
# SCRUM & AGILE
## Everything You Need To Know

Scrum differs from traditional "waterfall" approaches to project management in many ways, but is based on sound project-management principles. Our article on Scrum as Project Management dives deeper into the similarities and differences between Scrum and traditional project management methodologies.

4100 E. Third Ave., Suite 205, Foster City, CA 94404, United States of America
650-931-1650 Consulting | 650-931-1651 Training | www.cprime.com

# CONTENTS

# What is **Agile Development**?

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. The Manifesto says:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> **Individuals and interactions** over processes and tools
> **Working software** over comprehensive documentation
> **Customer collaboration** over contract negotiation
> **Responding to change** over following a plan
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

The agile philosophy holds that the best way to meet customer needs is through the collaboration of a committed group of people, who focus on achieving results quickly, with as little process overhead as possible.

A key element of this philosophy is that we must trust people and their ability to collaborate, more than we trust any particular process. This principle follows from the fact that people can succeed without a formal process, but no process can succeed without people. For this reason, we should design an agile process that best taps the abilities of team members by emphasizing collaboration, rather than relying on the structure of a process to guarantee success. The Agile Manifesto does not specify any particular practices that a development team should follow. Specific agile process frameworks, such as Scrum and XP, do define practices that must be followed.

# What is **Scrum**?

Scrum 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

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

As a result, Scrum projects achieve higher customer satisfaction rates.

# How do We Write **Requirementsin Scrum**?

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

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 de-

scriptive 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:

**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,

| |
|---|
| **Name**: Planner enters new contact into address book, so that he 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. He clicks "Save" to keep the data, and "Cancel" to discard data and return to previous screen. |
| **Screens and External Documents**: http://myserver/screens/contact-entry.html |
| **How to test**: Tester enters and saves the data, finds the name in the address book, and clicks on it. He sees a read-only view of the contact-entry screen, with all data previously entered. |

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).

### Technical 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 the Three **Scrum Roles**?

The three roles defined in Scrum are the ScrumMaster, the Product Owner, and the Team (which consists of Team members). The people who fulfill these roles work together closely, on a daily basis, to ensure the smooth flow of information and the quick resolution of issues.

### ScrumMaster

The ScrumMaster (sometimes written "Scrum Master," although the official term has no space after "Scrum") is the keeper of the process. He is responsible for making the process run smoothly, for removing obstacles that impact productivity, and for organizing and facilitating the critical meetings.

*The ScrumMasters responsibilities include*

- Removing the barriers between the development Team and the Product Owner so that the
- Product Owner directly drives development.
- Teach the Product Owner how to maximize return on investment (ROI), and meet his/her objectives through Scrum.
- Improve the lives of the development Team by facilitating creativity and empowerment.
- Improve the productivity of the development Team in any way possible.
- Improve the engineering practices and tools so that each increment of functionality is potentially shippable.
- Keep information about the Team's progress up to date and visible to all parties.

In practical terms, the ScrumMaster needs to understand Scrum well enough to train and mentor the other roles, and educate and assist other stakeholders who are involved in the process. He should maintain a constant awareness of the status of the project (its progress to date) relative to the expected progress, investigate and facilitate resolution of any roadblocks that hold back progress, and generally be flexible enough to identify and deal with any issues that arise, in any way that is required. He must protect the Team from disturbance from other people by acting as the interface between the two.

The ScrumMaster does not assign tasks to Team members, as task assignment is a Team responsibility. His general approach towards the Team is to encourage and facilitate their decision-making and problem-solving capabilities, so that they can work with increasing efficiency and decreasing need for supervision. His goal is to have a team that is not only empowered to make important decisions, but does so well and routinely.

## Product Owner

The Product Owner is the keeper of the requirements. He provides the "single source of truth" for the Team regarding requirements and their planned order of implementation.

