Using Scrum in Education: A Definitive Guide

Table of Contents

Version Control

Version	Author	Date	Changes
0.1	Vlad Krotov	07-12-2023	First draft created

Part 1: Scrum

Scrum: Overview

Scrum is the most widely used Agile Project Management methodology. Surveys routinely show that most organizations across all industries practice Agile - a new, flexible, and adaptive approach to Project Managements. Scrum is the most commonly used project management methodology among organizations using Agile.

Scrum has its origins in software development, but it can be used in other industries as well. It provides a flexible and iterative approach to managing and delivering complex projects. Scrum is not geared towards a particular industry or product; it promotes collaboration, self-organization, and continuous improvement within a cross-functional team regardless of what kind of a product or project the team is working on.

Scrum Explained in 100 words

Scrum is an Agile project management methodology with roots in software development. The focus of Scrum is on delivering products that meet customer expectations. Under Scrum, project work is broken down into small, assignable tasks or user stories. The team proceeds to work on these tasks in iterative fashion, where a demonstrable product is delivered at the end of each iteration. Scrum aims to empower people and foster flexibility and responsiveness of team work. The ultimate goal of Scrum is to build and sustain high performance teams that build products that satisfy client requirements.

Scrum Origins

Scrum was first introduced in the early 1990s by Jeff Sutherland and Ken Schwaber. They developed the framework based on their experiences working on complex software development projects and drawing inspiration from various management and team organization theories.

Figure 1. Jeff Sutherland (left) and Ken Schwaber
(Source: Scrum.org, 2023)

The term Scrum was initially coined by Hirotaka Takeuchi and Ikujiro Nonaka in their 1986 Harvard Business Review article titled "The New New Product Development Game." In this article, they argued that successful companies increasingly use a flexible, iterative approach to product development (as opposed to using a highly structured, “waterfall” approach that was widely practiced previously) (see Figure 2). They used the term “Scrum” to describe this highly flexible and iterative approach to product development that emphasized teamwork and collaboration. Sutherland and Schwaber found the concept of Scrum described in this article to be applicable to software development projects and adapted it to create the Scrum framework.

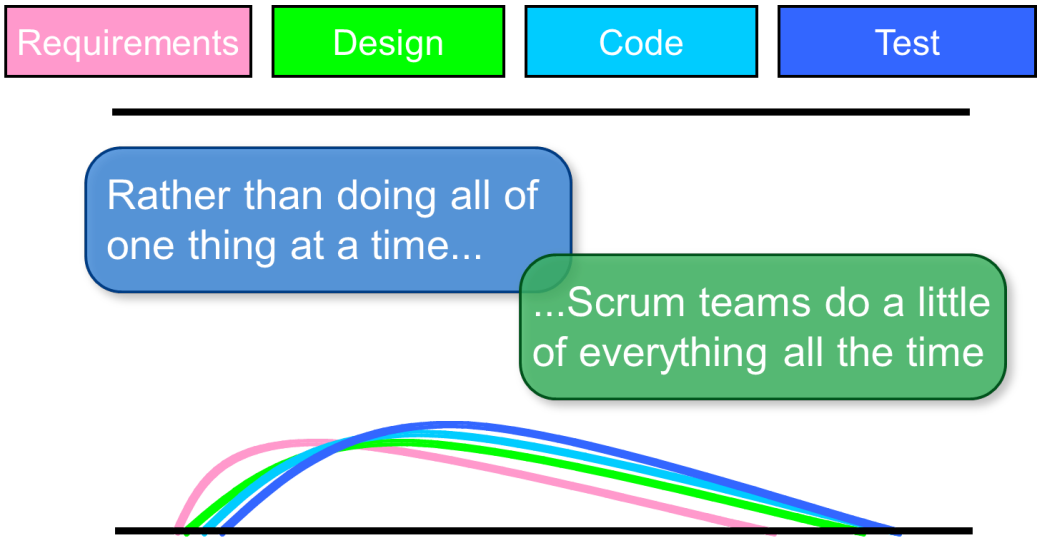


Figure 2. A New Approach to Product Development)

In the mid-1990s, Sutherland and Schwaber collaborated to formalize and document the Scrum framework. They published the first version of the Scrum Guide in 2010, which outlined the roles, events, artifacts, and principles of Scrum. The Scrum Guide has undergone several updates since then to refine and clarify the framework based on their ongoing experiences and feedback from the Scrum community.

Over time, Scrum has gained significant popularity and has been widely adopted by organizations around the world, not just in software development but also in various other industries, such as Education. It has become one of the most popular Agile project management frameworks, known for its flexibility, transparency, and emphasis on delivering value to clients in iterative increments.

Scrum and Agile

Scrum emerged from the Agile school of thought of Project Management. The Agile approach was conceived as a more realistic and effective alternative to the traditional, “waterfall”

development approach, which was the predominant approach to project management up to 1990's.

Waterfall development, also known as the "waterfall model", is a linear and sequential software development methodology (see Figure 3). It follows a structured approach, where each phase of the development process is completed before moving on to the next phase. Waterfall development is often contrasted with Agile methodologies, such as Scrum or Kanban, which emphasize flexibility and iterative development.

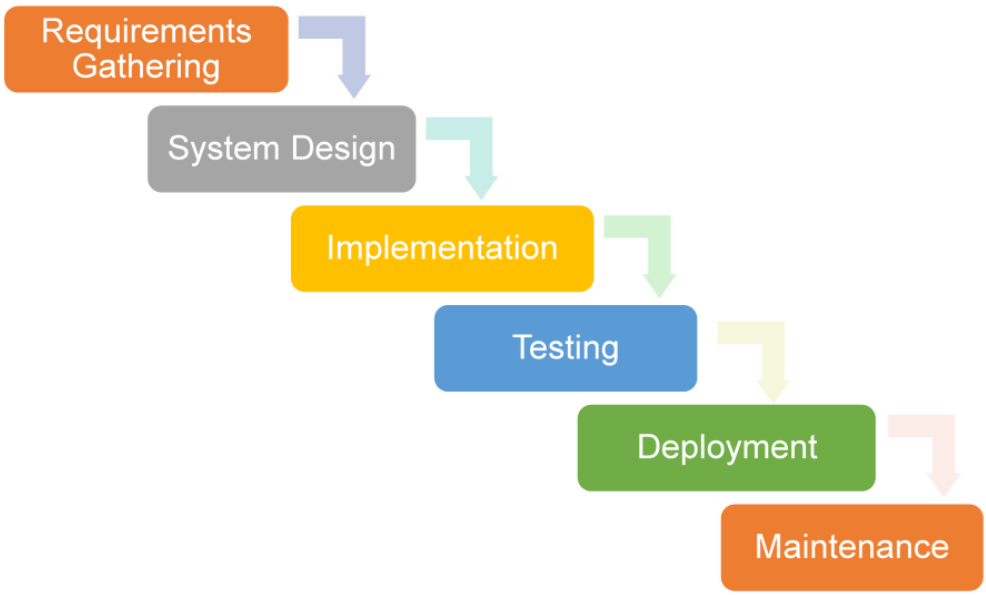


Figure 3. The Waterfall Approach to Software Development

The waterfall model typically consists of the following sequential phases:

- **Requirements Gathering**: The initial phase where the project requirements are gathered and documented in detail. This involves understanding the customer's needs and expectations.
- **System Design**: The requirements are translated into a system design, including architecture, modules, and data flow. This phase focuses on the overall structure and planning of the software system.
- **Implementation**: The actual development of the software takes place during this phase. The programming code is written, and the system design is transformed into a working product.
- **Testing**: The software is tested to ensure that it functions as expected and meets the specified requirements. Testing involves various activities, such as unit testing, integration testing, and system testing.
- **Deployment**: Once the software passes testing and quality assurance, it is deployed or released to the end-users or customers. This phase may involve installation, configuration, and user training.

- **Maintenance**: After deployment, the software enters the maintenance phase, where updates, bug fixes, and enhancements are addressed based on user feedback and evolving requirements.

The waterfall model assumes that all the requirements can be gathered and defined upfront, and there is little to no room for changes or iterations during development. Each phase has its defined deliverables, and progress flows sequentially from one phase to another.

While the waterfall model offers a structured and systematic approach to software development, it has certain limitations. The rigidity of the sequential process makes it challenging to accommodate changing requirements or feedback from users until later stages. Additionally, the long development cycle may result in delayed feedback and increased risks if any issues are identified late in the process.

Due to these limitations, many software development teams have shifted to more agile approaches that emphasize flexibility, collaboration, and iterative development, allowing for quicker feedback and adaptation.

The Agile approach was first captured and publicized in the form of the so-called “Agile Manifesto” (see https://agilemanifesto.org/) . Agile Manifesto outlines the values and principles of agile software development and is a foundational document for the Agile movement. It was created in 2001 by a group of seventeen software development thought leaders who sought to redefine the way software projects were approached.

The Agile Manifesto consists of four key values (see Figure 4) and twelve supporting principles. These values and principles emphasize collaboration, adaptability, customer satisfaction, and iterative development.

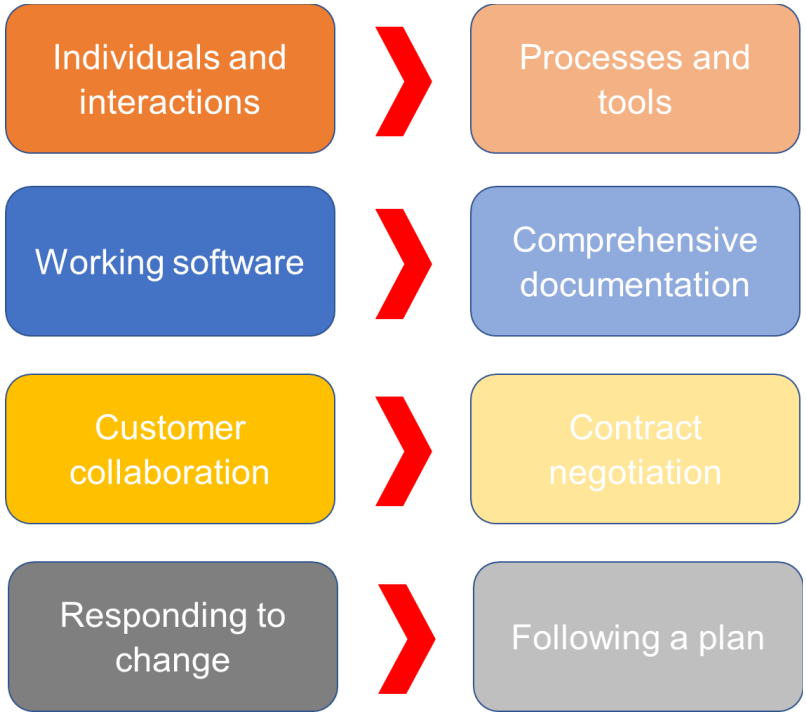


Figure 4. Key Agile Values

Here are the four key values of the Agile Manifesto:

- **Individuals and Interactions over Processes and Tools:** Agile values the importance of people and their interactions in the software development process. It emphasizes that effective communication, collaboration, and teamwork are crucial for project success.
- **Working Software over Comprehensive Documentation**: Agile prioritizes delivering working software that meets customer needs over excessive documentation. While documentation is important, the focus is on creating tangible value through functioning software.
- **Customer Collaboration over Contract Negotiation:** Agile encourages active customer involvement throughout the development process. Collaboration and feedback from customers enable better alignment with their requirements and enhance overall product quality.
- **Responding to Change over Following a Plan**: Agile recognizes that change is inevitable in software development. It promotes flexibility and the ability to respond to changing requirements and circumstances, rather than strictly adhering to a predetermined plan.

The Agile Manifesto also includes twelve supporting principles that further guide agile development practices. These principles emphasize continuous delivery, frequent customer feedback, self-organizing teams, and embracing change as a competitive advantage.

