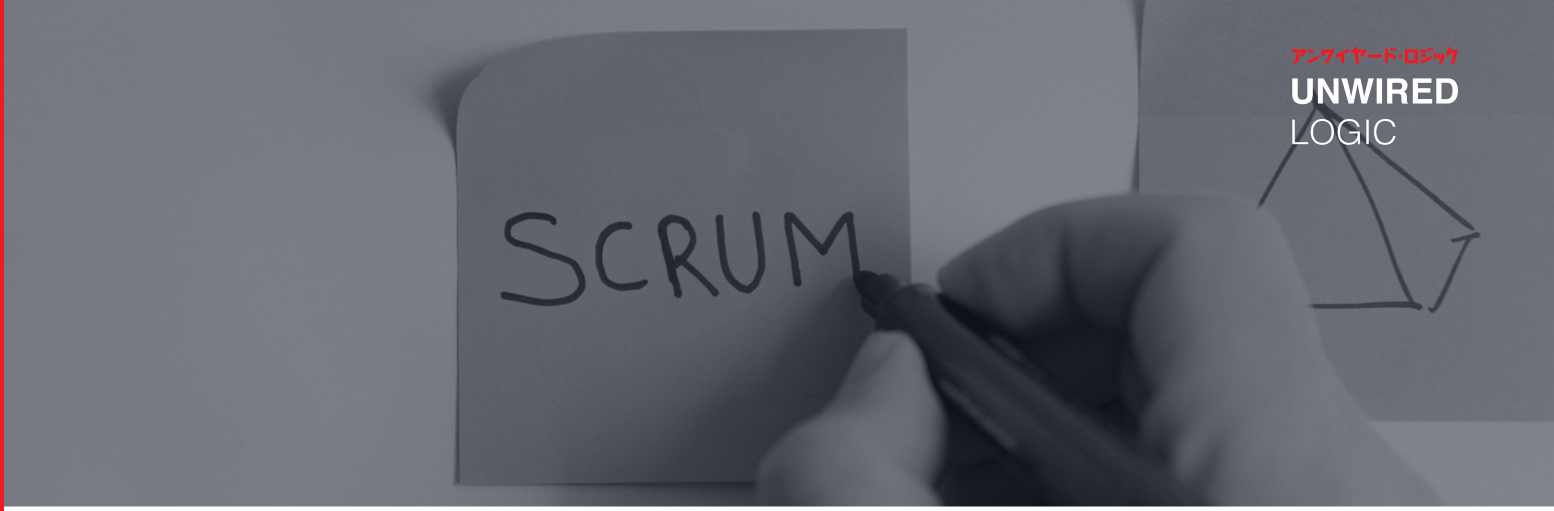
アンワイヤード·ロジック
UNWIRED
LOGIC

DO BETTER SCRUM

AUTHORS
ANNA SHENGELIA
AARON FARNEY

unwiredlogic.com

CONTENTS



SCRUM - UNDERSTANDING THE BACKGROUND

What is Agile & SCRUM

Invented in the early 2000s, Agile quickly gained popularity and became one of the most prolific project management styles. Before long, organizations were jumping at the framework and rethinking how their teams and projects worked. Agile is a great way to bring order into the chaos of a project, and to bring something from idea to working project. However, not everything labeled Agile is agile. This document will introduce the underlying concepts of Agile methodologies, particularly SCRUM, and provide practical advice for project managers to make their projects agile, not only in name.

To begin, it is important to understand the difference between Agile and SCRUM. Often the two terms are used as synonyms. While they are related, they do not mean the same. Agile is an umbrella term covering different approaches with the same principles and values. SCRUM is the most widely known Agile Framework. It serves to organize work to maximize value for the user: breaking down complex projects into smaller pieces to deliver value continuously and more frequently. In other words, we implement SCRUM at the product development level, while Agile concerns the organization.

The concept of Agile is simple to understand – yet deceptively difficult to implement correctly

The Agile Manifesto - A set of values and principles:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

While there is value in the items on the right, we value those on the left more.

Why Choose Agile / SCRUM

The 20/80 rule or the Pareto Principle states that 80% of the value comes from 20% of the features. SCRUM is a manifestation of this principle. We emphasize building the high-priority items first, bringing value early on. The methodology challenges you to question the business value of incremental improvements. If it doesn't deliver value – why build it? Because of this paradigm shift, SCRUM projects typically release a useable product to customers faster and with a higher overall user satisfaction rating. By breaking down the project in increments, the project team can tackle even complex and previously impossible projects. Overall agile teams report higher productivity, team morale, and better output quality.

