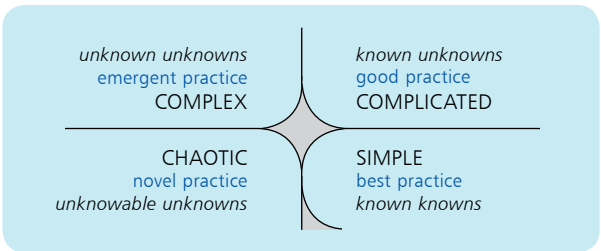

The *Pace-Layered Application Strategy* was developed as a decision support framework by Gartner Inc.




- **Systems of Innovation** are creative responses to new opportunities, and typically require an experimental approach that generates feedback or results quickly.
- **Systems of Differentiation** create a unique selling point for an existing market or product, and require careful thought combined with fast implementation.
- **Systems of Record** handle mission-critical administration and transactions and require standardized processes, operational efficiency, and compliance with governance.

## Understanding your market

The **Cynefin Framework** was developed to help us make sense of our situation and choose the right approach to managing change.





The *Cynefin Framework* was developed as a decision support framework by Dave Snowden. *Cynefin* is a Welsh word meaning habitats or domains.

- In **simple** situations, the connection between cause and effect is clear, and from that the course of action will be obvious – suiting a simpler checklist approach.
- In **complicated** situations, the link between cause and effect has to be uncovered, through expert analysis, to decide the right response – suiting a more thorough plan-driven process.
- In **complex** situations, there are no direct connections between cause and effect – we need to feel our way one step at a time – suiting a more exploratory feedback-driven process.
- Finally, in **chaotic** situations, we need to stabilize our organization urgently – suiting immediate action – typically not a safe place for product development.

## The right approach for your product in your market

Once you have understood your product type and the dynamics of your market, you can select an approach that is best suited to discovering and delivering your product. If the answer involves innovation and complexity, then a checklist or plan-driven approach is not suitable. Scrum is an ideal framework to follow.

However, even if you follow a plan-driven approach for most of your projects, you will find that you will benefit from adopting a feedback-driven approach some of the time.

While Dante's, our pizza company, is in an established market, they are unfamiliar with online ordering. They adopted Scrum so they get fast feedback, learn what works, and improve rapidly.


The use of the term *Scrum* originated from this 1986 article in the Harvard Business Review.

# A feedback-driven approach

The shift from a plan-driven to a feedback-driven approach to product development also requires a shift in mindset:

> "The traditional sequential *relay race* approach to product development – exemplified by phased program planning – may conflict with the goals of maximum speed and flexibility. Instead, a holistic or *rugby* approach – where a team tries to go the distance as a unit, passing the ball back and forth – may better serve today's competitive requirements."
> *Harvard Business Review*

## Plan-driven product development

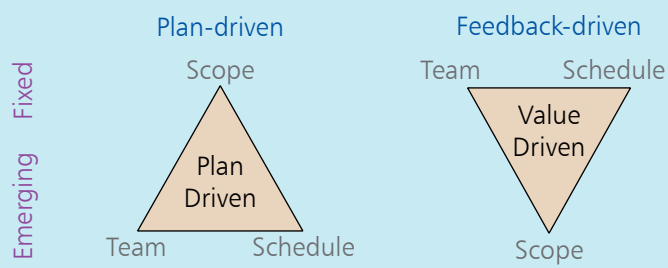
The plan-driven approach to product development starts by defining exactly what the product should do. This scope is signed off as a specification. The team then carries out further work to analyze, then design, then build, and then test a product they believe meets that specification. Finally, they present the finished product which may or may not be accepted as meeting the specification. This approach suits simpler products in stable markets where the requirements are unlikely to change and the technical solution is well-known.

## Feedback-driven product development

The feedback-driven approach to product development starts out with an expectation that scope will evolve and change as the product is progressively developed. Instead, we fix the size of the team and the amount of time we are willing to invest in delivering as many as possible of the prioritized features the customer wants. This approach suits novel products in emerging markets where the requirements cannot be fully known up front, or where the technology is still evolving.

The difference between the two approaches is illustrated below:

The mindset shift



To be effective, this shift also requires some new practices that enable teams to develop with speed, flexibility and feedback:

### The self-organizing autonomous team

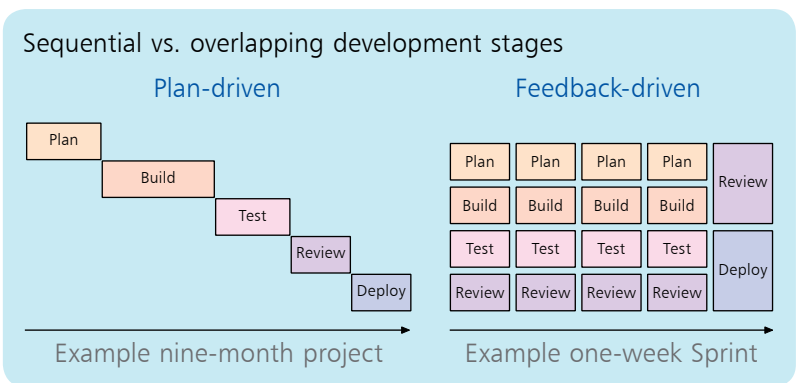
Teams should be free to determine the best technology, architecture, and approach to developing the product. They should think of themselves like a start-up, taking initiative and defining their own identity. To support and enable this level of autonomy, there should also be clear expectations and enough checkpoints to avoid instability, tension, and chaos.

