Quickscrum
Brings Agility

STEP BY STEP GUIDE TO SCRUM

Table of Contents

About us 3

CEO Message 4

Chapter 1: Introduction to Scrum

1.1 History 6

1.2 Scrum in Brief 7

Chapter 2: Scrum Artifacts

2.1 Product Backlog 10

2.2 Sprint Backlog 14

Chapter 3: Userstory

3.1 Userstory in Brief 17

3.2 Write Userstory 18

3.3 Estimate Userstory 22

Chapter 4: Scrum Team

4.1 Product Owner 31

4.2 Scrum Master 35

4.3 Team Member 40

Chapter 5: Scrum Events

5.1 Sprint Planning 44

5.2 Daily Stand-up 46

5.3 Sprint Review 51

5.4 Sprint Retrospective 53

About us

Started in 2013 as a simple scrum solution provider, Quickscrum has evolved as one of the leading work management solution today globally.
Having lean fundamentals within our DNA, we are on a mission to craft the most innovative work management solution and cater its benefits to the companies across multiple Industries.

Our passionate tech team uses **artificial intelligence**, **machine learning**, **business intelligence** and many more latest technologies to solve day to day work management issues.

Our Belief

In today's fast moving era, every organization is thriving for the great success every moment. Competition is inevitable. Resources are limited.
We believe the **Operational Efficiency** plays the key role in achieving great success

Our Life

Crafting the enterprise solution to make organization operationally efficient by better team collaboration and streamlining internal processes.

CEO Message

Hello Quickscrum family,

Scrum is everywhere. Everyone is trying to learn its framework and implement within their organization to get benefited. Since last few years, We have been working with clients globally with diverse culture and geography to train their teams for scrum fundamentals and implement our software to **improve their team collaboration** and **streamline workflows**.

We identified that many teams have learnt from internet about scrum and found their understanding quite contradict. As well saw them struggling to implement scrum.

I, Mrugesh PANCHAL founder of Quickscrum, and my team decided to write upon the scrum framework in detail to help teams to understand scrum framework in-depth. We have also incorporated our experience and introduces some of the new fundamentals to practically improve team efficiency.

In practical life, it's not just about implement any framework rightly. It's about getting benefits and improve operational efficiency of the organization to be successful. We are committed to bring series of e-books having case studies, framework fundamentals, practical guidance, individual and team improvements and research work to the Quickscrum family.

We hope you will like our work. I personally welcome your feedback to improve upon next time. You can send directly to me on mrugesh.panchal@quickscrum.com.

Have a nice transformation journey ☺.
Mrugesh PANCHAL, Founder

Chapter 1

Introduction to Scrum

1.1 History

Scrum is not only one of the greatest inventions in the agile world but also one of the most popular frameworks. But with this popularity comes a great responsibility, which when abused, leads to controversies.

Hirotaka Takeuchi and **Ikujiro Nonaka** came up with the initial concepts of scrum, including the word “Scrum” itself in their white paper in year 1986.

Jeff Sutherland fine tuned the concept and together with **Ken Schwaber** collectively presented their experiences during the OOPSLA 1995 conference. Subsequently, **Ken Schwaber** and **Mike Beedle** attempted to communicate scrum through the first scrum book - **Agile Software Development with Scrum**.

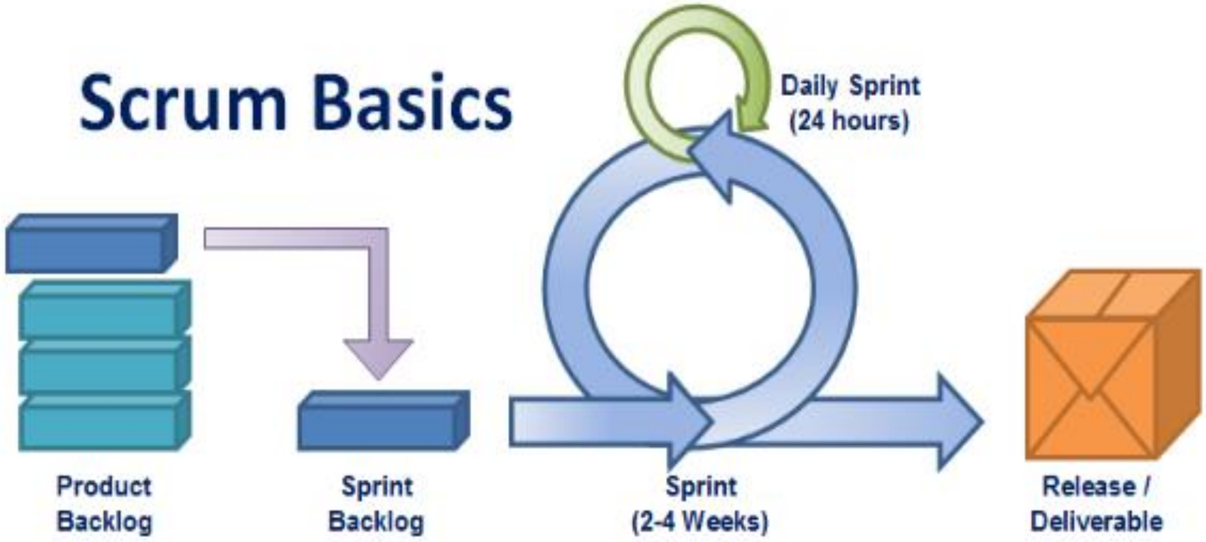
As the scrum community started growing, it was decided to create a platform to bring them together, which in turn lead to the birth of the **Scrum Alliance** (SA) and **Certified Scrum Master** (CSM) certification. However, controversies started brewing about transparency and the motive behind SA, resulting in the birth of **Scrum.org**.