Theory X & Theory Y

A critical difference between SCRUM and traditional project management is that with SCRUM, we empower the team to contribute through their intrinsic motivation. Traditional project management approaches projects with a command-and-control approach. The project manager and sponsor make most of the decisions and dole out tasks for the team.

McGregor's theory X and Y delves into human motivation and management techniques. Theory X states that workers dislike their work and have little inherent motivation. Therefore, they must be supervised to ensure that they get their work done. On the other hand, Theory Y implies that given the right environment and working conditions, people will perform well because they take pride in their work. It gives credit to intrinsic motivation and workers' ability to self-manage their work.

SCRUM essentially follows the approach of theory Y. It gives the team flexibility to make decisions which ultimately creates ownership and a higher level of engagement in their work. The superior results speak for themselves.

Defined vs. Empirical Planning

Every project starts with uncertainty and assumptions. A traditional waterfall project begins with listing the assumptions under which it will proceed. The project management team pools their experience, hoping that they covered everything. Then they formulate a watertight plan for the project under those conditions and proceed blindly. That is Defined Planning. This approach is suboptimal for many project types, where SCRUM ultimately became the preferred choice. Taking software development as an example, many factors are subject to oftentimes quick changes based on user feedback or market factors. Defined planning as an approach is not flexible and adaptable enough.

For this reason, SCRUM uses Empirical Planning. The team plans based on experience and the available data. Experimentation is key. As new inputs and insights come up, they integrate them. Consequently, progress is based on observation and implementation from the previous sprint. Rather than anticipating and estimating the unexpected, we expect the unexpected to materialize. Because there is no fixed definition of the outcome and features, empirical planning encourages learning and adaptation during the process. Change is not a risk, but a requirement of the approach.

The empirical approach builds on three pillars:

- Transparency – the process must be visible for everyone related to the project.
- Inspection – Before setting a goal, the team must inspect if the goal can be successful.
- Adaptation – After inspecting errors or issues in the project, the team must adapt to improve, even if it requires changing the project process to avoid further deviations.

ESSENTIAL SCRUM LINGO & CONCEPTS

SCRUM Roles

Product Owner

The product owner is the visionary and ultimate owner of the product and the pipeline to the stakeholders.

SCRUM Master

Essentially the mother of the team. The SCRUM Master ensures that the team follows the process, uses the available tools, and clears roadblocks.

SCRUM Team

The ideal team is between 3 and 7 people in size. Keeping the same people in the team is vital to ensure optimum velocity.

Typically, the team will undergo various stages of development over the course of the project:

- Forming – unable and unwilling to work together.
- Storming – unable but willing to move forward, team cohesion forms.
- Norming – able and willing but lack confidence in their abilities.
- Performing – able, willing, and confident – the team is now self-managing and cross-functional. The members effectively manage technical risk, enjoy collective ownership, are committed, estimate the work, and develop high priority features.

Experts

Subject matter experts brought in when necessary. They do not form a part of the core team.

SCRUM Structure

This section highlights the key structures of SCRUM, which are timeboxed rituals.

The Sprint

Sprints are typically 1, 2 or 4 weeks in length. 2 weeks is the most common length. Most training materials on SCRUM is based on a 2-week sprint as well. We recommend 1-week sprints as we have found that they produce a better result. It is easier to stay focused during a shorter sprint, as well.

If you are having difficulties finishing stories in a 2-weeksprint, it is likely that the sprint is too long for your team. Move to a 1-week sprint.

Sprint Planning

During the sprint planning, the product owner explains the vision to the team. Sprint planning allows the team to decompose that vision into tasks for that sprint. You should spend up to 2 hours maximum for a 1-week sprint.

Daily SCRUM

A daily 15-minute ritual that happens at the same time and same place. All team members ask and answer 3 questions:

What did you do yesterday? → What am I going to do today? → Are there any impediments or roadblocks?

The daily SCRUM is merely a place to report and share. If any discussions are needed, they are done outside of the Daily SCRUM meeting and not during.