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The Agile Manifesto has had a significant impact on software development methodologies and has influenced the widespread adoption of agile frameworks like Scrum, Kanban, and Extreme

Programming (XP), among many others. It promotes a mindset of adaptability, collaboration, and customer-centricity, enabling teams to deliver high-quality software in a more iterative and responsive manner.

The Scrum Framework

In comparison to traditional project management methodologies, the Scrum framework is deliberately "light" (see Figure 5). The framework is based on simple, generative rules that empower people and give them an opportunity to show their best performance. For example, the framework does not prescribe any technologies or tools; it is up to the team to decide which technologies and tools should be used to deliver a product. The Scrum framework is easy to understand, but takes years to master. In most cases, effective use of Scrum is a result of extensive experience and tacit knowledge rather than intellectual understanding of this framework's "theory".

Scrum emphasizes empirical process control, which means that decisions are based on observation and feedback rather than rigid planning. This allows for flexibility and adaptability throughout the project. The framework encourages regular reflection and improvement through retrospective meetings held after each Sprint, where the team discusses what went well, what could be improved, and decides on actions for the next iteration.

Overall, Scrum provides a framework that enables teams to deliver high-quality products in a collaborative and efficient manner, with a focus on continuous learning and adaptation.

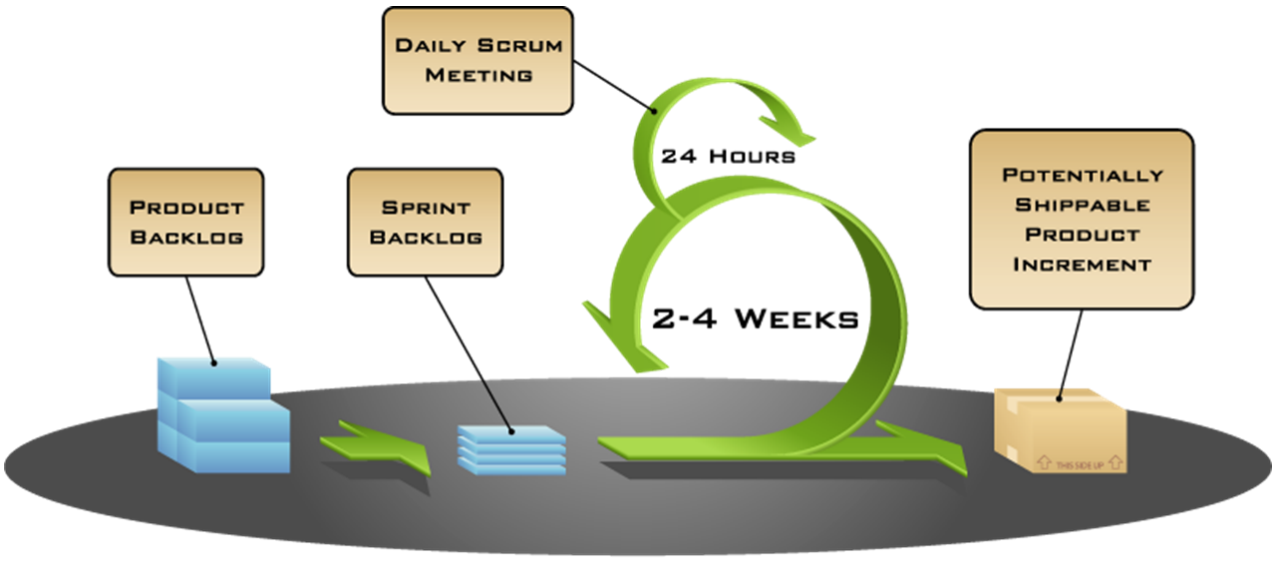


Figure 5. The Scrum Framework

The starting point of Scrum is forming a Scrum Team of three to nine people. The team is comprised of a Scrum Master (responsible for coordinating team work, removing impediments, and enforcing Scrum values and processes), Product Owner (represents the customer and serves a the ultimate authority on product requirements), and Development Team (team members with various technical and non-technical skills required for delivering the product). Once the team is formed, it creates a Product Backlog composed of tasks or User Stories. These user stories are phrases from the customer perspective (e.g. “as a traveler, I would like to be able to browse through various travel insurance options available through the website”). Collectively, the Product Backlog represents the total functionality of the product. The User Stories should be prioritized. The entire Product Backlog can be modified and reprioritized between Sprints.

After the Product Backlog is created, the team selects a few items from the Product Backlog for the Sprint. In software development, a Sprint is a work interval of approximately 2-4 weeks. For each Sprint, the Scrum Team commits to deliver a certain subset of the Product Backlog. During each Sprint, the team participates in Daily Scrums - brief, 15-minutes, stand-up meetings. During the meeting, everybody answers the following three questions:

1. What did you do yesterday;
2. What will you do today?
3. Is anything standing in your way?

These Daily Scrums are not for problem solving; team members should collaborate on problem solving outside of the Daily Scrum. Daily Scrums allow team members to “touch base” with each other and commit to work in front of all other team members. Daily Scrums are meant to eliminate unnecessary meetings.

At the end of each Sprint, the Scrum Team conducts a Sprint Review meeting. During this meeting, the team presents what is accomplished during the Sprint by providing a demo of the new product features delivered as a result of the Sprint. The whole team participates in Spring Reviews and these meetings should be open to anyone within the organization.

Periodically, the Scrum Team should conduct a Sprint Retrospective meeting in-between Sprints. The main goal of these meetings is continuous improvement of team work by looking at what is working well for the team, what is not working, and what needs to be improved. These Sprint Retrospectives can be done after each Sprint, if necessary.

Scrum Team

The first step in implementing Scrum is forming a Scrum Team. The ideal size of a Scrum team is between three and nine members. This range is based on the observation that smaller teams tend to have better communication, collaboration, and self-organization. However, if the team is too small, then not enough expertise and positive team dynamics may be available to deliver a quality product.

It is important to note that the ideal team size may vary depending on the specific context and requirements of the project. While three to nine members is the recommended range, it is not a strict rule. Organizations may have multiple Scrum teams working together on larger projects or programs, where each team adheres to the Scrum principles and maintains effective communication and collaboration across teams. Ultimately, the goal is to have a Scrum team size that enables effective communication, collaboration, and self-organization, while also ensuring the team's ability to deliver value and meet the project's objectives.

Consider these factors when deciding on the size and composition of your Scrum team:

- **Keeping the team small enough for effective communication**: Smaller teams generally find it easier to communicate, share information, and coordinate their efforts. With fewer members, it becomes more manageable to have frequent and meaningful interactions, ensuring everyone is on the same page.
- **Promoting collaboration and self-organization**: A small team fosters a sense of ownership and shared responsibility. It allows team members to contribute actively, participate in decision-making, and collaborate closely. Self-organization and cross-functional collaboration are easier to achieve with a smaller group.
- **Ensuring efficiency and flexibility**: A smaller team tends to be more agile and adaptable, enabling faster decision-making and quicker responses to changes. The team can easily reassign or redistribute work among themselves, ensuring a smooth workflow and flexibility to adapt to evolving requirements.
- **Avoiding coordination challenges:** As the team size increases, the complexity of coordinating efforts and managing dependencies tends to rise. Larger teams may face challenges related to aligning schedules, handling communication overhead, and maintaining a high level of transparency and collaboration.

Figure 6. The Scrum Team

The Scrum Team typically consists of a Product Owner, a Scrum Master, and the Development Team (see Figure 5). These roles are explained below:

- **Product Owner**: Represents the stakeholders, customers, and users. The Product Owner is responsible for defining and prioritizing the product backlog, which is a list of features, user stories, or requirements.
- **Scrum Master**: Facilitates the Scrum process and ensures adherence to Scrum principles. The Scrum Master helps the team remove any impediments, promotes collaboration, and facilitates the various Scrum events.
- **Development Team**: The cross-functional team responsible for delivering the product increment. The development team self-organizes and determines how to accomplish the work within the sprint.

Product Backlog and User Stories

After forming the Scrum Team, the Scrum framework requires breaking down a project into small, manageable tasks or product features. These items are usually called User Stories, since they are phrased from an end-user perspective (e.g. “As a hotel customer, I would like to be able to browse through pictures of the hotel”). These User Stories are added to the Product Backlog, which is a prioritized “to do” list for the entire project.

The Scrum team should provide estimates of the weight of each of the User Story in terms of its effort and complexity. In Scrum, the weight or effort estimation for user stories is typically done using a relative scale rather than precise time-based estimates. Here are a few common ways to provide weight for user stories in Scrum:

- **Person-hours**: User stories can be assigned quick estimates in person-hours. For each user story, a Scrum Master can ask the team to estimate how many hours it is likely to take a person on the team to complete this task. A Scrum Master can do these estimates himself or herself, ask team members to develop a consensus, or use a more elaborate and structured technique for these estimate, such as Planning Poker (please see below)
- **Planning Poker**: Planning Poker is a collaborative technique where the development team collectively estimates the effort or complexity of user stories. Each team member uses a deck of cards with numbers representing different levels of effort (e.g., Fibonacci sequence: 1, 2, 3, 5, 8, 13, etc.). They simultaneously reveal their chosen card, and if there is a significant difference in estimates, the team discusses the reasons and re-estimates until a consensus is reached.
- **T-Shirt Sizes**: T-Shirt sizing is a simplified method where user stories are assigned sizes based on relative comparisons. Sizes are represented by T-shirt sizes like XS, S, M, L, XL, etc., indicating the effort or complexity of each story. The team collectively decides the size by comparing the user story to others and agreeing on the most appropriate size.
- **Bucket System**: The Bucket System is a technique where user stories are grouped into different "buckets" based on their complexity or effort level. For example, stories can be categorized into buckets like Small, Medium, Large, or Low, Medium, High. This method

allows for a rough estimation of the stories without the need for precise numerical values.

- **Story Points**: Story points are a numerical value assigned to user stories to indicate their relative effort or complexity. The specific scale used for story points can vary, but often teams use a Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.) or a modified Fibonacci sequence. The team collectively assigns story points based on their judgment and experience, considering factors such as effort, complexity, risk, and uncertainty. These Story Points may or may not correspond to actual units of measurements, such as person-hours.

It is important to note that the primary purpose of these estimation techniques is to facilitate discussion, create a shared understanding within the team, and aid in prioritization and planning. The actual time required to complete a story may vary based on various factors, and the estimation serves as a guide rather than an absolute prediction. Over time, the team can calibrate their estimates based on their historical velocity and use that information for future planning and forecasting.

Sprints

The team proceeds to work on these tasks using Sprints. Sprints are time-boxed iterations, usually ranging from two to four weeks, during which a set of prioritized tasks or User Stories from the Backlog are completed. Each Sprint begins with Sprint Planning where the team selects the work to be done and defines the goals for the sprint. During a Sprint, the team holds daily stand-up meetings, known as Daily Scrums, to provide progress updates, discuss any impediments, and plan the day's activities. These meetings promote transparency and help the team stay focused and aligned. At the end of every Sprint, the Scrum Team delivers a Potentially Shippable Product Increment - which is a demonstrable subset of the entire product, something that can be shown and evaluated by the entire Scrum Team before embarking on another Sprint. A Sprint Review meeting is used to demonstrate the Potentially Shippable Product Increment and solicit feedback in relation to the product from relevant stakeholders. A Spring Retrospective meeting can also be held at the end of every Sprint to reflect and improve on teamwork.

All-in-all, scrum incorporates the following processes or ceremonies to facilitate communication and track progress:

- Sprint Planning: The team plans the upcoming sprint, selects the work to be done, and defines the sprint goal.
- Daily Scrum: A brief daily meeting where team members share progress, discuss any obstacles, and plan their work for the day.
- Sprint Review: A meeting held at the end of each sprint to demonstrate the completed work to stakeholders and gather feedback.
- Sprint Retrospective: A reflection session where the team discusses what went well, what can be improved, and identifies actionable items for the next sprint.

Each of these ceremonies is discussed in more detail in the sections below.