1.2 Scrum in Brief

Scrum is based on the fundamental of **iterative** and **incremental** development. It is designed to plan, execute and deliver iteratively. It emphasises on **smaller & frequent delivery cycle**, **continuous client feedback**, **adopt to the requirement changes**, **team collaboration** & **self organizing behaviour**.

Scrum is quite easy to understand but hard to implement. Since its inception, scrum has also helped quite complex projects to be successful with high return on investment (ROI).

Scrum team is generally made of 6-9 team members. Basic roles in a team are – Scrum Master, Product Owner and Team Member.



Chapter 2

Scrum Artifacts

In Scrum, artifacts are “information radiators” and they serve to capture the shared understanding of the team at a particular point in time.

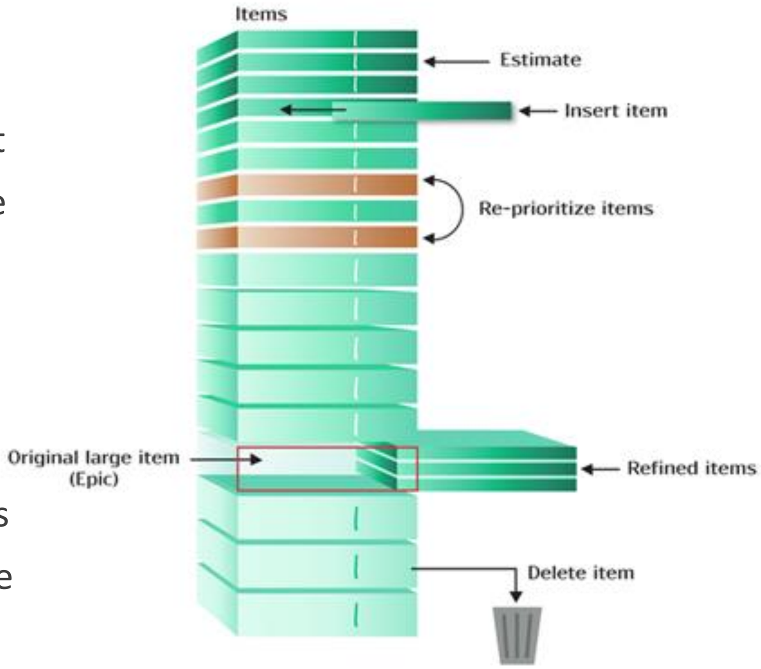
In a co-located Scrum team, artifacts play a vital role for the team to reflect themselves on how they are doing with the sprint goal. Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding.

Per the latest Scrum guide, Scrum framework defines 3 essential artifacts.

- Product Backlog
- Sprint Backlog
- Product Increments

2.1 Product Backlog

A product backlog is a **prioritized list of work** for the scrum team that is created from its requirement. The most important items are **shown at the top** of the product backlog so the team knows what to deliver first.



The **product owner** creates, maintains, and regularly re-orders the product backlog. It can also be updated anytime to adapt to the emerging requirement, customer feedback, and market changes.

What are the Product Backlog Items (PBI) ?

The product backlog is composed of product backlog items or PBIs. Most PBIs are **functionalities** that will have tangible value to the end user. These are often written as user stories. Other PBIs include **Defects**, **Technical work**, **Knowledge-acquisition work**, and any other work the product owner finds valuable.

PBI Types

The product backlog items can be categorized mainly into four fundamental types :

Feature: The most common type of PBI is something that is of value to a end user – the product features. It can be a **brand-new functionality** or **changes to existing functionality**. Features, for most teams, should represent the bulk of items in the backlog.

Defects: Testing team includes defects/bugs in their product backlog. High-performance agile teams never let defects live long so they are unlikely to ever have many defects in their product backlogs.

Technical Work: PBIs can also include technical items. Examples might include upgrading to the latest version of the Oracle DBMS or refactoring a section of previously completed code.

Knowledge Acquisition: Another type of PBIs are items that involve knowledge acquisition. During agile development when the team is presented with a high degree of uncertainty a common and effective solution is to **research information**. Examples include **prototype**, **proof-of-concept**, **research**, **experiment**, spike, etc.

Creating the Product backlog

Creating a product backlog is a **sole responsibility of product owner**. Depends on the type of the project, product owner starts creating a backlog **based on conversation with client** or **based on own project knowledge.**

For many projects, product owner acts as a client, as s/he fully responsible for the **market research** and **idea generation**. For example, start-up founders.

Maintaining the Product backlog

After this initial setup, the Product Backlog has to be maintained in an ongoing process that comprises the following steps :

- Ordering the product backlog, the most important items are moved to the top.
- Preparing the high-priority entries for the next sprint planning meeting.
- (Re-)Estimating the entries in the product backlog.

The product owner is responsible for making sure that the product backlog is in good shape. This is a collaborative process.

When using the scrum framework about 10% of the scrum team's total time should be reserved for maintaining the product backlog - discussion, estimation etc. The collaborative maintenance of the product backlog helps to clarify the requirements and create a buy-in from the scrum team.

Characteristics of a good product backlog

Good product backlogs share similar characteristics – **DEEP**: **Detailed appropriately**, **Emergent**, **Estimated**, and **Prioritized.**

Detailed Appropriately: The product backlog items will differ in their level of detail. Those that we will work on sooner, those at the top of the product backlog, will be more detailed.