Product Backlog Grooming

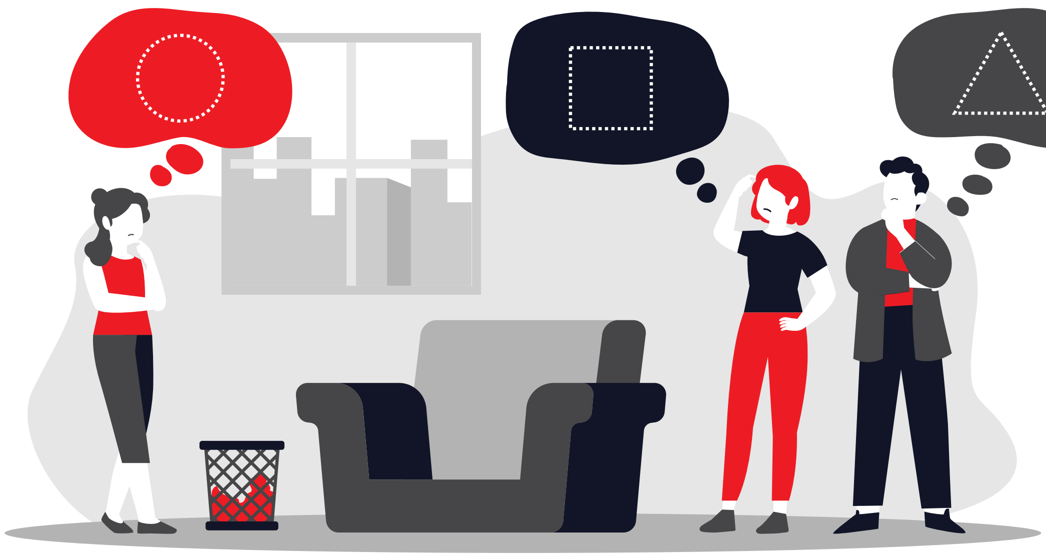
The project backlog is the amount of work not completed for the entire project and the sprint backlog is the amount of work not yet completed for a sprint.

In practice, backlog grooming helps ensure that the upcoming prints worth of user stories are ready for sprint planning. Maintaining good backlog grooming habits ensures that the right stories get prioritized. Regular backlog meetings help avoid rework and increase clarity on requirements for teammates. Consequently, they help to make the sprints more effective. Constantly and consistently clarifying future stories should take 5 to 15% of the total sprint time.

Sprint Review

The main goal of the sprint review is to demo the product to the stakeholders. It takes around an hour for a 7-day sprint. However, there is a bigger purpose, making it one of the most important ceremonies in SCRUM. Hand-on experience or a presentation helps the product owners and stakeholders understand the implemented changes. Additionally, it helps to assess if more work is needed and how it will factor into the planning – in other words updating the product backlog.

The sprint review also provides a regular opportunity to engage with stakeholders and collect their feedback. This improves team cohesion. Quality assurance and maximized responsiveness to customers are other benefits of the session.



Retrospective

If the Sprint Review focuses on the product, then Sprint Retrospective is all about the team. It is a regular opportunity to evaluate what needs improvement and what went well. Retrospectives help to make the team work better together, feel more motivated, and become faster.

How to SCRUM

There are many ways to implement SCRUM, but this section outlines what we found to be the most effective.

Sprint Zero

Sprint Zero is what we typically call a short meeting to create a vision and a rough product backlog, which helps to estimate the product release. There is no standard for how long Sprint Zero should take. Every project is unique, and the length varies. Sprint Zero is essentially prep time before project kick-off. Take a few hours to discuss what kick-off readiness means for this project with the team. Identify all required items for the project to start successfully.

Typically, in a Sprint Zero, the team discusses the following items:

- Architecture & Design – This is a valid exercise, but more importantly, it is essential to have this discussion at the beginning to avoid getting locked in further down the road. In a way, it is a "get it out of our systems" topic.
- Team training – Identify skill gaps early on and plan for necessary training.
- Product vision – Craft it during Sprint Zero and disseminate it to all stakeholders as early as possible.
- Setting up -Technical environments should be largely completed during Sprint Zero. This should include the development, testing, staging, and live environments. If possible, complete deployment tools, needed to move the product from environment to environment at this stage.

- Define done- Debate and finalize the definition of done. *Refer to DoD in the "Best Practices" section for more information*
- Initial story workshop – Stories or themes are created and planned out per the release planning strategy. Stories that will be happening in early sprints should be fleshed out in more detail.
- Minimum viable solution– Figure out the minimal required solution for launch and create an initial release plan.