In practice, the Product Owner is the interface between the business, the customers, and their product related needs on one side, and the Team on the other. He buffers the

Team from feature and bug-fix requests that come from many sources, and is the single point of contact for all questions about product requirements. He works closely with the team to define the user-facing and technical requirements, to document the requirements as needed, and to determine the order of their implementation. He maintains the Product Backlog (which is the repository for all of this information), keeping it up to date and at the level of detail and quality the Team requires.

The Product Owner also sets the schedule for releasing completed work to customers, and makes the final call as to whether implementations have the features and quality required for release.

### Team

The Team is a self-organizing and cross-functional group of people who do the hands-on work of developing and testing the product. Since the Team is responsible for producing the product, it must also have the authority to make decisions about how to perform the work. The Team is therefore self-organizing: Team members decide how to break work into tasks, and how to allocate tasks to individuals, throughout the Sprint. The Team size should be kept in the range from five to nine people, if possible. (A larger number make communication difficult, while a smaller number leads to low productivity and fragility.)

*Note: A very similar term, "Scrum Team," refers to the Team plus the ScrumMaster and Product Owner.*

## What are the **Scrum Time Boxes**?

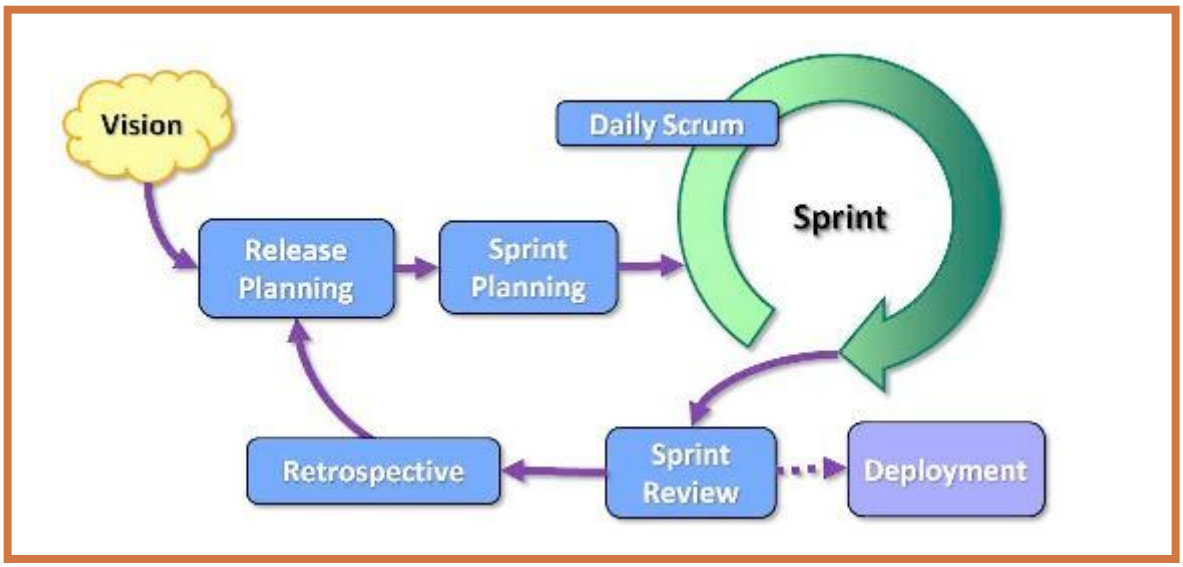
A "time box" is a span of time of fixed duration, dedicated to a particular purpose, whose boundaries are strictly enforced. Scrum defines several "official" time boxes, such as the Sprint and critical meetings, but the concept of a time box is an important theme that permeates Scrum projects.

Scrum is a schedule-oriented, rather than scope-oriented, approach towards managing projects. This means that the schedule overrides other considerations. Thus all meetings and activities should be "time-boxed," meaning kept within the planned duration.

The time-box attitude promotes focus and quick decision making. It encourages people to make decisions that are "good enough" now, rather than "perfect" later. It is a key element in the constant effort to remain pragmatic and flexible in the face of the constant temptation to strive for perfection.