Those that we won't work on or for some time will be less detailed. We want to refine (add detail to) backlog items as they rise in priority, adding details in a just-enough, just-in-time fashion.

Emergent: As discussed in earlier chapters, the product backlog is a living document, constantly changing as the product is being developed or maintained. As the team and product owner learn more about the product and the marketplace, they might add new items, discard some, and change others.

The **emergent nature of the product backlog** is not only expected but is a sign of a healthy and functioning product backlog.

Estimated: Each product backlog item should have a estimated size that corresponds to the **effort required to develop that item**. The product owner uses this estimation to prioritize the backlog items.

Ideally, most of the items at the top of the product backlog will be sprint-sized, small enough to be worked on during a single sprint. Large, high-priority items should be broken into smaller items prior to being declared sprint-ready.

Prioritized: A product backlog should be a prioritized list of PBIs, but not every PBI needs to be prioritized. We recommend prioritizing about a release worth of PBIs.

Prioritizing beyond that is likely a waste of effort, as too much might change by the time the first release is out.

Tools to manage product backlog

The scrum team uses different tools based on their choices. Many teams prefer sticky notes or spreadsheet. They are quite simple to use but does not help in advanced analysis and monitoring. It is best to adopt digital tool such as **Quickscrum** to rightly follow scrum and make your team highly effective.

2.2 Sprint Backlog

The **sprint backlog** represents all product backlog items (PBI) committed by the scrum team **to be delivered within a single sprint**.

How to create a sprint backlog

The sprint backlog is created during **sprint planning meeting**. During this meeting, product owner presents **high priority items** to the team and entire team agrees to complete a set of product backlog items within a sprint.

Team commits to the delivery based on their **average velocity** or **capacity** and **length of the sprint** – ideally 1 to 4 weeks.

Ideally, sprint planning meeting should be timeboxed to maximum 2 hours. Sometimes it is **too short duration** to **check item feasibility**, **identify the dependencies**, **break down into the task, assign to the resources** and **estimate them**.

Thus mature scrum team keeps at least **3-4 product backlog grooming** sessions prior to each sprint planning meeting.

Why sprint backlog is important?

Well defined sprint backlog has immense values such as,

- Deliver high value items
- Bring focus
- Provide clear goal to the team
- Bring Transparency
- Reduce impediments

- Keep everyone on the same page
- Measure the team velocity
- Improve quality
- Increase team efficiency

7 Tips for creating a good sprint backlog

- **Tips 1:** Define clear sprint goal. What exactly team aims to achieve within a sprint.

- **Tips 2:** Involve every team member in the process. For example, involving tester in each meeting will help to improve testing quality.

- **Tips 3:** Perform grooming session prior to the sprint planning meeting to have identified technical feasibility, analyse requirement, check dependencies, impact on architecture etc.

- **Tips 4:** Plan only high business value items.

- **Tips 5:** Discuss over every item in detail about how it should be implemented.

- **Tips 6:** Have a clear definition of done.

- **Tips 7:** Break each item into smaller tasks.

Chapter 3

Userstory

3.1 User Story in Brief

A user story is a short, simple description of a requirement written from the **perspective of the end user (person who desires the new capability)**.

You may find many different explanation about user story on internet. Simple word, user story helps,

- Shift focus on value creation. Help to assure that team doesn't work on something having no value.
- Avoid lengthy requirement documentation
- Emphasis on end user personas
- Explain end user requirement in just one or two sentences
- Keep everyone on the same understanding

Writing good user story is really an art and require lots of practices. Clear and precise story writing can help team to **improve their efficiency by at least 2x**.

With user stories, everyone in your team knows exactly “who,” “what,” and “why” they are building features. Each component adds a necessary layer of context to give your team a proper start.

A user story immediately directs the focus to a specific circumstance which provokes further discussion and careful revision. The end result is that your team becomes more focused on delivering solutions to user problems as opposed to merely delivering functional code.

3.2 Write User Story

User stories allow you to say why the need you're proposing to develop makes sense. A user story answers 3 important questions,

1. Who are you building this for?
2. What are you building?
3. Why are you building this?

There are different templates on how to define user stories. A most **popular template** is as follows :

As an [actor], I want [need] so that [business value]

Actor:

S/he is the 'owner' of the user story. This is often a user but it is recommended to be more specific here. By using specific actors (e.g. "Administrator", "Logged in Customer", "Unauthenticated visitor") it's easier to understand and sets the user story into context.

Need:

Actual requirement of the actor in terms of feature.

Business value:

It explains, what are the benefits end user will get once the need is satisfied.

User stories are often written on **index cards or sticky notes**, and arranged on walls or tables to **facilitate planning and discussion**. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

Example 1: *As a user, I want to filter items by item type so that I can see how my team's time is being used between features and bugs on a weekly basis*

Example 2: *As a user, I want to filter items by item type so that I can create a report on everything we did this month for my boss.*

Notice how changing one component of a user story would change your approach entirely?

In the first case, you would probably display this information in a chart or graph for a simple breakdown of your team's time.

In the second example, you would likely create a function to export the data so it can be shared and presented.

I.N.V.E.S.T rule to write good user story

The **I.N.V.E.S.T.** guideline to writing user stories is almost universally accepted as the standard to work by. The acronym was made popular by Bill Wake's original article from 2003. Our interpretation:

Independent

You should be able to prioritize and rearrange user stories in any way with no overlap or confusion.

Negotiable

As previously discussed, a good user story can be reworked or modified to best suit the business. User stories are not an explicit set of tasks.

Valuable

User stories need to be valuable. By this, we mean value for the business or the customer. If it’s not, why would you have your team work on it?