### The learning team

As teams grow and discover how to solve their problems, they will experience a range of learning across levels (e.g. individuals, the team, and the wider organization) and across functions (e.g. analysis, programming, and quality assurance). It is vital that this learning is encouraged and that opportunities are also found to transfer this to people outside the team.

### Overlapping development stages

The team should work together across all activities, testing one feature while they are building a second feature, at the same time as they are planning a third. This is in contrast to the plan-driven approach that makes a team wait for all requirements to be specified in a plan before starting the build, for all build work to be completed before starting any testing, and for all testing to be completed before anything can be reviewed.




This enables them to re-plan or do further build work immediately, when required, rather than waiting for a new project.

# Iterative and incremental

As we explored feedback-driven approaches to product development, we found a blend of approaches more appropriate.

## Cycling back through stages

One of the simplest ideas for incorporating feedback is to repeat as necessary the stages of analysis, design, programming, testing, and integration. Teams should actively pursue feedback from their stakeholders as they go, and if they find they have misunderstood anything, cycle back to the appropriate earlier stage to fix it. However, while this encourages use of feedback, it treats it as an exception rather than something that can be anticipated.

## Iterative design

Iterative design is a deliberately cyclical process of prototyping, testing, analyzing, and refining a product design. Changes are made to the most recent iteration of a design, based on feedback. With iterative design, feedback is encouraged, and repeating the whole cycle is assumed.

In early cycles, there is a greater concentration on analysis and design with a little programming to generate feedback. As work progresses, understanding of the product will stabilize, so that instead of reworking the analysis or design, more time is dedicated to programming and testing.

## Incremental development

Incremental development is a progressive process where the product is built a feature at a time (designed, built, and tested incrementally) until it is finished. In effect, the product is broken down into its component features, each of which is developed and delivered separately. This allows the earlier release of some features which should speed up return on investment.

## Iterative design with incremental development

In order for product development to be truly feedback-driven, it is essential that the product is released progressively and repeatedly. Each release should be complete and usable, each one adding more functionality, until the product is complete.

For this to be effective, you need to prioritize features so that the most useful and valuable ones are developed first. Features must be decomposed into small increments that can be delivered separately.

# The Scrum framework

Scrum is a feedback-driven framework for product development that incorporates iterative design, incremental development, self-organizing teams, and continual improvement.

## The Scrum Team

There are three key roles within the team:

- **Product Owner**: responsible for the business value of the product, selecting *what* gets done and explaining *why*.
- **Delivery Team**: the programmers, testers, analysts, etc. who self-organize to decide *how* the work gets done, and do it.
- **Scrum Master**: responsible for ensuring the team is motivated, productive, and following their working agreement.

## Events in Scrum

There are five events central to the Scrum framework:

- **Sprint**: the block of time within which all work and the events below take place – most often two weeks.
- **Sprint Planning**: in which the team agrees with the Product Owner the functionality to be developed in a Sprint, and then plan the work they need to do in order to deliver it.
- **Daily Scrum**: in which the team look at their progress, the work remaining, any impediments, and re-plan as required.
- **Sprint Review**: in which the team share the progress they have made with their stakeholders and elicit their feedback.
- **Sprint Retrospective**: in which the team consider how well the last Sprint went, and agree possible improvement actions.

## Artifacts of Scrum

There are three core artifacts:

- **Product Backlog**: a prioritized breakdown of the work required to build the desired product.
- **Sprint Backlog**: a sub-set of items from the Product Backlog that the team agrees to complete in a single Sprint, with Tasks.
- **Product Increment**: the product features completed within a Sprint, built on top of product features already released.


The Product Increment should be released at the end of each Sprint. However, Product Owners often choose to wait. For this reason, this is often described as a Potentially Shippable Increment (PSI).

# Summary

- Developing novel products in complex markets is a risky and costly business. It is difficult to plan and then build everything a customer might need or that might delight them.
- The development and support approach to use will depend on the type of product: while established and regulated products might suit a more plan-driven approach, more novel products suit a more exploratory feedback-driven approach.
- The type of market will also drive the choice of an appropriate development and support approach: stable mature markets might suit a high-volume simple approach, while newer emerging markets suit a more feedback-driven approach.
- With feedback-driven development, focus shifts more to fixing the time and resources, and flexing the scope based on early and ongoing feedback.
- A feedback-driven approach requires overlapping the development stages of analysis, build, and testing – by enabling these to happen at the same time, teams can respond more quickly to changing requirements.
- Feedback-driven development requires self-organizing teams who are able to reflect on their progress and learn as they go.
- Iterative design allows a product to be progressively defined as the team gets feedback and learns more.
- Incremental development allows the product to grow by adding and releasing features progressively.
- A feedback-driven approach benefits most from combining both iterative design and incremental development, as this allows new requirements to be discovered and progressively added to the product.
- Scrum is a feedback-driven product development framework that combines iterative design and incremental development, based on self-organizing learning teams.
- Like learning anything new, becoming proficient in Scrum goes through stages of competency – most commonly referred to using the martial arts concept of Shu Ha Ri.