It is paramount for all participants to understand that everything discussed in Sprint Zero is subject to change due to inspection and adaptation.

Story Writing

A story is an informal and general explanation of a product feature from the perspective of a user. They are not just a list of system requirements. User stories are essential to establish and maintain a user-centric mindset throughout the project. The story must help the team know the why and what they are building, and the value it will create for the user. Stories should encompass an independent working experience that one can test and deploy on its own.

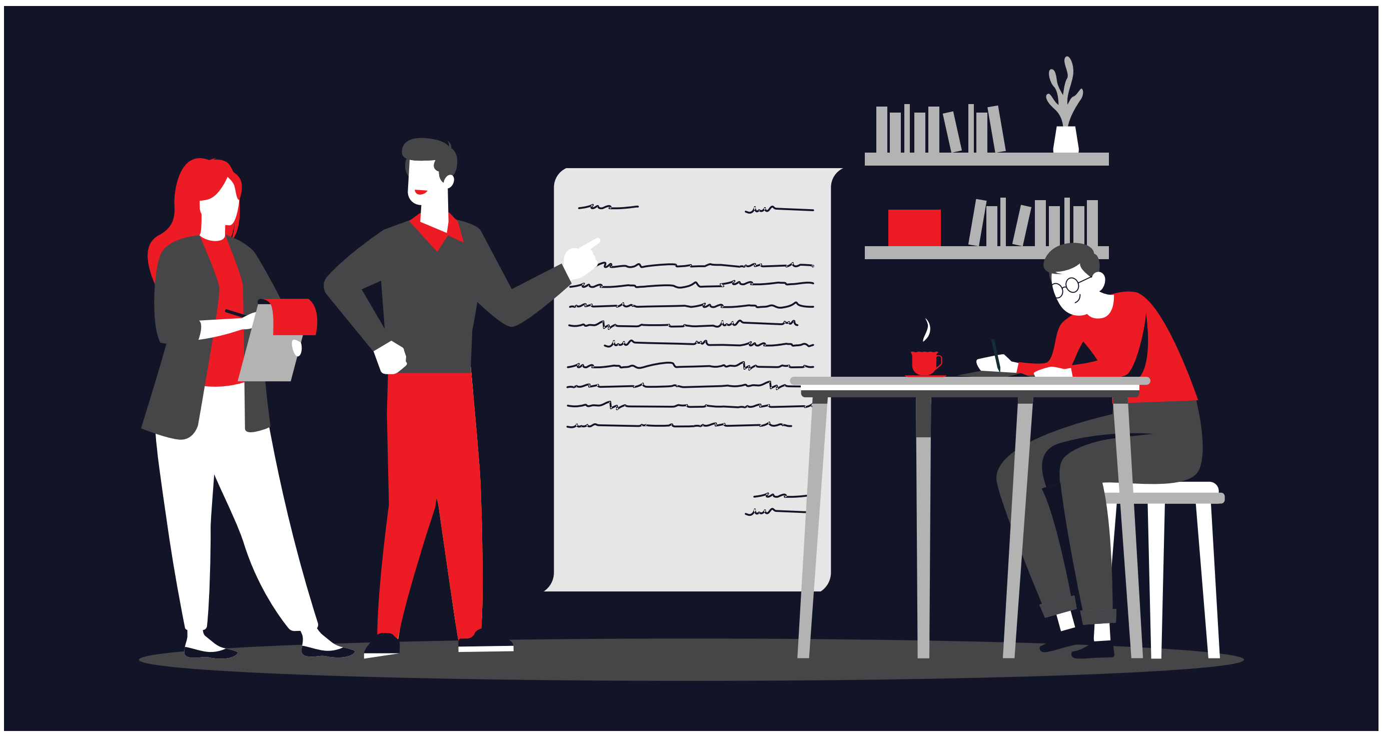
The typical structure of a story is as follows, and we will only attempt 2 or 3 stories in a one-week sprint:

- Title
- As a <user> I can/ want <action> so that <result>.

SCRUM focuses on results and a working product. The path to the result matters less. Stories follow a similar approach: focus on the experience you want to deliver with the story, not the features. To do this effectively, we need to understand the persona of the customer and their specific customer needs.