The Scrum time-boxes include the Sprint and critical meetings. The meetings are the Sprint Planning meeting, the Daily Scrum meeting, the Sprint Review meeting, and the Retrospective meeting. Each time box has a specified start and duration, and work is not allowed to extend beyond the duration.

*The diagram shows how the different meetings and activities relate to each other.*




# Sprint

The basic development cycle for a project is called a Sprint. Different organizations and projects select Sprint lengths that best meet their needs, most often in the 2-4 week range.

Experimentation with different Sprint lengths is common for new projects, but once an organization identifies a length that works well, the Sprint length should be frozen at that value, across all Teams. Uniform Sprints simplify planning. They help everyone to schedule their in a predictable fashion, and to develop a smooth rhythm for their work.

During the Sprint, Team members collaborate as much as possible on the top Story or Defect, in order to complete its implementation as quickly as possible. Team members "self-organize" to decide which members will perform which tasks, optimizing Team performance in real time by allocating tasks to the people best able to execute them.

Each Team member updates the status of a task when he starts or finishes it. At the end of the day, he also updates incomplete tasks with information about remaining effort. This information is extremely important, as it is used to track progress daily.

The principal goal for a Sprint is to produce release-quality increments in functionality. In most organizations, the product is not actually released to customers after each Sprint, because this is too frequent for customer comfort, and a single Sprint generally does not produce enough new functionality to justify release. Instead, releases usually incorporate the result of multiple Sprints, and occur at times dictated by customer and business needs. However, the quality of the product should be high enough to be releasable at the end of each Sprint. The chart below provides an example of how workdays can be grouped into Sprints, and Sprints grouped into Releases. The example assumes that each Sprint contains ten workdays, and each Release contains two Sprints.

| Release | Release 1 | | | | | | | | Release 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sprint | Sprint 1 | | | | Sprint 2 | | | | Sprint 3 | | | | Sprint 4 | | | |
| Workday | 1 | 2 | ... | 10 | 11 | 12 | ... | 20 | 21 | 22 | ... | 30 | 31 | 32 | ... | 40 |

## Sprint Planning Meeting

The ScrumMaster facilitates this meeting, which kicks off the Sprint, and which is attended by the Team members and the Product Owner. The purpose of this meeting is to select from the Product Backlog those Product Backlog Items (PBIs) the Team intends to implement in this Sprint. For this selection to be possible,

1. The Team's capacity to do work in this Sprint (its velocity) must be known
2. The amount of work required to implement each item must be known
3. Enough of the top-priority PBIs must have been ranked by the Product Owner, in advance, to fill the Team's capacity for this Sprint

The Team's velocity should have been estimated prior to the meeting. The sizes of the PBIs of interest are either estimated during the meeting, or already known from previous estimation.

The most common estimation technique is "Planning Poker," a voting approach designed to avoid influence bias. The Team discusses each PBI, asks clarifying questions of the Product Owner, and votes, in one or more rounds, until their estimates converge. The ScrumMaster facilitates this process, and moves the PBI to the Sprint Backlog after each estimate is completed, until the Team's capacity for this Sprint has been filled.

After the Sprint Backlog has been defined, the Team spends additional time creating a breakdown of all tasks required to implement each PBI. This work is done in whatever

fashion works best for the Team. Some Teams do this work in the Sprint Planning meeting (which should be time-boxed to no more than, say four hours), or, less formally, after the meeting.

A good rule of thumb for tasks is that they should take between four to sixteen hours to complete. All tasks are trackable (and tracked).

## Daily Scrum Meeting

The Daily Scrum meeting (sometimes, the Daily Standup meeting) is a minimalist status meeting, time-boxed to fifteen minutes. Its purpose is to ensure that questions are answered quickly, that issues are identified and addressed quickly, and to provide Team members with a common understanding of how the Sprint is progressing.