Estimable

A good user story can be estimated. It’s also important to differentiate time estimations from an exact timeframe. A rough estimate is beneficial to allow teams to rank and schedule their priorities.

Small

We definitely recommend keeping your user stories small. While we don’t suggest an exact timeframe to stay in, writing user stories that focus on smaller tasks allows for greater focus.

The larger a story is, the harder it is to estimate and easier it is to get caught up in sub items that should have probably been their own stories.

Testable

Before a user story is written, it is essential that criteria to test it is in place. Outlining the testability first ensures that the story actually accomplishes the goal you are trying to achieve.

A story is not finished until it is tested. For maximum productivity and team alignment, make sure your team knows how their work will be tested

Benefits of writing user stories

- Helps to express the business value.
- Avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution.

- Avoid the appearance of false completeness and clarity.
- Get to small enough chunks that invite negotiation and movement in the backlog.
- Stories are comprehensible.
- Developers and customers understand them.
- People are better able to remember events if they are organized into stories.
- They support and encourage iterative development.
- They can be easily started as epics and later disaggregated at the development time.
- Stories support opportunistic development.
- Stories support participatory design.
- Stories put the focus on the user’s goals.

3.3 Estimate User Story

Agile emphasis to estimate a story in **Story point** instead of hours.

What is story point?

Story points represent the relative sizing of the user story. It is a unit of estimation used by Agile teams to estimate User Stories.

When the product owner wants some features to be developed he/she desires to know how soon the team can complete the features and how many resources it will take to complete the work.

From the developer's perspective it's next to impossible to predict the exact time in which he/she can complete the work. The person can, however, give a rough estimates about total time.

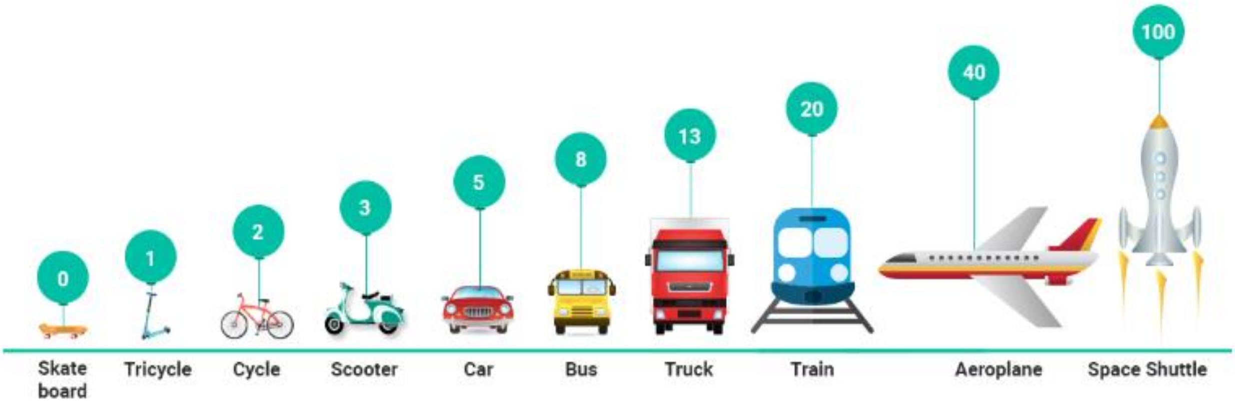
Note that instead of "will" the developer chose to use "might" because he/she is not absolutely "sure about the time factor but "feels" it might take that much time. This is user story estimation in a nut shell.

You don't give an exact number explaining how complex the story is and how long it'll take to develop – **you give a rough "estimate"**.

As a human, we are good at comparing size, so estimating a story using fibonacci series gives more **clarity of its complexity** and **relative sizing** in terms of development. It is helpful to have a set of stories nearby to make comparison and recommendation to set priority.

Here is an example of **relative sizing** and **its estimation points** to develop following vehicle:

Here is an example of **relative sizing** and **its estimation points** to develop following vehicle:



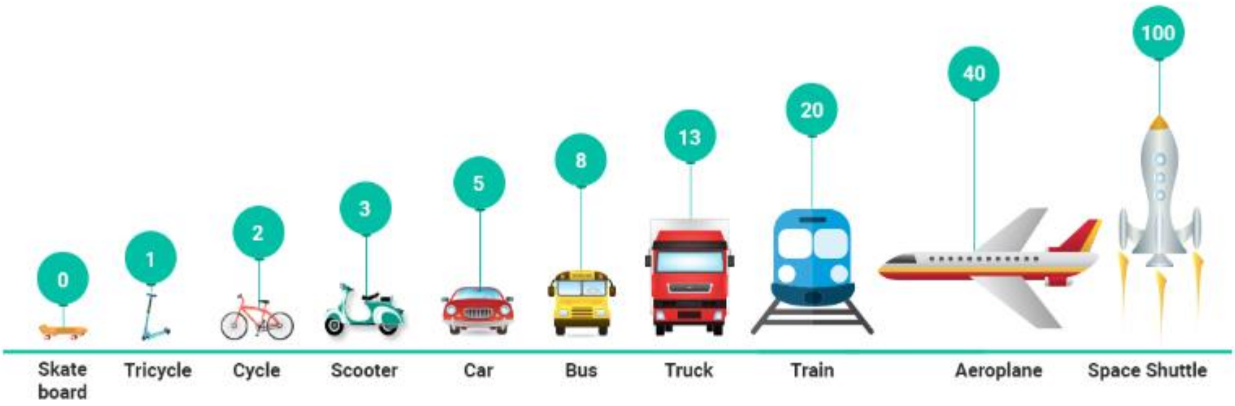
Factors to consider while estimating stories

- **Complexity :** Consider the complexity of the story.
- **Risk :** Consider the team's in-experience with developing the story.
- **Implementation :** Consider the implementation factors.
- **Deployment :** Consider the deployment requirements.
- **Interdependencies :** Consider dependencies on other stories.