Spring Planning

Sprint Planning is an important event in Scrum that occurs at the beginning of each Sprint. It involves the Scrum Team (Product Owner, Scrum Master, and Development Team) collaborating to determine the work to be undertaken during the sprint. It is essential to have the right stakeholders present to ensure a shared understanding of the upcoming work.

The Sprint Planning meeting should have a fixed duration or a “timebox” - depending on the nature and complexity of the project (e.g 1-3 hours). The Scrum Master ensures that the meeting stays within the allocated time to maintain focus and productivity.

Here's what typically happens in a Sprint Planning session:

- **Formulation the Sprint Goal**: The Product Owner discusses the overarching goal or objective for the sprint. This goal provides a clear direction for the team and serves as a guiding principle for their work during the sprint.
- **Performing Product Backlog Review**: The Product Owner reviews the product backlog with the team. They present the highest-priority items and ensure that they are well-refined and ready for selection. Any new or changed user stories are discussed, and clarifications are provided as needed.
- **Selecting User Stories**: The Development Team collaborates with the Product Owner to select user stories from the product backlog for the upcoming sprint. The team assesses the complexity, effort, and dependencies of each story, considering their capacity and capabilities.
- **Defining Sprint Backlog**: Once user stories are selected, the Development Team decomposes them into smaller, actionable tasks. They identify the necessary activities, determine the acceptance criteria, and estimate effort for each task. This breakdown creates the sprint backlog, which is a detailed plan for the work to be completed in the sprint.

By the end of the meeting, the team should have a Sprint Backlog consisting of selected User Stories (including their decompositions into smaller sub-tasks) and estimated effort for each of these User Stories. The team should have a shared understanding of what needs to be accomplished in the Sprint. At the end of the meeting, the team reconfirms the Sprint Goal, ensuring that it aligns with the selected User Stories and the overall purpose of the Sprint. This provides a unifying focus for the team's work and helps them stay on track throughout the Sprint.

To sum up, Sprint Planning is a collaborative process that enables the team to establish a clear plan for the sprint. It helps ensure that the team is aligned with the product vision, sets realistic expectations, and provides a foundation for successful sprint execution.

Daily Scrum

The Daily Scrum, also known as the “daily stand-up”, is a brief and time-boxed meeting that occurs every day during a sprint in Scrum. It provides an opportunity for the Development Team to synchronize their activities, share progress updates with each other, and plan their work for the upcoming day.

The Daily Scrum usually involves only the members of the Development Team, including developers, testers, and any other team members involved in the Sprint. The Product Owner and Scrum Master may attend as observers but do not actively participate unless they are also part of the Development Team.

The Daily Scrum is time-boxed to a maximum of 15 minutes, regardless of the team size. This time constraint encourages participants to focus on the most critical information and keep the meeting concise and efficient.

The Daily Scrum is often conducted with the team standing up to maintain energy and encourage brevity. The standing position helps keep the meeting short and encourages participants to share updates quickly without delving into lengthy discussions.

The Scrum Master facilitates the Daily Scrum, ensuring that the meeting stays focused, remains within the timebox, and promotes active participation. They may ask additional questions to foster collaboration, help remove any identified impediments, and observe the team's progress.

The Development Team members take turns answering three key questions during the Daily Scrum:

1. **What did I accomplish yesterday?** Each team member provides a brief update on the work they completed since the last Daily Scrum. They highlight any significant progress, completed tasks, or obstacles encountered.
2. **What will I do today?** Team members share their plans for the current day, outlining the tasks or user stories they will be working on. They focus on what they intend to accomplish to advance the sprint goal.
3. **Are there any obstacles or blockers?** If any team members are facing challenges or impediments that hinder their progress, they bring them up during this time. The purpose is to identify obstacles early and seek help from the Scrum Master or collaborate with other team members to address them.

The Daily Scrum fosters collaboration and synchronization among team members. It allows them to identify dependencies, align their work, and ensure that everyone is working towards the sprint goal. If necessary, team members can engage in quick discussions to resolve any issues or provide clarifications. But Daily Scrums should not be used for problem-solving. Team members should collaborate on solving problems during Sprints.

The Daily Scrum is not intended to be a status reporting meeting to the Product Owner or management. Instead, it serves as a short, daily synchronization and planning session for the Development Team. It enables the team to stay aligned, address potential issues proactively, and maintain a steady pace of progress throughout the sprint.

Sprint Review

The Sprint Review is a key process or ceremony in the Scrum framework that takes place at the end of each sprint. It provides an opportunity for the Scrum Team to showcase the work accomplished during the sprint and gather feedback from stakeholders.

The Sprint Review is attended by the Scrum Team, including the Product Owner, Scrum Master, and Development Team members. Additionally, stakeholders such as customers, users, managers, and other relevant individuals are invited to provide feedback and insights.

The Sprint Review is time-boxed, with the duration typically ranging from one to two hours, depending on the length of the Sprint. The Scrum Master ensures that the meeting stays within the allocated time to maintain focus and productivity.

Here's what typically happens in a Sprint Review:

- **Demonstrating the Increment:** The Development Team presents the completed increment, which includes the User Stories and features that were planned for the Sprint. They showcase the functionality, features, and any other relevant deliverables to the stakeholders. This demonstration can be done through live demonstrations, interactive sessions, or presentations.
- **Collecting Feedback**: Stakeholders have the opportunity to provide feedback on the increment presented. They can ask questions, provide suggestions, or express their satisfaction or concerns. This feedback is valuable for the Scrum Team to gather insights and identify areas for improvement.
- **Discussing the Product Backlog**: The Product Owner and stakeholders discuss the Product Backlog, reviewing the priorities and potential changes based on the feedback received during the Sprint Review. This discussion may include reprioritizing user stories, adding new items, or making adjustments to the future work plan based on the stakeholders' input.
- **Sprint Retrospective Preparation**: The Sprint Review often serves as a transition to the subsequent Sprint Retrospective meeting. The Scrum Team may identify topics or observations during the review that can be further discussed and addressed in the retrospective to improve their processes.

The Sprint Review serves as an opportunity for the Scrum Team to demonstrate progress, gather valuable feedback, and make adjustments to the product backlog based on stakeholder input. It promotes transparency, collaboration, and continuous improvement by incorporating stakeholders' perspectives into the development process.

Sprint Retrospective

A Sprint Retrospective meeting is a dedicated session that takes place at the end of each Sprint in Scrum. It provides an opportunity for the Scrum Team (including the Product Owner, Scrum Master, and Development Team) to reflect on the just-concluded sprint and identify areas for improvement. The primary goal of the Sprint Retrospective is to foster a culture of continuous improvement within the team.

The Sprint Retrospective is timeboxed, usually to a maximum of three hours for a four-week sprint. For shorter sprints, the timebox is adjusted accordingly. The time constraint ensures that the team remains focused and allows for efficient discussions.

The Sprint Retrospective is attended by the Scrum Team, including the Product Owner, Scrum Master, and Development Team members. The purpose is to encourage open communication and collaboration among all team members.

Here are the key aspects of a Sprint Retrospective meeting:

- **Reflection**: The team reflects on the completed Sprint, discussing what went well and what could have been improved. They examine the process, teamwork, communication, tools, and any other factors that influenced the sprint.
- **Feedback**: Each team member is encouraged to provide constructive feedback on various aspects of the Sprint. They can share their observations, challenges faced, and suggestions for improvement. The focus is on fostering a blame-free environment where open and honest feedback is encouraged.
- **Improvement**: The team collectively identifies specific improvements to implement in the upcoming Sprints. These improvements can relate to the development process, communication, collaboration, tools, or any other area that can enhance the team's effectiveness and productivity.
- **Action Items**: The team captures the agreed-upon improvements as actionable items or tasks. Each item is assigned to a responsible team member or shared collectively. These action items are later prioritized and integrated into the next sprint backlog or the team's working agreement.
- **Follow-up**: The Scrum Master tracks the progress of the identified improvements, ensuring that they are addressed in subsequent Sprints. The Sprint Retrospective serves as an input for continuous improvement, and the team's commitment to making changes is vital for its success.

The Sprint Retrospective is a critical Scrum ceremony that promotes transparency, reflection, and adaptation. It allows the team to learn from their experiences, make adjustments, and continuously refine their processes. By regularly conducting retrospectives, the team can steadily improve their productivity, quality, and overall performance.

Scrum Artifacts

Several artifacts are used by Scrum teams as a part of Scrum framework implementation. These artifacts include:

- **Product Backlog**: A prioritized list of features, user stories, or requirements that serve as the input for sprint planning.
- **Sprint Backlog**: The set of tasks or user stories selected from the product backlog for a specific sprint.
- **Burndown Chart**: A visual representation of the work remaining in a sprint, helping track progress and manage scope.

A Product Backlog is used to create and prioritize a set of features that are to be implemented as a part of the product development process. For each Sprint, a certain subset from the overall Product Backlog is selected for the Sprint Backlog and implemented during the Sprint. Product features or tasks are typically in the form of User Stories that express customer requirements or wishes in relation to a product from an end-user perspective. Each User Story is assigned a score that reflects the difficulty of the tasks involved and the time required to implement this User Story. A Burndown Chart is a graphical representation that visually tracks the progress of work completed during a sprint. It provides a snapshot of the remaining work versus time, helping the Scrum Team and stakeholders understand how the project is progressing and whether the team is on track to meet their sprint goal.

Scrum Board

Both the Product and Sprint Backlogs are typically implemented in the form of a Scrum Board. A Scrum Board, also known as a Kanban Board or Agile Board, is a visual tool used by Scrum Teams to visualize and manage their work during a Sprint. It provides a clear and transparent representation of the status of User Stories or tasks throughout the Sprint.

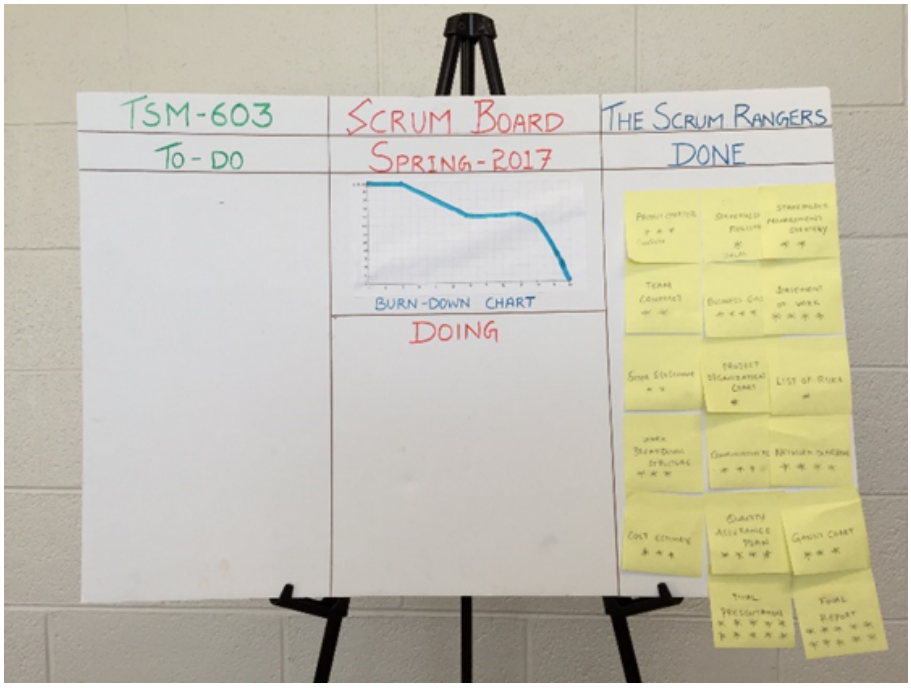


Figure 6. The Scrum Board

The Scrum Board typically consists of columns representing the different stages or states of work in the sprint. Common columns include "To Do," "In Progress," and "Done." Additional columns can be added based on the specific workflow and needs of the team. The “To Do” column represents the Product Backlog, while the “Doing” tab represents the Sprint Backlog.