The ScrumMaster faciilates this meeting, which all Team members attend. Attendance by the Product Owner is optional, and at the discretion of the Team. (Basically, the Product Owner should only attend if his presence is useful.) If the Product Owner does attend, his purpose is to take note of requirements-related issues, so that he can address them with the relevant Team members after the meeting.

*The ScrumMaster works to keep people focused and the meeting in its time box, and makes sure that each Team member describes these three things:*

- What I've done since the last Daily Scrum meeting
- What I plan to do before the next Daily Scrum meeting
- What issues I'm facing, that may need help to resolve

It is important that the issues be resolved, but afterwards, not in the meeting itself. Otherwise, the meeting grows, violates its time box, and wastes time for the Team members who are not involved with specific issues. The Daily Scrum meeting should be held at the same time each day, at a time that works best for the Team.

## Sprint Review Meeting

The Sprint Review, or Sprint Demo, meeting is held at the end of the Sprint. In this meeting, the Team demonstrates the Sprint's completed Product Backlog Items, to the Product Owner and other interested parties.

This meeting gives the Product Owner a final chance to make a go/no-go release decision, and gives the Team members a chance to show off their work. (However, some

organizations replace this formal review meeting with PBI-level demonstrations to the Product Owner during the Sprint, to achieve the same results.) While this meeting provides the Product Owner a final opportunity to decide whether the completed work is as desired for release, surprises should be rare at this point, as the Product Owner should be reviewing development status throughout the Sprint.

## Retrospective Meeting

The Retrospective meeting is held after the Sprint Review meeting. Its purpose is to provide the Team an opportunity to learn, and therefore improve, from the experience of the just-concluded Sprint. The Retrospective meeting is facilitated by the ScrumMaster, and attended by the Product

Owner, the Team members, and anyone else whose contribution is desired. The ScrumMaster works to keep people focused and the meeting in its time box (typically one to three hours), and makes sure that each participant describes these three things:

- What worked well, that we should do again
- What didn't work well
- What changes we should make for next time

The ScrumMaster records this information, and facilitates discussion during or after its collection to identify what changes the Team intends to implement for the next Sprint.

# What are the Three **Scrum Artifacts**?

The three artifacts defined by Scrum are the Product Backlog, the Sprint Backlog, and the Burndown Chart (also "Burndown graph," or simply "Burndown").

## Sprint Backlog

The Sprint Backlog is the set of Product Backlog Items, or PBIs (Stories and Defects) planned for implementation in a Sprint. The items in the Sprint Backlog must be ranked in the desired order of implementation (a Product Owner responsibility). The ranking reflects both the urgency (value) of the item, and any dependencies that exist between items.

In an ideal case, all team members work on PBI #1 until it is complete (meaning the implementation has passed all acceptance tests), and only then begin work on PBI

#2. This sequence continues until the team has worked through all items in the Sprint Backlog.