Who should be involved for story estimation?

All team members who are responsible for getting a story done should ideally be part of the estimation

Here is an example of relative sizing and its estimation points to develop following vehicle:



Advantages of using story points for estimating work

- Story points are a measure of relative size and complexity.
- Story points are unit less, meaning as a scale, they are only valuable to compare against each other **within the same team**.
- Estimating in story points **encourages everyone**, including business people to participate in the estimation process (using Planning Poker).
- Estimating in story points is **fast and easy**.
- Story points initiate **high-level discussions** about everything that is involved in a project.
- Story points of **completed stories** can be used to generate the **teams velocity** which can be used to **predict future work capacity**.

Best practices to estimate stories

Follow these best practices to estimate stories within your product backlog.

Step 1: Identify base stories

Identify one or multiple base or reference stories against which you would do relative sizing of the backlog. This story is picked from current product backlog or any other story which you have used earlier. But what is most important is the **understanding of this story must be same among everyone** on the team. Team should be confident of this base story.

Step 2: Talk through the detail requirements

Discuss with product owner about the story in detail.

Step 3: Discuss and note down points

These can be bullet points on the story card or text in the "notes" section of a tool. This is best done by scrum master who can add these details when discussion is on.

Step 4: Raise questions if any

During discussion, question may arise and must be clarified the same time. Such as,

Requirement : Any doubt about story requirement? Raise an alert. Ask product owner to give more clarity.

Technical Feasibility : Can story be delivered using current technology? Any unforeseen technical challenges must be surfaced.

Acceptance Criteria : Team must clarify the **check list to be fulfilled** to mark story as accepted.

Dependency : Does this story have external dependencies? If yes, that must be understood and resolved quickly.

Expertise : Do we have enough skills to deliver the story? Team must have internal skills to deliver story otherwise delivery might be delayed or not done properly.

Chapter 4

Scrum Team

A scrum team is a made of individuals working together to deliver the requested and committed product increments. It comprises of **cross-functional members** who are capable of achieving the sprint goal.

As per scrum framework, Team is made of three key roles,

- **Product Owner**
- **Scrum Master**
- **Team Members**

But practically, the scrum team can have additional roles such as,

- **Architects,**
- **Analysts,**
- **System admins,**
- **QA Lead,**
- **UXD Engineers**
- **UI designers** etc.

The most effective scrum teams are **tight-knit**, **co-located**, and usually **6 to 9 team members**. Team members have differing skill sets, and cross-train each other so no one person becomes a bottleneck in the delivery of work.

Strong scrum teams approach their project with a clear **"we"** attitude. All members of the team **help one another** to ensure a successful sprint completion.

Responsibilities of scrum team

Core Responsibilities

- Embraces the Scrum Values – Focus, Commitment, Openness, Respect, and Courage.
- Delivers working product on-time, on the scope and on quality.
- Shares development responsibilities, assists, trains and mentors team members to meet "Sprint" challenges.
- Strives to value the individual and increase team recognition over self-recognition.
- Builds consensus and applies sound judgment in a team-centric environment.
- Provides open and honest feedback within a "Scrum Team" environment.
- Always prepared to challenge the Scrum Team and improve continuously.

Sprint Creation and Execution

- Help to plan sprint and get clarity over planned user stories.
- Focus on user stories with high business value and priority.
- Complete the planned user stories as quickly as possible.
- Request clarification from the product owner when unclear about a user story.
- Conduct peer reviews on one another's work.
- Take on tasks beyond your normal role as the sprint demands.

- Fully develop functionality as agreed to in the definition of done.
- Report daily on the sprint progress.
- Alert the scrum master to any roadblocks and help to effectively remove it.
- Achieve the sprint goal as committed in sprint planning meeting.

4.1 Product Owner

A product owner is a person who represents the customer, business or user community and is responsible for working with the team to determine what features will be included in the product release.

As a liaison between the development team and customers, the product owner must collaborate closely with both groups to ensure there is a clear understanding of what features are needed in the product or application.

Because there may be a variety of types of customers and users, the product owner must have a firm understanding of the business domain and the varying needs of different types of users.

Required skills for product owner

A product owner is usually a **CEO**, **domain specialist**, **project manager** or even a **business analyst** with technical skills. A Product owner should ideally have a good balance of the following skills :

- Domain expertise
- Technical skill
- Decision making
- Fully available to the team
- Communication
- Business savvy
- Highly motivated
- Visionary and out of box thinking

Responsibilities of product owner

The Product owner decides what will be built and in which order. S/he takes following responsibilities.

Manage Product Backlog and Release Planning

- Write user stories (smaller chunks of requirement).
- Fully elaborates **acceptance criteria** for each user story.
- Prioritize user stories according to business value.
- Perform release planning and ensures that the release backlog is aligned with the vision and roadmap.
- Reviews product backlog in depth with the scrum master in preparation for future release planning.
- Update user stories as needed in order to have an up to date and comprehensive backlog.

Work closely with scrum team

- Collaborates with the team to ensure user stories are accurately elaborated and understood.
- Ensures that everyone in the scrum team understands what is required.
- Be available to answer any question arise during sprint execution and support them.