Each user story or task is represented by a sticky note or card on the Scrum Board. The card contains relevant information, such as the title, a brief description, the assigned team member(s), and any estimated effort or due dates.

As work progresses, team members physically or digitally move the sticky notes/cards across the columns to reflect the current status of each item. For example, a user story might be moved from the "To Do" column to "In Progress" when a team member starts working on it, and finally to the "Done" column when it is completed.

The Scrum Board provides real-time visibility into the progress of the sprint. It allows team members, the Product Owner, and stakeholders to quickly assess the status of work and identify any bottlenecks or impediments. This visualization of team work results in transparency that promotes collaboration and helps the team stay aligned.

The Scrum Board is often used during the Daily Scrum (daily stand-up) meetings. Team members can refer to the board to provide updates on the tasks they are working on, discuss any challenges, and plan their work for the day.

The Scrum Board serves as a visual representation of the team's workflow and can be used to identify areas for improvement. By analyzing the flow of work, the team can identify recurring patterns, optimize their processes, and identify opportunities for increasing efficiency and productivity.

Scrum Boards can be customized to fit the specific needs of the team. Additional columns, such as "Testing" or "Review," can be added to reflect the team's unique workflow and stages of work. Digital tools or software can also be used to create virtual Scrum Boards for distributed teams or for more advanced features like automated updates and metrics.

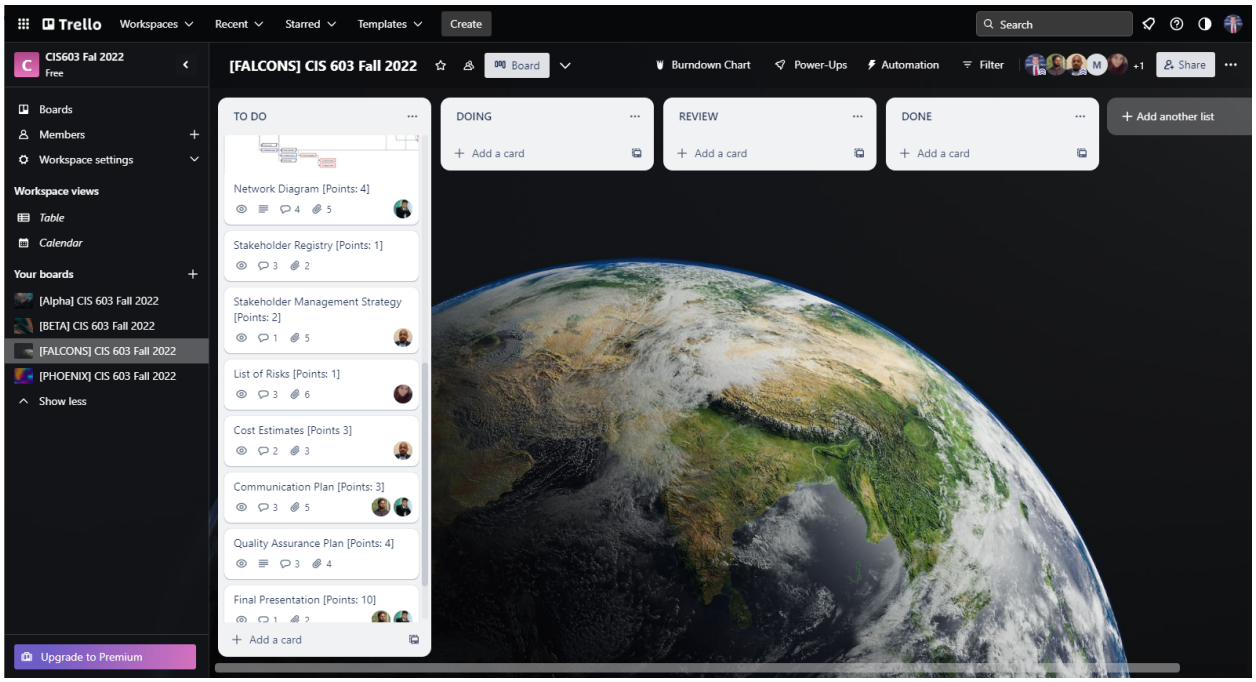


Figure 7. Digital Scrum Board Created in Trello

The Scrum Board helps teams maintain transparency, facilitate collaboration, and track the progress of work during a sprint. It serves as a powerful visual management tool that supports the team in achieving their sprint goals.

Burndown Chart

A Burndown Chart is a visual representation of the progress made by the Development Team during a sprint. It provides a clear indication of how much work remains to be completed and whether the team is on track to meet the sprint goal.

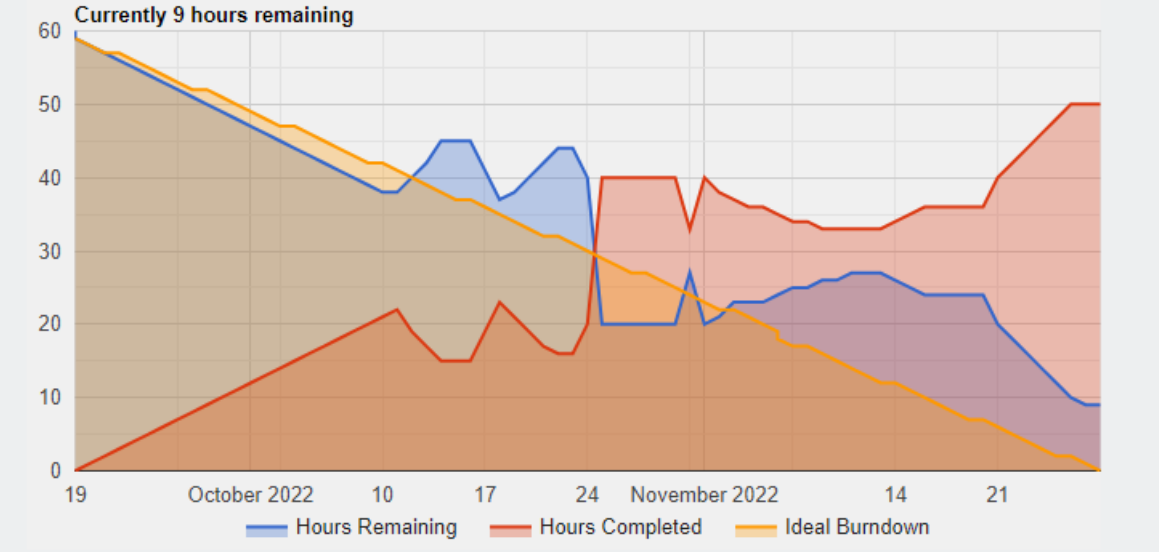


Figure 8. A Burndown Chart Created using a Power-Up in Trello

The Y-axis of the Burndown Chart represents the remaining work or effort, often measured in story points, ideal days, or any other unit of measurement chosen by the team. It shows the amount of work yet to be completed over time.

The X-axis represents the sprint duration, divided into increments such as days or iterations. It shows the progression of time throughout the sprint.

The Ideal Burndown line is a straight line that connects the total effort at the beginning of the sprint with zero effort at the end of the sprint. It represents the ideal progress of the Development Team, assuming that they complete the work evenly and consistently throughout the sprint.

The Hours Completed depicts the team work accomplished over time during the Sprint. It is based on the daily updates provided by the Development Team, reflecting the actual progress made.

The Hours Remaining line depicts the remaining work as it changes over time during the Sprint. It is also based on the daily updates provided by the Development Team, reflecting the actual progress made.

By analyzing the Burndown Chart, the Scrum Team can identify trends and variances. For example, if the Hours Completed line consistently falls below the ideal line, it may indicate that the team is facing challenges or is behind schedule. Conversely, if the Hours completed line is above Ideal Burndown line, it may indicate that the team is progressing faster than expected.

The Burndown Chart promotes transparency by providing a visual representation of the team's progress. It allows the Scrum Team, Product Owner, and stakeholders to make informed decisions regarding scope adjustments, resource allocation, and potential risks.

Scaling Scrum

One of the criticisms of Scrum is that the framework is suitable only for managing small software development teams in small organizations. Large projects composed of hundreds or even thousands of people may not benefit from using Scrum, as large-scale projects require a more structured and hierarchical approach.

Scrum has been “scaled” and used successfully for large-scale projects involving hundreds of people. Scaling Scrum for large projects involves implementing frameworks and practices that allow multiple Scrum teams to work together effectively. Here are some approaches commonly used to scale Scrum:

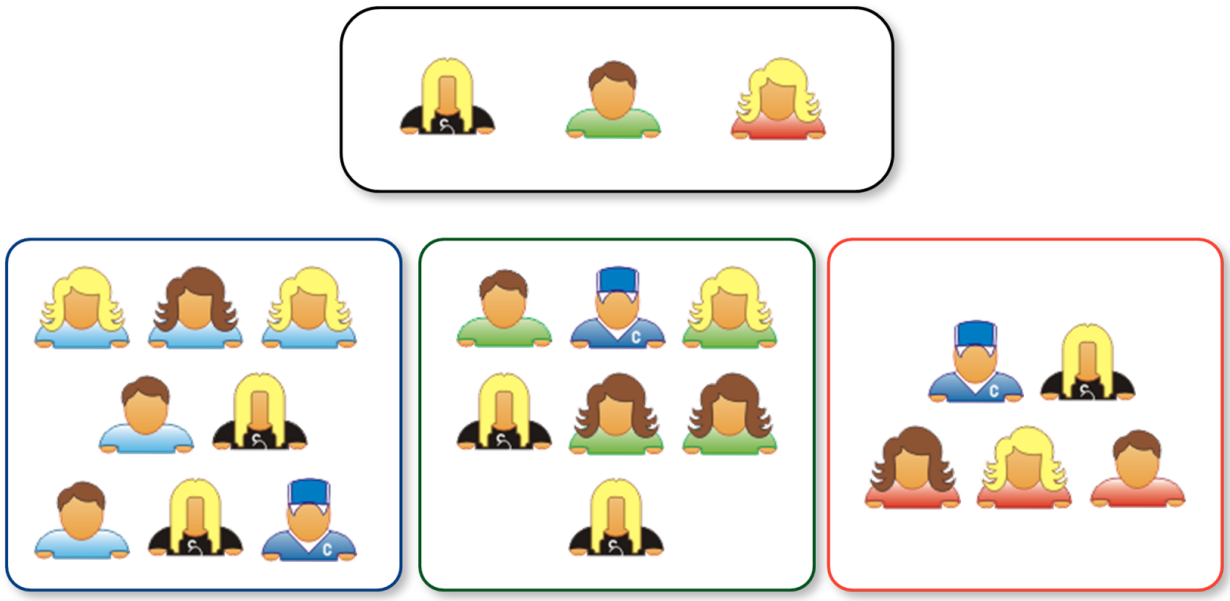


Figure 9. Scaling through Scrum of Scrums

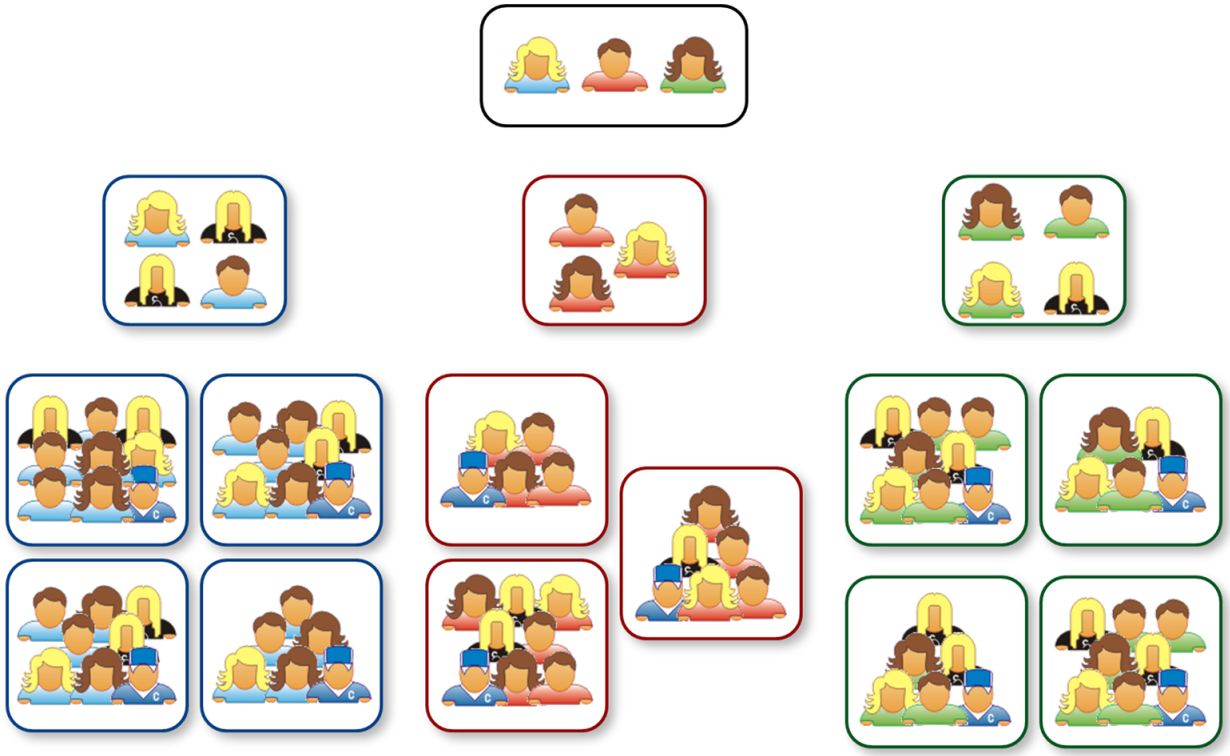
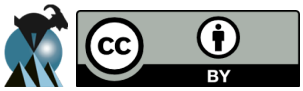


Figure 10. Scaling through Scrum of Scrums of Scrums



- **Scrum of Scrums**: In the Scrum of Scrums approach, representatives from individual Scrum teams form a higher-level Scrum team. They meet regularly to coordinate their work, discuss dependencies, and resolve any cross-team impediments. This approach helps ensure alignment and collaboration across multiple Scrum teams.
- **LeSS (Large-Scale Scrum)**: LeSS is a framework designed specifically for scaling Scrum. It provides guidelines for coordinating multiple Scrum teams working on the same product or project. LeSS promotes a single product backlog, shared sprint planning, and cross-team coordination, while maintaining the core principles of Scrum.
- **SAFe (Scaled Agile Framework)**: SAFe is an enterprise-level scaling framework that incorporates Scrum and other Agile practices. It provides a structured approach for aligning multiple teams across different levels of an organization. SAFe introduces additional roles, artifacts, and events to enable coordination, synchronization, and governance at scale.
- **Nexus Framework**: The Nexus framework, developed by Scrum co-creator Ken Schwaber, focuses on scaling Scrum specifically. It provides a set of practices and guidelines to help multiple Scrum teams work together seamlessly. The Nexus framework emphasizes shared goals, integrated sprint planning, and frequent collaboration among teams.
- **Disciplined Agile Delivery (DAD)**: DAD is an Agile framework that offers a flexible approach to scaling Scrum and other Agile practices. It provides guidance on tailoring

Agile methods to suit the needs of large, complex projects. DAD incorporates various Agile practices, including Scrum, Kanban, and Lean, to support teams working in a scaled environment.

When scaling Scrum, it is crucial to maintain the core principles and values of Scrum while adapting to the larger project context. Effective communication, collaboration, and alignment are key to successful scaling. It's important to establish clear roles, define shared goals, encourage cross-team coordination, and foster a culture of transparency and continuous improvement.

Choosing the right scaling approach depends on the specific needs and complexity of the project. Organizations often combine different frameworks and practices, tailoring them to their unique circumstances to achieve successful scaling and deliver value in large-scale Agile projects.

Part 2: Using Scrum in Education

Uses of Scrum in Education

Given the popularity of Scrum in organizations, practical knowledge of this agile methodology is a valuable skill for professionals in all industries. Because of that, educational institutions around the globe are increasingly integrating Scrum into their educational process.

One of the most popular uses of Scrum in education is for Project-Based Learning. Learners can form teams, define project goals, create backlogs of tasks or learning objectives, and work through sprints to accomplish their goals. Scrum helps learners learn to collaborate, manage their time effectively, and take ownership of their learning process.

On a broader note, Scrum can facilitate group work and collaboration among students. By adopting Scrum roles such as Scrum Master and Product Owner within student teams, they can organize their work, set goals, track progress, and conduct regular meetings. Scrum provides a framework for improved communication, task transparency, and accountability within student groups.

Some other uses of Scrum in Education include:

- **Curriculum Development**: Scrum can be applied to curriculum development processes. Educators and curriculum developers can define a product backlog containing learning objectives, activities, and assessments. Through sprints, they can plan and execute the development of curriculum materials, review and iterate on the content, and incorporate feedback from students and stakeholders.
- **Agile Teaching Practices**: Scrum principles and practices can be applied to teaching itself. Educators can plan their lessons and course materials in sprints, regularly review and adapt their teaching methods based on student feedback, and foster a collaborative

and interactive learning environment. Agile teaching encourages experimentation, continuous improvement, and adaptability to meet the diverse needs of students.
- **School Administration and Operations**: Scrum can be utilized in school administration and operations, such as organizing events, managing projects, or implementing new initiatives. By adopting Scrum practices, administrative teams can effectively plan, prioritize, and execute tasks, improve communication and collaboration, and ensure timely delivery of projects or initiatives.
- **Professional Development**: Scrum can be incorporated into professional development programs for educators. Teachers can form learning communities, set professional goals, and engage in collaborative learning activities. Applying Scrum principles, they can regularly reflect on their teaching practices, share experiences, and work towards continuous improvement in their professional development journey.

It is important to note that implementing Scrum in education requires adaptation to the specific context and needs of learners and educators. If implemented right, Scrum provides a framework for fostering collaboration, organization, and agility within educational settings, supporting student engagement, and enhancing the learning experience.

Scrum and Soft Skill Development

Scrum can contribute to the development of soft skills, which are non-technical, interpersonal skills that are essential for personal and professional success. Here is how Scrum can help foster the development of soft skills among learners and educators:

- **Collaboration and Teamwork**: Scrum emphasizes collaboration and cross-functional teamwork. Through the daily stand-up meetings, sprint planning, and sprint review sessions, team members learn to communicate effectively, listen to others, and work together towards a common goal. This promotes the development of skills such as communication, active listening, empathy, and conflict resolution.
- **Adaptability and Flexibility**: Scrum encourages adaptability and embracing change. The iterative nature of Scrum means that teams constantly evaluate and adjust their approach based on feedback and changing requirements. This cultivates skills such as resilience, flexibility, and openness to new ideas or perspectives.
- **Time Management and Prioritization**: Scrum emphasizes time-boxed iterations and prioritization of work. Team members learn to manage their time effectively, estimate task durations, and make informed decisions about task sequencing. This promotes skills like time management, organization, and prioritization.
- **Accountability and Ownership**: Scrum fosters a sense of ownership and accountability within the team. Each team member takes responsibility for their assigned tasks and commitments. This cultivates skills like responsibility, self-motivation, and a strong work ethic.
- **Communication and Transparency**: Scrum promotes open and transparent communication among team members. Daily stand-up meetings encourage individuals to share progress, challenges, and seek help if needed. The sprint review sessions facilitate feedback and collaboration with stakeholders. These practices enhance skills

such as clear communication, articulating ideas, and giving and receiving constructive feedback.

- **Problem-Solving and Critical Thinking**: Scrum encourages teams to find creative solutions to challenges and obstacles. Through collaborative problem-solving, team members develop skills in critical thinking, analysis, and decision-making.
- **Leadership and Facilitation**: Scrum roles, such as the Scrum Master and Product Owner, provide opportunities for individuals to develop leadership and facilitation skills. The Scrum Master guides the team, removes impediments, and facilitates the Scrum process. The Product Owner acts as a liaison with stakeholders and represents their interests. These roles cultivate skills like leadership, facilitation, and stakeholder management.

By integrating Scrum practices into projects and teams, learners have the opportunity to enhance their soft skills, which are vital in various aspects of personal and professional life. The collaborative and iterative nature of Scrum creates an environment that encourages individuals to develop and apply these skills continuously, leading to improved teamwork, productivity, and personal growth.

Translating Scrum to the Educational Context

Effective implementing Scrum in education requires adaptation of the framework to the specific context and needs of learners and educators. More specifically, Scrum's artifacts and processes have to be modified to suit the academic environment and pedagogical goals of the educator. This section details a possible adaptation of Scrum to the educational context for project-based learning.

The adaptation consists of two main frameworks: The Educational Scrum Processes (see Figure 1) outlining the main steps or phases of Scrum use in the educational context and the so-called Educational Scrum Translation Matrix listing and defining the main elements Scrum adapted for the educational context (see Table 1).

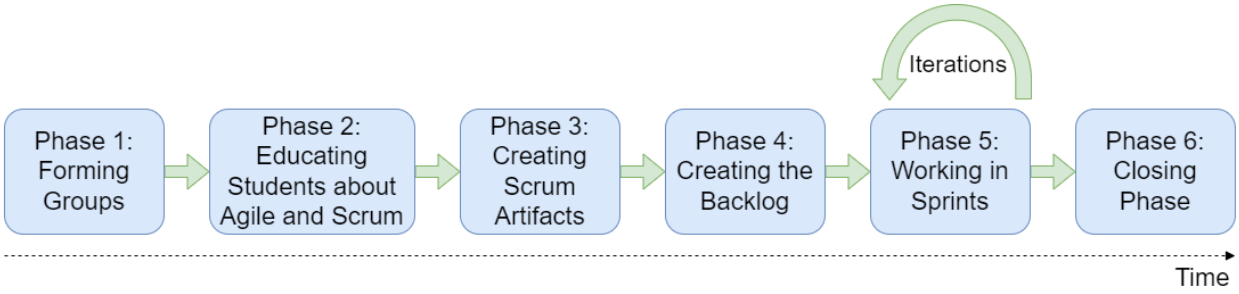


Figure 11.

Table 1. Educational Scrum Translation Matrix

Element	Description
Scrum Board	A physical or digital board containing: (1) Group name; (2) class code; (3) To Do (also known as the Backlog), Doing, and Done columns. The board is populated with “sticky notes” containing information about Deliverables.
Deliverable	A Deliverable is a subset of the entire group project. A template is provided to students for each Deliverable. Students are free to modify these templates. Each Deliverable is described using a “sticky note”. At the minimum, each note should contain: (1) the name of the Deliverable; (2) the value of the Deliverable in Story Points; and (3) the name of the person responsible for the Deliverable. Small “sticky notes” can be used to add additional information about the Deliverables.
Story Points	Scale-free estimates of the relative workload required to complete a particular Deliverable. The story points are also used to “weigh” grades associated with individual Deliverables.
Burndown Chart	A Burndown Chart shows team progress over time. The line on the chart starts at the number representing the total value (e.g. in person-hours) of all items in the Product Backlog. When a Deliverable is completed (see the Definition of Done), the line “goes down”. The slope of the line gives information about the “velocity” of the group work, meaning how productive the team is in completing the work. A “flat line” may suggest that not much work is being done and students may miss the final deadlines. A “steep line” with a negative slope may suggest good progress.
Backlog	The To Do column on the Scrum Board. Contains all the group project Deliverables that need to be completed. The Deliverables in the Backlog are arranged in an order that reflects dependencies among the Deliverables and the overall plan for completing the project created by each Group.
Scrum Master	A person conducting Weekly Scrums and making sure each group member has all the tools and information to be productive. For the first few Sprints, the role can be performed by the instructor, but will eventually be transferred to a leader of each of the groups.
Product Owner	This is the instructor teaching the course who explains to students what needs to be done for each of the Deliverables, provides feedback on the submitted Deliverables, and assigns grades.
Group	A permanent group of 4-7 students working together on a group project for the duration of the entire semester.
Sprint	A period of time (usually 1-2 weeks) during which students work on a particular portion of the group project. By default, each Sprint is one week long, although it can be longer if a weekly session is canceled due to holidays or exams.
Weekly Scrum	An in-class group work session that consists of two meetings. The first meeting is among the Group members. During this meeting, the Group members discuss what has been done and what needs to be done for the next Sprint. Based on this discussion, the Scrum Board and the Burndown Chart are updated. Other issues can be discussed during the Group meeting. The second meeting is a stand-up meeting between the group and the instructor. During these meetings students report to the instructor on what has (or has not) been done, the issues they are facing, and the items they commit to work on for the next Sprint.