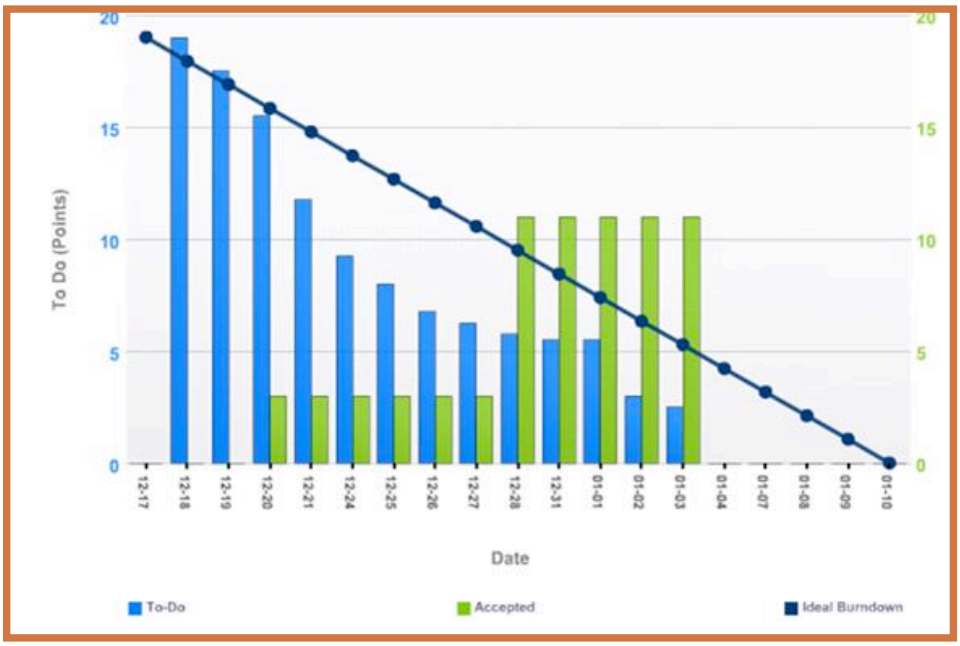
The importance of ranking becomes clear when progress does not go as well as expected. It may be that only 50% of the actual work required by the Sprint Backlog can actually be done in the time available. By working on PBIs in rank order, we've at least completed the most important half of Sprint Backlog. A different kind of ranking, or parallel work across items, would have resulted in less or even no value delivered by the Sprint.

## ProductBacklog

The Product Backlog is the set of all un-implemented Product Backlog Items (requirements, in the form of Stories and Defects) that have not been assigned to the current Sprint. Unlike the Sprint Backlog, there is no requirement that all PBIs be assigned a rank. In practice, Product Owners usually have at least a rough concept of ranking for items likely to be implemented in the next couple of months, and less so beyond that time. Prior to the Sprint Planning meeting, the Product Owner must finalize the ranking of the top items in the Product Backlog.

## Burndown Chart

At the start of the Sprint, the team breaks down each item in the Sprint backlog into a set of tasks, with a time estimate for each, such that executing the tasks results in a completed implementation. During the Sprint, the team member who completes a particular task marks that task as complete. Plotting the amount of uncompleted task work against time from the start of the Sprint produces a Burndown Chart (see picture,



The diagonal line from the top left to lower right shows the ideal "burn down" of work versus time, and ends with zero remaining work on the last day of the Sprint. The blue bars in the sample chart show the amount of work actually remaining each day. If the bars are below the diagonal line, the project is ahead of schedule; above, and it is behind schedule.

*(Note: A Burndown Chart is essentially the same thing as the "Estimate to Complete" chart familiar to project managers.)*

Just as a Burndown Chart shows work remaining, versus time, a "Burnup Chart" shows work accomplished versus time. The green bars in the sample picture show the burnup status of the Sprint, but recorded at PBI (Story or Defect) granularity, rather than task granularity. Burndown and Burnup charts can also be plotted for multi-Sprint Releases, when longer time scales are of interest.

# What are the **Benefits of Scrum**?

*The benefits are different for different people.*

## 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

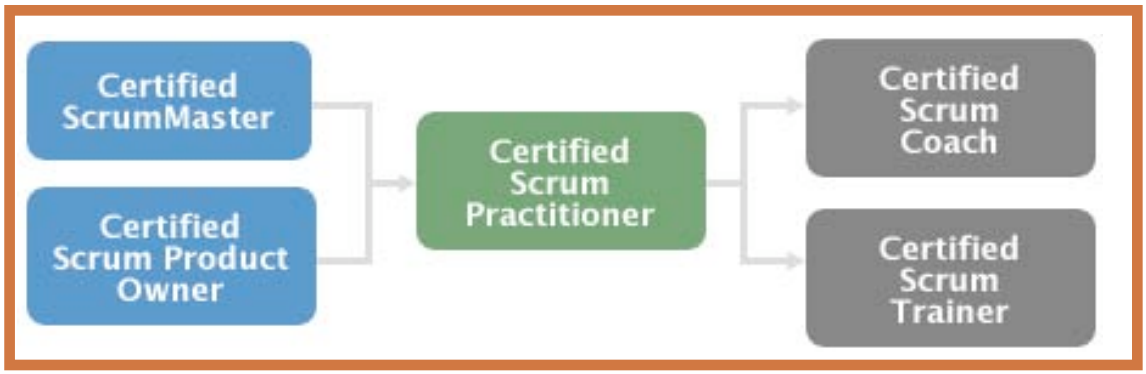
Product Managers, who typically fill the Product Owner role, are responsible for making customers happy by ensuring that development work is aligned with customer needs. Scrum makes this alignment easier by providing frequent opportunities to re-prioritize work, to ensure maximum delivery of value.