Sprint Planning and Execution

- Attends **sprint planning meeting** to plan and elaborate each user story to the team.
- Attends daily stand-up to remain engaged and up-to-date on the team's progress.
- Completes incremental reviews of stories as they are completed.

- Adjust priorities of stories mid-sprint based on impediments or dependencies.
- Attend **sprint review meeting** to get demonstration of what has been achieved within a sprint, provide feedback and accept the stories.

Stakeholder Management

- Collaborates with stakeholders outside of the team to review progress.
- Collect and discuss required functionalities with the stakeholders.

5 qualities of product owner

1. Leadership

As every team member is responsible for the product success, It is must for the product owner to provide guidance and direction to everyone involved in the development effort and ensures that the project goes well.

2. Decision making

S/he should be a good decision maker especially in deciding which product features will bring the highest ROI.

3. Communication

S/he must be able to communicate different messages to different people about the project at any given time.

4. Serve

Adopt a “customer service” mentality, make yourself available in-person whenever possible, and be open to questions.

5. Let Go

Allowing the scrum team to oversee their own task-execution will save time and make everyone’s life easier.

4.2 Scrum Master

The Scrum Master is responsible for making sure a Scrum team lives by the values and practices of Scrum. He/she is often considered a coach for the team, helping the team do the best work it possibly can.

The Scrum Master can also be thought of as a **process owner** for the team, creating a balance with the project's key stakeholder, who is referred to as the product owner. The Scrum Master does everything possible to help the team perform at their highest level. This involves **removing any impediments** to progress, facilitating meetings, and doing things like working with the product owner to make sure the product backlog is in good shape and ready for the next sprint.

The Scrum Master's role is commonly filled by a former project manager or a technical team leader but can be anyone.

Main role activities carried out by a scrum master includes:

- He/she is a facilitator and servant leader who encourages and demands self-organization from the development team.
- He/she enables close cooperation across all roles and functions, addresses resource issue and disobedience of scrum practices.
- He/she protects the team from external and internal distractions.
- He/she removes impediments so the team can focus on the work at hand and follow scrum practices.
- He/she is not typically a manager or lead, but he is an influential leader and coach who does not do direct command and control.

Responsibilities of scrum master

Facilitation of Scrum events

The scrum framework defines several meetings those have to be organized and facilitated by the scrum master. Following are the scrum meetings,

- Daily scrum meetings
- Sprint planning meetings
- Sprint review meetings
- Sprint retrospective meeting

Maintain team dynamics

- Coaching team members.
- Mediating through conflicts.
- Helping the team to make decisions.
- Fostering the team’s self-organization.
- Mediating the general conflict of goals between the development team and product owner.

Support learning

- Encourage team for continuous learning .
- Provide learning resources.
- Helping the team to be information radiators.
- Encouraging the use of best engineering practices within the development team (e.g. one click releases, continuous

delivery etc.).

Help in product-related activities

- Help to write or split user stories.
- Help to write or adapt product visions.
- Help to prioritize product backlog items.
- Help in release planning.
- Being familiar with the team’s work (i.e. the product).

Supporting the big picture

- Bringing people together who should talk to each other.
- Keeping in touch with every stakeholder regularly.
- Helping the team to report to management.
- Help to further the agile community within the organization.
- Sharing insights throughout the company (micro-blogging, blogging, internal conferences, etc.).
- Being a contact person for everyone in the team and stakeholders regarding agile.
- Giving learning opportunities to people in the organization (e.g. talks or workshops) and letting them learn important agile concepts like e.g. technical debt.

Facilitating change

- Helping the team to get rid of impediments.
- Suggesting new metrics for the team as catalysts for change.

Mirror

- Reflecting agile and scrum values to the team.
- Reminding the team of their arrangements (e.g. policies).

- Helping the team to continuously improve their process.
- Reflecting issues to the team through observation from outside of the team.
- Asking open questions.
- Checking the artifacts used by the team such sprint backlog, metrics, etc.

Miscellaneous

- Helping the team to keep focus (e.g. by acting as a buffer between external distractions and the team).
- Helping the team to maintain their scrum tools (storyboard, action board, charts, backlogs, etc.).
- Helping team and product owner to find a suitable Definition of Done.

Skills required for scrum master

A Scrum Master is usually the team leader or the project manager. A scrum master should ideally have a good balance of the following skills,

- Technical expertise
- Understands the product owner's vision
- A good team player and mentor
- Understands the team's capabilities
- Motivate and coach the team
- Problem solver

6 Qualities of scrum master

Responsible

A scrum master's role is a highly important in scrum. He/she ensures that the **team follows scrum in a right manner** and is responsible for **facilitating the work process** so the team can deliver successful product increments.

Humble

One of the best ways of gaining the respect of people is to **be humble** and **remain true to work**. S/he should play a **servant-leader role** and **talk humbly with people** and make genuine efforts to understand their problems and resolve them.

Collaborative

The scrum master should collaboratively work with the team to **resolve impediments** and help team to **achieve sprint goal**.

Committed

Being an scrum master is not easy – it is generally a **full-time job** and requires a lot of efforts. S/he must remain committed to the team in terms of **being available whenever needed** and help team **effectively deal with their problems** as they arise.

Influential

The scrum master bridges the gap between the scrum team and the management. At times s/he might be required to **negotiate for better working conditions** or **quick resolutions to problems** with the management. S/he should be influential and able to convince the management to resolve issues quickly.

Knowledgeable

The scrum master also functions as a mentor and coach for the team. S/he should be **conversant with the latest trends** and **updates in scrum** and share them with the team.

