# Agile Scrum Development

## IV. LESSON PROPER

### Agile Scrum Development

Agile software development refers to  software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change. Scrum and Kanban are two of the most widely used Agile methodologies. Below are the most frequently asked questions around Agile and Scrum, answered by experts.

### WHAT IS AGILE?

DELIVER
INCREMENTALLY
INSTEAD OF ALL
AT ONCE

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. Agile development refers to any development process that is aligned with the concepts of the Agile Manifesto. The Manifesto was developed by a group fourteen leading figures in the software industry, and reflects their experience of what approaches do and do not work for software development.

### WHAT IS SCRUM?

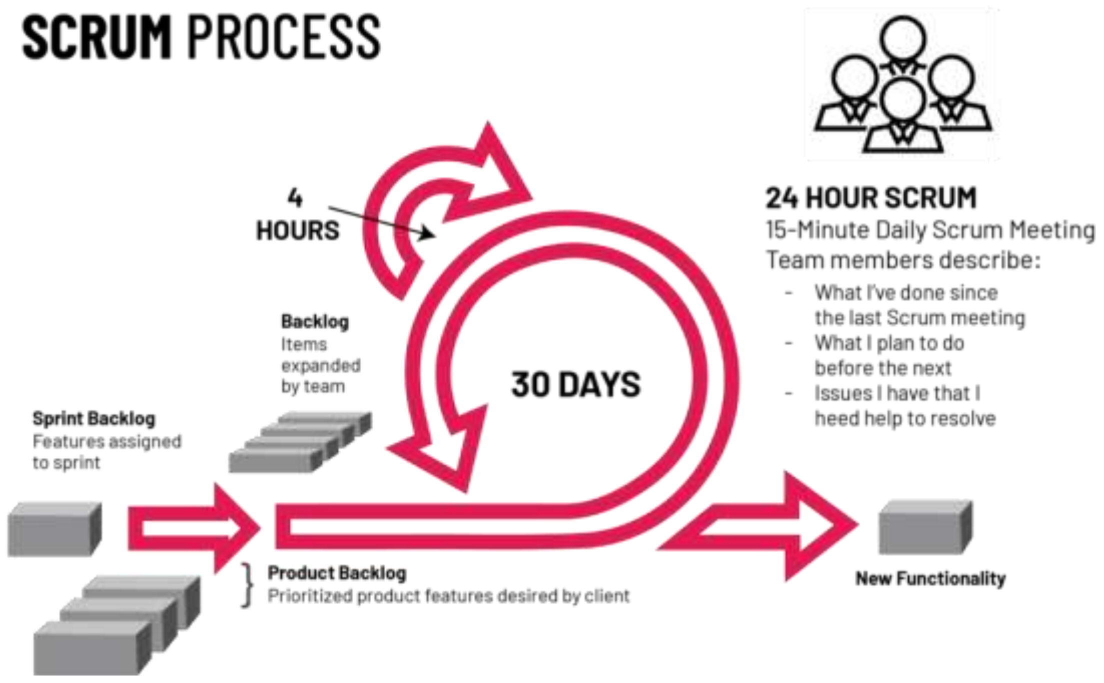
Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one.

- A “process framework” is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints, the XP framework requires pair programming, and so forth.)

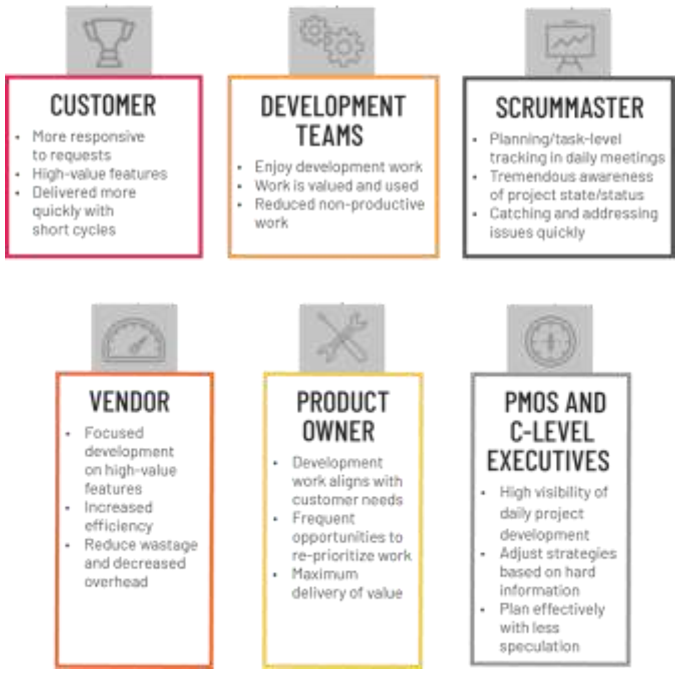
- "Lightweight" means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.