Definition of Done	A Deliverable is done when it is submitted via the Learning Management System (LMS) for grading; only then the Scrum Board and the Burndown Chart can be updated.
Peer Evaluation	An online survey asking students to provide quantitative and qualitative evaluation of their own contribution to group work together with the contribution of each of their group members. This feedback is used to assign the peer-evaluation grade component for each of the students.

Each of the phases of the Educational Scrum Process described in more detail below, together with explanations of how various Educational Scrum elements described in Table 1 can be used in each of the phases.

Phase 1: Forming Groups

Some time should be allocated to let students form groups of 4-7 people.

In a classroom environment, students may need time to find a group of people they would like to work with. An instructor can announce a “firm” deadline of when all the groups should be formed (with some slack time added given that some students may not be able to find their groups by this deadline). An easy in-class group activity can be used to facilitate the creation of these groups. For example, an instructor can ask students to get together into groups (or put them into groups randomly) and come up with an idea for a new product. This will force students to collaborate and to get to know each other.

To facilitate the creation of teams, the instructor can also ask students to play several rounds of the so-called “10 things in common” game.

10 Things in Common Game

Divide people into small teams and give each team a piece of paper. Teams have an allotted time to find 10 things in common with one another. Tell everyone not to use easy cop-outs, such as "we both have hands". Once teams have found 10 things in common, they share their discoveries with the class. Several rounds of this game can be played, with each new round involving new groups of people. This will allow students to get to know someone new each time this game is played.

If some students do not find their groups by the deadline, the instructor should assign them to groups “manually”. The instructor should avoid forming a “leftover group” composed of all those who could not find a group before the deadline. These “leftover teams” are almost guaranteed to show poor performance throughout the entire duration of a project, leading to unpleasant experiences and bad grades for students. Instead, they should be assigned to other groups. Sometimes, this may require increasing the size of some teams that were already formed.

While an instructor should be flexible in relation to team size in order to accommodate all students, going below 4 people in a group or above 7 people is not recommended. If there are less than 4 people on a team, then some of the valuable team dynamics may not be present. Having more than 7 people in a team can lead to a situation where coordination and cooperation becomes difficult and makes it easier for some team members to “slack off”.

It can be announced initially, that the minimum group size is 4 and the maximum group size is 6. If necessary, the instructor can assign people to groups that have less than 6 people. It is also possible to add one more person to a group of 6. Going above 7 is not recommended.

Once groups are finalized, the instructor can consider conducting more focused and performance-oriented team building activities to help students get to know each other better and get into a working mode as a team. For example, the instructor can play one or more rounds of the so-called “Marshmallow Tower Game” to promote teamwork, problem-solving, creativity, and experimentation within the teams.

The Marshmallow Tower Game

The Marshmallow Tower game is a popular team-building activity that involves building the tallest possible tower using spaghetti sticks, tape, and a marshmallow. The goal is to construct a freestanding structure that can support the marshmallow at the top. Here are the basic rules of the game:

- Materials: Each team is provided with dry spaghetti sticks, tape, and a marshmallow. The specific quantities may vary depending on the facilitator's instructions.
- Team Formation: Divide participants into teams of usually three to five members. The ideal team size can be determined based on the number of participants and the available resources.
- Time Limit: Set a specific time limit for the activity, typically ranging from 10 to 20 minutes. This time constraint encourages teams to work efficiently and make quick decisions.
- Construction Constraints: Teams are allowed to use only the provided materials (spaghetti sticks and tape) to build their tower. The marshmallow must be placed on top of the tower.
- Tower Height: The height of the tower is measured from the base to the highest point where the marshmallow is supported. The tower must be freestanding, meaning it cannot be propped against walls or other structures.
- Winning Criteria: The team that builds the tallest freestanding tower with the marshmallow on top, as measured at the end of the time limit, is declared the winner.

Additional Guidelines:

- Teams should plan and collaborate on their tower design before starting construction.
- Spaghetti sticks can be broken into smaller pieces if desired, but they cannot be taped together.
- Teams should use the tape strategically to stabilize the structure and connect the spaghetti sticks.

- The marshmallow must be placed on the top of the tower and remain there without any additional support.

The Marshmallow Tower game promotes teamwork, problem-solving, creativity, and experimentation. It encourages participants to think outside the box, iterate on their designs, and learn from failures. The game emphasizes the importance of collaboration, communication, and effective resource utilization to achieve the objective of building a tall and stable structure.

Phase 2: Educating Students about Agile and Scrum

Once groups are formed, the instructor should educate students about Agile philosophy and Scrum methodology. This can be done in the form of a lecture or workshop that relies on instructor PowerPoint slides, videos, and hands-on team activities.

A good starting point for introducing Agile to students is contrasting it with the so-called traditional, “waterfall” approach to project management, where the development process is highly structured and sequential. After that, the instructor can introduce students to the so-called “Agile Manifesto” available at https://agilemanifesto.org/.

There are also several educational videos available on YouTube that introduce one to both Agile and Scrum and draw the connection between the two. One of these videos is a TED talk by Jeff Sutherland, one of the original creators of Scrum, titled “Scrum: How to do twice as much in half the time”. Currently, the video is available here:

Scrum: How to do twice as much in half the time | Jeff Sutherland | TEDxAix

After introducing students to Agile, the instructor can ask students to play a team game that allows them to explore the main principles of Agile and learn about teamwork from various group dynamics (both positive and negative) that happen during a game. The recommended game is called “The Ball Game”. The main rules and procedures of the game are described below.

The Ball Game

Overview: Exposes you to the main principles of Agile. You will work as a team to pass as many balls as you can through all team members during each round

Getting Started:

- Form teams of ~4-6 people
- Get a bucket with balls of different color

The main goal: Work as a team to pass as many balls of different color as possible through all team members

Protocol:

The facilitator should introduce students to the following game protocol

- You will have 2 minutes to prepare
- There will be 3-4 iterations of 2 minutes each
- After each iteration, you will tell me how many balls you managed to pass and there will be a 2 minute discussion among the team members on how to improve team work
- There will be a debriefing session after the iterations

Rules:

- The facilitator can start with the rules listed below. To make things interesting, these rules can be modified between rounds. Additional rules can be added as well.
- Each team member must touch each ball
- Each ball must have air time each time it is passed from one member to another one
- The color of the ball has to change for every passing

Artifacts:

To play the game, the instructor needs to have lightweight, colorful balls. The number of balls should be proportional to the number of teams participating in the game. At least 50 balls per team can be recommended. Also, each team needs to have two buckets: one bucket for keeping the balls at the start of the game, and one bucket for moving the balls into. It would be great if the buckets that you use have different colors for each team (e.g. red and blue). Also, you may want to use a stopwatch and a scoreboard to keep track of games scores:

Color Balls

Buckets

	BLUE	RED
Round 1		
Round 2		
Round 3		
Round 4		

Game Scoreboard

The Scrum framework can be introduced by asking students to read the Scrum Guide by Jeff Sutherland and Ken Scwaber, which is usually available online in the open access. This white paper is fairly concise and provides a good and simple introduction into the main principles, processes, and artifacts of Scrum. An instructor can also use the slides on Scrum created and distributed in an open source fashion by Mountain Goat Software LLC: https://www.mountaingoatsoftware.com/agile/scrum/resouces/a-reusable-scrum-presentation

After Scrum is introduced, the instructor can conduct a few rounds (or Sprints) of the “New Video Game Ad”

The Video Game Ad

Overview: This game exposes you to the main values, processes, and artifacts of Scrum YOu will work as a team to create a magazine ad for a new video game.

Getting Started:

- Form teams of ~4-6 people
- Create a Scrum Board for your team
- Place User Stories in the form of post-it notes into the To-Do" column
- Appoint people to roles: Scrum Master, Cutter, Writer, Gluer, Drawer(s)
- Place a few User Stories in the "Doing" tab for the first Sprint

The main goal: to create a magazine ad for a new video game.

Protocol:

- You will have 15 minutes to prepare
- There will be 3 iterations (or sprints) 12 minutes each
- For each Sprint, you will select items from the "to do" column to work on by placing them into the "doing" column; you cannot change your Sprint backlog during a Sprint
- After each iteration, you will tell the facilitator what you have done, what you are planning to do during the next Sprint, and whether you have any questions.
- There will be a debriefing session at the end of the game

Rules:

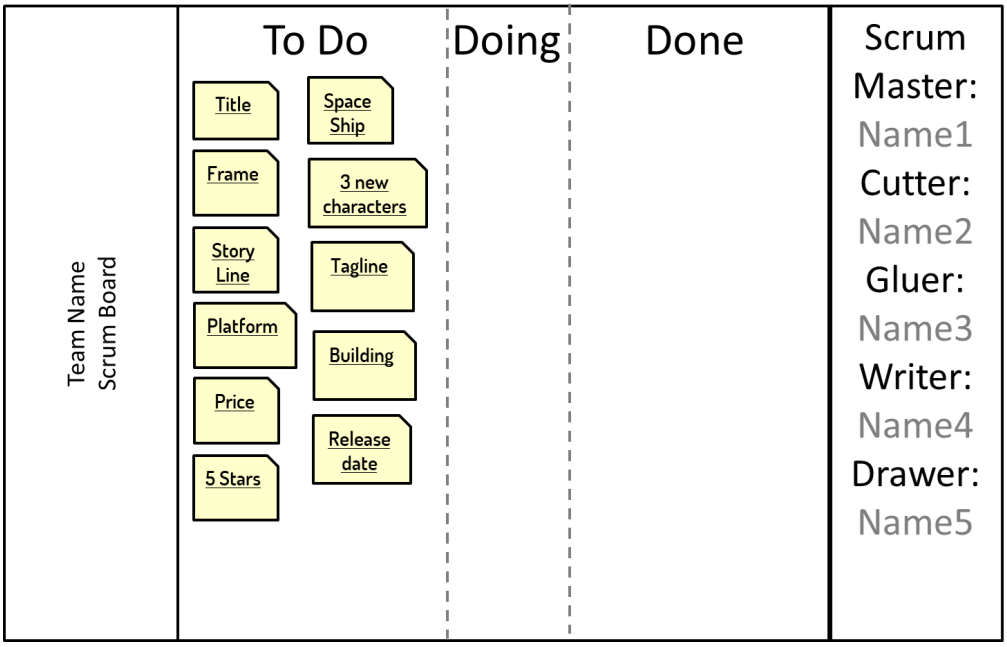
- Work has to be carried out in accordance with assigned roles:
    - None of the team members can do anything unless the Scrum Master tells them to do something; everybody is free to discuss things
    - Only a Gluer can glue things using tape and glue
    - Only a Drawer can draw things on the poster using markers and pencils
    - Only a Writer can write things on the poster using markers and pencils
    - Only a Cutter can cut things using scissors
- You cannot move additional tasks into the "doing" tab during an interaction
- Tasks can only be moved into the "done" tab when they are final
- You cannot use any devices like cell phones
- You must use all three design elements: markers, pencils, and colored paper

Artifacts:

The following items are needed for each team:

- Markers with assorted colors
- Pencils with assorted colors
- Child-safe scissors
- A ruler
- Colored paper
- Clear tape
- Eraser
- A poster board for creating a Scrum Board

- Post-it notes



Sample Scrum Board

Phase 3: Creating Scrum Artifacts

After forming groups and learning about Scrum and Agile, students should be able to create their own Scrum artifacts: Scrum Board, Burndown Chart, and “sticky notes” for the Deliverables or User Stories.. All these artifacts can be created electronically by either students or the instructor, using such tools as Trello (see Figure 12).