4.3 Team Member

The Scrum Master is responsible for making sure a Scrum team lives by the values and practices of Scrum. He/she is often considered a coach for the team, helping the team do the best work it possibly can.

The Scrum Master can also be thought of as a **process owner** for the team, creating a balance with the project's key stakeholder, who is referred to as the product owner. The Scrum Master does everything possible to help the team perform at their highest level. This involves **removing any impediments** to progress, facilitating meetings, and doing things like working with the product owner to make sure the product backlog is in good shape and ready for the next sprint.

The Scrum Master's role is commonly filled by a former project manager or a technical team leader but can be anyone.

Main role activities carried out by a scrum master includes:

- He/she is a facilitator and servant leader who encourages and demands self-organization from the development team.
- He/she enables close cooperation across all roles and functions, addresses resource issue and disobedience of scrum practices.
- He/she protects the team from external and internal distractions.

- He/she is a facilitator and servant leader who encourages and demands self-organization from the development team.
- He/she removes impediments so the team can focus on the work at hand and follow scrum practices.
- He/she is not typically a manager or lead, but he is an influential leader and coach who does not do direct command and control.

Chapter 5

Scrum Events

Meetings or “ceremonies” are an important part of Scrum. They help to broadcast information to all team members, bring common goal and vision, and share team progress appropriately.

Every project has two elements:

- **The content aspect:** What you commit to achieving, such as writing software.

- **The process aspect:** The activities you perform to keep the content work on a track, such as updating gantt charts and writing status reports.

Ceremonies or Events in Scrum:

- Sprint Planning
- Daily Stand-up
- Sprint Review
- Sprint Retrospective

5.1 Sprint Planning

Sprint planning meeting is a time-boxed session that lasts roughly 1 hour for every week of a sprint. In sprint planning, the entire **team agrees** to complete a **set of product backlog items** within a sprint.

Those set of items called as the **sprint backlog.** Team commits based on past **velocity** or **capacity** and the **duration of the sprint**.

Every project has two elements:

Time-box:

4 hours for 2 weeks sprint or 8 hours for 4 weeks sprint

Attendees:

- **Scrum Master:** who facilitates the meeting.
- **Product Owner:** who clarifies the details of the product backlog items and their respective acceptance criteria.
- **Development Team:** who splits the items into smaller tasks and estimates required efforts to meet the sprint commitment.

When:

Before starting a new sprint.

Inputs:

- Product Backlog
- Team Capacity

Outputs:

- **Sprint Backlog:** A sprint backlog is a list of the items the team commits to deliver within a sprint.

- **Sprint Goal:** A sprint goal is a short, one- or two-sentence, description of what the team plans to achieve during the sprint. It is written collaboratively by the team and the product owner.

Sprint planning process:

- Define precise goal for a sprint
- Choose the user stories that support the sprint goal
- Estimate user stories
- Commits to the stories based on team capacity
- Break user stories into smaller tasks
- Assign tasks to individuals

5.2 Daily Stand-up

In Scrum, on each day of a sprint, the team holds a scrum meeting called the "Daily Scrum". Meetings are typically held in the same location and at the same time each day.

Ideally, the Daily Scrum meeting is **held in the morning** as it helps to set the context for the day's work. It is strictly time-boxed to **15 minutes**.

Meeting Specifics

Time-box:

- 15 Minutes

Attendees:

- **Scrum Master:** who facilitates the meeting.

- **Product Owner (Optional):** who clarifies the details of the product backlog items and their respective acceptance criteria.

- **Development Team:** who splits the items into smaller tasks and estimates required efforts to meet the sprint commitment.

When:

It is held each working day at the same time and at the same place.

How is the meeting held?

The Daily Scrum meeting is not used as a problem-solving or issue resolution meeting. Issues are taken offline and usually dealt by the relevant personnel after the meeting. During the Daily Scrum, each team member answers the following three questions :

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

Or alternatively, as an advancement of traditional Scrum practice and/or market trends, each team member may explain:

- How much of work has been completed by him/her in the sprint?
- How much work is still remaining in the Sprint?
- Does anything impede progress? If so what it is?

By focusing on what each person accomplished yesterday and will accomplish today, the team gains an **excellent understanding** of what work has been done and what work remains towards Sprint commitment.

The Daily Scrum meeting is not a status update meeting in which a boss is collecting information about who is behind schedule. Rather, it is a meeting in which team members make commitments to each other.

Objective of daily stand-up

The Daily Scrum Meeting is held so the team can self-organize and achieve its sprint commitment.

The key objectives are:

- **Team Sync:** To review progress toward the Sprint goal.

- **Assess Risks:** To assess any risks to the Sprint commitment.

- **Adjust Plan:** To make any adjustments if required to the plan so the sprint commitment can be met.

- **Accountability:** The team should hold each other accountable for achieving their daily commitments.

Key benefits

• The team stays in sync with how the project is going.

• There is an opportunity for swift course correction if needed.

• By consistently committing and delivering, trust is built among team members.

• Action can be taken with those who are consistently unable to make their commitments (not always pleasant, but usually necessary).

Do's

Timebox your meeting duration: Without a specific time box in place, a Daily Scrum meeting can drag on and actually waste the time of the team. A typical daily meeting should get over in 15 minutes – but larger teams could agree upon a longer duration. The idea is to stop when that time runs out, and keep all discussions short and to the point.

Start the meeting on time: Ensure that the Stand-up starts on time, even if some development team members are missing. A delay of even five minutes, when multiplied by the number of team members, counts for a lot of potential work wasted.