Afterward, break down stories into tasks.

Refer to "Invest in your Product Backlog" to understand how to get the most out of your stories.



Estimating

Estimating story size in story points will feed into the release planning. This way, we can understand when we will likely attempt to build a specific story and when it can be completed.

Planning poker is used to estimate story points, and all members must agree on the values of the story points before even considering estimating a story size. It is important to note that points are relative sizing and don't correspond to hours. The goal is to group similar size items together for comparison rather than providing a precise estimate.

You must consider3 different aspects when estimating story size: Complexity, Effort, and Doubt.

Complexity covers the things we need to figure out. We know we can solve the, but we must determine exactly what we want to do.

Effort is the amount of work required. You may know how to complete the job, but it may take time to do so.

Doubt are the items we are unsure about - either because we are looking for the solution in the wrong place or because we don't consider the technology as up to the task yet.

Release Planning

This is the exercise of defining the minimum viable solution. What experience do we want to deliver to the customer? This experience or vision is the glue for the release, and all the stories should support that vision. This will determine the minimum set of functions required for the release and answer the question "when will we be done?"

It is important to note that without team velocity, release planning is difficult.

The team will also conduct capacity planning and commit to the amount of work in the sprint. If something must be completed but doesn't fit into a specific story, creating a task, and working on it in a sprint is ok.

Grooming

Grooming is typically conducted mid-sprint and should take between 1 to 3 hours. Grooming is used to talk about upcoming stories and helps ensure a common understanding within the team. The definition of ready is used to guide items that need to be completed.

Understanding the Definition of Ready

Here are the things needed for a story to be moved from the product backlog to the sprint backlog:

- Acceptance tests written
- Any necessary spikes completed
- UX is ready
- Story has been groomed
- Story is estimated – ideally, stories are all similar in size

Sprint Planning

At the beginning of every project, the product owner needs to describe their vision to the team. This will be the inspiration to the team in developing the product. In addition to the product vision, visions must be updated or explained – *as needed* – for specific sprints or releases.

The product owner will present the vision alongside any high value/high-risk stories. The team will take that vision and stories and decompose the stories into tasks. Unlike stories, tasks are estimated in hours.

Spikes

A spike, sometimes also called proof of concept, is a timeboxed task or user story to research, gather information, or answer a question. Rather than creating a shippable product, the focus is on a user story. Usually, a spike is needed when a team can't adequately estimate a task or story without further research. The product owner can proactively allocate time and team capacity so that when the story comes into the sprint, the team already knows what to do. The spike must take place well before the sprint where that story is planned. Never do a spike during the sprint of that story – that is like attempting to build a plane mid-flight.

Retrospective

The retrospective is a time for the team to reflect and ultimately improve on their methods and teamwork. It is vital that a plan for the retrospective is created and verbalized during every retrospective. The retrospective should be fun and interactive so that people want to be there and participate constructively. These meetings are a chance for everyone to list out what we like, don't like, and a chance to rant. Create three columns on a whiteboard for each section, and each person must write down 5 to 10 items and put them in a column. Each person gets three votes – write initials on three stickies that you think are most important to prioritize them. Talk about the items with the most votes first. If the retrospectives aren't useful or people aren't putting up the real issues – then it is an indicator of team trust issues.

BEST PRACTICES

INVEST in Your Product Backlog

INVEST is an acronym and guideline for a widely accepted checklist for story quality assessment. If a story does not meet any criteria, the team must reword or rewrite it. Stories are the essence and nucleus of agile methodology. INVEST helps to make sure that the stories are written correctly. Good user stories facilitate collaboration and keep everyone in sync by promoting a mutual understanding between team members and users. Stories are also essential for the value-added delivery of agile methodologies.

- Independent – stories that are dependent on others inherently lead to problems when estimating and prioritizing.
- Negotiable – items are not written in stone. Make sure you leave or imply some flexibility.
- Valuable – ensure that items are written in a customer-orientated manner, so they are valuable to the end-user,not the developer.
- Estimable – you must be able to estimate the time or size of a story.
- Small–complex items are naturally large. Items that contain multiple items are probably multiple stories.
- Testable – each story should have a user acceptance test associated with it

Prototyping

Prototypes have several benefits. A prototype is an early sample or model of the final product. We use it to test a concept or gain valuable insights from the creation process for the real deal. A single prototype can help the team visualize and understand the product better than dozens of user stories. It is an effective way to communicate the designer's intent and ensure that all members have a common understanding of the product. By creating something that everyone can touch and experience – you get better and more concrete feedback. Some might think of prototyping as investing time in making a disposable product. But SCRUM encourages us to experiment and learn from our mistakes.