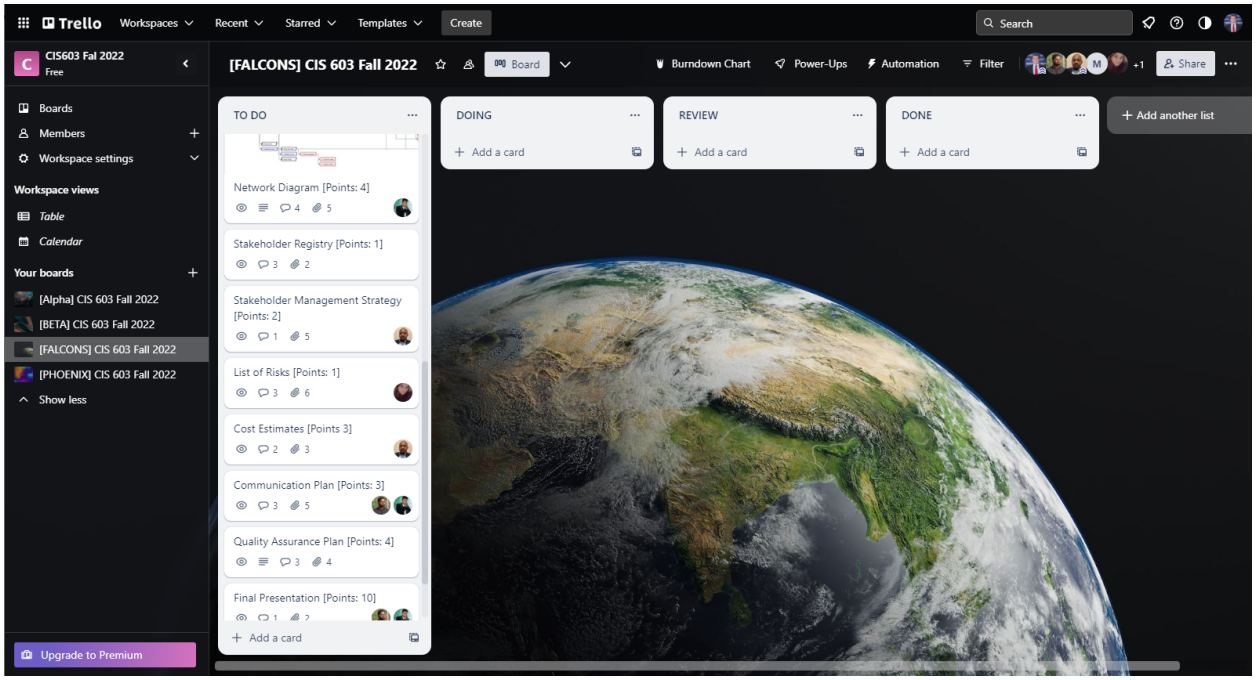


Figure 12. A Scrum Board Created In Trello

Trello features a number of useful add-ons or “Power-Ups” that can be used to create additional Scrum Artifacts, such as the Burndown Chart (see Figure 13) or an online communication channel via Slack. However, availability of these add-ons is often a function of the type of Trello account (free or premium) that the instructor has. The terms associated with these accounts are subject to constant change too.

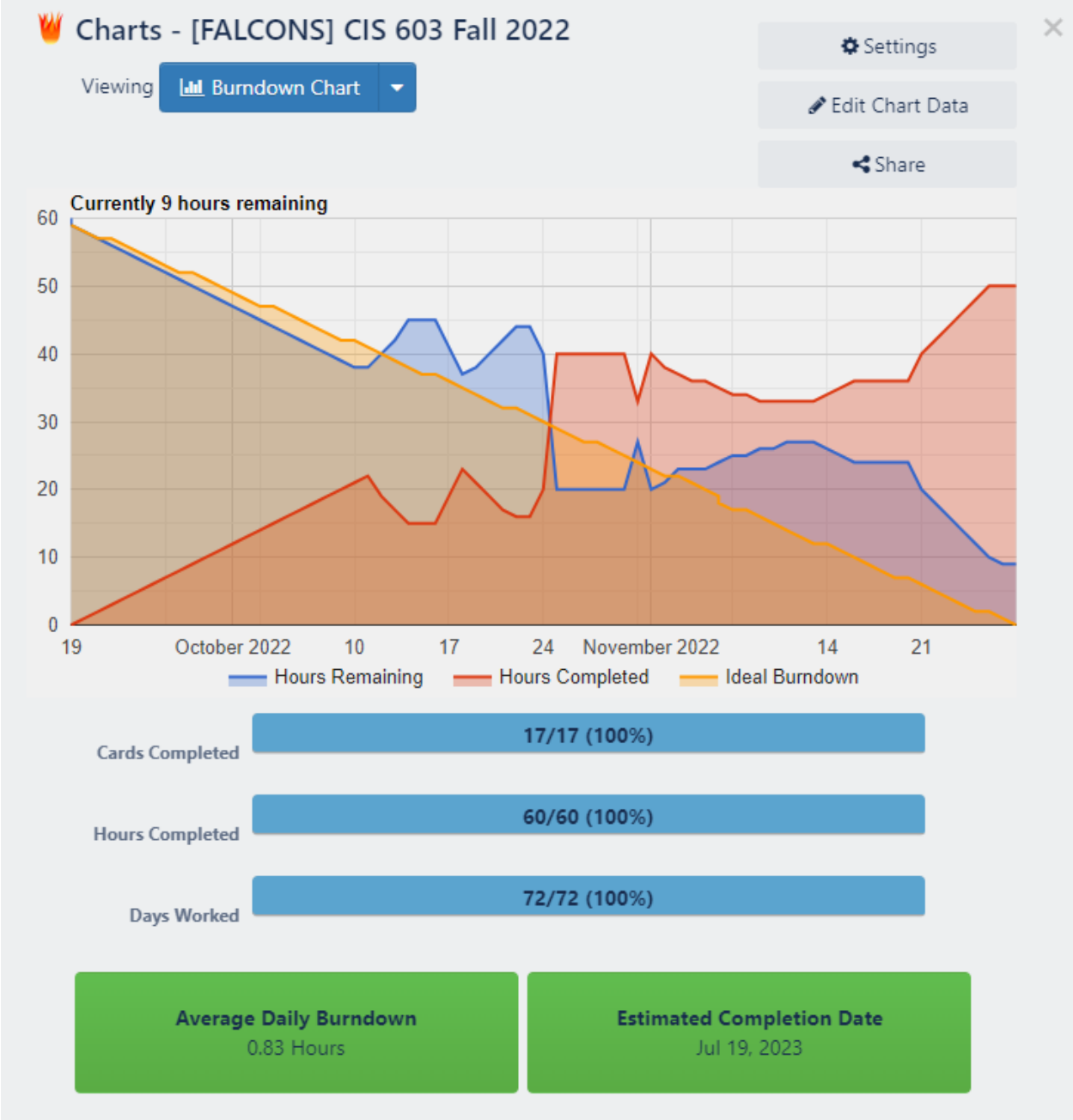


Figure 13. Burndown Chart Power-Up in Trello

The instructor may choose to create these artifacts himself or herself for each team to save time and also to add more structure to team work from the very beginning. This can be easily done by reusing Scrum Boards and cards from previous classes.

While many digital tools can be used to create Scrum artifacts and facilitate teamwork under Scrum, preference should be given to physical artifacts (e.g. physical Scrum Board and physical Burndown Chart made from poster boards) to create opportunities for quality, face-to-face interactions in class.

Creating physical Scrum artifacts requires acquiring certain supplies (see Table 3). Students should be asked to create these artifacts themselves, since this facilitates hands-on learning and better retention of theoretical material in relation to Scrum.

Table 2. Supplies for Creating Scrum Artifacts

Item	Quantity	Additional Notes
Poster Boards	1-2 per team	These poster boards will be used for creating Scrum Boards – one per team. If an instructor wishes to have a separate board for a Burndown Chart, then additional poster boards (one per team) should be acquired. Preferably, the board used for the Burndown Chart should contain a square grid. This will make plotting easier.
Markers	1 pack per team	At the minimum, the instructor should have one marker per team. Ideally, each team should have access to a set of markers of various colors. Students usually like to use several colors for creating their Scrum Boards and Burndown Charts.
Large “Sticky Notes”	1 stack per team	“Sticky notes” (e.g. Post-it® Notes) are to be used by students to capture the information about the Deliverables (or “user stories”) comprising their Backlog. At the minimum, these notes should contain: (1) the name of the Deliverable; (2) the value in terms of Story Points (can be indicated by putting thick dots using a marker; and (3) the person in charge.
Small “Sticky Notes”	1 stack per team	Small “sticky notes” can be used by students to add additional information to their “user stories” (captured with the help of large “sticky notes”).
Mobile Tripods	1 per team	Mobile tripods are used for mounting Scrum Boards and Burndown Charts during “stand-up” meetings between team members and the instructor.
Rulers	1 per team	Long rulers (depending on the size of the poster boards used) to assist students with drawing lines on their Scrum Boards and Burndown Charts.
Pencils and Erasers	1 per team	Pencils are used by students to sketch layouts of their Scrum Boards and Burndown Charts.
Storage Box	1	A box for storing rulers, “sticky notes”, markers, pencils, etc. The instructor can give out these supplies during Weekly Scrum sessions and put them away once group work is over.
Folders	1 per team	These folders are used for storing physical copies of templates for each of the Deliverables. This allows students to have something in front of them when they discuss the details of each of the Deliverables.

To encourage experimentation and innovation, the instructor can deliberately give students only rough guidelines on how these artifacts should look (e.g. by showing some examples from previous courses). Students can create different versions of Scum Boards and Burndown Charts

of the course of four semesters. One representative example of a Scrum Board and Burndown Chart is provided in Figure 2.