Peer to peer interaction: All interactions should be on a peer-to-peer basis, and not in the form of a status report made to the Scrum Master. Teamwork is always emphasized in Scrum, and the team should communicate well with each other.

Ask the 3 main questions and answer them: There are three questions that are sought to be asked and answered by each team member. What did you complete yesterday? What will you do today? Did you find any impediments in your way? Answers should be concise and precise.

Stay focused: The Scrum Master must keep the team focused on answering the three main questions and prevent it from straying into random discussions. Each team member should focus on the meeting and not get distracted by phone calls or emails. This can extend the time box, or worse, remove the focus of the team from the meeting.

Don'ts

Don't use it as a status reporting meeting: Apart from answering the three basic questions, the meeting should not be used to discuss other topics or problems even if they're identified as impediments by the team members. The main purpose of the stand-up meeting is to make the team conversant about the sprint current progress and set the focus upon what work's going to be done on that day.

Don't micromanage the meeting: The product owner, scrum master or any other team member should not micromanage the meeting. The stand-up should not be used as a venue for issuing instructions to team members or deciding how the team should carry out its work. Scrum teams are empowered to decide on their own how work should be taken up and completed. No efforts should be made to monopolize or dominate the meeting.

Don't remain seated during the stand-up meeting: As the name suggests, the entire team should remain standing during the Stand-up. The idea is to keep the team members on their toes so the meeting does not exceed the time box and they get charged up for the day’s work.

Don’t hold the stand-up meeting away from the work location: With the exception of remote or distributed teams, the daily stand-up should be ideally held on or near the workplace or development room, preferably near the Scrum board. The idea is to keep the team charged up to tackle the day’s work and start with work immediately after the meeting. By holding the meeting away from the workplace, the team could lose its focus by engaging in other discussions while reaching the workplace.

5.3 Sprint Review

A Sprint Review (Demo meeting) is held at the end of each sprint to inspect the product increment. The Product owner reviews and accepts or reject the user stories.

Meeting Specifics

Time-box:

- 1.5 hours (2 weeks sprint) / 3 hours (4 weeks sprint)

Attendees:

- Scrum Master
- Product Owner
- Scrum Team
- Stakeholder and Sponsors
- Customer

When:

It is held at the end of each sprint.

How is the meeting held?

During sprint review meeting, the development team demonstrate the product increment done in a sprint.

The scrum team goes through the forecasted sprint backlog items and reviews how the requirements (user stories) have been realized in the product increment.

After the demonstration, the product owner and stakeholders tell their impressions and clarify if a requirement is not implemented rightly. They also identifies what has been done and what hasn’t been done according to the Definition of Done. The product owner also accept or reject stories.

The result of this meeting can helps team to plan for the subsequent sprint.

5.4 Sprint Retrospective

The sprint retrospective is a meeting facilitated by the Scrum Master at which the team discusses the just-concluded sprint and determines what could change that might make the next sprint more productive.

The sprint review looks at what the team is building, whereas the retrospective looks at how they are building it.

The sprint retrospective is an important mechanism that allows a team to continuously evolve and improve throughout the life of a project. It is important that everyone, including the team members, product owner, and scrum Master, get a chance to air their opinions in an open, honest, yet constructive atmosphere.

Meeting Specifics

Time-box:

- 2 hours (2 weeks sprint) / 4 hours (4 weeks sprint)

Attendees:

- Scrum Master
- Product Owner
- Scrum Team

When:

It is held at the end of each sprint.

How is the meeting held?

The Sprint Retrospective is held after the sprint review at the end of each sprint. During the retrospective, the team self-identifies elements of the process that did or did not work during the sprint, along with potential solutions. It aims to continuously improve the processes.

Sprint Retrospective meetings can be facilitated by asking each person in the team to answer the following questions:

- What went well during the sprint?
- What would we like to change?
- How can we implement that change?

Alternatively, instead of asking what went well and what didn’t go well, the following questions may be asked:

- What should we start doing?
- What should we stop doing?
- What should we continue to do?

Additional topics can also be considered for discussion:

- **Results:** Compare the amount of work planned with what the development team actually completed. Review the sprint burndown chart and identify work team couldn’t completed and find the reason behind that.

- **People:** Discuss team composition and alignment.

- **Relationships:** Talk about communication, collaboration, and working in pairs.

- **Processes:** Go over getting support, development, and code review processes.

- **Tools:** How are the different tools working for the scrum team? Think about the artifacts, electronic tools, communication tools, and technical tools.

- **Productivity:** How can the team improve productivity and get the most work done within the next sprint?

Teams are asked to be specific in their answers so that effective actions can be taken.

The retrospective meeting is usually conducted immediately after the Sprint Review meeting. It is recommended that the entire team participate in this exercise so that any issues or concerns that the teams face during the previous Sprint are addressed during the teaming and avoided in upcoming Sprints.

Benefits of Retrospective Meeting

Process improvements are made at the end of every sprint. This ensures that the project team is always improving the way it works.

- Identifies what went well and what could be improved
- Improve process continuously
- Bring transparency
- Increase team bonding
- Create open culture
- build the team’s sense of ownership and its self-management

Tips for effective retrospective meeting

- Review notes and actions from the previous retrospective
- Ask what problems, successes and opportunities you have with the team, the product, and the process
- Let each team member speak without discussion
- Vote on the most important items to take action on
- Use “five whys” to discover the root cause of problems.
- Create an action plan for your top priority items.
- Explicitly allocate time for improvement actions
- Record your retrospectives & actions

quickscrum
Brings Agility

Thank you