A Scrum process is distinguished from other agile processes by specific concepts and practices, divided into the three categories of Roles, Artifacts, and Time Boxes. These and other terms used in Scrum are defined below. Scrum is most often used to manage complex software and product development, using iterative and incremental practices. Scrum significantly increases productivity and reduces time to benefits relative to classic "waterfall" processes. Scrum processes enable organizations to adjust smoothly to rapidly-changing requirements, and produce a product that meets evolving business goals. An agile Scrum process benefits the organization by helping it to

- Increase the quality of the deliverables
- Cope better with change (and expect the changes)
- Provide better estimates while spending less time creating them
- Be more in control of the project schedule and state


SCRUM PROCESS
4 HOURS
Backlog
Items expanded by team
30 DAYS
24 HOUR SCRUM
15-Minute Daily Scrum Meeting
Team members describe:
- What I've done since the last Scrum meeting
- What I plan to do before the next
- Issues I have that I heed help to resolve
Sprint Backlog
Features assigned to sprint
Product Backlog
Prioritized product features desired by client
New Functionality

## WHAT ARE THE BENEFITS OF AGILE?


CUSTOMER
More responsive to requests
High-value features
Delivered more quickly with short cycles
DEVELOPMENT TEAMS
Enjoy development work
Work is valued and used
Reduced non-productive work
SCRUMMASTER
Planning/task-level tracking in daily meetings
Tremendous awareness of project state/status
Catching and addressing issues quickly
VENDOR
Focused development on high-value features
Increased efficiency
Reduce wastage and decreased overhead
PRODUCT OWNER
Development work aligns with customer needs
Frequent opportunities to re-prioritize work
Maximum delivery of value
PMOS AND C-LEVEL EXECUTIVES
High visibility of daily project development
Adjust strategies based on hard information
Plan effectively with less speculation

Benefits to Customer
Customers find that the vendor is more responsive to development requests. High-value features are developed and delivered more quickly with short cycles, than with the longer cycles favored by classic "waterfall" processes.

Benefits to Vendors
Vendors reduce wastage by focusing development effort on high-value features, and reduce time-to-market relative to waterfall processes due to decreased overhead and increased efficiency. Improved customer satisfaction translates to better customer retention and more positive customer references.

Benefits to Development Teams
Team members enjoy development work, and like to see their work used and valued. Scrum benefits Team members by reducing non-productive work (e.g., writing specifications or other artifacts that no one uses), and giving them more time to do the work they enjoy. Team members also know their work is valued, because requirements are chosen to maximize value to customers.

Benefits to Product Managers
Product Managers, who typically fill the Product Owner role, are responsible for making customers happy by ensuring that development work is aligned with customer needs. Scrum makes this alignment easier by providing frequent opportunities to re-prioritize work, to ensure maximum delivery of value.

Benefits to Project Managers
Project Managers (and others) who fill the ScrumMaster role find that planning and tracking are easier and more concrete, compared to waterfall processes. The focus on task-level tracking, the use of Burndown Charts to display daily progress, and the Daily Scrum meetings, all together give the Project Manager

tremendous awareness about the state of the project at all times. This awareness is key to monitoring the project, and to catching and addressing issues quickly.

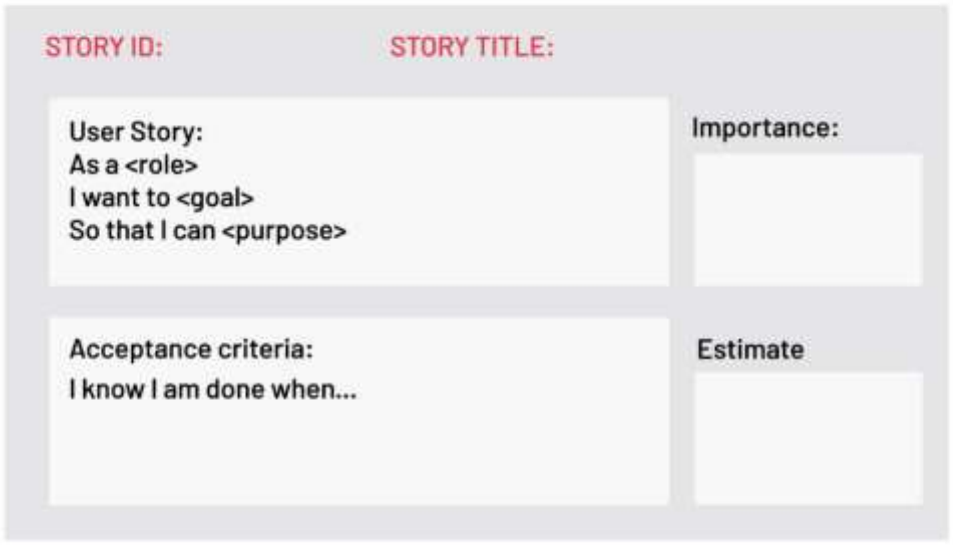
### Benefits to PMOs and C-Level Executives

Scrum provides high visibility into the state of a development project, on a daily basis. External stakeholders, such as C-Level executives and personnel in the Project Management Office, can use this visibility to plan more effectively, and adjust their strategies based on more hard information and less speculation.