## 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 affectively, and adjust their strategies based on more hard information and less speculation.




What are The Scrum Certification? How do I become Certified, and why should I? The Scrum Alliance has defined five certifications, as indicated in the chart.

*The primary value of a certification is always in the opportunities it opens up for the holder.*

- Companies increasingly seek Project Managers with CSM certifications, to organize and facilitate their Scrum development teams.
- Product Managers who have CSPO certifications are more attractive to companies that develop with Scrum, than those who don't.
- The CSP provides a strong statement to potential employers that the holder has

deep practical experience with Scrum (as a ScrumMaster or Product Owner), and corresponding value.

- The CSC and CST certifications are useful for firms and consultants who provide Scrum coaching and teaching services, as they confirm that the holder has strong experience in these areas. (A CST is also required for anyone who will teach a CSM or CSPO course.)

All Scrum certifications are offered by the Scrum Alliance. For information on how to apply for certifications, consult the Scrum Alliance certification page.

## Certified Scrum Master (CSM)

The CSM certification is the starting point for people who wish to fill the ScrumMaster role, either as their primary goal, or as a step towards other certifications. A CSM indicates that the holder has attended a CSM class, and passed the CSM exam.

After sucessfully completing ScrumMaster course, the student receives an email from the Scrum Alliance. The message contains information about membership, and how to create an online profile and take the CSM exam.

The CSM exam consists of 60 multiple-choice questions, to be answered within 60 minutes. The exam focuses on the following information (taught in the class):

- Scrum Basics - The principles of Scrum, Scrum framework, and artifacts and roles
- Meetings - Sprint Review, Sprint Planning Meeting, Daily Scrum, and Sprint Retrospective
- Roles - Scrum Team, Product Owner, ScrumMaster roles and responsibilities
- Artifacts - Product Backlog, Sprint Backlog, and Burndown charts, and how they are used

The cost of the exam is included in the cost of the ScrumMaster course. There is no exam charge for the first and subsequent attempts to pass the exam.

The CSM exam is currently under design, and the ScrumAlliance is collecting information about responses to the exam questions. Until the exam is finalized, everyone who takes it automatically receives a passing grade.

## Certified Scrum Product Owner (CSPO)

The CSPO certification is the starting point for people who wish to fill the Product Owner role, either as their primary goal, or as a step towards other certifications.

### Certified Scrum Practitioner (CSP)

The CSP certification is the next step for ScrumMasters and Product Owners. A CSP indicates that the holder has successfully served in one of these roles for at least one year. The CSP candidate must submit an application containing a detailed description what he has done in his role. The CSP may be obtained immediately after the CSM or CSPO, if the applicant has been actively practicing his Scrum role for the required duration.

### Certified Scrum Coach (CSC)

The CSC certification indicates that the holder has acted as a Scrum coach, meaning coached Teams through their adoption and mastery of Scrum, for at least 1500 hours in the past 5 years. It is intended for those who focus on coaching.

### Certified Scrum Trainer (CST)

The CST certification is required for anyone who wants to teach CSM or CSPO classes. Applicants must have a CSP for at least a year before applying, along with either a CSM or CSPO.