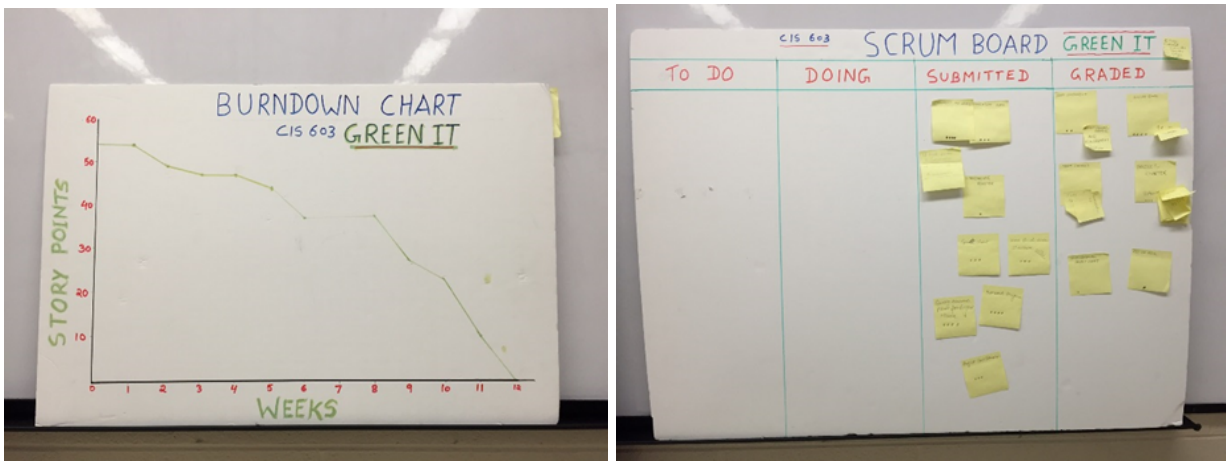


Figure 14. Scrum Board and Burndown Chart Example

Over the years, many teams converged on the following version of a combined Scrum Board and Burndown Chart:

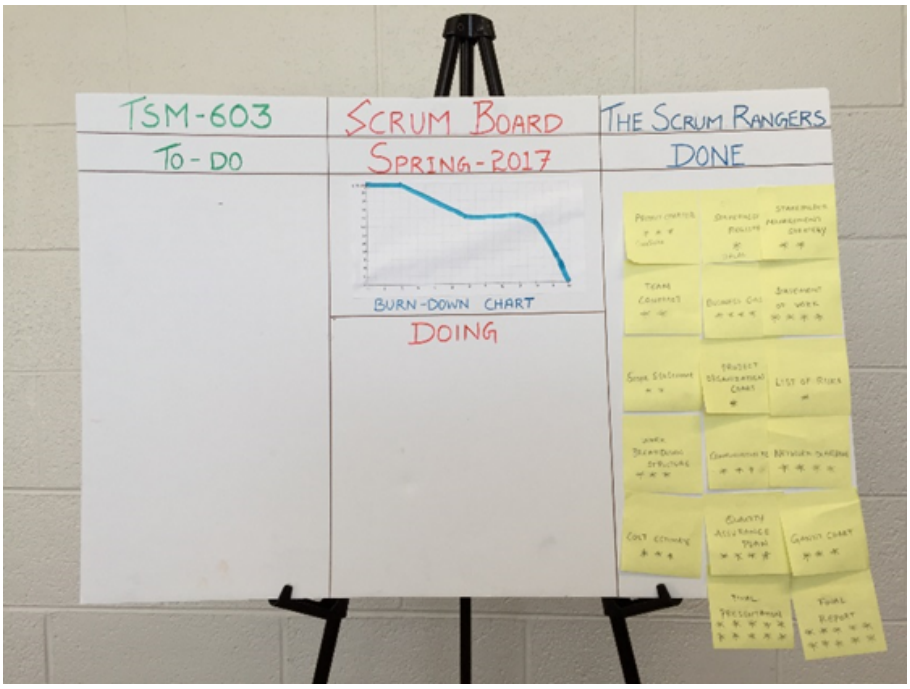


Figure 15. Scrum Board and Burndown Chart Combined

Note that the Scrum Board in Figure 15 is simpler (it has three columns) and also includes a small Burndown Chart in the middle (made from a sheet of graphing paper). This makes it

cheaper and simplifies the logistics associated with Weekly Scrums (e.g. only one tripod is needed to mount this board).

Phase 4: Creating a Backlog

To start working on the group project, students need to familiarize themselves with the Deliverables comprising the group project and to create a Backlog. An example of Deliverables comprising a Backlog for a group project involving the creation of an IT Project Management Plan is provided in Table 3 below.

Table 3. Group Project Deliverables

Deliverable	Story Points
1. Team Contract	2
2. Project Organization Chart	1
3. Project Charter	3
4. Business Case	4
5. Statement of Work	4
6. Scope Statement	2
7. Work Breakdown Structure	3
8. Gantt Chart	3
9. Network Diagram	4
10. Cost Estimate	3
11. Stakeholder Registry	1
12. Stakeholder Management Strategy	2
13. List of Risks	1
14. Communication Plan	3
15. Quality Assurance Plan	4
16. Final Presentation	10
17. Final Report	10
Total:	60

Each of the Deliverables listed in Table 3 is a stand-alone document representing a Project Management document or artifact. Working on these Deliverables enables students to get hands-on understanding of the main artifact and processes in modern Project Management. At the end of the semester, students are asked to compile all these Deliverables into a comprehensive Project Management Plan for an IT project of their choice. Thus, these deliverables become sections of a Project Management Plan.

The instructor can provide templates for each of the Deliverables together with examples of completed deliverables either in digital (e.g. by uploading files to a Learning Management System) or physical (e.g. by creating printouts and putting them into team folders) formats. Physical copies are needed so that each team has copies of the required Deliverables in front of

them in class. Usually, students bring smartphones or laptops to class, so they can also access these templates online via Canvas.

Creating a Backlog requires using a large “sticky note” to capture information (or “user story”, as it is called in Scrum) for each of the Deliverables comprising the group project. At the minimum, each of these large notes should contain: (1) the name of the Deliverable; (2) the value in terms of Story Points (can be indicated by putting thick dots using a marker) (3) the person in charge of the Deliverable. If a Trello board is used, then these “user stories” can be captured via “cards” in trello; the same task information can be added to these cards. Moreover, in Trello, a user has numerous formatting options for these cards (e.g. using images or color-coding).

The instructor explains to students that several people can collaborate on a particular Deliverable. In fact, this is often necessary as individual Deliverables overlap or rely on several other Deliverables. However, there should be one person in charge of each Deliverable. This person will be responsible for finalizing the Deliverable and submitting it to the instructor via Canvas for grading. Small “sticky notes” can be used to subdivide large Deliverables into sub-parts or to simply add additional notes to each of the Deliverables.

Once a note or digital card is created for each Deliverable, students need to place it into the “To Do” column of their Scrum Boards in the order in which these Deliverables should be completed. This requires students to develop a shared understanding of what it will take to complete the entire project. The instructor acts as a consultant to make sure students arrange the Deliverables in an order that reflects the actual dependencies among them. For example, students cannot work on their Gantt Chart or Network Diagram unless their Work Breakdown Structure is finalized. Ordering the notes in the “To Do” column completes the creation of the Backlog.

Once each team creates a Backlog, the instructor asks students to select a few items to work on for the next week (or Sprint). The “sticky notes” or Trello cards devoted to these items are placed into the “Doing” column of the Scrum Board. Some Groups make very ambitious plans for the first week selecting several labor-intensive items. The instructor can advise them to start with a Project Charter and a Scope Statement. Spending more time on these brief yet important documents can put the entire project on the right track.

Phase 5: Working in Sprints

Once the Backlog is created, Groups should be ready to start working in Sprints. It is communicated to students that, by default, each Sprint will be one week long. Groups are asked to commit to a certain amount of work for each Sprint (week). Sometimes Sprints are longer than 1 week due to class cancellations (e.g. due to holidays or exams). Students seem to prefer longer Sprints, as this gives them more time to work on their Deliverable. Yet the instructor has found that one week Sprints allow students to intervene earlier in case a student is “stuck” or simply fails to do any work.

At the end of each Sprint, the instructor asks each team to have a Weekly Scrum. The Weekly Scrum consists of two meetings. First, students are asked to convene with their Group members to discuss what has (or has not) been done the week prior to that and what needs to be done for next week. Before asking students to break into Groups, the instructor gives students an overall feedback on the submissions. During the Group meeting, students need to develop a shared understanding of what has been done and what needs to be done. After this, Groups need to update their Scrum Board and Burndown Chart (if some items were actually completed). The second meeting happens closer to the end of the Weekly Scrum. It is a “stand-up” meeting between the Group and the instructor. This requires the entire Group to come forward to the instructor and place their Scrum Board and Burndown Chart in front of the Instructor. During the first Sprints, the instructor acts as a Scrum Master and Product Owner. The instructor asks the team the following three questions:

1. What have you done during this Sprint?
2. What issues are you facing in your work?
3. What are you planning to do for the next Sprint?

Going through these questions is usually enough to generate a brief yet useful discussion among the Group members and the instructor. After a few weeks, the instructor can usually identify a “natural leader” within each Group and transfer the responsibilities of a Scrum Master to him or her. This can be done by telling the leader that he or she needs to go through these three questions with the Group in front of the instructor. The instructor can help the student Scrum Master “interrogate” the Group members who do not actively participate in these “strand-up” meetings. It is especially important for the instructor to politely ask those Group members who have not accomplished anything during the Sprint or whether they need any help. This is usually enough to put “peer pressure” on the lagging team members without the Group leader having to do any disciplining in front of everyone. The instructor remains the Product Owner for the rest of the semester, acting as the ultimate authority on what needs to be done for the group project.

In order for a student to say that something has been done during a Sprint, a specific Deliverable needs to be submitted via Canvas. Submission to Canvas constitutes the so-called Definition of Done for this Scrum adaptation. This rule is reiterated and strictly enforced by the instructor throughout the semester. Sometimes students forget the rule and report things as being done without actually submitting them to Canvas. This probably happens due to the fact that these students are embarrassed to admit in front of everyone that they have not done anything during the Sprint.

Sometimes students bring up some grading issues during Weekly Scrums. For example, they may remind the instructor that an important Deliverable has not been graded. They may also ask the instructor to clarify some of the feedback that was given to them within the document (the instructor grades each document in canvas by inserting in-text notes and highlights for each of the documents submitted). The instructor may open this assignments on his class computer and discuss those issues with students. The instructor makes sure that all submissions for the

week are graded at least one day prior to each class. This gives students some time to look at the feedback and raise meaningful questions during Weekly Scrum meetings.

If students wish to resubmit a particular Deliverable and have it re-graded, they need to obtain instructor permission. The instructor grants this permission it is important to redo a particular Deliverable to increase the quality of the entire project. Ideally, the instructor wants to give unlimited submission attempts for each of the Deliverables. This will help the students to proceed with their group project in a truly iterative fashion. However, this is simply not possible when a class is large. But students can “resubmit” all of their Deliverables when they submit their Final Report, which is a compilation of all Deliverables submitted during the semester.
At the end of each stand-up meeting with each Group, the instructor gives the team a concluding message to motivate them. For example, if the instructor feels that the team is making good progress, then the instructor will praise the entire team. If the instructor thinks that the team is lagging and has noticed certain quality issues with the submitted work (e.g. poor writing style or lack of understanding of some fundamental concepts in project management), the instructor will bring it up and emphasize the need to take some corrective actions from the team. If the class is too big or the instructor simply cannot remember the details of the feedback given to each of the teams, then notes should be taken prior to the class and used as memory aids for formulating this “weekly message” to the team.

Some corrective actions may be required on the instructor side as well. For example, poor writing quality has been an issue with several teams due to the fact that the majority of students enrolled in the class are international students. The instructor addresses this problem by conducting additional in-class workshops on effective technical writing and proper referencing using APA style.

Phase 5: Closing Phase

After going through several Sprints, teams should be done with the project. Once the final deliverables for the project are submitted and the Burndown Chart is “brought to zero”, the instructor can ask students to complete a simple online survey (created using Google Forms) where they provide quantitative and qualitative Peer Evaluations of their own contribution to the group project and the contributions of their peers. The feedback is provided using a “single blind” procedure. While each student sees his or her grade for the peer evaluation component of the group project, the student will not see individual scores assigned by students and the qualitative feedback.

This feedback is used by the instructor to calculate the peer evaluation grade component for each of the students. Typically, the instructor averages out the quantitative scores given to a particular student by each of the group members. Qualitative feedback is used by the instructor to validate the numeric scores given. For example, if somebody gives a very low score to another student, then it will be explained that the student being evaluated has not contributed much to the group project at all due to poor attendance. The instructor informs the students that

he reserves the right to delete “outliers” – scores that are either too high or too low relative to others and are not backed by convincing justifications.

References

Manifesto for Agile Software Development (2021). Retrieved from https://agilemanifesto.org/

Mountain Goat Software (2023). Reusable scrum presentation. Retrieved from https://www.mountaingoatsoftware.com/agile/scrum/resources/a-reusable-scrum-presentation

Schwaber, K., & Sutherland, J. (2020). The scrum guide. Scrum.org. Retrieved from https://scrumguides.org/

Takeuchi, H., & Nonaka, I. (1986). The new product development game. *Harvard business review*, *64*(1), 137-146.