Understanding the Definition of Done (DoD)

The definition of done gives the team a fundamental guideline when to consider a user story as actually finished. Each project is unique, so this must be defined well in advance. The task level is the most detailed, but each step builds on the previous DoD. One size DOES NOT fit all! The sections below provide examples and general guidelines for DoD for each part of the project or sprint.

Discipline	Area	Details
Prep	Environmental most ready	Development environment is ready with all third-party tools configured. Staging environment is ready. Test data for the selected features has been created. Live environment prepared.
	Design drafted	Strategy for design created and agreed upon by SCRUM team. Design completes by using wireframes or prototypes. UML diagrams created.
	Documentation	Documentation strategy defined and agreed upon by team. Definition of Done tailored for the specific project. Process maps created or updated. Policy or regulations identified. Training manuals updated.
General	User Story Clarity	Stories selected have been selected to fit into the product or sprint theme. Stories have been discussed by the team to ensure everyone knows what it means.
	Tasks Identified	Tasks are typically the DoD – any exceptions should be documented for a particular story.
Coding	Unit testing is done	Unit testing complete and bug-free:
	Code Refactoring	Source code has been refactored to make it compre-hensive, maintainable, and amenable to change. *A common mistake is not to keep refactoring in the definition of done. If not taken seriously, refactoring normally spills out to the next sprint or, worse, is com-pletely ignored.*

	Error Handling/ Validation	Enumerate error cases and how each should be handled. For example, if a user performs the steps in the wrong order, how will the software handle it? Specify which fields need validation and what kind of validation is required *Ideally, the story contains this information as it is important to get the product owner's blessing. However, if not defined, the SCRUM team can make decisions or recommendations to the product owner at any time during the sprint.*
	Performance Expectations	If performance is of paramount concern, the story should specify the response time: If it doesn't contain it, the developer should apply best effort performance if they think that is appropriate or seek feedback from the team.
Review		Automated code review has been completed using the supported tools / technologies. Violations have been shared with the team, and the team has resolved all discrepancies to adhere to the coding standard. (Automated code reviews should be hooked up with CI builds.)
	Peer reviews	Peer reviews are done *If pair programming is used, a separate peer review session might not be required.*
	Release Build	Build and packaging • A Build (successful) is done using a continuous integration framework. • Change log report has been generated from the code library, and release notes have been created. Deliverables have been moved to the release area. Build deployment in the staging environment • Build deliverables are deployed in the staging environment.

Testing	Transaction testing done Automated Testing	Unit test cases are written - In most sprints, the unit test will be the story. Automated testing - All types of automated test cases have been executed, and a test report has been generated. All incidents/defects are reported. Manual testing – the SCRUM team has reviewed the reports generated from automation testing and conducted necessary manual test cases to ensure that tests are passing. All incidents/defects are reported. Build issues - If any integration or build issues are found, the necessary steps are taken to rectify the issues – if not, the issues are entered into the product backlog.
	Integration testing	Regression testing ensures that defects have not been introduced in the unchanged area of the software. Performance and/or stress testing ensures to describe how the system responds when it is under stress, such as many users or transactions. Acceptance test – testers have discretion when conducting the tests, but the story should be referred to. The following questions should be satisfied: 1. Each story is tested for usability – is it easy to use? 2. Each story tested for performance – is it responsive? *A common mistake is not to keep performance testing in the definition of done. This is an important aspect. Most performance issues are design issues and are hard to fix at a later stage.*
Closure		All finished user stories/tasks are marked complete/resolved. Remaining hours for the task are set to zero before closing the task. Other necessary/optional activities such as things that are very specific to the project – security audit, deployable on multiple platforms (OS versions, browser versions, etc.).

FINAL WORDS

Agile is more than just a project management methodology that simplifies software development. It is a mindset. A SCRUM is successful when the stakeholders are on the same page and not afraid of making (and learning) from mistakes. Change will happen, and nothing is set in stone, but transparency, inspection, and adaptation are key to success. By focusing on the customer and on delivering value-added products in small increments, even the most mammoth of projects can be tackled.