## WHAT ARE THE SCRUM REQUIREMENTS?

Scrum does not define just what form requirements are to take, but simply says that they are gathered into the Product Backlog, and referred to generically as "Product Backlog Items," or "PBIs" for short. Given the time-boxed nature of a Sprint, we can also infer that each set should require significantly less time to implement than the duration of the Sprint. Most Scrum projects borrow the "XP" (Extreme Programming) practice of describing a feature request as a "User Story," although a minority uses the older concept of a "Use Case." We will go with the majority view here, and describe three reasonably-standard requirements artifacts found in Product Backlogs.

### User Story



STORY ID:

STORY TITLE:

User Story:
As a <role>
I want to <goal>
So that I can <purpose>

Importance:

Acceptance criteria:
I know I am done when...

Estimate

A User Story describes a desired feature (functional requirement) in narrative form. User Stories are usually written by the Product Owner, and are the Product Owner's responsibility. The format is not standardized, but typically has a name, some descriptive text, references to external documents (such as screen shots), and information about how the implementation will be tested. For example, a Story might resemble the following:

**Name:** Planner enters new contact into address book, so that one can contact the person later by postal or electronic mail

**Description:** Planner enters standard contact information (first and last name, two street address lines, city, state, zip / postal code, country, etc.) into contact-entry screen. One clicks "Save" to keep the data, and "Cancel" to discard data and return to previous screen.

**How to test:** Tester enters and saves the data, finds the name in the address book, and clicks on it. One sees a read-only view of the contact-entry screen, with all data previously entered.

**The elements in this User Story are:**

1. **Name:** The Name is a descriptive phrase or sentence. The example uses a basic "Role-Action-Reason" organization. Another common style, popularized by Mike Cohn, follows the template "As a <type of user>, I want <some goal> so that <some reason>." The choice of template is less important than having a workable standard of some kind.
2. **Description:** This is a high-level (low-detail) description of the need to be met. For functional (user-facing) requirements, the description is put in narrative form. For non-functional requirements, the description can be worded in any form that is easy to understand. In both cases, the key is that the level of detail is modest, because the fine details are worked out during the implementation phase, in discussions between team members, product owners, and anyone else who is involved. (This is one of the core concepts of Scrum: Requirements are specified at a level that allows rough estimation of the work required to implement them, not in detail.)
3. **Screens and External Documents:** If the Story requires user-interface changes (especially non-trivial ones), the Story should contain or link to a prototype of the changes. Any external documents required to implement the Story should also be listed.
4. **How to test:** The implementation of a Story is defined to be complete if, and only if, it passes all acceptance tests developed for it. This section provides a brief description of how the story will be tested. As for the feature itself, the description of testing methods is short, with the details to be worked out during implementation, but we need at least a summary to guide the estimation process.

There are two reasons for including the information about how to test the Story. The obvious reason is to guide development of test cases (acceptance tests) for the Story. The less-obvious, but important, reason, is that the Team will need this information in order to estimate how much work is required to implement the story (since test design and execution is part of the total work).

## Story

Not all requirements for new development represent user-facing features, but do represent significant work that must be done. These requirements often, but not always, represent work that must be done to support user-facing features. We call these non-functional requirements "Technical Stories." Technical Stories have the same elements as User Stories, but need not be cast into narrative form if there is no benefit in doing so. Technical Stories are usually written by Team members, and are added to the Product Backlog. The Product Owner must be familiar with these Stories, and understand the dependencies between these and User Stories in order to rank (sequence) all Stories for implementation.

## Defect

A Defect, or bug report, is a description of a failure of the product to behave in the expected fashion. Defects are stored in a bug-tracking system, which may or may not be physically the same system used to store the Product Backlog. If not, then someone (usually the Product Owner) must enter each Defect into the Product Backlog, for sequencing and scheduling.

## WHAT ARE SOME AGILE METRICS I CAN USE FOR REPORTING?

What to measure in agile is the enduring question. There should be two primary filters we should ask ourselves before we measure anything; "***will this measurement accelerate value delivery?***," and "***will this measurement enhance trust?***".

Below is an example of a fitting agile measurement. Typically an organization will create a goal to increase story point velocity, and this seems rational because we always strive to deliver more where possible. This perspective is looking at the problem from the wrong angle because what we want is value delivery not higher output. Those are not the same; one is outcome focused, and the other is production focused. Agile stresses outcome; which is value delivery, not output. Looking through the lens that equates increases in velocity to output assumes a few things; the teams are not working hard enough, and that output equals value. Of which both assumptions are typically untrue.

Sample Operational Metrics

- Lead Time
- Cycle Time
- Burndown Charts

Sample Output Metrics

- Throughput
- Agility Assessment Model
- Technical quality / defect measurements / code coverage
- # of features, etc.

We had just finished the discussion on Agile Scrum Development. Let's move on to the next higher level of activity/ies or exercise/s that demonstrate your potential skills/knowledge of what you have learned.